Reference Manual Pages for SunOS 4.1.2

Sun Microsystems, Inc. • 2550 Garcia Avenue • Mountain View, CA 94043 • 415-960-1300

Part No: 800-6641-10
Revision A of December 1991

Reference Manual Pages for SunOS 4.1.2

This package provides "man pages" for new and updated topics in the *SunOS Reference Manual.* Replace the appropriate pages in your copy of the *SunOS Reference Manual*, if you have one.

The following man pages are new or changed for SunOS 4.1.2:

man/man1/SCCS/s.Intro.1
man/man1/SCCS/s.arch.1
man/man1/SCCS/s.mps.1 (new)
man/man1/SCCS/s.mpstat.1 (new)
man/man3/SCCS/s.mblen.3
man/man4/SCCS/s.Intro.4
man/man4/SCCS/s.audio.4
man/man4/SCCS/s.bwtwo.4s
man/man4/SCCS/s.cgsix.4s
man/man4/SCCS/s.cgtwelve.4s
man/man4/SCCS/s.gt.4s
man/man4/SCCS/s.id.4s
man/man4/SCCS/s.ipi.4s
man/man4/SCCS/s.is.4s
man/man4/SCCS/s.le.4s
man/man4/SCCS/s.mcp.4s
man/man4/SCCS/s.mem.4s
man/man4/SCCS/s.mtio.4
man/man4/SCCS/s.openprom.4s
man/man4/SCCS/s.sd.4s
man/man4/SCCS/s.sr.4s
man/man4/SCCS/s.st.4s
man/man4/SCCS/s.vx.4s
man/man4/SCCS/s.zs.4s
man/man5/SCCS/s.fstab.5
man/man8/SCCS/s.Intro.8
man/man8/SCCS/s.boot.8s
man/man8/SCCS/s.colldef.8
man/man8/SCCS/s.config.8
man/man8/SCCS/s.devinfo.8s

man/man8/SCCS/s.eeprom.8s
man/man8/SCCS/s.fsck.8
man/man8/SCCS/s.gt_lpconfig.8
man/man8/SCCS/s.gtconfig.8
man/man8/SCCS/s.modload.8
man/man8/SCCS/s.monitor.8s
man/man8/SCCS/s.mount.8
man/man8/SCCS/s.openboot.8s (new)
man/man8/SCCS/s.sundiag.8
man/man8/SCCS/s.unixname2bootname.8

NAME

intro – introduction to commands

DESCRIPTION

This section describes publicly accessible commands in alphabetic order. Commands of general utility, many with enhancements from 4.3 BSD.

Pages of special interest have been categorized as follows:

1C Commands for communicating with other systems.

1G Commands used primarily for graphics and computer-aided design.

1V System V commands. One or more of the following are true:

- The man page documents System V behavior only.
- The man page documents default SunOS behavior, and System V behavior as it differs from the default behavior. These System V differences are presented under **SYSTEM V** section headers.
- The man page documents behavior compliant with *IEEE Std 1003.1-1988* (POSIX.1).

SEE ALSO

- Section 8 in this manual for system administration procedures, system maintenance and operation commands, local daemons, and network-services servers.
- Section 7 in this manual for descriptions of publicly available files and macro packages for document preparation.
- Section 6 in this manual for computer games.
- *SunOS User's Guide: Getting Started*
- *SunOS User's Guide: Customizing Your Environment*
- *SunView User's Guide*
- *SunOS User's Guide: Doing More*
- *Programming Utilities and Libraries*

DIAGNOSTICS

Upon termination each command returns two bytes of status, one supplied by the system giving the cause for termination, and (in the case of "normal" termination) one supplied by the program, see **wait**(2V) and **exit**(2V). The former byte is 0 for normal termination, the latter is customarily 0 for successful execution, nonzero to indicate troubles such as erroneous parameters, bad or inaccessible data, or other inability to cope with the task at hand. It is called variously "exit code," "exit status" or "return code," and is described only where special conventions are involved.

LIST OF COMMANDS

Name	Appears on Page	Description
acctcom	**acctcom**(1)	search and print process accounting files
adb	**adb**(1)	general-purpose debugger
addbib	**addbib**(1)	create or extend a bibliographic database
adjacentscreens	**adjacentscreens**(1)	connect multiple screens to SunView window driver
admin	**sccs-admin**(1)	create and administer SCCS history files
aedplot	**plot**(1G)	graphics filters for various plotters
alias	**csh**(1)	C shell built-in commands, see **csh**(1)
align_equals	**textedit_filters**(1)	filters provided with **textedit**(1)
apropos	**apropos**(1)	locate commands by keyword lookup
ar	**ar**(1V)	create library archives, and add or extract files
arch	**arch**(1)	display the architecture of the current host
as	**as**(1)	assembler
at	**at**(1)	execute a command or script at a specified time
atq	**atq**(1)	display the queue of jobs to be run at specified times
atrm	**atrm**(1)	remove jobs spooled by at or batch
awk	**awk**(1)	pattern scanning and processing language
banner	**banner**(1V)	display a string in large letters
bar	**bar**(1)	create tape archives, and add or extract files
basename	**basename**(1V)	display portions of pathnames and filenames
batch	**at**(1)	execute a command or script at a specified time
bc	**bc**(1)	arbitrary-precision arithmetic language
bg	**csh**(1)	C shell built-in commands, see **csh**(1)
bgplot	**plot**(1G)	graphics filters for various plotters
biff	**biff**(1)	give notice of incoming mail messages
bin-mail	**bin-mail**(1)	an early program for processing mail messages
break	**csh**(1)	C shell built-in commands, see **csh**(1)
breaksw	**csh**(1)	C shell built-in commands, see **csh**(1)
cal	**cal**(1)	display a calendar
calendar	**calendar**(1)	a simple reminder service
cancel	**lp**(1)	send/cancel requests to a printer
capitalize	**textedit_filters**(1)	filters provided with **textedit**(1)
case	**csh**(1)	C shell built-in commands, see **csh**(1)
cat	**cat**(1V)	concatenate and display
cb	**cb**(1)	a simple C program beautifier
cc	**cc**(1V)	C compiler
cd	**cd**(1)	change working directory
cdc	**sccs-cdc**(1)	change the delta commentary of an SCCS delta
cflow	**cflow**(1V)	generate a flow graph for a C program
checkeq	**eqn**(1)	typeset mathematics
checknr	**checknr**(1)	check nroff and troff input files for errors
chfn	**passwd**(1)	change local or NIS password information
chgrp	**chgrp**(1)	change the group ownership of a file
chkey	**chkey**(1)	create or change encryption key
chmod	**chmod**(1V)	change the permissions mode of a file
chsh	**passwd**(1)	change local or NIS password information
clear	**clear**(1)	clear the terminal screen
clear_colormap	**clear_colormap**(1)	clear the colormap to make console text visible
clear_functions	**clear_functions**(1)	reset the selection service to clear stuck function keys
click	**click**(1)	enable or disable the keyboard's keystroke click
clock	**clock**(1)	display the time in an icon or window

cluster	**cluster**(1)	find optional cluster containing a file
cmdtool	**cmdtool**(1)	run a shell (or program) using the SunView text facility
cmp	**cmp**(1)	perform a byte-by-byte comparison of two files
col	**col**(1V)	filter reverse paper motions for terminal diplay
colcrt	**colcrt**(1)	filter nroff output for a terminal lacking overstrike capability
coloredit	**coloredit**(1)	alter color map segment
colrm	**colrm**(1)	remove characters from specified columns within each line
comb	**sccs-comb**(1)	combine SCCS deltas
comm	**comm**(1)	display lines in common between two sorted lists
compress	**compress**(1)	compress or expand files, display expanded contents
continue	**csh**(1)	C shell built-in commands, see **csh**(1)
cp	**cp**(1)	copy files
cpio	**cpio**(1)	copy file archives in and out
cpp	**cpp(1)**	the C language preprocessor
crontab	**crontab**(1)	install, edit, remove or list a user's crontab file
crtplot	**plot**(1G)	graphics filters for various plotters
crypt	**crypt**(1)	encode or decode a file
csh	**csh**(1)	shell with a C-like syntax and advanced interactive features
csplit	**csplit**(1V)	split a file with respect to a given context
ctags	**ctags**(1)	create a tags file for use with ex and vi
ctrace	**ctrace**(1V)	generate a C program execution trace
cu	**cu**(1C)	connect to remote system
cut	**cut**(1V)	remove selected fields from each line of a file
cxref	**cxref**(1V)	generate a C program cross-reference
date	**date**(1V)	display or set the date
dbx	**dbx**(1)	source-level debugger
dbxtool	**dbxtool**(1)	SunView interface for dbx source-level debugger
dc	**dc**(1)	desk calculator
dd	**dd**(1)	convert and copy files with various data formats
default	**csh**(1)	C shell built-in commands, see **csh**(1)
defaultsedit	**defaultsedit**(1)	edit default settings for SunView utilities
defaults_from_input	**defaultsedit**(1)	edit default settings for SunView utilities
defaults_from_input	**input_from_defaults**(1)	update current state of the mouse and keyboard
defaults_to_indentpro	**defaultsedit**(1)	edit default settings for SunView utilities
defaults_to_mailrc	**defaultsedit**(1)	edit default settings for SunView utilities
delta	**sccs-delta**(1)	make a delta to an SCCS file
deroff	**deroff**(1)	remove nroff, troff, tbl and eqn constructs
des	**des**(1)	encrypt or decrypt data using Data Encryption Standard
desktop	**desktop**(1)	switch the window system to be invoked upon login
df	**df**(1V)	report free disk space on file systems
diff	**diff**(1)	display line-by-line differences between pairs of text files
diff3	**diff3**(1V)	display line-by-line differences between 3 files
diffmk	**diffmk**(1)	mark differences between versions of a troff input file
dircmp	**dircmp**(1V)	compare directories
dirname	**basename**(1V)	display portions of pathnames and filenames
dirs	**csh**(1)	C shell built-in commands, see **csh**(1)
dis	**dis**(1)	object code disassembler for COFF
disablenumlock	**enablenumlock**(1)	enable or disable the numlock key
domainname	**domainname**(1)	set or display name of the current NIS domain
dos	**dos**(1)	SunView window for IBM PC/AT applications
dos2unix	**dos2unix**(1)	convert text file from DOS format to ISO format
du	**du**(1V)	display the number of disk blocks used per directory or file
dumbplot	**plot**(1G)	graphics filters for various plotters

dumpkeys	**loadkeys**(1)	load and dump keyboard translation tables
e	**ex**(1)	line editor
echo	**echo**(1V)	echo arguments to the standard output
ed	**ed**(1)	basic line editor
edit	**ex**(1)	line editor
egrep	**grep**(1V)	search a file for a string or regular expression
eject	**eject**(1)	eject media device from drive
else	**csh**(1)	C shell built-in commands, see **csh**(1)
enablenumlock	**enablenumlock**(1)	enable or disable the numlock key
end	**csh**(1)	C shell built-in commands, see **csh**(1)
endif	**csh**(1)	C shell built-in commands, see **csh**(1)
endsw	**csh**(1)	C shell built-in commands, see **csh**(1)
enroll	**xsend**(1)	send or receive secret mail
env	**env**(1)	obtain or alter environment variables
eqn	**eqn**(1)	typeset mathematics
error	**error**(1)	categorize compiler error messages, insert at source file lines
eval	**csh**(1)	C shell built-in commands, see **csh**(1)
ex	**ex**(1)	line editor
exec	**csh**(1)	C shell built-in commands, see **csh**(1)
exit	**csh**(1)	C shell built-in commands, see **csh**(1)
expand	**expand**(1)	expand TAB characters to SPACE characters, and vice versa
expr	**expr**(1V)	evaluate expressions as logical, arithmetic, or string
false	**true**(1)	provide truth values
fdformat	**fdformat**(1)	format diskettes
fg	**csh**(1)	C shell built-in commands, see **csh**(1)
fgrep	**grep**(1V)	search a file for a string or regular expression
file	**file**(1)	determine the type of a file by examining its contents
find	**find**(1)	find files by name, or by other characteristics
finger	**finger**(1)	display information about users
fmt	**fmt**(1)	simple text and mail-message formatters
fmt_mail	**fmt**(1)	simple text and mail-message formatters
fold	**fold**(1)	fold long lines for display
fontedit	**fontedit**(1)	a vfont screen-font editor
foption	**foption**(1)	determine available floating-point code generation options
foreach	**csh**(1)	C shell built-in commands, see **csh**(1)
from	**from**(1)	display the sender and date of newly-arrived mail messages
ftp	**ftp**(1C)	file transfer program
gcore	**gcore**(1)	get core images of running processes
get	**sccs-get**(1)	retrieve a version of an SCCS file
get_alarm	**set_alarm**(1)	SunView programmable alarms
getoptcvt	**getopts**(1)	parse command options in shell scripts
getopt	**getopt**(1V)	parse command options in shell scripts
getopts	**getopts**(1)	parse command options in shell scripts
get_selection	**get_selection**(1)	copy contents of SunView selection to the standard output
gfxtool	**gfxtool**(1)	run graphics programs in a SunView window
gigiplot	**plot**(1G)	graphics filters for various plotters
glob	**csh**(1)	C shell built-in commands, see **csh**(1)
goto	**csh**(1)	C shell built-in commands, see **csh**(1)
gprof	**gprof**(1)	display call-graph profile data
graph	**graph**(1G)	draw a graph
grep	**grep**(1V)	search a file for a string or regular expression
groups	**groups**(1)	display a user's group memberships
hashcheck	**spell**(1)	report spelling errors

hashmake	**spell**(1)	report spelling errors
hashstat	**csh**(1)	C shell built-in commands, see **csh**(1)
head	**head**(1)	display first few lines of specified files
help	**sccs-help**(1)	help regarding SCCS error or warning messages
help_viewer	**help_viewer**(1)	provide help with SunView applications and desktop
history	**csh**(1)	C shell built-in commands, see **csh**(1)
hostid	**hostid**(1)	print the numeric identifier of the current host
hostname	**hostname**(1)	set or print name of current host system
hpplot	**plot**(1G)	graphics filters for various plotters
i386	**machid**(1)	return a true exit status if the processor is of the indicated type
iAPX286	**machid**(1)	return a true exit status if the processor is of the indicated type
iconedit	**iconedit**(1)	create and edit images for icons, cursors and panel items
id	**id**(1V)	print the user name and ID, and group name and ID
if	**csh**(1)	C shell built-in commands, see **csh**(1)
implot	**plot**(1G)	graphics filters for various plotters
indent	**indent**(1)	indent and format a C program source file
indentpro_to_defaults	**defaultsedit**(1)	edit default settings for SunView utilities
indxbib	**indxbib**(1)	create an inverted index to a bibliographic database
inline	**inline**(1)	in-line procedure call expander
input_from_defaults	**defaultsedit**(1)	edit default settings for SunView utilities
input_from_defaults	**input_from_defaults**(1)	update the current state of the mouse and keyboard
insert_brackets	**textedit_filters**(1)	filters provided with **textedit**(1)
install	**install**(1)	install files
ipcrm	**ipcrm**(1)	remove a message queue, semaphore set, or shared memory ID
ipcs	**ipcs**(1)	report interprocess communication facilities status
jobs	**csh**(1)	C shell built-in commands, see **csh**(1)
join	**join**(1)	relational database operator
keylogin	**keylogin**(1)	decrypt and store secret key
keylogout	**keylogout**(1)	delete stored secret key
kill	**kill**(1)	send a signal to a process, or terminate a process
label	**csh**(1)	C shell built-in commands, see **csh**(1)
last	**last**(1)	indicate last logins by user or terminal
lastcomm	**lastcomm**(1)	show the last commands executed, in reverse order
ld	**ld**(1)	link editor, dynamic link editor
ldd	**ldd**(1)	list dynamic dependencies
ld.so	**ld**(1)	link editor, dynamic link editor
leave	**leave**(1)	remind you when you have to leave
lex	**lex**(1)	lexical analysis program generator
limit	**csh**(1)	C shell built-in commands, see **csh**(1)
line	**line**(1)	read one line
lint	**lint**(1V)	a C program verifier
ln	**ln**(1V)	make hard or symbolic links to files
load	**load**(1)	load clusters
loadc	**load**(1)	load clusters
loadkeys	**loadkeys**(1)	load and dump keyboard translation tables
lockscreen_default	**defaultsedit**(1)	edit default settings for SunView utilities
lockscreen_default	**lockscreen**(1)	maintain SunView context and prevent unauthorized access
lockscreen	**lockscreen**(1)	maintain SunView context and prevent unauthorized access
logger	**logger**(1)	add entries to the system log
login	**login**(1)	log in to the system
logname	**logname**(1)	get the name by which you logged in
logout	**csh**(1)	C shell built-in commands, see **csh**(1)
look	**look**(1)	find words in the system dictionary or lines in a sorted list

lookbib	**lookbib**(1)	find references in a bibliographic database
lorder	**lorder**(1)	find an ordering relation for an object library
lp	**lp**(1)	send/cancel requests to a printer
lpq	**lpq**(1)	display the queue of printer jobs
lpr	**lpr**(1)	send a job to the printer
lprm	**lprm**(1)	remove jobs from the printer queue
lpstat	**lpstat**(1)	display the printer status information
lptest	**lptest**(1)	generate lineprinter ripple pattern
ls	**ls**(1V)	list the contents of a directory
lsw	**lsw**(1)	list TFS whiteout entries
m4	**m4**(1V)	macro language processor
m68k	**machid**(1)	return a true exit status if the processor is of the indicated type
mach	**mach**(1)	display the processor type of the current host
machid	**machid**(1)	return a true exit status if the processor is of the indicated type
Mail	**mail**(1)	read or send mail messages
mailrc_to_defaults	**defaultsedit**(1)	edit default settings for SunView utilities
mailtool	**mailtool**(1)	SunView interface for the mail program
make	**make**(1)	maintain, update, and regenerate related programs and files
man	**man**(1)	display reference manual pages
mesg	**mesg**(1)	permit or deny messages on the terminal
mkdir	**mkdir**(1)	make a directory
mkstr	**mkstr**(1)	create an error message file by massaging C source files
more	**more**(1)	browse or page through a text file
mps	**mps**(1)	display status of current processes on an MP system
mpstat	**mpstat**(1)	show multi-processor usage
mt	**mt**(1)	magnetic tape control
mv	**mv**(1)	move or rename files
nawk	**nawk**(1)	pattern scanning and processing language
neqn	**eqn**(1)	typeset mathematics
newgrp	**newgrp**(1)	log in to a new group
nice	**nice**(1)	run a command at low priority
nl	**nl**(1V)	line numbering filter
nm	**nm**(1)	print symbol name list
nohup	**nohup**(1V)	run a command immune to hangups and quits
notify	**csh**(1)	C shell built-in commands, see **csh**(1)
nroff	**nroff**(1)	format documents for display or line-printer
objdump	**objdump**(1)	dump selected parts of a COFF object file
od	**od**(1V)	octal, decimal, hexadecimal, and ascii dump
old-ccat	**old-compact**(1)	compress and uncompress files, and cat them
old-compact	**old-compact**(1)	compress and uncompress files, and cat them
old-eyacc	**old-eyacc**(1)	modified yacc allowing much improved error recovery
old-filemerge	**old-filemerge**(1)	window-based file comparison and merging program
old-make	**old-make**(1)	maintain, update, and regenerate groups of programs
old-prmail	**old-prmail**(1)	display waiting mail
old-pti	**old-pti**(1)	phototypesetter interpreter
old-setkeys	**old-setkeys**(1)	modify interpretation of the keyboard
old-sun3cvt	**old-sun3cvt**(1)	convert Sun-2 system executables to Sun-3 system executables
old-syslog	**old-syslog**(1)	make a system log entry
old-uncompact	**old-compact**(1)	compress and uncompress files, and cat them
old-vc	**old-vc**(1)	version control
on	**on**(1C)	execute command on a remote system with local environment
onintr	**csh**(1)	C shell built-in commands, see **csh**(1)
organizer	**organizer**(1)	file and directory manager

overview	**overview**(1)	run a program from SunView that takes over the screen
pack	**pack**(1V)	compress and expand files
page	**more**(1)	browse or page through a text file
pagesize	**pagesize**(1)	display the size of a page of memory
passwd	**passwd**(1)	change local or NIS password information
paste	**paste**(1V)	join corresponding lines of files, subsequent lines of one
pax	**pax**(1V)	portable archive exchange
paxcpio	**paxcpio**(1V)	copy file archives in and out
pcat	**pack**(1V)	compress and expand files
pdp11	**machid**(1)	return a true exit status if the processor is of the indicated type
perfmeter	**perfmeter**(1)	display system performance values in a meter or strip chart
pg	**pg**(1V)	page through a file on a soft-copy terminal
plot	**plot**(1G)	graphics filters for various plotters
popd	**csh**(1)	C shell built-in commands, see **csh**(1)
pr	**pr**(1V)	prepare file(s) for printing, perhaps in multiple columns
printenv	**printenv**(1)	display environment variables currently set
prof	**prof**(1)	display profile data
prs	**sccs-prs**(1)	display selected portions of an SCCS history
prt	**sccs-prt**(1)	display delta table information from an SCCS file
ps	**ps**(1)	display the status of current processes
ptx	**ptx**(1)	generate a permuted index
pushd	**csh**(1)	C shell built-in commands, see **csh**(1)
pwd	**pwd**(1)	display the pathname of the current working directory
quota	**quota**(1)	display a user's disk quota and usage
ranlib	**ranlib**(1)	convert archives to random libraries
rasfilter8to1	**rasfilter8to1**(1)	convert an 8-bit deep rasterfile to a 1-bit deep rasterfile
rastrepl	**rastrepl**(1)	magnify a raster image by a factor of two
rcp	**rcp**(1C)	remote file copy
rdist	**rdist**(1)	remote file distribution program
red	**ed**(1)	basic line editor
refer	**refer**(1)	expand and insert references from a bibliographic database
rehash	**csh**(1)	C shell built-in commands, see **csh**(1)
remove_brackets	**textedit_filters**(1)	filters provided with **textedit**(1)
repeat	**csh**(1)	C shell built-in commands, see **csh**(1)
reset	**tset**(1)	establish or restore terminal characteristics
rev	**rev**(1)	reverse the order of characters in each line
ring_alarm	**set_alarm**(1)	SunView programmable alarms
rlogin	**rlogin**(1C)	remote login
rm	**rm**(1)	remove (unlink) files or directories
rmdel	**sccs-rmdel**(1)	remove a delta from an SCCS file
rmdir	**rm**(1)	remove (unlink) files or directories
roffbib	**roffbib**(1)	format and print a bibliographic database
rpcgen	**rpcgen**(1)	RPC protocol compiler
rsh	**rsh**(1C)	remote shell
rup	**rup**(1C)	show host status of local machines (RPC version)
ruptime	**ruptime**(1C)	show host status of local machines
rusers	**rusers**(1C)	who's logged in on local machines (RPC version)
rwall	**rwall**(1C)	write to all users over a network
rwho	**rwho**(1C)	who's logged in on local machines
sact	**sccs-sact**(1)	show editing activity status of an SCCS file
sccs	**sccs**(1)	front end for the Source Code Control System (SCCS)
sccs-admin	**sccs-admin**(1)	create and administer SCCS history files
sccs-cdc	**sccs-cdc**(1)	change the delta commentary of an SCCS delta

sccs-comb	**sccs-comb**(1)	combine SCCS deltas
sccs-delta	**sccs-delta**(1)	make a delta to an SCCS file
sccsdiff	**sccs-sccsdiff**(1)	compare two versions of an SCCS file
sccs-get	**sccs-get**(1)	retrieve a version of an SCCS file
sccs-help	**sccs-help**(1)	ask for help regarding SCCS error or warning messages
sccs-prs	**sccs-prs**(1)	display selected portions of an SCCS history
sccs-prt	**sccs-prt**(1)	display delta table information from an SCCS file
sccs-rmdel	**sccs-rmdel**(1)	remove a delta from an SCCS file
sccs-sact	**sccs-sact**(1)	show editing activity status of an SCCS file
sccs-sccsdiff	**sccs-sccsdiff**(1)	compare two versions of an SCCS file
sccs-unget	**sccs-unget**(1)	undo a previous get of an SCCS file
sccs-val	**sccs-val**(1)	validate an SCCS file
screenblank	**screenblank**(1)	turn off the screen when the mouse and keyboard are idle
screendump	**screendump**(1)	dump a frame-buffer image to a file
screenload	**screenload**(1)	load a frame-buffer image from a file
script	**script**(1)	make typescript of a terminal session
scrolldefaults	**defaultsedit**(1)	edit default settings for SunView utilities
sdiff	**sdiff**(1V)	contrast two text files by displaying them side-by-side
sed	**sed**(1V)	stream editor
selection_svc	**selection_svc**(1)	SunView selection service
set_alarm	**set_alarm**(1)	SunView programmable alarms
setenv	**csh**(1)	C shell built-in commands, see **csh**(1)
set	**csh**(1)	C shell built-in commands, see **csh**(1)
sh	**sh**(1)	standard UNIX system shell and command-level language
shelltool	**shelltool**(1)	run a shell in a SunView terminal window
shift	**csh**(1)	C shell built-in commands, see **csh**(1)
shift_lines	**textedit_filters**(1)	filters provided with **textedit**(1)
size	**size**(1)	display the size of an object file
sleep	**sleep**(1)	suspend execution for a specified interval
snap	**snap**(1)	SunView application for system and network administration
soelim	**soelim**(1)	resolve and eliminate .so requests from nroff or troff input
sort	**sort**(1V)	sort and collate lines
sortbib	**sortbib**(1)	sort a bibliographic database
source	**csh**(1)	C shell built-in commands, see **csh**(1)
sparc	**machid**(1)	return a true exit status if the processor is of indicated type
spell	**spell**(1)	report spelling errors
spellin	**spell**(1)	report spelling errors
spline	**spline**(1G)	interpolate smooth curve
split	**split**(1)	split a file into pieces
stop	**csh**(1)	C shell built-in commands, see **csh**(1)
strings	**strings**(1)	find printable strings in an object file or binary
strip	**strip**(1)	remove symbols and relocation bits from an object file
stty	**stty**(1V)	set or alter the options for a terminal
stty_from_defaults	**defaultsedit**(1)	edit default settings for SunView utilities
stty_from_defaults	**stty_from_defaults**(1)	set terminal editing characters from the defaults database
su	**su**(1V)	super-user, temporarily switch to a new user ID
sum	**sum**(1V)	calculate a checksum for a file
sun	**machid**(1)	return a true exit status if the processor is of indicated type
sunview	**sunview**(1)	the SunView window environment
suspend	**csh**(1)	C shell built-in commands, see **csh**(1)
sv_acquire	**sv_acquire**(1)	change owner, group, mode of window devices
sv_release	**sv_acquire**(1)	change owner, group, mode of window devices
swin	**swin**(1)	set or get SunView user input options

switcher	**switcher**(1)	switch between multiple desktops on the same physical screen
switch	**csh**(1)	C shell built-in commands, see **csh**(1)
symorder	**symorder**(1)	rearrange a list of symbols
sync	**sync**(1)	update the super block; force changed blocks to the disk
sysex	**sysex**(1)	start the system exerciser
syswait	**syswait**(1)	execute a command, suspending termination until user input
t300	**plot**(1G)	graphics filters for various plotters
t300s	**plot**(1G)	graphics filters for various plotters
t4013	**plot**(1G)	graphics filters for various plotters
t450	**plot**(1G)	graphics filters for various plotters
tabs	**tabs**(1V)	set tab stops on a terminal
tail	**tail**(1)	display the last part of a file
talk	**talk**(1)	talk to another user
tar	**tar**(1)	create tape archives, and add or extract files
tbl	**tbl**(1)	format tables for nroff or troff
tcopy	**tcopy**(1)	copy a magnetic tape
tcov	**tcov**(1)	construct test coverage analysis
tee	**tee**(1)	replicate the standard output
tek	**plot**(1G)	graphics filters for various plotters
tektool	**tektool**(1)	SunView Tektronix 4014 terminal-emulator window
telnet	**telnet**(1C)	user interface to a remote system using the TELNET protocol
test	**test**(1V)	return true or false according to a conditional expression
textedit	**textedit**(1)	SunView window- and mouse-based text editor
textedit_filters	**textedit_filters**(1)	filters provided with **textedit**(1)
tftp	**tftp**(1C)	trivial file transfer program
then	**csh**(1)	C shell built-in commands, see **csh**(1)
time	**time**(1V)	time a command
tip	**tip**(1C)	connect to remote system
toolplaces	**toolplaces**(1)	display current window locations, sizes, and other attributes
touch	**touch**(1V)	update the access and modification times of a file
tput	**tput**(1V)	initialize a terminal or query the terminfo database
tr	**tr**(1V)	translate characters
trace	**trace**(1)	trace system calls and signals
traffic	**traffic**(1C)	SunView program to display Ethernet traffic
troff	**troff**(1)	typeset or format documents
true	**true**(1)	provide truth values
tset	**tset**(1)	establish or restore terminal characteristics
tsort	**tsort**(1)	topological sort
tty	**tty**(1)	display the name of the terminal
u3b	**machid**(1)	return a true exit status if the processor is of the indicated type
u3b2	**machid**(1)	return a true exit status if the processor is of the indicated type
u3b5	**machid**(1)	return a true exit status if the processor is of the indicated type
u3b15	**machid**(1)	return a true exit status if the processor is of the indicated type
ul	**ul**(1)	do underlining
umask	**csh**(1)	C shell built-in commands, see **csh**(1)
unalias	**csh**(1)	C shell built-in commands, see **csh**(1)
uname	**uname**(1)	display the name of the current system
uncompress	**compress**(1)	compress or expand files, display expanded contents
unexpand	**expand**(1)	expand TAB characters to SPACE characters, and vice versa
unget	**sccs-unget**(1)	undo a previous get of an SCCS file
unhash	**csh**(1)	C shell built-in commands, see **csh**(1)
unifdef	**unifdef**(1)	resolve and remove ifdef'ed lines from cpp input
uniq	**uniq**(1)	remove or report adjacent duplicate lines

units	**units**(1)	conversion program
unix2dos	**unix2dos**(1)	convert text file from ISO format to DOS format
unlimit	**csh**(1)	C shell built-in commands, see **csh**(1)
unload	**unload**(1)	unload optional clusters
unloadc	**unload**(1)	unload optional clusters
unpack	**pack**(1V)	compress and expand files
unset	**csh**(1)	C shell built-in commands, see **csh**(1)
unsetenv	**csh**(1)	C shell built-in commands, see **csh**(1)
unwhiteout	**unwhiteout**(1)	remove a TFS whiteout entry
uptime	**uptime**(1)	show how long the system has been up
users	**users**(1)	display a compact list of users logged in
ustar	**ustar**(1V)	process tape archives
uucp	**uucp**(1C)	system to system copy
uudecode	**uuencode**(1C)	encode a binary file, or decode its ASCII representation
uuencode	**uuencode**(1C)	encode a binary file, or decode its ASCII representation
uulog	**uucp**(1C)	system to system copy
uuname	**uucp**(1C)	system to system copy
uupick	**uuto**(1C)	public system-to-system file copy
uusend	**uusend**(1C)	send a file to a remote host
uustat	**uustat**(1C)	UUCP status inquiry and job control
uuto	**uuto**(1C)	public system-to-system file copy
uux	**uux**(1C)	remote system command execution
vacation	**vacation**(1)	reply to mail automatically
val	**sccs-val**(1)	validate an SCCS file
vax	**machid**(1)	return a true exit status if the processor is of the indicated type
vedit	**vi**(1)	visual display editor based on **ex**(1)
vfontinfo	**vfontinfo**(1)	inspect and print out information about fonts
vgrind	**vgrind**(1)	grind nice program listings
vi	**vi**(1)	visual display editor based on **ex**(1)
view	**vi**(1)	visual display editor based on **ex**(1)
vplot	**vplot**(1)	plot graphics for a Versatec printer
vswap	**vswap**(1)	convert a foreign font file
vtroff	**vtroff**(1)	troff to a raster plotter
vwidth	**vwidth**(1)	make a troff width table for a font
w	**w**(1)	who is logged in, and what are they doing
wait	**wait**(1)	wait for a process to finish
wall	**wall**(1)	write to all users logged in
wc	**wc**(1)	display a count of lines, words and characters
what	**what**(1)	extract SCCS version information from a file
whatis	**whatis**(1)	display a one-line summary about a keyword
whereis	**whereis**(1)	locate the binary, source, and manual page for a command
which	**which**(1)	locate a command; display its pathname or alias
while	**csh**(1)	C shell built-in commands, see **csh**(1)
who	**who**(1)	who is logged in on the system
whoami	**whoami**(1)	display the effective current username
whois	**whois**(1)	TCP/IP Internet user name directory service
write	**write**(1)	write a message to another user
xargs	**xargs**(1V)	construct the arguments list(s) and execute a command
xget	**xsend**(1)	send or receive secret mail
xsend	**xsend**(1)	send or receive secret mail
xstr	**xstr**(1)	extract strings from C programs to implement shared strings
yacc	**yacc**(1)	yet another compiler-compiler: parsing program generator
yes	**yes**(1)	be repetitively affirmative

ypcat	**ypcat**(1)	print values in a NIS data base
ypmatch	**ypmatch**(1)	print the value of one or more keys from a NIS map
yppasswd	**yppasswd**(1)	change your network password in the NIS database
ypwhich	**ypwhich**(1)	return hostname of NIS server or map master
zcat	**compress**(1)	compress or expand files, display expanded contents

NAME

acctcom – search and print process accounting files

SYNOPSIS

acctcom [**–abfhikmqrtv**] [**–C** *sec*] [**–e** *time*] [**–E** *time*] [**–g** *group*] [**–H** *factor*] [**–I** *chars*] [**–l** *line*] [**–n** *pattern*] [**–o** *output-file*] [**–O** *sec*] [**–s** *time*] [**–S** *time*] [**–u** *user*]

DESCRIPTION

acctcom reads *filename*, the standard input, or **/var/adm/pacct**, in the form described by **acct**(5) and writes selected records to the standard output. Each record represents the execution of one process. The output shows the **COMMAND NAME**, **USER**, **TTYNAME**, **START TIME**, **END TIME**, **REAL (SEC)**, **CPU (SEC)**, **MEAN SIZE(K)**, and optionally, **F** (the *fork/exec* flag: **1** for **fork()** without **exec()**), **STAT** (the system exit status), **HOG FACTOR**, **KCORE MIN**, **CPU FACTOR**, **CHARS TRNSFD**, and **BLOCKS READ** (total blocks read and written).

A '**#**' is prepended to the command name if the command was executed with super-user privileges. If a process is not associated with a known terminal, a '**?**' is printed in the **TTYNAME** field.

If no *filename*s are specified, and if the standard input is associated with a terminal or **/dev/null** (as is the case when using '**&**' in the shell), **/var/adm/pacct** is read; otherwise, the standard input is read.

If any *filename* arguments are given, they are read in their respective order. Each file is normally read forward, that is, in chronological order by process completion time. The file **/var/adm/pacct** is usually the current file to be examined; a busy system may need several such files of which all but the current file are found in **/var/adm/pacct**?.

OPTIONS

–a Show some average statistics about the processes selected. The statistics will be printed after the output records.

–b Read backwards, showing latest commands first. This option has no effect when the standard input is read.

–f Print the *fork/exec* flag and system exit status columns in the output. The numeric output for this option will be in octal.

–h Instead of mean memory size, show the fraction of total available CPU time consumed by the process during its execution. This "hog factor" is computed as:
(total CPU time)/(elapsed time)

–i Print columns containing the I/O counts in the output.

–k Instead of memory size, show total kcore-minutes.

–m Show mean core size (the default).

–q Do not print any output records, just print the average statistics as with the **–a** option.

–r Show CPU factor (user time/(system-time + user-time).

–t Show separate system and user CPU times.

–v Exclude column headings from the output.

–C *sec* Show only processes with total CPU time, system plus user, exceeding *sec* seconds.

–e *time* Select processes existing at or before *time*.

–E *time* Select processes ending at or before *time*. Using the same *time* for both **–S** and **–E** shows the processes that existed at *time*.

–g *group* Show only processes belonging to *group*. The *group* may be designated by either the group ID or group name.

–H *factor* Show only processes that exceed *factor*, where factor is the "hog factor" as explained in the **–h** option above.

NAME

arch – display the architecture of the current host

SYNOPSIS

arch
arch –k
arch *archname*

DESCRIPTION

arch displays the application architecture of the current host system.

Sun systems can be broadly classified by their *architectures*, which define what executables will run on which machines. A distinction can be made between *kernel architecture* and *application architecture* (or, commonly, just "architecture"). Machines that run different kernels due to underlying hardware differences may be able to run the same application programs. For example, the MC68020-based Sun-3/160 system and the MC68030-based Sun-3/80 system run different SunOS kernels, but can execute the same applications. These machines could be described as having "the same application architecture" but "different kernel architectures."

Executing **arch** will display the application architecture of the machine, such as **sun3**, **sun4**, etc. All machines of the same application architecture will execute the same application programs, or user binaries.

Current Architectures

Application Architecture	Kernel Architecture	Current Sun System Models
sun3	sun3	3/50, 3/60, 3/75, 3/110, 3/140, 3/160, 3/180, 3/260, 3/280
sun3	sun3x	3/80, 3/460, 3/470, 3/480
sun4	sun4	4/110, 4/260, 4/280, SPARCsystem 300 series, SPARCsystem 400 series
sun4	sun4c	SPARCsystem 1, SPARCsystem 1+, SPARCsystem 2, SPARCsystem IPC, SPARCsystem SLC
sun4	sun4m	SPARCsystem 600MP series
sun386	sun386	386i/150, 386i/250

OPTIONS

–k Display the kernel architecture, such as **sun3**, **sun3x**, **sun4c**, etc. This defines which specific SunOS kernel will run on the machine, and has implications only for programs that depend on the kernel explicitly (for example, **ps**(1), **vmstat**(8), etc.).

archname Return "true" (exit status 0) if *application* binaries for *archname* can run on the current host system, otherwise, return "false" (exit status 1). This is the preferred method for installation scripts to determine the environment of the host machine; that is, which architecture of a multi-architecture release to install on this machine. *archname* must be a valid application architecture.

SEE ALSO

mach(1), **machid**(1)

Installing the SunOS System Software

NAME

as – Sun-1, Sun-2 and Sun-3, Sun-4 and Sun386i assemblers

SUN-1, SUN-2 and SUN-3 SYNOPSIS

as [**–L**] [**–R**] [**–o** *objfile*] [**–d2**] [**–h**] [**–j**] [**–J**] [**–O**] [**–mc68010**] [**–mc68020**] *filename*

SUN-4 SYNOPSIS

as [**–L**] [**–R**] [**–o** *objfile*] [**–O**[*n*]] [**–P** [[**–I***path*] [**–D***name*] [**–D***name=def*] [**–U***name*]] . . .]
[**–S**[**C**]] *filename* . . .

Sun386i SYNOPSIS

as [**–k**] [**–o** *objfile*] [**–R**] [**–V**] [**–i386**]

DESCRIPTION

as translates the assembly source file, *filename* into an executable object file, *objfile*. The Sun-4 assembler recognizes the filename argument '–' as the standard input.

All undefined symbols in the assembly are treated as global.

The output of the assembly is left in the file *objfile*.

OPTIONS

The following options are common to all Sun architectures. Options for specific Sun architectures are listed below.

–L Save defined labels beginning with an **L**, which are normally discarded to save space in the resultant symbol table. The compilers generate many such temporary labels.

–R Make the initialized data segment read-only by concatenating it to the text segment. This eliminates the need to run editor scripts on assembly code to make initialized data read-only and shared.

–o *objfile*
The next argument is taken as the name of the object file to be produced. If the **–o** flag is not used, the object file is named **a.out**.

Sun-1, Sun-2 and Sun-3 Options

–d2 Instruction offsets that involve forward or external references, and with unspecified size, are two bytes long. (See also the **–j** option.)

–h Suppress span-dependent instruction calculations. Restrict branches to medium length. Force calls to take the most general form. This option is used when the assembly must be minimized, even at the expense of program size and run-time performance. It results in a smaller and faster program than one produced by the **–J** option, but some very large programs may be unable to use it due to the limits of medium-length branches.

–j Use short (pc-relative) branches to resolve jump and jump-to-subroutine instructions to external routines. This is for compact programs for which the **-d2** option is inappropriate due to large-program relocation.

–J Suppress span-dependent instruction calculations and force branches and calls to take the most general form. This is useful when assembly time must be minimized, even at the expense of program size and run-time performance.

–O Perform span-dependent instruction resolution over entire files rather than just individual procedures.

Sun-4 Options

–O[*n*] Enable peephole optimization corresponding to optimization level *n* (1 if *n* not specified) of the Sun high-level language compilers. This option can be used safely only when assembling code produced by a Sun compiler.

–P Run **cpp**(1), the C preprocessor, on the files being assembled. The preprocessor is run separately on each input file, not on their concatenation. The preprocessor output is passed to the assembler.

SEE ALSO

ps(1)

BUGS

It would be better to use a real 56-bit key rather than an ASCII-based 56-bit pattern. Knowing that the key was derived from ASCII radically reduces the time necessary for a brute-force cryptographic attack.

RESTRICTIONS

Software encryption is disabled for programs shipped outside of the U.S. The program will still be able to encrypt files if one can obtain an encryption chip, legally or otherwise.

NAME

desktop – switch the window system to be invoked upon login

SYNOPSIS

desktop [**openwin**] [**sunview**]

AVAILABILITY

Available beginning with SunOS 4.1.1 Rev B.

DESCRIPTION

desktop switches the window system to be invoked upon your login between OpenWindows and SunView. This only applies if the user is using the default **.cshrc** file supplied from **/usr/lib/Cshrc** and has not modified the window system invocation code section. With no arguments, **desktop** prompts you to enter the window system you want invoked. The desired window system can also be supplied on the command line.

If the **add_user (8)** program is used to add a new user's account, the **/usr/lib/Cshrc** file is copied into the file **.cshrc** in the user's home directory. Alternatively, this may be done manually. In either case, if this file's window system invocation code is not thereafter modified, **desktop** switches window systems as specified.

A confirmation message is displayed for the window system chosen. An instructional message is then displayed indicating the steps to execute to activate this: exiting the system and logging back in.

OPTIONS

openwin Modify window system invocation code in **~/.cshrc** to execute OpenWindows.

sunview Modify window system invocation code in **~/.cshrc** to execute SunView.

FILES

/usr/lib/Cshrc	default file for setting up a user's environment
~/.cshrc	user's own file for setting up their environment

NAME

df – report free disk space on file systems

SYNOPSIS

df [**–a**] [**–i**] [**–t** *type*] [*filesystem*...] [*filename*...]

SYSTEM V SYNOPSIS

/usr/5bin/df [**–t**] [*filesystem*...] [*filename*...]

AVAILABILITY

The System V version of this command is available with the *System V* software installation option. Refer to *Installing the SunOS System Software* for information on how to install optional software.

DESCRIPTION

df displays the amount of disk space occupied by currently mounted file systems, the amount of used and available space, and how much of the file system's total capacity has been used. Used without arguments, **df** reports on all mounted file systems, producing something like:

```
example% df
Filesystem  kbytes  used  avail  capacity  Mounted on
/dev/ip0a    7445   4714  1986   70%       /
/dev/ip0g   42277  35291  2758   93%       /usr
```

Note: **used+avail** is less than the amount of space in the file system (kbytes); this is because the system reserves a fraction of the space in the file system to allow its file system allocation routines to work well. The amount reserved is typically about 10%; this may be adjusted using **tunefs**(8). When all the space on a file system except for this reserve is in use, only the super-user can allocate new files and data blocks to existing files. When a file system is overallocated in this way, **df** may report that the file system is more than 100% utilized.

If arguments to **df** are disk partitions (for example, **/dev/ip0a**s or path names, **df** produces a report on the file system containing the named file. Thus '**df .**' shows the amount of space on the file system containing the current directory.

SYSTEM V DESCRIPTION

The *System V* version of **df** works in the same manner as above but prints only the amount of available space (in 512 byte units) and the number of free inodes.

OPTIONS

–a Report on all filesystems including the uninteresting ones which have zero total blocks. (that is, *automounter*)

–i Report the number of used and free inodes. Print '*' if no information is available.

–t *type* Report on filesystems of a given *type* (for example, **nfs** or **4.2**).

SYSTEM V OPTIONS

–t Report the total allocated space figures.

FILES

/etc/mtab List of filesystems currently mounted.

SEE ALSO

du(1V), **mtab**(5), **quot**(8), **tunefs**(8)

Copying default configuration files into your home directory.
This is the first time you have run the **dos** command. A **~/pc** directory is being set up, and DOS-related files are being copied into it.

Another DOS window already has access to *device*

IRQ level *number* **is still in use by another DOS window.**
Your PC configuration file (normally **~/pc/setup.pc**) is requesting access to a physical device that another DOS window is using.

Port number *number* **out of range for** *board* **board.**
The port number specified in the **/etc/dos/defaults/boards.pc** is invalid.

IRQ value *number* **out of range for** *board* **board.**
The interrupt level specified in the **/etc/dos/defaults/boards.pc** is invalid.

IRQ level *number* **is in use by a Unix driver.**
There is a Unix driver servicing the board you are trying to attach to DOS. You are using the wrong IRQ level or you should use the driver instead.

Interrupt level *number* **is used by DOS to support** *device*
The interrupt level specified in the **/etc/dos/defaults/boards.pc** conflicts with an interrupt value currently being used by either a physical or emulated DOS device.

I/O address range *address-address* **requested for** *name* **board already in use by** *device*.
The address range specified in the **/etc/dos/defaults/boards.pc** conflicts with range currently being used by either a physical or emulated DOS device.

Cannot share *device* **with a hardware interrupt or DMA channel.**
A shared device specified in the **/etc/dos/defaults/boards.pc** was also assigned an interrupt level in this file. Shared devices cannot be assigned interrupt levels.

Couldn't find *name* **board in boards.pc.**
A file specified in the PC setup file (normally **~/pc/setup.pc**) is not listed in the **/etc/dos/defaults/boards.pc** file. Check the **setup.pc** file, or add an entry for the board in **boards.pc**.

ROM is newer than .quickpc. Rebooting *program_name*.
Save a new **.quickpc** file by issuing the command **dos –s**.

Warning: Your personal drive C (*pathname***) is not protected against simultaneous access by more than one workstation. Ask your system administrator to upgrade** *server* **to use the lock manager. Until your home directory server is updated with this program, do not use** *program_name* **when you are logged into more than one workstation.**

The system on the network where your drive C is stored has not protected the drive against access by DOS windows in other workstations on the network. This usually means that the server where your home directory is stored does not provide an NFS locking service. To avoid this error message, set the environment variable DOS_LOCKING to off.

SEE ALSO

dos2unix(1), **unix2dos**(1)
Sun386i User's Guide
Sun386i Advanced Skills
Sun MS-DOS Reference Manual

NAME

dos2unix – convert text file from DOS format to ISO format

SYNOPSIS

dos2unix [**–ascii**] [**–iso**] [**–7**] *originalfile convertedfile*

DESCRIPTION

dos2unix converts characters in the DOS extended character set to the corresponding ISO standard characters.

This command can be invoked from either DOS or SunOS. However, the filenames must conform to the conventions of the environment in which the command is invoked.

If the original file and the converted file are the same, **dos2unix** will rewrite the original file after converting it.

OPTIONS

–ascii Removes extra carriage returns and converts end of file characters in DOS format text files to conform to SunOS requirements.

–iso This is the default. It converts characters in the DOS extended character set to the corresponding ISO standard characters.

–7 Convert 8 bit DOS graphics characters to 7 bit space characters so that SunOS can read the file.

DIAGNOSTICS

File *filename* **not found, or no read permission**

The input file you specified does not exist, or you do not have read permission (check with the SunOS **ls –l** command).

Bad output filename *filename*, **or no write permission**

The output file you specified is either invalid, or you do not have write permission for that file or the directory that contains it. Check also that the drive or diskette is not write-protected.

Error while writing to temporary file

An error occurred while converting your file, possibly because there is not enough space on the current drive. Check the amount of space on the current drive using the DIR command. Also be certain that the default diskette or drive is write-enabled (not write-protected). Note that when this error occurs, the original file remains intact.

Could not rename temporary file to
Translated temporary file name = *filename*.

The program could not perform the final step in converting your file. Your converted file is stored under the name indicated on the second line of this message.

SEE ALSO

dos(1), **unix2dos**(1)

Sun386i Advanced Skills
Sun MS-DOS Reference Manual

NAME

fdformat – format diskettes for use with SunOS

SYNOPSIS

Sun386i Systems

fdformat [**–L**] [**–2**]

Sun-3/80 and Desktop SPARC Systems

fdformat [**–deflv**] [**–b** *label*] [*device*]

AVAILABLITY

This command is available on Sun386i, Sun-3/80, and Desktop SPARC systems only.

DESCRIPTION

fdformat is a utility for formatting floppy diskettes. All new blank diskettes must be formatted before they can be used. **fdformat** formats and verifies each track on the diskette, and terminates if it finds any bad sectors. All existing data on the diskette, if any, is destroyed by formatting.

By default, **fdformat** formats a high density diskette with a capacity of 1.44 megabytes. Use the **–L** or **–l** option to format low density diskettes (720K capacity). Note: it is not possible to put a low density format onto a high density floppy diskette. High density diskettes can be recognized by the high density detect hole in the lower right corner of the diskette, opposite the write protect hole in the lower left corner.

Use the **–d** option to put a **MS-DOS** file system on the diskette after format is done. Otherwise, the Sun-3/80 and Desktop SPARC systems version of **fdformat** writes a SunOS label on the diskette in logical block 0 after it has been formatted. The label is required on Sun-3/80 and Desktop SPARC systems if you intend to put a UNIX file system on the floppy diskette.

On Sun386i systems, use the **–2** option to format diskettes in the optional external 5 1/4" floppy drive. To format diskettes for use under MS-DOS, use the MS-DOS FORMAT command in a DOS window.

OPTIONS

Sun386i Systems

–L Format a low density diskette.

–2 Format a diskette in the optional external 5.25-inch floppy drive.

Sun-3/80 and Desktop SPARC Systems

–b Put a **MS-DOS** *label* on the disk after formatting it. This option is only meaningful when **–d** is also set.

–d Install a **MS-DOS** file system and boot sector on the disk after formatting. This is equivalent to the MS-DOS FORMAT command.

–e Eject the diskette when done.

–f Force. Do not ask for confirmation before starting format.

–l Format a low density (720K) diskette.

–v Verify the floppy diskette after formatting.

NOTES

A diskette formatted using the **–d option for MS-DOS** won't have the necessary system files, and is therefore not bootable. Trying to boot from it on a **PC** will result in the following message:

Non-System disk or disk error
Replace and strike any key when ready

FILES

Sun386i Systems

/dev/rfd0c	1.44 megabyte 3.5-inch high density diskette drive
/dev/rfdl0c	720 kilobyte 3.5-inch low density diskette drive

/dev/rfd2c High density 5.25-inch floppy drive
/dev/rfdl2c Low density 5.25-inch floppy drive

Sun-3/80 and Desktop SPARC Systems
/dev/rfd0c

SEE ALSO

dos(1), **fd**(4S), **pcfs**(4S)

BUGS

Currently bad sector mapping is not supported on floppy diskettes. Therefore, a diskette is unusable if **fdformat** finds an error (bad sector).

NAME

lockscreen, lockscreen_default – maintain SunView context and prevent casual access

SYNOPSIS

lockscreen [**–enr**] [**–b** *program*] [**–t** *seconds*] [*gfx-program* [*gfx-program-arguments*]]

AVAILABILITY

This command is available with the *SunView User's Guide* software installation option. Refer to *Installing the SunOS System Software* for information about installing optional software.

DESCRIPTION

lockscreen is a SunView utility that locks the screen against casual access while preserving the state of the SunView display. It clears the workstation screen to black, and then runs *gfx-program*, which typically provides a moving graphics display to reduce phosphor burn. When no *gfx-program* is provided, a suitable default program is run.

lockscreen requires the user's password before restoring the window context. When any keyboard or mouse button is pressed, the graphics screen is replaced by a password screen that displays the user name, a small box with a bouncing logo, and a prompt for the user's password. If the user has no password, or if the **–n** option is used, the window context is simply restored.

When the password screen appears:

- Restore the window context by entering the user's password followed by a RETURN (this password is not echoed on the screen) or,
- Point to the black box and click the left button to return to the graphics display.

If neither of the above actions is taken, *gfx_program* resumes execution after the interval specified with the **–t** option.

OPTIONS

–e Add the **Exit Desktop** choice to the password screen. If pointed to and clicked, the SunView environment is exited and the current user is logged out. This option is unreliable and can be a security hole, its use is strongly discouraged. See BUGS.

–n Require no password to reenter the window environment.

–r Allow the use of the user name **root** in the '**Name:**' field of the password screen. Normally, **root** is not accepted as a valid user name.

–b *program*
Allow an additional program to be run as a child process of **lockscreen**. This background process could be a compile server or some other useful program that the user wants run while lockscreen is running. No arguments are passed to this program.

–t *seconds*
After *seconds* seconds, clear the password screen and restart *gfx-program*. The default is 5 minutes (300 seconds).

[*gfx-program*] [*gfx-program-arguments*]
Run this program after clearing the screen to black. When no *program* argument is present, **lockscreen** tries to run **lockscreen_default** if it is in the standard search path; otherwise, a bouncing Sun logo appears. When *gfx-program-arguments* are specified and the *gfx-program* is not, the arguments are passed to **lockscreen_default**. **lockscreen_default** is typically a non-interactive graphics program (see **graphics_demos**(6)). **lockscreen** does not search for **lockscreen_default** when the *gfx-program* is specified explicitly as an empty argument (an adjacent pair of quote-marks).

FILES

/usr/bin/lockscreen_default
Default *gfx-program*. This displays a series of **life**(6) patterns. If a file named **lockscreen_default** appears earlier in the search path, that file is used instead.

SEE ALSO

login(1), **sunview**(1)

BUGS

lockscreen is more secure when *gfx-program* are not specified and when **lockscreen_default** is not in the search path. To improve security, enter the following command as root, or start **lockscreen** with an empty argument (see OPTIONS, above).

example# mv /usr/bin/lockscreen_default /usr/bin/lockscreen_default-

The **–e** option does not consistently log the current user out by clicking on the **Exit Desktop** button. Occassionally, users will be returned to the shell or the Desktop *without requiring a password*. This option is unreliable, and its use could allow unsecure access to the system.

:f Display the current filename and line number.

:q
:Q Exit from **more** (same as **q** or **Q**).

. Dot. Repeat the previous command.

^ Halt a partial display of text. **more** stops sending output, and displays the usual **--More--** prompt. Unfortunately, some output is lost as a result.

FILES

/etc/termcap	terminal data base
/usr/lib/more.help	help file

SEE ALSO

cat(1V), **csh**(1), **man**(1), **script**(1), **sh**(1), **environ**(5V), **termcap**(5)

BUGS

Skipping backwards is too slow on large files.

NAME

mps – display the status of current processes on an MP system

SYNOPSIS

/usr/kvm/mps [[–]**acCegjklnrSuUvwx**] [–**t***x*] | [*num*] [*kernel-name*] [*c-dump-file*] [*swap-file*]

AVAILABILITY

This program is only available on sun4m architectures.

DESCRIPTION

mps displays information about processes on an MP system. **mps** is identical to **ps(1)** except that the **CPU** field was added to show the cpu number the process is or was running on. Normally, only those processes that are running with your effective user ID and are attached to a controlling terminal (see **termio**(4)) are shown. Additional categories of processes can be added to the display using various options. In particular, the –**a** option allows you to include processes that are not owned by you (that do not have your user ID), and the –**x** option allows you to include processes without control terminals. When you specify both –**a** and –**x**, you get processes owned by anyone, with or without a control terminal. The –**r** option restricts the list of processes printed to "running" processes: runnable processes, those in page wait, or those in short-term non-interruptible waits.

mps displays the process ID, under PID; the control terminal (if any), under TT; the cpu time used by the process so far, including both user and system time), under TIME; the state of the process, under STAT; and finally, an indication of the COMMAND that is running.

The state is given by a sequence of four letters, for example, '**RWNA**'.

First letter indicates the runnability of the process:

- **R** Runnable processes.
- **T** Stopped processes.
- **P** Processes in page wait.
- **D** Processes in non-interruptible waits; typically short-term waits for disk or NFS I/O.
- **S** Processes sleeping for less than about 20 seconds.
- **I** Processes that are idle (sleeping longer than about 20 seconds).
- **Z** Processes that have terminated and that are waiting for their parent process to do a **wait**(2V) ("zombie" processes).

Second letter indicates whether a process is swapped out;

- *blank* Represented as a SPACE character, in this position indicates that the process is loaded (in memory).
- **W** Process is swapped out.
- **>** Process has specified a soft limit on memory requirements and has exceeded that limit; such a process is (necessarily) not swapped.

Third letter indicates whether a process is running with altered CPU scheduling priority (**nice**(1)):

- *blank* Represented as a SPACE character, in this position indicates that the process is running without special treatment.
- **N** The process priority is reduced,
- **<** The process priority has been raised artificially.

Fourth letter indicates any special treatment of the process for virtual memory replacement. The letters correspond to options to the **vadvise**(2) system call. Currently the possibilities are:

- *blank* Represented as a SPACE character, in this position stands for **VA_NORM**.
- **A** Stands for **VA_ANOM**. An **A** typically represents a program which is doing garbage collection.
- **S** Stands for **VA_SEQL**. An **S** is typical of large image processing programs that are using virtual memory to sequentially address voluminous data.

kernel-name specifies the location of the system namelist. If the **–k** option is given, *c-dump-file* tells **mps** where to look for the core dump. Otherwise, the core dump is located in the file **/vmcore** and this argument is ignored. *swap-file* gives the location of a swap file other than the default, **/dev/drum**.

OPTIONS

Options must all be combined to form the first argument.

–a Include information about processes owned by others.

–c Display the command name, as stored internally in the system for accounting purposes, rather than the command arguments, which are kept in the process address space. This is more reliable, if less informative, as the process is free to destroy the latter information.

–C Display raw CPU time instead of the decaying average in the **%CPU** field.

–e Display the environment as well as the arguments to the command.

–g Display all processes. Without this option, **mps** prints only "interesting" processes. Processes are deemed to be uninteresting if they are process group leaders. This normally eliminates top-level command interpreters and processes waiting for users to login on free terminals.

–j Display a listing useful for job control information, with fields PPID, PID, PGID, SID, TT, TPGID, STAT, UID, TIME, and COMMAND as described below.

With this option, the STAT field has three additional letters:

C indicates the process does not want SIGCHLD when a child changes state done to job control.

E The process has completed an exec, and the parent can no longer change the process group of this process.

O The process is an orphan, with no parent process to handle job control signals.

–k Normally, *kernel-name* defaults to **/vmunix**, *c-dump-file* is ignored, and *swap-file* defaults to **/dev/drum**. With the **–k** option in effect, these arguments default to **/vmunix**, **/vmcore**, and **/dev/drum**, respectively.

–l Display a long listing, with fields **F**, **PPID**, **CP**, **PRI**, **NI**, **SZ**, **RSS**, and **WCHAN**, as described below.

–n Produce numeric output for some fields. In a long listing, the **WCHAN** field is printed numerically rather than symbolically, or, in a user listing, the **USER** field is replaced by a **UID** field.

–r Restrict output to "running" processes.

–S Display accumulated CPU time used by this process and all of its reaped children.

–u Display user-oriented output. This includes fields **USER**, **%CPU**, **%MEM**, **SZ**, **RSS** and **START** as described below.

–U Update a private database where **mps** keeps system information. Include '**mps –U**' in the **/etc/rc** file.

–v Display a version of the output describing virtual memory information. This includes fields **RE**, **SL**, **PAGEIN**, **SIZE**, **RSS**, **LIM**, **%CPU** and **%MEM**, described below.

–w Use a wide output format (132 columns rather than 80); if repeated, that is, **–ww**, use arbitrarily wide output. This information is used to decide how much of long commands to print.

–x Include processes with no controlling terminal.

The following two options are mutually exclusive. When specified, these options must appear immediately following the last option.

–t*x* Restrict output to processes whose controlling terminal is *x* (which should be specified as printed by **mps**; for example, **t3** for **/dev/tty3**, **tco** for **/dev/console**, **td0** for **/dev/ttyd0**, **t?** for processes with no terminal, etc). This option must be the last one given.

num A process number may be given, in which case the output is restricted to that process. This option must also be last, and must appear with no white space between it and the previous option.

DISPLAY FORMATS

Fields that are not common to all output formats:

USER Name of the owner of the process.

%CPU CPU use of the process; this is a decaying average over up to a minute of previous (real) time. Because the time base over which this is computed varies (since processes may be very young) it is possible for the sum of all **%CPU** fields to exceed 100%.

NI Process scheduling increment (see **getpriority**(2) and **nice**(3V)).

SIZE

SZ The combined size of the data and stack segments (in kilobytes)

RSS Real memory (resident set) size of the process (in kilobytes).

LIM Soft limit on memory used, specified using a call to **getrlimit**(2); if no limit has been specified, this is shown as *xx*.

%MEM Percentage of real memory used by this process.

RE Residency time of the process (seconds in core).

SL Sleep time of the process (seconds blocked).

PAGEIN Number of disk I/Os resulting from references by the process to pages not loaded in core.

UID Numeric user-ID of process owner.

PPID Numeric ID of parent of process.

SID Numeric ID of the session to which the process belongs. SID = PGID = PID indicates a session leader.

PGID Numeric ID of the process group of the process.

TPGID Numeric ID of the process group associated with the terminal specified under TT (distinguished process group, see **termio**(4)).

CP Short-term CPU utilization factor (used in scheduling).

PRI Process priority (non-positive when in non-interruptible wait).

START Time the process was created if today, or the date it was created if before today.

WCHAN Event on which process is waiting (an address in the system). A symbol is chosen that classifies the address, unless numeric output is requested (see the **n** flag). In this case, the address is printed in hexadecimal.

CPU Relationship between process and processor. Under the **CPU** field, the cpu number the process is or was running on is displayed.

F Flags (in hex) associated with process as in **<sys/proc.h>**:

SLOAD	00000001	in core
SSYS	00000002	swapper or pager process
SLOCK	00000004	process being swapped out
SSWAP	00000008	save area flag
STRC	00000010	process is being traced
SWTED	00000020	parent has been told that this process stopped
SULOCK	00000040	user can set lock in core
SPAGE	00000080	process in page wait state
SKEEP	00000100	another flag to prevent swap out
SOMASK	00000200	restore old mask after taking signal
SWEXIT	00000400	working on exiting
SPHYSIO	00000800	doing physical I/O

SVFORK	00001000	process resulted from vfork()
SVFDONE	00002000	another vfork flag
SNOVM	00004000	no vm, parent in a vfork()
SPAGI	00008000	init data space on demand, from vnode
SSEQL	00010000	user warned of sequential vm behavior
SUANOM	00020000	user warned of anomalous vm behavior
STIMO	00040000	timing out during sleep
SORPHAN	00080000	process is orphaned
STRACNG	00100000	process is tracing another process
SOWEUPC	00200000	process is being profiled and has a pending count increment
SSEL	00400000	selecting; wakeup/waiting danger
SFAVORD	02000000	favored treatment in swapout and pageout
SLKDONE	04000000	record-locking has been done
STRCSYS	08000000	tracing system calls
SNOCLDSTOP	10000000	SIGCHLD not sent when child stops
SEXECED	20000000	process has completed an exec
SRPC	40000000	sunview window locking

A process that has exited and has a parent, but has not yet been waited for by the parent, is marked **<defunct>**; a process that is blocked trying to exit is marked **<exiting>**; otherwise, **mps** makes an educated guess as to the file name and arguments given when the process was created by examining memory or the swap area.

ENVIRONMENT

The environment variables **LC_CTYPE**, **LANG**, and **LC_default** control the character classification throughout **mps**. On entry to **mps**, these environment variables are checked in the following order: **LC_CTYPE**, **LANG**, and **LC_default.** When a valid value is found, remaining environment variables for character classification are ignored. For example, a new setting for **LANG** does not override the current valid character classification rules of **LC_CTYPE**. When none of the values is valid, the shell character classification defaults to the POSIX.1 "C" locale.

FILES

/vmunix	system namelist
/dev/kmem	kernel memory
/dev/drum	swap device
/vmcore	core file
/dev	searched to find swap device and terminal names
/etc/psdatabase	system namelist, device, and wait channel information

SEE ALSO

kill(1), **w**(1), **getpriority**(2), **getrlimit**(2), **wait**(2V), **vadvise**(2), **nice**(3V), **termio**(4), **locale**(5), **pstat**(8), **mpstat**(1)

BUGS

Things can change while **mps** is running; the picture it gives is only a close approximation to the current state.

NAME

mpstat – show multi-processor usage

SYNOPSIS

/usr/kvm/mpstat [*interval* [*count*]]

AVAILABILITY

This program is only available on sun4m architectures.

DESCRIPTION

mpstat enters the system and shows average and per-processor percentage usage data during a particular time interval. The first group of data represents the average of all the processors in the system while the second and subsequent data groups represent particular processors, with their particular name designations above each group.

Without an *interval* or *count* option, **vmstat** displays a one-line summary of MP system activity since the system has been booted. If *interval* is specified, **vmstat** summarizes activity over the last *interval* seconds. If a *count* is given, the statistics are repeated *count* times.

The particular fields of each data group give a breakdown of percentage usage of CPU time:

us user time for normal processes

ni time for processes with an altered scheduling priority (*nice(1)*)

sy system time

id CPU idle

FILES

/dev/kmem
/vmunix

SEE ALSO

mps(1), nice(1), ps(1), vmstat(8)

NAME

mt – magnetic tape control

SYNOPSIS

mt [**–f** *tapename*] *command* [*count*]

DESCRIPTION

mt sends commands to a magnetic tape drive. If *tapename* is not specified, the environment variable **TAPE** is used. If **TAPE** does not exist, **mt** uses the device **/dev/rmt12**. *tapename* refers to a raw tape device. By default, **mt** performs the requested operation once; multiple operations may be performed by specifying *count*.

The available commands are listed below. Only as many characters as are required to uniquely identify a command need be specified.

mt returns a 0 exit status when the operation(s) were successful, 1 if the command was unrecognized or if **mt** was unable to open the specified tape drive, and 2 if an operation failed.

OPTIONS

eof, weof Write *count* EOF marks at the current position on the tape.

fsf Forward space over *count* EOF marks. The tape is positioned on the first block of the file.

fsr Forward space *count* records.

bsf Back space over *count* EOF marks. The tape is positioned on the beginning-of-tape side of the EOF mark.

bsr Back space *count* records.

nbsf Back space *count* files. The tape is positioned on the first block of the file. This is equivalent to *count+1* **bsf**'s followed by one **fsf**.

asf Absolute space to *count* file number. This is equivalent to a **rewind** followed by a **fsf** *count.*

For the following commands, *count* is ignored:

eom Space to the end of recorded media on the tape. This is useful for appending files onto previously written tapes.

rewind Rewind the tape.

offline, rewoffl Rewind the tape and, if appropriate, take the drive unit off-line by unloading the tape.

status Print status information about the tape unit.

retension Rewind the cartridge tape completely, then wind it forward to the end of the reel and back to beginning-of-tape to smooth out tape tension.

erase Erase the entire tape.

FILES

/dev/rmt*	magnetic tape interface
/dev/rar*	Archive cartridge tape interface
/dev/rst*	SCSI tape interface

SEE ALSO

ar(4S), **mtio**(4), **st**(4S), **tm**(4S), **xt**(4S) **environ**(5V)

BUGS

Not all devices support all options. Some options are hardware-dependent. Refer to the corresponding device manual page.

NAME

sunview – the SunView window environment

SYNOPSIS

sunview [**–i**] [**–p**] [**–B** | **–F** | **–P**] [**–S**] [**–8bit_color_only**] [**–overlay_only**] [**–toggle_enable**]
[**–b** *red green blue*] [**–d** *display-device*] [**–f** *red green blue*] [**–k** *keyboard-device*]
[**–m** *mouse-device*] [**–n** | **–s** *startup-filename*] [**–background** *raster-filename*]
[**–pattern on** | **off** | **gray** | *iconedit-filename*]

DESCRIPTION

sunview starts up the SunView environment and (unless you have specified otherwise) a default layout of a few useful "tools," or window-based applications.

See **Start-up Processing** below to learn how to specify your own initial layout of tools. Some of the behavior of **sunview** is controlled by settings in your defaults database; see **SunView Defaults** below, and **defaultsedit**(1) for more information.

To exit **sunview** use the **Exit SunView** menu item. In an emergency, type CTRL-D then CTRL-Q (there is no confirmation in this case).

OPTIONS

–i Invert the background and foreground colors used on the screen. On a monochrome monitor, this option provides a video reversed image. On a color monitor, colors that are not used as the background and foreground are not affected.

–p Print to the standard output the name of the window device used for the **sunview** background.

–B Use the "background color" (**–b**) for the background.

–F Use the "foreground color" (**–f**) for the background.

–P Use a stipple pattern for the background. This option is assumed unless **–F** or **–B** is specified.

–S Set **Click-to-type** mode, allowing you to select a window by clicking in it. Having done so, input is directed to that window regardless of the position of the pointer, until you click to select some other window.

–8bit_color_only
For multiple plane group frame buffers, only let windows be created in the 8 bit color plane group. This frees up the black and white overlay plane to have a separate desktop running on it. This option is usually used with the **–toggle_enable** option. See **Multiple Desktops on the Same Screen**, below.

–overlay_only For multiple plane group frame buffers, only let windows be created in the black and white overlay plane group. This frees up the 8 bit color plane group to have a separate desktop running in it. This option is usually used with the **–toggle_enable** option. See **Multiple Desktops on the Same Screen**, below.

–toggle_enable For multiple plane group frame buffers, when sliding the pointer between different desktops running within different plane groups on the same screen, change the enable plane to allow viewing of the destination desktop. See **Multiple Desktops on the Same Screen**, below.

–b *red green blue*
Specify values for the *red*, *green* and *blue* components of the background color. If this option is not specified, each component of the background color is 255 (white). Sun 3/110 system users that use this option should use the **–8bit_color_only** option as well.

–d *display-device* Use *display-device* as the output device, rather than **/dev/fb** the default frame buffer device.

–f *red green blue* Specify values for the *red*, *green* and *blue* components of the foreground color. If this option is not specified, each component of the foreground color is 0 (black). Sun 3/110 system users that use this option should use the **–8bit_color_only** option as well.

–k *keyboard-device*
Accept keyboard input from *keyboard-device*, rather than **/dev/kbd**, the default keyboard device.

–m *mouse-device* Use *mouse-device* as the system pointing device (locator), rather than **/dev/mouse**, the default mouse device.

–n Bypass startup processing by ignoring the **/usr/lib/.sunview** and **~/.sunview** (and **~/.suntools**) files.

–s *startup-filename*
Read startup commands from *startup-filename* instead of **/usr/lib/.sunview** or **~/.sunview**).

–background *raster-filename*
Use the indicated raster file as the image in your background. The raster file can be created with **screendump**(1). Screen dumps produced on color monitors currently do not work as input to this option. Small images are centered on the screen.

–pattern on | **off** | **gray** | *iconedit-filename*
Use the indicated "pattern" to cover the background. **on** means to use the default desktop gray pattern. **off** means to not use the default desktop gray pattern. **gray** means to use a 50% gray color on color monitors. *iconedit-filename* is the name of a file produced with **iconedit**(1) which contains an image that is to be replicated over the background.

USAGE

Windows

The SunView environment always has one window open, referred to as the background, which covers the whole screen. A solid color or pattern is its only content. Each application is given its own window which lies on top of some of the background (and possibly on top of other applications). A window obscures any part of another window which lies below it.

Input to Windows

Mouse input is always directed to the window that the pointer is in at the time. Keyboard input can follow mouse input or, it can remain within a designated window using the **Click-to-Type** default setting. If you are not using **Click-to-Type**, and the pointer is on the background, keyboard input is discarded. Input actions (mouse motions, button clicks, and keystrokes) are synchronized, which means that you can "type-ahead" and "mouse-ahead," even across windows.

Mouse Buttons

LEFT mouse button	Click to select or choose objects.
MIDDLE mouse button	In text, click once to shorten or lengthen your selection. In graphic applications or on the desktop, press and hold to move objects.
RIGHT mouse button	Press and hold down to invoke menus.

Menus

sunview provides pop-up menus. There are two styles of pop-up menus: an early style, called "stacking menus," and a newer style, called "walking menus" (also known as "pull-right menus"). In the current release, walking menus are the default; stacking menus are still available as a defaults option.

Usually, a menu is invoked by pressing and holding the RIGHT mouse button. The menu remains on the screen as long as you hold the RIGHT mouse button down. To choose a menu item, move the pointer onto it (it is then highlighted), then release the RIGHT mouse button.

Another available option is "stay-up menus." A stay-up menu is invoked by pressing and releasing the RIGHT mouse button. The menu appears on the screen after you release the RIGHT mouse button. To choose a menu item, move the pointer onto it (it is then highlighted), then press and release the RIGHT mouse button a second time. Stay-up menus are an option in your defaults database; see **SunView Defaults** below.

With walking menus, any menu item can have an arrow pointing (⇒) to the right. Moving the pointer onto this arrow pops up a "sub-menu," with additional items. Choosing the item with an arrow (the "pull-right item") invokes the first item on the sub-menu.

The SunView Menu

You can use the default SunView menu to start SunView applications and perform some useful functions. To invoke it, hold down the RIGHT mouse button when the pointer is anywhere in the background.

The default SunView menu is defined in the file **/usr/lib/.rootmenu**. It consists of four sub-menus, labeled **Shells**, **Editors**, **Tools**, and **Services**, along with items for **Remote Login, Redisplay All, Lock Screen** and **Exit Sunview.** These sub-menus contain the following items:

Shells

Shell Tool Bring up a **shelltool**(1), an tty-based terminal emulator that supports a shell.

Command Tool Bring up a **cmdtool**(1), a scrollable window-based terminal emulator that supports a shell.

Graphics Tool Bring up a **gfxtool**(1), for running graphics programs.

Console Bring up a Console window, a **cmdtool** with the **−C** flag, to act as the system console. Since many system messages can be directed to the console, there should always be a console window on the screen.

Remote Login

This will create a terminal emulator that prompts for a machine name and then starts a shell on that machine.

Editors

Text Editor Bring up a **textedit**(1), for reading and editing text files.

Defaults Editor Bring up a **defaultsedit**(1), for browsing or changing your defaults settings.

Icon Editor Bring up a new **iconedit**(1).

Font Editor Bring up a **fontedit**(1).

Tools

Mail Tool Bring up a **mailtool**(1), for reading and sending mail.

Dbx (Debug) Tool

Bring up a **dbxtool**(1), a window-based source debugger.

Performance Meter

Bring up a **perfmeter**(1) to monitor system performance.

Clock Bring up a new **clock**(1).

Services

Eject There are two items on this submenu, "cdrom" and "floppy". Use this to eject cdrom or floppy media from the drive.

Printing There are two items on this submenu, **Check Printer Queue** and **Print Selected Text**. **Check Printer Queue** displays the printer queue in your console; **Print Selected Text** sends selected text to the

standard printer.

Save Layout Writes out a **~/.sunview** file that **sunview** can then use when starting up again. An existing **~/.sunview** file is saved as **~/.sunview–**.

Redisplay All
Redraw the entire screen. Use this to repair damage done by processes that wrote to the screen without consulting the SunView system.

Lock Screen
Completely covers the screen with a graphics display, and "locks" the workstation until you type your password. When you "unlock" the workstation, the screen is restored as it was when you locked it. See **lockscreen**(1) for details.

Exit SunView
Exit from **sunview**, including all windows, and kill processes associated with them. You return to the shell from which you started **sunview**.

You can specify your own SunView menu; see **SunView Defaults** below for details.

The Frame Menu

A small set of universal functions are available through the Frame menu. There are also accelerators for some of these functions, described under **Frame Menu Accelerators**, below.

You can invoke the Frame menu when the cursor is over a part of the application that does not provide an application-specific menu, such as the frame header (broad stripe holding the application's name), the border stripes of the window, and the icon.

Close

Open Toggle the application between closed (iconic) and open state. Icons are placed on the screen according to the icon policy in your defaults database; see **SunView Defaults** below. When a window is closed, its underlying processes continue to run.

Move Moves the application window to another spot on the screen. **Move** has a sub-menu with two items: **Unconstrained** and **Constrained**.

Unconstrained Move the window both horizontally and vertically.

Constrained Moves are either vertical or horizontal, but not both.

Choosing **Move** invokes an **Unconstrained** move.

Resize Shrink or stretch the size of a window on the screen. **Resize** has a sub-menu containing:

Unconstrained Resize the window both horizontally and vertically.

Constrained Resize vertically or horizontally, but not both.

Choosing **Resize** invokes an **Unconstrained** resize.

UnZoom

Zoom **Zoom** expands a window vertically to the full height of the screen. **UnZoom** undoes this.

FullScreen Make a window the full height and width of the screen.

Front Bring the window to "the top of the pile." The whole window becomes visible, and hides any window it happens to overlap on the screen.

Back Put the window on the "bottom of the pile". The window is hidden by any window which overlaps it.

Props Display the property sheet. (Only active for applications that provide a property sheet.)

Redisplay Redraw the contents of the window.

Quit Notify the application to terminate gracefully. Requires confirmation.

Frame Menu Accelerators

Accelerators are provided for some Frame menu functions. You can invoke these functions by pushing a single button in the window's frame header or outer border. See the **SunView Beginner's Guide** for more details.

Open Click the LEFT mouse button when the pointer is over the icon.

Move Press and hold the MIDDLE mouse button while the pointer is in the frame header or outer border. A bounding box that tracks the mouse is displayed while you hold the button down. When you release the button, the window is redisplayed within the bounding box. If the pointer is near a corner, the move is **Unconstrained**. If it is in the center third of an edge, the move is **Constrained**.

Resize Hold the CTRL key and press and hold the MIDDLE mouse button while the pointer is in the frame header or outer border. A bounding box is displayed, and one side or corner tracks the mouse. If the pointer is near a corner when you press the mouse button, the resize is **Unconstrained**; if in the middle third of an edge, the resize is **Constrained**.

Zoom
UnZoom Hold the CTRL key and click the LEFT mouse button while the pointer is in the frame header or outer border.

Front Click the LEFT mouse button while the pointer is on the frame header or outer border.

Back Hold the SHIFT key and click the LEFT mouse button while the pointer is on the frame header or outer border.

In addition, you can use two function keys as even faster accelerators. To expose a window that is partially hidden, press the **Front** function key (normally L5) while the pointer is anywhere in that window. Or, if the window is completely exposed, use the **Front** key to hide it. Similarly, to close an open window, press the **Open** key (normally L7) while the pointer is anywhere in that window. If the window is iconic, use the **Open** key to open it.

In applications with multiple windows, you can often adjust the border between two windows up or down, without changing the overall size of the application: hold the CTRL key, press the MIDDLE mouse button over the boundary between the two windows, and adjust the size of the (bounded) subwindow as with **Resize**.

Startup Processing: The .sunview File

Unless you override it, **sunview** starts up with a predefined layout of windows. The default layout is specified in the file **/usr/lib/.sunview**. If there is a file called **.sunview** in your home directory, it is used instead. For compatibility with earlier releases, if there is no **.sunview** file in your home directory, but a **.suntools** file instead, the latter file is used.

SunView Defaults

SunView allows you to customize the behavior of applications and packages by setting options in a defaults database (one for each user). Use **defaultsedit**(1) to browse and edit your defaults database. Select the "SunView" category to see the following items (and some others):

Walking_menus If enabled, the SunView menu, the Frame menu, and many applications will use walking menus. Applications that have not been converted will still use stacking menus. If disabled, applications will use stacking menus. The default value is "Enabled."

Click_to_Type If enabled, keyboard input will stay in a window until you click the LEFT or MIDDLE mouse button in another window. If disabled, keyboard input will follow the mouse. The default value is "Disabled."

Font You can change the SunView default font by giving the full pathname of the font you want to use. Some alternate fonts are in the directory **/usr/lib/fonts/fixedwidthfonts**. The default font from the SunOS 2.0 release was **/usr/lib/fonts/fixedwidthfonts/screen.r.13**. The default value is null, which has the same effect as specifying **/usr/lib/fonts/fixedwidthfonts/screen.r.11**.

Rootmenu_filename You can change the SunView menu by giving the full pathname of a file that specifies your own menu. See **The SunView Menu File** below for details. The default value is null, which gives you the menu found in **/usr/lib/.rootmenu**.

Icon_gravity Determine which edge of the screen ("North", "South", "East", or "West") icons will place themselves against. The default value is "North."

Audible_bell If enabled, the "bell" command will produce a beep. The default value is "Enabled."

Visible_bell If enabled, the "bell" command will cause the screen to flash. The default value is "Enabled."

Root_Pattern Used to specify the "pattern" that covers the background. "on" means to use the default desktop gray pattern. "off" means to not use the default desktop gray pattern. "gray" means to use a 50% gray color on color monitors. Anything else is the name of a file produced with **iconedit**(1) which contains an image that is replicated all over the background. The default value is "on."

After you have set the options you want in the "SunView" category, click on the **Save** button in **defaultsedit**; then exit **sunview** and restart it.

Select the "Menu" category to see the following items (and some others):

Stay_up If enabled, menus are invoked by pressing and releasing the RIGHT mouse button; the menu appears after you release the RIGHT mouse button. To choose a menu item, point at it, then press and release the RIGHT mouse button a second time. The default value is "False".

Items_in_column_major
If enabled, menus that have more than one column are presented in "column major" order (the way **ls**(1V) presents file names). This may make a large menu easier to read. The default value is "False."

After you have set the options you want in the "Menu" category, click on the **Save** button in **defaultsedit**. Any applications you start after saving your changes will be affected by your new choices. For all defaults categories except for "SunView", you do *not* need to exit **sunview** and restart it.

The SunView Menu File

The file called **/usr/lib/.rootmenu** contains the specification of the default SunView menu. You can change the SunView menu by creating your own file and giving its name in the **Rootmenu_filename** item in the SunView Defaults.

Lines in the file have the following format: The left side is a menu item to be displayed, and the right side is a command to be executed when that menu item is chosen. You can also include comment lines (beginning with a '#') and blank lines.

The menu item can be a string, or the full pathname of an icon file delimited by angle brackets (unless **Walking_menus** is disabled in the SunView defaults). Strings with embedded blanks must be delimited by double quotes.

There are four reserved-word commands that can appear on the right side.

EXIT Exit **sunview** (requires confirmation).

REFRESH Redraw the entire screen.

MENU This menu item is a pull-right item with a submenu. If a full pathname follows the **MENU** command, the submenu contents are taken from that file.

Otherwise, all the lines between a **MENU** command and a matching **END** command are added to the submenu.

END Mark the end of a nested submenu. The left side of this line should match the left side of a line with a **MENU** command.

If the command is not one of these four reserved-word commands, it is treated as a command line and executed. No shell interpretation is done, although you can run a shell as a command.

Here is a menu file that demonstrates some of these features:

```
Quit                        EXIT
"Mail reader"               mailtool
"My tools"                  MENU /home/me/mytools.menu
"Click to type"             swin -c
"Follow mouse"              swin -m
"Print selection"           sh -c get_selection | lpr
"Nested menu"               MENU
    "Command Tool"          cmdtool
    "Shell Tool"            shelltool
"Nested menu"               END
"Icon menu"                 MENU
    <images/textedit.icon>  textedit
    <images/dbxtool.icon>   dbxtool
"Icon menu"                 END
```

Multiple Screens

The **sunview** program runs on either a monochrome or color screen. Each screen on a machine with multiple screens may have a separate **sunview** running. The keyboard and mouse input devices can be shared between screens. Using **adjacentscreens**(1) you can set up the pointer to slide from one screen to another when you move it off the edge of a screen.

To set up an instance of **sunview** on two screens:

- Invoke **sunview** on the first display as you normally would. This starts an instance of **sunview** on the default frame buffer (**/dev/fb**).
- In a **shelltool**, run:

  sunview -d *device* **&**

  This starts another device. A typical choice might be **/dev/cgone**.
- In that same **shelltool**, run:

  adjacentscreens /dev/fb -r *device*

  This sets up the cursor to switch between screens as it crosses the right or left edge of the respective screens.

Multiple Desktops on the Same Screen

Machines that support multiple plane groups, such as the Sun-3/110 system, can support independent **sunview** processes on each plane group. They can share keyboard and mouse input in a manner similar to that for multiple screens. To set up two plane groups:

- Start **sunview** in the color plane group by running:

  sunview -8bit_color_only -toggle_enable

This starts **sunview** on the default frame buffer named **/dev/fb**, but limits access to the color plane group.

- In a **shelltool**, run:

  sunview –d /dev/bwtwo0 –toggle_enable –n &

  This starts **sunview** in the overlay plane accessed by **/dev/bwtwo0**.

- Run:

  adjacentscreens –c /dev/fb –l /dev/bwtwo0

  This sets up the pointer to switch between desktops as it crosses the right or left edge of the respective desktops.

Pre-3.2 applications cannot be run on the **–8bit_color_only** desktop, because they do not write to the overlay plane.

switcher(1), another application for switching between desktops, uses some amusing video wipe animation. It can also be used to toggle the enable plane. See **switcher**(1) for details.

Generic Tool Arguments

Most window-based tools take the following arguments in their command lines:

FLAG	(LONG FLAG)	ARGUMENTS	NOTES
–Ww	**(–width)**	columns	
–Wh	**(–height)**	lines	
–Ws	**(–size)**	*x y*	*x* and *y* are in pixels
–Wp	**(–position)**	*x y*	*x* and *y* are in pixels
–WP	**(–icon_position)**	*x y*	*x* and *y* are in pixels
–Wl	**(–label)**	*string*	
–Wi	**(–iconic)**		makes the application start iconic (closed)
–Wt	**(–font)**	*filename*	
–Wn	**(–no_name_stripe)**		
–Wf	**(–foreground_color)**	**red green blue**	0-255 (no color-full color)
–Wb	**(–background_color)**	**red green blue**	0-255 (no color-full color)
–Wg	**(–set_default_color)**		(apply color to subwindows too)
–WI	**(–icon_image)**	*filename*	(for applications with non-default icons)
–WL	**(–icon_label)**	*string*	(for applications with non-default icons)
–WT	**(–icon_font)**	*filename*	(for applications with non-default icons)
–WH	**(–help)**		print this table

Each flag option may be specified in either its short form or its long form; the two are completely synonymous.

SunView Applications

Some of the applications that run in the SunView environment:

clock(1), **cmdtool**(1), **dbxtool**(1), **defaultsedit**(1), **fontedit**(1), **gfxtool**(1), **iconedit**(1), **lockscreen**(1), **mailtool**(1), **overview**(1), **perfmeter**(1), **shelltool**(1), **tektool**(1), **textedit**(1), **traffic**(1C)

Some of the utility programs that run in or with the SunView environment:

adjacentscreens(1), **clear_functions**(1), **get_selection**(1), **stty_from_defaults**(1), **swin**(1), **switcher**(1), **toolplaces**(1)

ENVIRONMENT

DEFAULTS_FILE The value of this environment variable indicates the file from which SunView defaults are read. When it is undefined, defaults are read from the **.defaults** file in your home directory.

FILES

~/.sunview
/usr/lib/.sunview
/usr/lib/.rootmenu
/usr/lib/fonts/fixedwidthfonts/*
/dev/win*x*
/dev/ptyp*x*
/dev/ttyp*x*
/dev/fb
/dev/kbd
/dev/mouse
/etc/utmp

SEE ALSO

adjacentscreens(1), **clear_functions**(1), **clock**(1), **cmdtool**(1), **dbxtool**(1), **defaultsedit**(1), **fontedit**(1), **get_selection**(1), **gfxtool**(1), **iconedit**(1), **lockscreen**(1), **mailtool**(1), **overview**(1), **perfmeter**(1), **screendump**(1), **shelltool**(1), **stty_from_defaults**(1), **swin**(1), **switcher**(1), **tektool**(1), **textedit**(1), **toolplaces**(1), **traffic**(1C), **fbtab**(5), **svdtab**(5)

BUGS

Console messages ignore window boundaries unless redirected to a console window. This can disrupt the **sunview** desktop display. The display can be restored using the **Redisplay All** item on the SunView menu. To prevent this, use the **Console** item to start a console window.

With an optical mouse, sometimes the arrow-shaped cursor does not move at start-up; moving the mouse in large circles on its pad normally brings it to life.

sunview requires that the **/etc/utmp** file be given read and write permission for all users.

On a color display, colors may "go strange" when the cursor is in certain windows that request a large number of colors.

When running multiple desktops, only one console window can be used.

In **Click-to-type** mode, it is impossible to exit from **sunview** by typing CTRL-D CTRL-Q.

NAME

units – conversion program

SYNOPSIS

units

DESCRIPTION

units converts quantities expressed in various standard scales to their equivalents in other scales. It works interactively in this fashion:

You have: **inch**
You want: **cm**
* 2.540000e+00
/ 3.937008e–01

A quantity is specified as a multiplicative combination of units optionally preceded by a numeric multiplier. Powers are indicated by suffixed positive integers, division by the usual sign:

You have: **15 lbs force/in2**
You want: **atm**
* 1.020689e+00
/ 9.797299e–01

units only does multiplicative scale changes. Thus it can convert Kelvin to Rankine, but not Celsius to Fahrenheit. Most familiar units, abbreviations, and metric prefixes are recognized, together with a generous leavening of exotica and a few constants of nature including:

pi Ratio of circumference to diameter,
c Speed of light,
e Charge on an electron,
g Acceleration of gravity,
force Same as **g**,
mole Avogadro's number,
water Pressure head per unit height of water,
au Astronomical unit.

pound is not recognized as a unit of mass; **lb** is. **pound** refers to a British pound. Compound names are run together (for instance, **lightyear**). British units that differ from their U.S. counterparts are prefixed thus: **brgallon**. Currency is denoted **belgiumfranc**, **britainpound**, ... For a complete list of units, type:

cat /usr/lib/units

FILES

/usr/lib/units

BUGS

Do not base your financial plans on the currency conversions.

NAME

unix2dos – convert text file from ISO format to DOS format

SYNOPSIS

unix2dos [**–ascii**] [**–iso**] [**–7**] *originalfile convertedfile*

DESCRIPTION

unix2dos converts ISO standard characters to the corresponding characters in the DOS extended character set.

This command may be invoked from either DOS or SunOS. However, the filenames must conform to the conventions of the environment in which the command is invoked.

If the original file and the converted file are the same, **unix2dos** will rewrite the original file after converting it.

OPTIONS

–ascii Adds carriage returns and converts end of file characters in SunOS format text files to conform to DOS requirements.

–iso This is the default. Converts ISO standard characters to the corresponding character in the DOS extended character set.

–7 Convert 8 bit SunOS characters to 7 bit DOS characters.

DIAGNOSTICS

File *filename* **not found, or no read permission**

The input file you specified does not exist, or you do not have read permission (check with the SunOS command **ls –l**).

Bad output filename *filename*, **or no write permission**

The output file you specified is either invalid, or you do not have write permission for that file or the directory that contains it. Check also that the drive or diskette is not write-protected.

Error while writing to temporary file

An error occurred while converting your file, possibly because there is not enough space on the current drive. Check the amount of space on the current drive using the DIR command. Also be certain that the default diskette or drive is write-enabled (not write-protected). Note that when this error occurs, the original file remains intact.

Could not rename tmpfile to *filename*.
Translated tmpfile name = *filename*.

The program could not perform the final step in converting your file. Your converted file is stored under the name indicated on the second line of this message.

SEE ALSO

dos(1), **dos2unix**(1)

Sun386i Advanced Skills
Sun MS-DOS Reference Manual

When **MAP_FIXED** is not set, the system uses *addr* as a hint in an implementation-defined manner to arrive at *pa*. The *pa* so chosen will be an area of the address space which the system deems suitable for a mapping of *len* bytes to the specified object. All implementations interpret an *addr* value of zero as granting the system complete freedom in selecting *pa*, subject to constraints described below. A non-zero value of *addr* is taken to be a suggestion of a process address near which the mapping should be placed. When the system selects a value for *pa*, it will never place a mapping at address 0, nor will it replace any extant mapping, nor map into areas considered part of the potential data or stack ''segments''.

The parameter *off* is constrained to be aligned and sized according to the value returned by **getpagesize** (2). When **MAP_FIXED** is specified, the parameter *addr* must also meet these constraints. The system performs mapping operations over whole pages. Thus, while the parameter *len* need not meet a size or alignment constraint, the system will include in any mapping operation any partial page specified by the range [*pa, pa + len*).

mmap( ) allows [*pa, pa + len*) to extend beyond the end of the object, both at the time of the **mmap( )** and while the mapping persists, for example if the file was created just prior to the **mmap( )** and has no contents, or if the file is truncated. Any reference to addresses beyond the end of the object, however, will result in the delivery of a **SIGBUS** signal.

The system will always zero-fill any partial page at the end of an object. Further, the system will never write out any modified portions of the last page of an object which are beyond its end. References to whole pages following the end of an object will result in a **SIGBUS** signal. **SIGBUS** may also be delivered on various filesystem conditions, including quota exceeded errors.

If the process calls **mlockall**(3) with the **MCL_FUTURE** flag, the pages mapped by all future calls to **mmap( )** will be locked in memory. In this case, if not enough memory could be locked, **mmap( )** fails and sets **errno** to EAGAIN.

RETURN VALUES

mmap( ) returns the address at which the mapping was placed (*pa*) on success. On failure, it returns −1 and sets **errno** to indicate the error.

ERRORS

EACCES	*fd* was not open for read and **PROT_READ** or **PROT_EXEC** were specified.
	fd was not open for write and **PROT_WRITE** was specified for a **MAP_SHARED** type mapping.
EAGAIN	Some or all of the mapping could not be locked in memory.
EBADF	*fd* was not open.
EINVAL	The arguments *addr* (if **MAP_FIXED** was specified) and *off* were not multiples of the page size as returned by **getpagesize** (2).
	The **MAP_TYPE** field in *flags* was invalid (neither **MAP_PRIVATE** nor **MAP_SHARED**).
ENODEV	*fd* refered to an object for which **mmap( )** is meaningless, such as a terminal.
ENOMEM	**MAP_FIXED** was specified, and the range [*addr, addr + len*) exceeded that allowed for the address space of a process.
	MAP_FIXED was not specified and there was insufficient room in the address space to effect the mapping.
ENXIO	Addresses in the range [*off, off + len*) are invalid for *fd*.

SEE ALSO

fork(2V), **getpagesize**(2), **mprotect**(2), **munmap**(2), **mlockall**(3)

NAME

mount – mount file system

SYNOPSIS

```
#include <sys/mount.h>

int mount(type, dir, M_NEWTYPE | flags, data)
char *type;
char *dir;
int flags;
caddr_t data;
```

SYSTEM V SYNOPSIS

```
int mount(spec, dir, rdonly)
char *spec;
char *dir;
int rdonly;
```

DESCRIPTION

mount( ) attaches a file system to a directory. After a successful return, references to directory *dir* will refer to the root directory on the newly mounted file system. *dir* is a pointer to a null-terminated string containing a path name. *dir* must exist already, and must be a directory. Its old contents are inaccessible while the file system is mounted.

mount( ) may be invoked only by the super-user.

The *flags* argument is constructed by the logical OR of the following bits (defined in **<sys/mount.h>**):

M_RDONLY	mount filesystem read-only.
M_NOSUID	ignore set-uid bit on execution.
M_NEWTYPE	this flag must always be set.
M_GRPID	use BSD file-creation semantics (see **open**(2V)).
M_REMOUNT	change options on an existing mount.
M_NOSUB	disallow mounts beneath this filesystem.

Physically write-protected and magnetic tape file systems must be mounted read-only or errors will occur when access times are updated, whether or not any explicit write is attempted.

The *type* string indicates the type of the filesystem. *data* is a pointer to a structure which contains the type specific arguments to mount. Below is a list of the filesystem types supported and the type specific arguments to each:

```
4.2
    struct ufs_args {
            char        *fspec;     /* Block special file to mount */
    };
"lo"
    struct lo_args {
            char        *fsdir;     /* Pathname of directory to mount */
    };
"nfs"
    #include            <nfs/nfs.h>
    #include            <netinet/in.h>
    struct nfs_args {
            struct sockaddr_in  *addr; /* file server address */
            fhandle_t   *fh;        /* File handle to be mounted */
            int         flags;      /* flags */
```

```
                int     wsize;      /* write size in bytes */
                int     rsize;      /* read size in bytes */
                int     timeo;      /* initial timeout in .1 secs */
                int     retrans;    /* times to retry send */
                char    *hostname;  /* server's hostname */
                int     acregmin;   /* attr cache file min secs */
                int     acregmax;   /* attr cache file max secs */
                int     acdirmin;   /* attr cache dir min secs */
                int     acdirmax;   /* attr cache dir max secs */
                char    *netname;   /* server's netname */

        };
    rfs
        struct rfs_args {
                char    *rmtfs      /* name of remote resource */
                struct token {
                        int         t_id;/* token id */
                        char        t_uname[64];/* domain.machine name */
                } *token;           /* Identifier of remote machine */
        };
```

SYSTEM V DESCRIPTION

mount( ) requests that a file system contained on the block special file identified by *spec* be mounted on the directory identified by *dir*. *spec* and *dir* point to path names. When **mount( )** succeeds, subsequent references to the file named by *dir* refer to the root directory on the mounted file system.

The **M_RDONLY** bit of *rdonly* is used to control write permission on the mounted file system. If the bit is set, writing is not allowed. Otherwise, writing is permitted according to the access permissions of individual files.

RETURN VALUES

mount( ) returns:

0 on success.

−1 on failure and sets **errno** to indicate the error.

ERRORS

EACCES	Search permission is denied for a component of the path prefix of *dir*.
EBUSY	Another process currently holds a reference to *dir*.
EFAULT	*dir* points outside the process's allocated address space.
EIO	The UFS file system has been tuned to use contiguous blocks that are larger than the maximum block size than the device drivers can support.
ELOOP	Too many symbolic links were encountered in translating the path name of *dir*.
ENAMETOOLONG	The length of the path argument exceeds {PATH_MAX}.
	A pathname component is longer than {NAME_MAX} (see **sysconf**(2V)) while {_POSIX_NO_TRUNC} is in effect (see **pathconf**(2V)).
ENODEV	The file system type specified by *type* is not valid or is not configured into the system.
ENOENT	A component of *dir* does not exist.
ENOTDIR	The file named by *dir* is not a directory.
EPERM	The caller is not the super-user.

For a 4.2 file system, **mount( )** fails when one of the following occurs:

EACCES	Search permission is denied for a component of the path prefix of *fspec*.
EFAULT	*fspec* points outside the process's allocated address space.
EINVAL	The super block for the file system had a bad magic number or an out of range block size.
EIO	An I/O error occurred while reading from or writing to the file system.
ELOOP	Too many symbolic links were encountered in translating the path name of *fspec*.
EMFILE	No space remains in the mount table.
ENAMETOOLONG	The length of the path argument exceeds {PATH_MAX}.
	A pathname component is longer than {NAME_MAX} (see **sysconf**(2V)) while {_POSIX_NO_TRUNC} is in effect (see **pathconf**(2V)).
ENOENT	A component of *fspec* does not exist.
ENOMEM	Not enough memory was available to read the cylinder group information for the file system.
ENOTBLK	*fspec* is not a block device.
ENOTDIR	A component of the path prefix of *fspec* is not a directory.
ENXIO	The major device number of *fspec* is out of range (this indicates no device driver exists for the associated hardware).

SYSTEM V ERRORS

EBUSY	The device referred to by *spec* is currently mounted.
	There are no more mount table entries.
ENOENT	The file referred to by *spec* or *dir* does not exist.
ENOTBLK	*spec* is not a block special device.
ENOTDIR	A component of the path prefix of *dir* or *spec* is not a directory.
ENXIO	The device referred to by *spec* does not exist.

SEE ALSO

unmount(2V), **open**(2V), **lofs**(4S), **fstab**(5), **mount**(8)

BUGS

Some of the error codes need translation to more obvious messages.

NAME

issecure – indicates whether system is running secure

SYNOPSIS

int issecure( )

DESCRIPTION

This function tells whether the system has been configured to run in secure mode. It returns 0 if the system is not running secure, and non-zero if the system is running secure.

NAME

kvm_getu, kvm_getcmd – get the u-area or invocation arguments for a process

SYNOPSIS

```
#include <kvm.h>
#include <sys/param.h>
#include <sys/user.h>
#include <sys/proc.h>

struct user *kvm_getu(kd, proc)
kvm_t *kd;
struct proc *proc;

int kvm_getcmd(kd, proc, u, arg, env)
kvm_t *kd;
struct proc *proc;
struct user *u;
char ***arg;
char ***env;
```

DESCRIPTION

kvm_getu( ) reads the u-area of the process specified by *proc* to an area of static storage associated with *kd* and returns a pointer to it. Subsequent calls to **kvm_getu( )** will overwrite this static area.

kd is a pointer to a kernel identifier returned by **kvm_open**(3K). *proc* is a pointer to a copy (in the current process' address space) of a *proc* structure (obtained, for instance, by a prior **kvm_nextproc**(3K) call). As a side effect, **kvm_getu( )** sets the address space used to resolve user addresses in subsequent **kvm_read( )** and **kvm_write( )** calls to be the space belonging to *proc*.

kvm_getcmd( ) constructs a list of string pointers that represent the command arguments and environment that were used to initiate the process specified by *proc*.

kd is a pointer to a kernel identifier returned by **kvm_open**(3K). *u* is a pointer to a copy (in the current process' address space) of a *user* structure (obtained, for instance, by a prior **kvm_getu( )** call). If *arg* is not NULL, then the command line arguments are formed into a null-terminated array of string pointers. The address of the first such pointer is returned in *arg*. If *env* is not NULL, then the environment is formed into a null-terminated array of string pointers. The address of the first of these is returned in *env*.

The pointers returned in *arg* and *env* refer to data allocated by **malloc**(3V) and should be freed (by a call to **free** (see **malloc**(3V)) when no longer needed. Both the string pointers and the strings themselves are deallocated when freed.

Since the environment and command line arguments may have been modified by the user process, there is no guarantee that it will be possible to reconstruct the original command at all. Thus, **kvm_getcmd( )** will make the best attempt possible, returning –1 if the user process data is unrecognizable.

RETURN VALUES

On success, **kvm_getu( )** returns a pointer to a copy of the u-area of the process specified by *proc*. On failure, it returns NULL.

kvm_getcmd( ) returns:

0 on success.

–1 on failure.

SEE ALSO

execve(2V), **kvm_nextproc**(3K), **kvm_open**(3K), **kvm_read**(3K), **malloc**(3V)

NOTES

If **kvm_getcmd( )** returns −1, the caller still has the option of using the command line fragment that is stored in the u-area.

NAME

kvm_getproc, kvm_nextproc, kvm_setproc – read system process structures

SYNOPSIS

```
#include <kvm.h>
#include <sys/param.h>
#include <sys/time.h>
#include <sys/proc.h>

struct proc *kvm_getproc(kd, pid)
kvm_t *kd;
int pid;

struct proc *kvm_nextproc(kd)
kvm_t *kd;

int kvm_setproc(kd)
kvm_t *kd;
```

DESCRIPTION

kvm_nextproc() may be used to sequentially read all of the system process structures from the kernel identified by *kd* (see **kvm_open**(3K)). Each call to **kvm_nextproc()** returns a pointer to the static memory area that contains a copy of the next valid process table entry. There is no guarantee that the data will remain valid across calls to **kvm_nextproc()**, **kvm_setproc()**, or **kvm_getproc()**. Therefore, if the process structure must be saved, it should be copied to non-volatile storage.

For performance reasons, many implementations will cache a set of system process structures. Since the system state is liable to change between calls to **kvm_nextproc()**, and since the cache may contain obsolete information, there is no guarantee that *every* process structure returned refers to an active process, nor is it certain that *all* processes will be reported.

kvm_setproc() rewinds the process list, enabling **kvm_nextproc()** to rescan from the beginning of the system process table. **kvm_setproc()** will always flush the process structure cache, allowing an application to re-scan the process table of a running system.

kvm_getproc() locates the **proc** structure of the process specified by *pid* and returns a pointer to it. **kvm_getproc()** does not interact with the process table pointer manipulated by **kvm_nextproc**, however, the restrictions regarding the validity of the data still apply.

RETURN VALUES

On success, **kvm_nextproc()** returns a pointer to a copy of the next valid process table entry. On failure, it returns NULL.

On success, **kvm_getproc()** returns a pointer to the **proc** structure of the process specified by *pid*. On failure, it returns NULL.

kvm_setproc() returns:

0 on success.

−1 on failure.

SEE ALSO

kvm_getu(3K), **kvm_open**(3K), **kvm_read**(3K)

NAME

mblen, mbstowcs, mbtowc, wcstombs, wctomb – multibyte character handling

SYNOPSIS

```
#include <stdlib.h>

int mblen(s, n)
char *s;
size_t n;

size_t mbstowcs(s, pwcs, n)
char *s;
wchar_t *pwcs;
size_t n;

int mbtowc(pwc, s, n)
wchar_t *pwc;
char *s;
size_t n;

int wcstombs(s, pwcs, n)
char *s;
wchar_t *pwcs;
size_t n;

int wctomb(s, wchar)
char *s;
wchar_t wcar;
```

DESCRIPTION

The behavior of these functions is affected by the **LC_CTYPE** category of the program's locale. For a stat-dependent encoding, each function is placed into its initial state by a call for which its character pointer argument, *s*, is a NULL pointer. Subsequent calls with *s* as other than a NULL pointer cause the internal stste of the function to be altered as necessary. A call with a *s* as a NULL pointer causes these functions to return a nonzero value if encodings have state dependency, and zero otherwise. After the **LC_CTYPE** category is changed, the shift state of these functions is indeterminate.

If *s* is not a NULL pointer, these functions work as follows:

mblen( )

Determines the number of bytes comprising the multibyte character pointed to by *s*.

mbstowcs( )

Converts a sequence of multibyte characters that begins in the initial shift state from the array pointed to by *s* into a sequence of corresponding codes and stores no more than *n* codes into the array pointed to by *pwcs*. No multibyte characters that follow a null character (which is converted into a code with value zero) will be examined or converted. Each multibyte character is converted as if by a call to **mbtowc( )**, except that the shift state of **mbtowc( )** is not affected.

No more than *n* elements will be modified in the array pointed to by *pwcs*. If copying takes place between objects that overlap, the behavior is undefined.

mbtowc( )

Determines the number of bytes that comprise the multibyte character pointed to by *s*. **mbtowc( )** then determines the code for value of type **wchar_t** that corresponds to that multibyte character. The value of the code corresponding to the null caharacter is zero. If the multibyte character is valid and *pwc* is not a null pointer, **mbtowc( )** stores the code in the object pointed to by *pwc*. At most *n* bytes of the array pointed to by *s* will be examined.

wcstowcs( )

Converts a sequence of codes that correspond to multibyte characters from the array pointed to by *pwcs* into a sequence of multibyte characters that begins in the initial shift state and stores these multibyte characters into the array pointed to by *s*, stopping if a multibyte character would exceed the limit of *n* total bytes or if a null character is stored. Each code is converted as if by a call to **wctomb( )**, except that the shift state of **wctomb( )** is not affected.

wctomb( )

Determines the number of bytes needed to represent the multibyte character corresponding to the code whose value is *wchar* (including any change in shift state). **wctomb( )** stores the multibyte character representation in the array object pointed to by *s* (if *s* is not a null pointer). At most, **MB_CUR_MAX** characters are stored. If the value of *wchar* is zero, **wctomb( )** is left in the initial shift state.

RETURN VALUES

If *s* is a null pointer, **mblen( )**, **mbtowc( )**, and **wctomb( )** return a nonzero or zero value, if multibyte character encodings, respectively, do or do not have state dependent encodings.

If *s* is not a null pointer, **mblen( )** and **mbtowc( )** either return 0 (if *s* points to the null character), or return the number of bytes that comprise the converted multibyte character (if the next *n* or fewer bytes form a valid multibyte character), or return −1 (if they do not form a valid multibyte character).

In no case will the value returned by **mbtowc( )** be greater than *n* or the value of the **MB_CUR_MAX** macro. If *s* is not a null pointer, **wctomb( )** returns −1 (if the value does not correspond to a valid multibyte character), or returns the number of bytes that comprise the multibyte character corresponding to *wchar*.

If an invalid multibyte character is encountered, **mbstowcs( )** and **wcstombs( )** return **(size_t)** −1. Otherwise, they return the number of bytes modified, not including a terminating null character, if any.

NOTE

When an application program using these routines is statically linked, user defined multibyte character handling routines in a user provided shared library will not be used.

NAME

intro – introduction to device drivers, protocols, and network interfaces

DESCRIPTION

This section describes device drivers, high-speed network interfaces, and protocols available under SunOS. The system provides drivers for a variety of hardware devices, such as disks, magnetic tapes, serial communication lines, mice, and frame buffers, as well as virtual devices such as pseudo-terminals and windows. SunOS provides hardware support and a network interface for the 10-Megabit Ethernet, along with interfaces for the IP protocol family and a STREAMS-based Network Interface Tap (NIT) facility.

In addition to describing device drivers that are supported by the 4.3BSD operating system, this section contains subsections that describe:

- SunOS-specific device drivers, under '4S'.
- Protocol families, under '4F'.
- Protocols and raw interfaces, under '4P'.
- STREAMS modules, under '4M'.
- Network interfaces, under '4N'.

Configuration

The SunOS kernel can be configured to include or omit many of the device drivers described in this section. The CONFIG section of the manual page gives the line(s) to include in the kernel configuration file for each machine architecture on which a device is supported. If no specific architectures are indicated, the configuration syntax applies to all Sun systems.

The GENERIC kernel is the default configuration for SunOS. It contains all of the optional drivers for a given machine architecture. See **config**(8), for details on configuring a new SunOS kernel.

The manual page for a device driver may also include a DIAGNOSTICS section, listing error messages that the driver might produce. Normally, these messages are logged to the appropriate system log using the kernel's standard message-buffering mechanism (see **syslogd**(8)); they may also appear on the system console.

Ioctls

Various special functions, such as querying or altering the operating characteristics of a device, are performed by supplying appropriate parameters to the **ioctl**(2) system call. These parameters are often referred to as ''ioctls.'' Ioctls for a specific device are presented in the manual page for that device. Ioctls that pertain to a class of devices are listed in a manual page with a name that suggests the class of device, and ending in 'io', such as **mtio**(4) for magnetic tape devices, or **dkio**(4S) for disk controllers. In addition, some ioctls operate directly on higher-level objects such as files, terminals, sockets, and streams:

- Ioctls that operate directly on files, file descriptors, and sockets are described in **filio**(4). Note: the **fcntl**(2V) system call is the primary method for operating on file descriptors as such, rather than on the underlying files. Also note that the **setsockopt** system call (see **getsockopt**(2)) is the primary method for operating on sockets as such, rather than on the underlying protocol or network interface. Ioctls for a specific network interface are documented in the manual page for that interface.
- Ioctls for terminals, including pseudo-terminals, are described in **termio**(4). This manual page includes information about both the BSD **termios** structure, as well as the System V **termio** structure.
- Ioctls for STREAMS are described in **streamio**(4).

Devices Always Present

Device drivers present in every kernel include:

- The paging device; see **drum**(4).
- Drivers for accessing physical, virtual, and I/O space in memory; see **mem**(4S).
- The data sink; see **null**(4).

Terminals and Serial Communications Devices

Serial communication lines are normally supported by the terminal driver; see **tty**(4). This driver manages serial lines provided by communications drivers, such as those described in **mti**(4S) and **zs**(4S). The terminal driver also handles serial lines provided by virtual terminals, such as the Sun console monitor described in **console**(4S), and true pseudo-terminals, described in **pty**(4).

Disk Devices

Drivers for the following disk controllers provide standard block and raw interfaces under SunOS:

- SCSI controllers, in **sd**(4S),
- Xylogics 450 and 451 SMD controllers, in **xy**(4S),
- Xylogics 7053 SMD controllers, in **xd**(4S),
- IPI controllers, in **id**(4S).

Ioctls to query or set a disk's geometry and partitioning are described in **dkio**(4S).

Magnetic Tape Devices

Magnetic tape devices supported by SunOS include those described in **ar**(4S), **tm**(4S), **st**(4S), and **xt**(4S). Ioctls for all tape-device drivers are described in **mtio**(4S).

Frame Buffers

Frame buffer devices include color frame buffers described in the **cg***(4S) and **gt**(4S) manual pages, monochrome frame buffers described in the **bw***(4S) manual pages, graphics processor interfaces described in the **gp***(4S) manual pages, and an indirect device for the console frame buffer described in **fb**(4S). Ioctls for all frame-buffer devices are described in **fbio**(4S).

Miscellaneous Devices

Miscellaneous devices include the console keyboard described in **kbd**(4S), the console mouse described in **mouse**(4S), window devices described in **win**(4S), and the DES encryption-chip interface described in **des**(4S).

Network-Interface Devices

SunOS supports the 10-Megabit Ethernet as its primary network interface; see **ie**(4S) and **le**(4S) for details. However, a software loopback interface, **lo**(4) is also supported. General properties of these network interfaces are described in **if**(4N), along with the ioctls that operate on them.

Support for network routing is described in **routing**(4N).

Protocols and Protocol Families

SunOS supports both socket-based and STREAMS-based network communications. The Internet protocol family, described in **inet**(4F), is the primary protocol family primary supported by SunOS, although the system can support a number of others. The raw interface provides low-level services, such as packet fragmentation and reassembly, routing, addressing, and basic transport for socket-based implementations. Facilities for communicating using an Internet-family protocol are generally accessed by specifying the **AF_INET** address family when binding a socket; see **socket**(2) for details.

Major protocols in the Internet family include:

- The Internet Protocol (IP) itself, which supports the universal datagram format, as described in **ip**(4P). This is the default protocol for SOCK_RAW type sockets within the **AF_INET** domain.
- The Transmission Control Protocol (TCP); see **tcp**(4P). This is the default protocol for SOCK_STREAM type sockets.
- The User Datagram Protocol (UDP); see **udp**(4P). This is the default protocol for SOCK_DGRAM type sockets.
- The Address Resolution Protocol (ARP); see **arp**(4P).
- The Internet Control Message Protocol (ICMP); see **icmp**(4P).

The Network Interface Tap (NIT) protocol, described in **nit**(4P), is a STREAMS-based facility for accessing the network at the link level.

SEE ALSO

fcntl(2V), **getsockopt**(2), **ioctl**(2), **socket**(2), **ar**(4S), **arp**(4P), **dkio**(4S), **drum**(4), **fb**(4S), **fbio**(4S), **filio**(4), **icmp**(4P), **if**(4N), **inet**(4F), **ip**(4P), **kbd**(4S), **le**(4S), **lo**(4), **mem**(4S), **mti**(4S), **mtio**(4), **nit**(4P), **null**(4), **pty**(4), **routing**(4N), **sd**(4S), **st**(4S) **streamio**(4), **tcp**(4P), **termio**(4), **tm**(4S), **tty**(4), **udp**(4P), **win**(4S), **xd**(4S), **xy**(4S), **zs**(4S)

LIST OF DEVICES, INTERFACES AND PROTOCOLS

Name	Appears on Page	Description
alm	**mcp**(4S)	ALM-2 Asynchronous Line Multiplexer
ar	**ar**(4S)	Archive 1/4 inch Streaming Tape Drive
arp	**arp**(4P)	Address Resolution Protocol
atbus	**mem**(4S)	main memory and bus I/O space
audio	**audio**(4)	telephone quality audio device
bwtwo	**bwtwo**(4S)	black and white memory frame buffer
cdromio	**cdromio**(4S)	CDROM control operations
cgeight	**cgeight**(4S)	24-bit color memory frame buffer
cgfour	**cgfour**(4S)	Sun-3 color memory frame buffer
cgnine	**cgnine**(4S)	24-bit VME color memory frame buffer
cgsix	**cgsix**(4S)	accelerated 8-bit color frame buffer
cgthree	**cgthree**(4S)	8-bit color memory frame buffer
cgtwelve	**cgtwelve**(4S)	24-bit SBus color memory frame buffer
cgtwo	**cgtwo**(4S)	color graphics interface
console	**console**(4S)	console driver and terminal emulator
db	**db**(4M)	SunDials STREAMS module
des	**des**(4S)	DES encryption chip interface
dkio	**dkio**(4S)	generic disk control operations
drum	**drum**(4)	paging device
eeprom	**mem**(4S)	main memory and bus I/O space
fb	**fb**(4S)	driver for Sun console frame buffer
fbio	**fbio**(4S)	frame buffer control operations
fd	**fd**(4S)	Disk driver for Floppy Disk Controllers
filio	**filio**(4)	ioctls that operate directly on files, file descriptors, and sockets
fpa	**fpa**(4S)	Sun-3 floating-point accelerator
gpone	**gpone**(4S)	graphics processor
gt	**gt**(4S)	double buffered 24-bit SBus graphics accelerator
icmp	**icmp**(4P)	Internet Control Message Protocol
id	**id**(4S)	disk driver for IPI disk controllers
ie	**ie**(4S)	Intel 10 Mb/s Ethernet interface
if	**if**(4N)	general properties of network interfaces
inet	**inet**(4F)	Internet protocol family
ip	**ip**(4P)	Internet Protocol
ipi	**ipi**(4S)	IPI driver
is	**is**(4S)	IPI channel driver for Sun IPI string controllers
kb	**kb**(4M)	Sun keyboard STREAMS module
kbd	**kbd**(4S)	Sun keyboard
kmem	**mem**(4S)	main memory and bus I/O space
ldterm	**ldterm**(4M)	standard terminal STREAMS module
le	**le**(4S)	LANCE 10Mb/s Ethernet interface
lo	**lo**(4N)	software loopback network interface
lofs	**lofs**(4S)	loopback virtual file system
mcp	**mcp**(4S)	MCP Multiprotocol Communications Processor

mem	**mem**(4S)	main memory and bus I/O space
mouse	**mouse**(4S)	Sun mouse
ms	**ms**(4M)	Sun mouse STREAMS module
mti	**mti**(4S)	Systech MTI-800/1600 multi-terminal interface
mtio	**mtio**(4)	general magnetic tape interface
NFS	**nfs**(4P)	network file system
nit	**nit**(4P)	Network Interface Tap
nit_buf	**nit_buf**(4M)	STREAMS NIT buffering module
nit_if	**nit_if**(4M)	STREAMS NIT device interface module
nif_pf	**nit_pf**(4M)	STREAMS NIT packet filtering module
null	**null**(4)	data sink
openprom	**openprom**(4S)	PROM monitor configuration interface
pp	**pp**(4)	Centronics-compatible parallel printer port
pty	**pty**(4)	pseudo-terminal driver
rfs	**rfs**(4)	remote file sharing service
root	**root**(4S)	pseudo-driver for Sun386i root disk
routing	**routing**(4N)	system supporting for local network packet routing
sbus	**mem**(4S)	main memory and bus I/O space
sd	**sd**(4S)	driver for SCSI disk devices
sockio	**sockio**(4)	ioctls that operate directly on sockets
sr	**sr**(4S)	driver for CDROM SCSI controller
st	**st**(4S)	driver for SCSI tape devices
streamio	**streamio**(4)	STREAMS ioctl commands
taac	**taac**(4S)	Sun applications accelerator
tcp	**tcp**(4P)	Internet Transmission Control Protocol
tcptli	**tcptli**(4P)	TLI-Conforming TCP Stream-Head
termio	**termio**(4)	general terminal interface
tfs	**tfs**(4S)	translucent file service
tm	**tm**(4S)	Tapemaster 1/2 inch tape controller
tmpfs	**tmpfs**(4S)	memory based filesystem
ttcompat	**ttcompat**(4M)	V7 and 4BSD STREAMS compatibility module
tty	**tty**(4)	controlling terminal interface
udp	**udp**(4P)	Internet User Datagram Protocol
unix	**unix**(4F)	UNIX domain protocol family
vd	**vd**(4)	loadable modules interface
vme16d16	**mem**(4S)	main memory and bus I/O space
vme16d32	**mem**(4S)	main memory and bus I/O space
vme24d16	**mem**(4S)	main memory and bus I/O space
vme24d32	**mem**(4S)	main memory and bus I/O space
vme32d16	**mem**(4S)	main memory and bus I/O space
vme32d32	**mem**(4S)	main memory and bus I/O space
vpc	**vpc**(4S)	Systech VPC-2200 Versatec printer/plotter
vx	**vx**(4S)	Sun applications accelerator
win	**win**(4S)	Sun window system
xd	**xd**(4S)	Disk driver for Xylogics 7053 SMD Disk Controller
xt	**xt**(4S)	Xylogics 472 1/2 inch tape controller
xy	**xy**(4S)	Disk driver for Xylogics 450 and 451 SMD Disk Controllers
zero	**mem**(4S)	main memory and bus I/O space
zero	**zero**(4S)	source of zeroes
zs	**zs**(4S)	Zilog 8530 SCC serial communications driver

coming from other machines. This allows a host to act as an "ARP server" which may be useful in convincing an ARP-only machine to talk to a non-ARP machine.

ARP is also used to negotiate the use of trailer IP encapsulations; trailers are an alternate encapsulation used to allow efficient packet alignment for large packets despite variable-sized headers. Hosts which wish to receive trailer encapsulations so indicate by sending gratuitous ARP translation replies along with replies to IP requests; they are also sent in reply to IP translation replies. The negotiation is thus fully symmetrical, in that either or both hosts may request trailers. The **ATF_USETRAILERS** flag is used to record the receipt of such a reply, and enables the transmission of trailer packets to that host.

ARP watches passively for hosts impersonating the local host (that is, a host which responds to an ARP mapping request for the local host's address).

SEE ALSO

ec(4S), **ie**(4S), **inet**(4F), **arp**(8C), **ifconfig**(8C)

Plummer, Dave, "*An Ethernet Address Resolution Protocol -or- Converting Network Protocol Addresses to 48.bit Ethernet Addresses for Transmission on Ethernet Hardware*," RFC 826, Network Information Center, SRI International, Menlo Park, Calif., November 1982. (Sun 800-1059-10)

Leffler, Sam, and Michael Karels, "*Trailer Encapsulations*," RFC 893, Network Information Center, SRI International, Menlo Park, Calif., April 1984.

DIAGNOSTICS

duplicate IP address!! sent from ethernet address: %x:%x:%x:%x:%x:%x.

ARP has discovered another host on the local network which responds to mapping requests for its own Internet address.

BUGS

ARP packets on the Ethernet use only 42 bytes of data, however, the smallest legal Ethernet packet is 60 bytes (not including CRC). Some systems may not enforce the minimum packet size, others will.

NAME

audio – telephone quality audio device

CONFIG — SUN-4c, SUN-4m SYSTEMS

device-driver audio

DESCRIPTION

The **audio** device plays and records a single channel of sound using the AM79C30A Digital Subscriber Controller chip. The chip has a built-in analog-to-digital converter (ADC) and digital-to-analog converter (DAC) that can drive either the built-in speaker or an external headphone jack, selectable under software control. Digital audio data is sampled at a rate of 8000 samples per second with 12-bit precision, though the data is compressed, using *u*–law encoding, to 8-bit samples. The resulting audio data quality is equivalent to that of standard telephone service.

The **audio** driver is implemented as a STREAMS device. In order to record audio input, applications **open**(2V) the **/dev/audio** device and read data from it using the **read**(2V) system call. Similarly, sound data is queued to the audio output port by using the **write**(2V) system call.

Opening the Audio Device

The audio device is treated as an exclusive resource: only one process may typically open the device at a time. However, two processes may simultaneously access the device if one opens it read-only and the other opens it write-only.

When a process cannot open **/dev/audio** because the requested access mode is busy:

- if the **O_NDELAY** flag is set in the **open( )** *flags* argument, then **open( )** returns −1 immediately, with *errno* set to EBUSY.
- if **O_NDELAY** is not set, then **open( )** hangs until the device is available or a signal is delivered to the process, in which case **open( )** returns −1 with *errno* set to EINTR.

Since the audio device grants exclusive read or write access to a single process at a time, long-lived audio applications may choose to close the device when they enter an idle state, reopening it when required. The *play.waiting* and *record.waiting* flags in the audio information structure (see below) provide an indication that another process has requested access to the device. This information is advisory only; background audio output processes, for example, may choose to relinquish the audio device whenever another process requests write access.

Recording Audio Data

The **read( )** system call copies data from the system buffers to the application. Ordinarily, **read( )** blocks until the user buffer is filled. The **FIONREAD ioctl** (see **filio**(4)) may be used to determine the amount of data that may be read without blocking. The device may alternatively be set to a non-blocking mode, in which case **read( )** completes immediately, but may return fewer bytes than requested. Refer to the **read**(2V) manual page for a complete description of this behavior.

When the audio device is opened with read access, the device driver immediately starts buffering audio input data. Since this consumes system resources, processes that do not record audio data should open the device write-only (**O_WRONLY**).

The transfer of input data to STREAMS buffers may be paused (or resumed) by using the **AUDIO_SETINFO ioctl** to set (or clear) the *record.pause* flag in the audio information structure (see below). All unread input data in the STREAMS queue may be discarded by using the **I_FLUSH** STREAMS **ioctl** (see **streamio**(4)).

Input data accumulates in STREAMS buffers at a rate of 8000 bytes per second. If the application that consumes the data cannot keep up with this data rate, the STREAMS queue may become full. When this occurs, the *record.error* flag is set in the audio information structure and input sampling ceases until there is room in the input queue for additional data. In such cases, the input data stream contains a discontinuity. For this reason, audio recording applications should open the audio device when they are prepared to begin reading data, rather than at the start of extensive initialization.

Playing Audio Data

The **write( )** system call copies data from an applications buffer to the STREAMS output queue. Ordinarily, **write( )** blocks until the entire user buffer is transferred. The device may alternatively be set to a non-blocking mode, in which case **write( )** completes immediately, but may have transferred fewer bytes than requested (see **write**(2V)).

Although **write( )** returns when the data is successfully queued, the actual completion of audio output may take considerably longer. The **AUDIO_DRAIN ioctl** may be issued to allow an application to block until all of the queued output data has been played. Alternatively, a process may request asynchronous notification of output completion by writing a zero-length buffer (end-of-file record) to the output stream. When such a buffer has been processed, the *play.eof* flag in the audio information structure (see below) is incremented.

The final **close( )** of the file descriptor hangs until audio output has drained. If a signal interrupts the **close( ) ,** or if the process exits without closing the device, any remaining data queued for audio output is flushed and the device is closed immediately.

The conversion of output data may be paused (or resumed) by using the **AUDIO_SETINFO ioctl** to set (or clear) the *play.pause* flag in the audio information structure. Queued output data may be discarded by using the **I_FLUSH** STREAMS **ioctl**.

Output data is played from the STREAMS buffers at a rate of 8000 bytes per second. If the output queue becomes empty, the *play.error* flag is set in the audio information structure and output ceases until additional data is written.

Asynchronous I/O

The **I_SETSIG** STREAMS **ioctl** may be used to enable asynchronous notification, via the **SIGPOLL** signal, of input and output ready conditions. This, in conjunction with non-blocking **read( )** and **write( )** requests, is normally sufficient for applications to maintain an audio stream in the background. Alternatively, asynchronous reads and writes may be initiated using the **aioread**(3) functions.

Audio Data Encoding

The data samples processed by the audio device are encoded in 8 bits. The high-order bit is a sign bit: 1 represents positive data and 0 represents negative data. The low-order 7 bits represent signal magnitude and are inverted (1's complement). The magnitude is encoded according to a *u*-law transfer function; such an encoding provides an improved signal-to-noise ratio at low amplitude levels. In order to achieve best results, the audio recording gain should be set so that typical amplitude levels lie within approximately three-fourths of the full dynamic range.

Audio Control Pseudo-Device

It is sometimes convenient to have an application, such as a volume control panel, modify certain characteristics of the audio device while it is being used by an unrelated process. The **/dev/audioctl** minor device is provided for this purpose. Any number of processes may open **/dev/audioctl** simultaneously. However, **read( )** and **write( )** system calls are ignored by **/dev/audioctl**. The **AUDIO_GETINFO** and **AUDIO_SETINFO ioctl** commands may be issued to **/dev/audioctl** in order to determine the status or alter the behavior of **/dev/audio**.

Audio Status Change Notification

Applications that open the audio control pseudo-device may request asynchronous notification of changes in the state of the audio device by setting the **S_MSG** flag in an **I_SETSIG** STREAMS **ioctl**. Such processes receive a **SIGPOLL** signal when any of the following events occurs:

- An **AUDIO_SETINFO ioctl** has altered the device state.
- An input overflow or output underflow has occurred.
- An end-of-file record (zero-length buffer) has been processed on output.
- An **open( )** or **close( )** of **/dev/audio** has altered the device state.

Audio Information Structure

The state of the audio device may be polled or modified using the **AUDIO_GETINFO** and **AUDIO_SETINFO ioctl** commands. These commands operate on the **audio_info** structure, defined in **<sun/audioio.h>** as follows:

```
/* Data encoding values, used below in the encoding field */
#define AUDIO_ENCODING_ULAW     (1)     /* u-law encoding */
#define AUDIO_ENCODING_ALAW     (2)     /* A-law encoding */

/* These ranges apply to record, play, and monitor gain values */
#define AUDIO_MIN_GAIN          (0)             /* minimum gain value */
#define AUDIO_MAX_GAIN          (255)           /* maximum gain value */

/* Audio I/O channel status, used below in the audio_info structure */
struct audio_prinfo {
        /* The following values describe the audio data encoding */
        unsigned                sample_rate;    /* samples per second */
        unsigned                channels;       /* number of interleaved channels */
        unsigned                precision;      /* number of bits per sample */
        unsigned                encoding;       /* data encoding method */

        /* The following values control audio device configuration */
        unsigned                gain;           /* gain level */
        unsigned                port;           /* selected I/O port */

        /* The following values describe the current device state */
        unsigned                samples;        /* number of samples converted */
        unsigned                eof;            /* End Of File counter (play only) */
        unsigned char           pause;          /* non-zero if paused, zero to resume */
        unsigned char           error;          /* non-zero if overflow/underflow */
        unsigned char           waiting;        /* non-zero if a process wants access */

        /* The following values are read-only device state flags */
        unsigned char           open;           /* non-zero if open access granted */
        unsigned char           active;         /* non-zero if I/O active */
};

/* This structure is used in AUDIO_GETINFO and AUDIO_SETINFO ioctl commands */
typedef struct audio_info {
        struct audio_prinfo     record;         /* input status information */
        struct audio_prinfo     play;           /* output status information */
        unsigned                monitor_gain;   /* input to output mix */
} audio_info_t;
```

The *play.gain* and *record.gain* fields specify the output and input volume levels. A value of **AUDIO_MAX_GAIN** indicates maximum gain. The device also allows input data to be monitored by mixing audio input onto the output channel. The *monitor_gain* field controls the level of this feedback path. The *play.port* field controls the output path for the audio device. It may be set to either **AUDIO_SPEAKER** or **AUDIO_HEADPHONE** to direct output to the built-in speaker or the headphone jack, respectively.

The *play.pause* and *record.pause* flags may be used to pause and resume the transfer of data between the audio device and the STREAMS buffers. The *play.error* and *record.error* flags indicate that data underflow or overflow has occurred. The *play.active* and *record.active* flags indicate that data transfer is currently active in the corresponding direction.

The *play.open* and *record.open* flags indicate that the device is currently open with the corresponding access permission. The *play.waiting* and *record.waiting* flags provide an indication that a process may be waiting to access the device. These flags are set automatically when a process blocks on **open( )**, though they may also be set using the **AUDIO_SETINFO ioctl** command. They are cleared only when a process relinquishes access by closing the device.

The *play.samples* and *record.samples* fields are initialized, at **open( )**, to zero and increment each time a data sample is copied to or from the associated STREAMS queue. Applications that keep track of the number of samples read or written may use these fields to determine exactly how many samples remain in the STREAMS buffers. The *play.eof* field increments whenever a zero-length output buffer is synchronously processed. Applications may use this field to detect the completion of particular segments of audio output.

The *sample_rate*, *channels*, *precision*, and *encoding* fields report the audio data format in use by the device. For now, these values are read-only; however, future audio device implementations may support more than one data encoding format, in which case applications might be able to modify these fields.

Filio and STREAMS IOCTLS

All of the **filio**(4) and **streamio**(4) **ioctl** commands may be issued for the **/dev/audio** device. Because the **/dev/audioctl** device has its own STREAMS queues, most of these commands neither modify nor report the state of **/dev/audio** if issued for the **/dev/audioctl** device. The **I_SETSIG ioctl** may be issued for **/dev/audioctl** to enable the notification of audio status changes, as described above.

Audio IOCTLS

The audio device additionally supports the following **ioctl** commands:

AUDIO_DRAIN

The argument is ignored. This command suspends the calling process until the output STREAMS queue is empty, or until a signal is delivered to the calling process. It may only be issued for the **/dev/audio** device. An implicit **AUDIO_DRAIN** is performed on the final **close( )** of **/dev/audio**.

AUDIO_GETINFO

The argument is a pointer to an **audio_info** structure. This command may be issued for either **/dev/audio** or **/dev/audioctl**. The current state of the **/dev/audio** device is returned in the structure.

AUDIO_SETINFO

The argument is a pointer to an **audio_info** structure. This command may be issued for either **/dev/audio** or **/dev/audioctl**. This command configures the audio device according to the structure supplied and overwrites the structure with the new state of the device. [Note: The *play.samples*, *record.samples*, *play.error*, *record.error*, and *play.eof* fields are modified to reflect the state of the device when the **AUDIO_SETINFO** was issued. This allows programs to atomically modify these fields while retrieving the previous value.]

Certain fields in the information structure, such as the *pause* flags, are treated as read-only when **/dev/audio** is not open with the corresponding access permission. Other fields, such as the gain levels and encoding information, may have a restricted set of acceptable values. Applications that attempt to modify such fields should check the returned values to be sure that the corresponding change took effect.

Once set, the following values persist through subsequent **open( )** and **close( )** calls of the device: *play.gain*, *record.gain*, *monitor_gain*, *play.port*, and *record.port*. All other state is reset when the corresponding I/O stream of **/dev/audio** is closed.

The **audio_info** structure may be initialized through the use of the **AUDIO_INITINFO** macro. This macro sets all fields in the structure to values that are ignored by the **AUDIO_SETINFO** command. For instance, the following code switches the output port from the built-in speaker to the headphone jack without modifying any other audio parameters:

```
audio_info_t    info;

AUDIO_INITINFO(&info);
info.play.port = AUDIO_HEADPHONE;
err = ioctl(audio_fd, AUDIO_SETINFO, &info);
```

This technique is preferred over using a sequence of **AUDIO_GETINFO** followed by **AUDIO_SETINFO**.

Unsupported Device Control Features

The AM79C30A chip is capable of performing a number of functions that are not currently supported by the device driver, many of which were designed primarily for telephony applications. For example, the chip can generate ringer tones and has a number of specialized filtering capabilities that are designed to compensate for different types of external speakers and microphones.

Ordinarily, applications do not need to access these capabilities and, further, altering the chip's characteristics may interfere with its normal behavior. However, knowledgeable applications may use the unsupported **AUDIOGETREG** and **AUDIOSETREG ioctl** commands to read and write the chip registers directly. The description of this interface may be found in **<sbusdev/audio_79C30.h>**. Note: these commands are supplied for prototyping purposes only and may become obsolete in a future release of the audio driver.

FILES

/dev/audio
/dev/audioctl
/usr/demo/SOUND

SEE ALSO

ioctl(2), **poll**(2), **read**(2V), **write**(2V), **aioread**(3), **filio**(4), **streamio**(4)

AMD data sheet for the AM79C30A Digital Subscriber Controller, Publication number 09893.

BUGS

Due to a *feature* of the STREAMS implementation, programs that are terminated or exit without closing the **audio** device may hang for a short period while audio output drains. In general, programs that produce audio output should catch the **SIGINT** signal and flush the output stream before exiting.

The current driver implementation does not support the A-law encoding mode of the AM79C30A chip. Future implementations may permit the **AUDIO_SETINFO ioctl** to modify the *play.encoding* and *record.encoding* fields of the device information structure to enable this mode.

FUTURE DIRECTIONS

Workstation audio resources should be managed by a networked audio server, in the same way that the video monitor is manipulated by a window system server. For the time being, we encourage you to write your programs in a modular fashion, isolating the **audio** device-specific functions, so that they may be easily ported to such an environment.

NAME

bwtwo – black and white memory frame buffer

CONFIG — SUN-3, SUN-3x SYSTEMS

device bwtwo0 at obmem 1 csr 0xff000000 priority 4
device bwtwo0 at obmem 2 csr 0x100000 priority 4
device bwtwo0 at obmem 3 csr 0xff000000 priority 4
device bwtwo0 at obmem 4 csr 0xff000000
device bwtwo0 at obmem 7 csr 0xff000000 priority 4
device bwtwo0 at obmem ? csr 0x50300000 priority 4

The first synopsis line given above is used to generate a kernel for Sun-3/75, Sun-3/140 or Sun-3/160 systems; the second, for a Sun-3/50 system; the third, for a Sun-3/260 system; the fourth, for a Sun-3/110 system; the fifth, for a Sun-3/60 system; and the sixth for Sun-3/80 and Sun-3/470 systems.

CONFIG — SUN-4 SYSTEMS

device bwtwo0 at obio 1 csr 0xfd000000 priority 4
device bwtwo0 at obio 2 csr 0xfb300000 priority 4
device bwtwo0 at obio 3 csr 0xfb300000 priority 4
device bwtwo0 at obio 4 csr 0xfb300000 priority 4

The first synopsis line given above should be used to generate a kernel for a Sun-4/260 or Sun-4/280 system; the second, for a Sun-4/110 system; the third for a Sun-4/330 system; and the fourth for a Sun-4/460 system.

CONFIG — SUN-4c, SUN-4m SYSTEMS

device-driver bwtwo

CONFIG — Sun386i SYSTEM

device bwtwo0 at obmem ? csr 0xA0200000

DESCRIPTION

The **bwtwo** interface provides access to Sun monochrome memory frame buffers. It supports the ioctls described in **fbio**(4S).

If **flags 0x1** is specified, frame buffer write operations are buffered through regular high-speed RAM. This "copy memory" mode of operation speeds frame buffer accesses, but consumes an extra 128K bytes of memory. Only Sun-3/75, Sun-3/140, and Sun-3/160 systems support copy memory; on other systems a warning message is printed and the flag is ignored.

Reading or writing to the frame buffer is not allowed — you must use the **mmap**(2) system call to map the board into your address space.

FILES

/dev/bwtwo[0-9] device files

SEE ALSO

mmap(2), **cgfour**(4S), **fb**(4S), **fbio**(4S)

BUGS

Use of vertical-retrace interrupts is not supported.

NAME

cdromio – CDROM control operations

DESCRIPTION

The Sun CDROM device driver supports a set of **ioctl**(2) commands for audio operations and CDROM specific operations. It also supports the **dkio**(4S) operations — generic disk control operation for all Sun disk drivers. See **dkio(4S)** Basic to these **cdromio ioctl( )** requests are the definitions in **<scsi/targets/srdef.h>** or **<sundev/srreg.h>**

```
/*
 * CDROM I/O controls type definitions
 */

/* definition of play audio msf structure */
struct cdrom_msf {
        unsigned char   cdmsf_min0;     /* starting minute */
        unsigned char   cdmsf_sec0;     /* starting second */
        unsigned char   cdmsf_frame0;   /* starting frame  */
        unsigned char   cdmsf_min1;     /* ending minute   */
        unsigned char   cdmsf_sec1;     /* ending second   */
        unsigned char   cdmsf_frame1;   /* ending frame    */
};

/* definition of play audio track/index structure */
struct cdrom_ti {
        unsigned char   cdti_trk0;      /* starting track */
        unsigned char   cdti_ind0;      /* starting index */
        unsigned char   cdti_trk1;      /* ending track */
        unsigned char   cdti_ind1;      /* ending index */
};

/* definition of read toc header structure */
struct cdrom_tochdr {
        unsigned char   cdth_trk0;      /* starting track */
        unsigned char   cdth_trk1;      /* ending track */
};

/* definition of read toc entry structure */
struct cdrom_tocentry {
        unsigned char   cdte_track;
        unsigned char   cdte_adr        :4;
        unsigned char   cdte_ctrl       :4;
        unsigned char   cdte_format;
        union {
                struct {
                        unsigned char   minute;
                        unsigned char   second;
                        unsigned char   frame;
                } msf;
                int     lba;
        } cdte_addr;
        unsigned char   cdte_datamode;
};
```

NAME

cgnine – 24-bit VME color memory frame buffer

CONFIGURATION

device cgnine0 at vme32d32 ? csr 0x08000000 priority 4 vector cgnineintr 0xaa

DESCRIPTION

cgnine is a 24-bit double-buffered VME-based color frame buffer. It provides the standard frame buffer interface defined in **fbio**(4S), and and can be paired with the **GP2** graphics accelerator board using **gpconfig**(8).

cgnine has two bits of overlay planes, each of which is a 1-bit deep frame buffer that overlays the 24-bit plane group. When either bit of the two overlay planes is non-zero, the pixel shows the color of the overlay plane. If both bits are zero, the color frame buffer underneath is visible.

The 24-bit frame buffer pixel is organized as one longword (32 bits) per pixel. The pixel format is defined in **<pixrect/pixrect.h>** as follows:

```
union fbunit {
unsigned int    packed; /* whole-sale deal */
struct {
        unsigned int    A:8;    /* unused, for now */
        unsigned int    B:8;    /* blue channel */
        unsigned int    G:8;    /* green channel */
        unsigned int    R:8;    /* red channel */
        }
                channel;        /* access per channel */
};
```

When the board is in double-buffer mode, the low 4 bits of each channel are ignored when written to, which yields 12-bit double-buffering.

The higher bit of the overlay planes ranges from offset 0 to 128K (0x20000) bytes. The lower bit ranges from 128K to 256K bytes. The 4MB (0x400000) of the 24-bit deep pixels begins at 256K. The addresses of the control registers start at the next page after the 24-bit deep pixels.

FILES

/dev/cgnine0	device special file
/dev/gpone0a	**cgnine** bound with **GP2**
/dev/fb	default frame buffer

SEE ALSO

mmap(2), **fbio**(4S), **gpone**(4S) **gpconfig**(8)

NAME

cgsix – accelerated 8-bit color frame buffer

CONFIG — SUN-3, SUN-3x, SUN-4 SYSTEMS

device cgsix0 at obmem ? csr 0xff000000 priority 4
device cgsix0 at obmem ? csr 0x50000000 priority 4
device cgsix0 at obio ? csr 0xfb000000 priority 4

The first synopsis line given should be used for Sun-3/60 systems, the second for Sun-3x systems, and the third for Sun-4 systems.

CONFIG — SUN-4c, SUN-4m SYSTEMS

device-driver cgsix

DESCRIPTION

The **cgsix** is a low-end graphics accelerator designed to enhance vector and polygon drawing performance. It has an 8-bit color frame buffer and provides the standard frame buffer interface as defined in **fbio**(4S).

The cgsix has registers and memory that may be mapped with **mmap**(2), using the offsets defined in **<sundev/cg6reg.h>**.

FILES

/dev/cgsix0

SEE ALSO

mmap(2), **fbio**(4S)

NAME

cgthree – 8-bit color memory frame buffer

CONFIG — Desktop SPARCsystems

device-driver cgthree

CONFIG — Sun386i SYSTEM

device cgthree0 at obmem ? csr 0xA0400000

AVAILABILITY

Desktop SPARCsystems and Sun386i systems only.

DESCRIPTION

cgthree is a color memory frame buffer. It provides the standard frame buffer interface as defined in **fbio**(4S).

FILES

/dev/cgthree[0-9]

SEE ALSO

mmap(2), **fbio**(4S)

NAME

cgtwelve – 24-bit SBus color memory frame buffer and graphics accelerator

CONFIGURATION

device-driver cgtwelve

DESCRIPTION

cgtwelve is a 24-bit SBus-based color frame buffer and graphics accelerator, with 12-bit double buffering, 8-bit colormap, and overlay/enable planes. It provides the standard frame buffer interface defined in **fbio**(4S), paired with microcode which can be downloaded using **gpconfig**(8). Application acceleration is achieved using the Pixwin and SunPHIGS Application Programmer Interfaces (APIs).

The cgtwelve has registers and memory that may be mapped with **mmap**(2), using the offsets defined in **<sbusdev/cg12reg.h>**.

When in double-buffer mode, each channel is dithered to 4 bits, yielding 12-bit double-buffering.

FILES

/dev/cgtwelve0	device special file
/dev/fb	default frame buffer
/usr/include/sbusdev/cg12reg.h	device-specific definitions

SEE ALSO

mmap(2), **fbio**(4S), **gpconfig**(8)

NAME

cgtwo – color graphics interface

CONFIG — SUN-3, SUN-3x, SUN-4 SYSTEMS

cgtwo0 at vme24d16 ? csr 0x400000 priority 4 vector cgtwointr 0xa8

DESCRIPTION

The **cgtwo** interface provides access to the color graphics controller board, which is normally supplied with a 19'' 66 Hz non-interlaced color monitor. It provides the standard frame buffer interface as defined in **fbio**(4S).

The hardware consumes 4 megabytes of VME bus address space. The board starts at standard address 0x400000. The board must be configured for interrupt level 4.

FILES

/dev/cgtwo[0-9]

SEE ALSO

mmap(2), **fbio**(4S)

NAME

fb – driver for Sun console frame buffer

CONFIG

None; included in standard system.

DESCRIPTION

The **fb** driver provides indirect access to a Sun frame buffer. It is an indirect driver for the Sun workstation console's frame buffer. At boot time, the workstation's frame buffer device is determined from information from the PROM monitor and set to be the one that **fb** will indirect to. The device driver for the console's frame buffer must be configured into the kernel so that this indirect driver can access it.

The idea behind this driver is that user programs can open a known device, query its characteristics and access it in a device dependent way, depending on the type. **fb** redirects **open**(2V), **close**(2V), **ioctl**(2), and **mmap**(2) calls to the real frame buffer. All Sun frame buffers support the same general interface; see **fbio**(4S).

FILES

/dev/fb

SEE ALSO

close(2V), **ioctl**(2), **mmap**(2), **open**(2V), **fbio**(4S)

NAME

fbio – frame buffer control operations

DESCRIPTION

All Sun frame buffers support the same general interface that is defined by **<sun/fbio.h>**. Each responds to an **FBIOGTYPE ioctl(2)** request which returns information in a **fbtype** structure.

Each device has an **FBTYPE** which is used by higher-level software to determine how to perform graphics functions. Each device is used by opening it, doing an **FBIOGTYPE ioctl( )** to see which frame buffer type is present, and thereby selecting the appropriate device-management routines.

Full-fledged frame buffers (that is, those that run SunView1) implement an **FBIOGPIXRECT ioctl( )** request, which returns a pixrect. This call is made only from inside the kernel. The returned pixrect is used by **win**(4S) for cursor tracking and colormap loading.

FBIOSVIDEO and **FBIOGVIDEO** are general-purpose **ioctl( )** reuqests for controlling possible video features of frame buffers. These **ioctl( )** requests either set or return the value of a flags integer. At this point, only the **FBVIDEO_ON** option is available, controlled by **FBIOSVIDEO**. **FBIOGVIDEO** returns the current video state.

The **FBIOSATTR** and **FBIOGATTR ioctl( )** requests allow access to special features of newer frame buffers. They use the **fbsattr** and **fbgattr** structures.

Some color frame buffers support the **FBIOPUTCMAP** and **FBIOGETCMAP ioctl( )** reuqests, which provide access to the colormap. They use the **fbcmap** structure.

SEE ALSO

ioctl(2), **mmap**(2), **bw***(4S), **cg***(4S), **gp***(4S), **fb**(4S), **win**(4S)

BUGS

The **FBIOSATTR** and **FBIOGATTR ioctl( )** requests are only supported by frame buffers which emulate older frame buffer types. For example, **cgfour**(4S) frame buffers emulate **bwtwo**(4S) frame buffers. If a frame buffer is emulating another frame buffer, **FBIOGTYPE** returns the emulated type. To get the real type, use **FBIOGATTR**.

NAME

fd – disk driver for Floppy Disk Controllers

CONFIG — Sun386i SYSTEMS

controller fdc0 at atmem ? csr 0x1000 dmachan 2 irq 6 priority 2
disk fd0 at fdc0 drive 0 flags 0

CONFIG — SUN-3/80 SYSTEMS

controller fdc0 at obio ? csr 0x6e000000 priority 6 vector fdintr 0x5c
disk fd0 at fdc0 drive 0 flags 0

CONFIG — Desktop SPARCsystems

device-driver fd

AVAILABILITY

Sun386i, Sun-3/80, and Desktop SPARCsystems only.

DESCRIPTION

The **fd** driver provides an interface to floppy disks using the Intel 82072 disk controller on Sun386i, Sun-3/80 and Desktop SPARCsystems.

The minor device number in files that use the floppy interface encodes the unit number as well as the partition. The bits of the minor device number are defined as **rrruuppp** where **r**=reserved, **u**=unit, and **p**=partition. The unit number selects a particular floppy drive for the controller. The partition number picks one of eight partitions [**a-h**].

When the floppy is first opened the driver looks for a label in logical block 0 of the diskette. If a label is found, the geometry and partition information from the label will be used on each access thereafter. The driver first assumes high density characteristics when it tries to read the label. If the read fails it will try the read again using low density characteristics. If both attempts to read the label fail, the open will fail. Use the **FNDELAY** flag when opening an unformatted diskette as a signal to the driver that it should not attempt to access the diskette. If block 0 is read successfully, but a label is not found, the open will fail for the block interface. Using the raw interface, the open will succeed even if the diskette is unlabeled. Default geometry and partitioning are assumed if the diskette is unlabeled.

The default partitions are:

a –> 0, N-1

b –> N-1, N

c –> 0, N

where N is the number of cylinders on the diskette.

The **fd** driver supports both block and raw interfaces. The block files access the disk using the system's normal buffering mechanism and may be read and written without regard to physical disk records. There is also a "raw" interface that provides for direct transmission between the disk and the user's read or write buffer. A single **read**(2V) or **write**(2V) call usually results in one I/O operation; therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files conventionally begin with an extra '**r**'.

FILES — Sun386i SYSTEMS

1.44 MB Floppy Disk Drives:

/dev/fd0a	block file
/dev/fd0c	block file
/dev/rfd0a	raw file
/dev/rfd0c	raw file

720 K Floppy Disk Drives:

/dev/fdl0a	block file
/dev/fdl0c	block file
/dev/rfdl0a	raw file
/dev/rfdl0c	raw file

FILES — SUN-3/80 and Desktop SPARCsystems

Note: the **fd** driver on Sun-3/80 and Desktop SPARCsystems auto-senses the density of the floppy.

/dev/fd0[a-c]	block file
/dev/fd0	block file (same as **/dev/fd0c**)
/dev/rfd0[a-c]	raw file
/dev/rfd0	raw file (same as **/dev/rfd0c**)

SEE ALSO

read(2V), **write**(2V), **dkio**(4S)

DIAGNOSTICS — Sun386i SYSTEMS

fd drv %d, trk %d: %s

A command such as read or write encountered a format-related error condition. The value of *%s* is derived from the error number given by the controller, indicating the nature of the error. The track number is relative to the beginning of the partition involved.

fd drv %d, blk %d: %s

A command such as read or write encountered an error condition related to I/O. The value of *%s* is derived from the error number returned by the controller and indicates the nature of the error. The block number is relative to the start of the partition involved.

fd controller: %s

An error occurred in the controller. The value of *%s* is derived from the status returned by the controller and specifies the error encountered.

fd(%d):%s please insert

I/O was attempted while the floppy drive door was not latched. The value of *%s* indicates which disk was expected to be in the drive.

DIAGNOSTICS — SUN-3/80 and Desktop SPARCsystems

fd%d: %s failed (%x %x %x)

The command, %s, failed after several retries on drive %d. The three hex values in parenthesis are the contents of status register 0, status register 1, and status register 2 of the Intel 82072 Floppy Disk Controller on completion of the command as documented in the data sheet for that part. This error message is usually followed by one of the following, interpreting the bits of the status register:

fd%d: not writable
fd%d: crc error
fd%d: overrun/underrun
fd%d: bad format
fd%d: timeout

NOTES

Floppy diskettes have 18 sectors per track, and can cross a track (though not a cylinder) boundary without lossing data, so when using **dd**(1) to or from a diskette, you should specify **bs=18k** or multiples thereof.

Returns the true minor device. **argp** points to a **char** which is returned from the driver.

The graphics processor driver also responds to the **FBIOGTYPE**, ioctl which a program can use to inquire as to the characteristics of the display device, the **FBIOGINFO**, ioctl for passing generic information, and the **FBIOGPIXRECT** ioctl so that SunWindows can run on it. See **fbio**(4S).

FILES

/dev/fb
/dev/gpone[0-3][abcd]

SEE ALSO

fbio(4S), **mmap**(2), **gpconfig**(8)

SunCGI Reference Manual

DIAGNOSTICS

The Graphics Processor has been restarted. You may see display garbage as a result.

NAME

gt – double buffered 24-bit SBus color memory frame buffer and graphics accelerator

CONFIGURATION

device-driver gt

DESCRIPTION

gt is a 24-bit SBus-based color frame buffer and graphics accelerator. The frame buffer consists of 108 video memory planes of 1280×1024 pixels including 24-bit double buffering, 16 alpha/overlay planes, 24 z-buffer planes, 10 window id planes, 8 fast clear planes, and 2 cursor planes. It provides the standard frame buffer interface defined in **fbio**(4S), paired with microcode which can be downloaded using **gtconfig (8).** Application acceleration is achieved via the Pixrect/Pixwin and SunPHIGS Application Programmer Interfaces (APIs).

FILES

/dev/gt0	device special file
/dev/fb	default frame buffer

SEE ALSO

gtconfig(8)

NAME

icmp – Internet Control Message Protocol

SYNOPSIS

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip_icmp.h>

s = socket(AF_INET, SOCK_RAW, proto);
```

DESCRIPTION

ICMP is the error and control message protocol used by the Internet protocol family. It is used by the kernel to handle and report errors in protocol processing. It may also be accessed through a "raw socket" for network monitoring and diagnostic functions. The protocol number for ICMP, used in the *proto* parameter to the socket call, can be obtained from **getprotobyname** (see **getprotoent**(3N)). ICMP sockets are connectionless, and are normally used with the *sendto* and *recvfrom* calls, though the **connect**(2) call may also be used to fix the destination for future packets (in which case the **read**(2V) or **recv**(2) and **write**(2V) or **send**(2) system calls may be used).

Outgoing packets automatically have an Internet Protocol (IP) header prepended to them. Incoming packets are provided to the holder of a raw socket with the IP header and options intact.

ICMP is an unreliable datagram protocol layered above IP. It is used internally by the protcol code for various purposes including routing, fault isolation, and congestion control. Receipt of an ICMP "redirect" message will add a new entry in the routing table, or modify an existing one. ICMP messages are routinely sent by the protocol code. Received ICMP messages may be reflected back to users of higher-level protocols such as TCP or UDP as error returns from system calls. A copy of all ICMP message received by the system is provided using the ICMP raw socket.

ERRORS

A socket operation may fail with one of the following errors returned:

EISCONN	when trying to establish a connection on a socket which already has one, or when trying to send a datagram with the destination address specified and the socket is already connected;
ENOTCONN	when trying to send a datagram, but no destination address is specified, and the socket hasn't been connected;
ENOBUFS	when the system runs out of memory for an internal data structure;
EADDRNOTAVAIL	when an attempt is made to create a socket with a network address for which no network interface exists.

SEE ALSO

connect(2), **read**(2V), **recv**(2), **send**(2), **write**(2V), **getprotoent**(3N), **inet**(4F), **ip**(4P), **routing**(4N)

Postel, Jon, *Internet Control Message Protocol — DARPA Internet Program Protocol Specification*, RFC 792, Network Information Center, SRI International, Menlo Park, Calif., September 1981. (Sun 800-1064-01)

BUGS

Replies to ICMP "echo" messages which are source routed are not sent back using inverted source routes, but rather go back through the normal routing mechanisms.

NAME

id – disk driver for IPI disk controllers

CONFIG — SUN-4, SPARCsystem 600MP SERIES

```
controller      idc0     at ipi ? csr 0x0000ff priority 2  # channel 0 slave 0
disk            id 0x000 at idc0 drive 0   # facility 0
disk            id 0x001 at idc0 drive 1
disk            id 0x002 at idc0 drive 2
disk            id 0x003 at idc0 drive 3
disk            id 0x004 at idc0 drive 4
disk            id 0x005 at idc0 drive 5
disk            id 0x006 at idc0 drive 6
disk            id 0x007 at idc0 drive 7

controller      idc1     at ipi ? csr 0x0001ff priority 2 # channel 0 slave 1
disk            id 0x010 at idc1 drive 0   # facility 0
disk            id 0x011 at idc1 drive 1
disk            id 0x012 at idc1 drive 2
disk            id 0x013 at idc1 drive 3
disk            id 0x014 at idc1 drive 4
disk            id 0x015 at idc1 drive 5
disk            id 0x016 at idc1 drive 6
disk            id 0x017 at idc1 drive 7

controller      idc2     at ipi ? csr 0x0002ff priority 2 # channel 0 slave 2
disk            id 0x020 at idc2 drive 0   # facility 0
disk            id 0x021 at idc2 drive 1
disk            id 0x022 at idc2 drive 2
disk            id 0x023 at idc2 drive 3
disk            id 0x024 at idc2 drive 4
disk            id 0x025 at idc2 drive 5
disk            id 0x026 at idc2 drive 6
disk            id 0x027 at idc2 drive 7

controller      idc3     at ipi ? csr 0x0003ff priority 2 # channel 0 slave 3
disk            id 0x030 at idc3 drive 0   # facility 0
disk            id 0x031 at idc3 drive 1
disk            id 0x032 at idc3 drive 2
disk            id 0x033 at idc3 drive 3
disk            id 0x034 at idc3 drive 4
disk            id 0x035 at idc3 drive 5
disk            id 0x036 at idc3 drive 6
disk            id 0x037 at idc3 drive 7

controller      idc4     at ipi ? csr 0x0004ff priority 2  # channel 0 slave 4
disk            id 0x040 at idc4 drive 0   # facility 0
disk            id 0x041 at idc4 drive 1
disk            id 0x042 at idc4 drive 2
disk            id 0x043 at idc4 drive 3
disk            id 0x044 at idc4 drive 4
disk            id 0x045 at idc4 drive 5
disk            id 0x046 at idc4 drive 6
disk            id 0x047 at idc4 drive 7
```

The first four **controller** lines given in the synopsis section above correspond to the first, second, third, and fourth IPI disk controllers in a Sun system; the fifth controller is currently available on SPARCsystem 600MP series machines only. These controllers are on IPI channels or combined on integrated

channel/disk-controller cards such as the ISP-80 IPI disk controller. The **csr** value gives the channel, slave, and facility address of the controller. The facility address should always be **0xff**. See **is**(4S).

DESCRIPTION

In order to accommodate a large number of disk drives, a multiple major number device addressing scheme is used. The minor number is formed using the low order 5 bits of the unit number and the 3-bit partition number. The low-order 3 bits of the minor number give the partition number on the drive. The high order bits of the unit number are added to the first major number for **id** devices to give the major number of the particular device. The unit number itself is formed as follows: the 4-bit facility number forms the low order 4-bits, next the 3-bit slave number, and then the channel number.

The standard device names begin with **id** followed by three hex digits giving the channel number, slave number, and facility number, and then a letter **a-h**, for partitions 0-7 respectively.

The block files access the disk using the system's normal buffering mechanism and may be read and written without regard to physical disk records. There is also a *raw* interface which provides for direct transmission between the disk and the user's read or write buffer. A single read or write call usually results in only one I/O operation; therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files conventionally begin with an extra '**r**'.

In raw I/O, counts should be a multiple of 512 bytes (a disk sector). Likewise **directory**(3) calls should specify a multiple of 512 bytes. Depending on the channel adaptor, the buffer for raw reads or writes may be required to be on a 2-byte or 4-byte boundary.

The **ioctl( )** interface described in **dkio**(4S) is supported by this driver. The DKIOCSCMD ioctl can be used to issue certain IPI commands to the drive. The argument structure is:

```
struct dk_cmd {
        u_short dkc_cmd;        /* command to be executed */
        int     dkc_flags;      /* execution flags */
        daddr_t dkc_blkno;      /* disk address for command */
        int     dkc_secnt;      /* sector count for command */
        caddr_t dkc_bufaddr;    /* user's buffer address */
        u_int   dkc_buflen;     /* size of user's buffer */
};
```

The lower 8-bits of the **dkc_cmd** field indicate one of the supported commands listed below. The upper 8-bits indicate the IPI Opcode modifier. These commands are defined in **/usr/include/sundev/ipi3.h**. Block numbers are not remapped by the partition map when these commands are used. The supported commands are:

IP_READ
IP_WRITE

Read or write data. The addressing is always by logical block (ignoring [**a-h**] logical partition information); the Opcode modifier is ignored.

IP_READ_DEFLIST
IP_WRITE_DEFLIST

Read or write one of the defect lists. The defect list is selected by the Opcode modifier in bits <15:8> of the **dkc_cmd**.

IP_FORMAT

Format a range of cylinders. For this command, the block number and sector count fields must both be a multiple of the number of blocks per cylinder. The **dk_buflen** field must be zero for this command.

IP_REALLOC

Reallocate a block. The controller attempts to recover the data from the old block being reallocated. If the old data cannot be recovered, a conditional success status is presented and a message may be printed. The **dk_buflen** field must be zero for this command.

DISK SUPPORT

This driver handles all supported IPI drives by reading controller attributes and a label from sector 0 of the drive which describes the disk geometry and partitioning.

The **id000a** partition is normally used for the root file system on a disk, the **id0??b** partition as a paging area, and the **id???c** partition for pack-pack copying (it normally maps the entire disk). The **id???g** partition is generally a partition suitable for holding the **/usr** file system. The rest of the disk is normally the **id???h** partition.

FILES

/dev/id[0-7][0-7][0-7][a-h]	block files
/dev/rid[0-7][0-7][0-7][a-h]	raw files

SEE ALSO

lseek(2), **read**(2V), **write**(2V), **directory**(3), **dkio**(4S) **is**(4S)

DIAGNOSTICS

The following messages are usually preceded by either **idc***n*, indicating a message associated with controller *n*, or **id***csf*, indicating a message associated with the drive *csf*, where *csf* is the drive's channel, slave, and facility address.

idc*n*: **configured with bad IPI address** *addr*
The IPI address of the controller should end with '**ff**'.

idc*n*: **unknown ctlr vendor id: manuf** '*name*' **model** '*model*'
The controller type is unsupported. The driver has not been tested with the controller being used.

idc*n*: **unknown string ctlr type**
The controller type is unsupported. The driver has not been tested with the controller being used.

id*csf*: **out of memory for label info**
The disk could not be initialized because the driver could not allocate kernel memory for the label.

id*csf*: **warning: starting block address nonzero:** *addr*
The disk attributes indicate a non-zero first block number. None of the supported controllers do this.

id*csf*: **zero blocks per track**
The disk attributes indicate zero blocks per track. None of the supported controllers do this.

id*csf*: **inconsistent geometry: blk/cyl** *num* **blk/trk** *num* **head** *num*
The disk attributes indicate different number of blocks per cylinder than would be expected from the blocks per track and the number of tracks (heads).

id*csf*: **inconsistent geometry: tot blks** *num* **blks/cyl** *num* **pcyl** *num*
The disk attributes indicate different number of total blocks than would be expected from the blocks per cylinder and the number of cylinders.

id*csf*: **invalid logical block size** *num*
The disk logical block size, given by the logical block size attribute parameter, is not a power of two less than 2**31.

idintr: bad rupt NULL q_dev
The interrupt routine was called with a request that cannot be identified.

idc*n*: **ctlr message:** '*message-text*'
The firmware in the disk controller issued the message *message-text*. The controller manual should describe the message.

idc*n*: **ctlr failure fru** *num* **message:** '*message-text*'
The firmware in the disk controller issued the failure message *message-text*, and indicates a likely field-replaceable unit number (printed in hex). The controller manual should describe the message and **FRU** number.

id*csf*: **corrupt label**
The label checksum mismatched.

id*csf*: **<***ASCII-label***>**
The label contains an ASCII string which is printed after the label is read. The ASCII string should give the disk type and number of cylinders, heads, and sectors.

id*csf*: **warning: label doesn't match IPI attribute info**
id*csf*: **label ncyl** *num* **pcyl** *num* **acyl** *num* **head** *num* **sect** *num* **rpm** *num*
id*csf*: **geom ncyl** *num* **pcyl** *num* **acyl** *num* **head** *num* **sect** *num* **rpm** *num*
Some of the geometry information from the disk label does not match the geometry information from the controller attributes. This may or may not be a problem, but indicates a non-standard label.

id*csf*: **id_rdwr: block** *num* **past end of disk**
An **ioctl( )** read or write was issued to a block in a cylinder with a number greater than or equal to the number of physical cylinders indicated by the disk geometry from the label or attributes.

id*csf*: **id_format failed. result** *num*
A format operation failed. The code number printed, *num*, is defined in **/usr/include/sundev/ipi_driver.h**.

id*csf*: **id_rdwr_deflist failed. result** *num*
A read or write defect list operation failed. The code number printed, *num*, is defined in **/usr/include/sundev/ipi_driver.h**.

id*csf*: **id_realloc failed. result** *num*
A reallocation operation failed. The code number printed, *num*, is defined in **/usr/include/sundev/ipi_driver.h**.

id*csf***[a-h]:** *opcode* **block** *rel-block* **(***block* **abs) idintr: bad rupt NULL qdev**
An interrupt occurred for which the associated disk could not be located.

id*csf***[a-h]:** *opcode* **block** *rel-block* **(***block* **abs) idintr: neg cmd_q_len**
The driver's count of commands outstanding to the drive became less than zero. System operation should continue normally, but this message indicates a minor driver problem.

id*csf***[a-h]:** *opcode* **block** *rel-block* **(***block* **abs) poll timeout**
A command hung or took longer than the driver expected.

id*csf***[a-h]:** *opcode* **block** *rel-block* **(***block* **abs) id_cmd_intr: ctlr cmd_q_len negative**
The driver's count of commands outstanding to the controller became less than zero. System operation should continue normally, but this message indicates a minor driver problem.

id*csf***[a-h]:** *opcode* **block** *rel-block* **(***block* **abs) id_error: slave IML not supported**
An error response from the controller indicates that it requires an initial microcode load. If this message occurs, controllers which issue this error are not supported.

id*csf***[a-h]:** *opcode* **block** *rel-block* **(***block* **abs) id_error: slave reset not supported**
An error response from the controller indicates that it requires a reset. If this message occurs, controllers which issue this error are not supported.

id*csf***[a-h]:** *opcode* **block** *rel-block* **(***block* **abs) id_error: full queue: not handled**
An error response from the controller indicates that its internal command queue is full, and the command cannot be accepted. This message should not occur because the driver respects the queue limits determined from the controller attributes.

id*csf***[a-h]:** *opcode* **block** *rel-block* **(***block* **abs) id_error: log read not supported**
The controller presented status indicating that its internal log has overflowed and should be read by the host. If this message occurs, controllers which issue this status are not supported.

id*csf***[a-h]:** *opcode* **block** *rel-block* **(***block* **abs) id_error: aborted command not supported**
An interrupt indicated that the associated command had been terminated by abort. If this message

occurs, the driver does not issue aborts or expect this status.

id*csf*[a-h]: *opcode* **block** *rel-block* (*block* **abs) id_error_parse: no response where expected**
An interrupt occurred which should have been accompanied by a response packet, but none was found.

id*csf*[a-h]: *opcode* **block** *rel-block* (*block* **abs) missing interrupt – attempting recovery**
The command took much longer than expected, indicating a possible problem with the controller or drive. Recovery of the controller will be started, and the command will be retried if possible.

id*csf*[a-h]: *opcode* **block** *rel-block* (*block* **abs) missing interrupt – recovery in progress**
The command took much longer than expected, indicating a possible problem with the controller or drive. Since recovery was already in progress, the command was added to the list or commands to be retried after recovery.

id*csf*[a-h]: *opcode* **block** *rel-block* (*block* **abs) no memory for label**
The driver could not obtain a temporary memory area for the disk label.

id*csf*[a-h]: *opcode* **block** *rel-block* (*block* **abs) cannot abort yet**
The recovery code cannot abort specific commands. It will assume the command was cleared by a reset and simply reissue the command.

id*csf*: id_recover *state-number state-name*
id*cnum*: id_recover *state-number state-name*
id*csf*: id_recover_intr *state-number state-name result*
id*cnum*: id_recover_intr *state-number state-name result*
These messages trace the recovery action during missing interrupt handling and controller recovery. These are for diagnosing problems in the event that recovery is unsuccessful.

id*cnum*: synchronous command setup failed
During recovery or initialization, the setup for a command failed, probably due to an internal error or lack of free memory.

id*csf*[a-h]: write check disabled
Write checking for the partition has been disabled via the *DKIOCWCHK* ioctl() function. See **dkio**(4S) and **dkctl**(8). This message is issued whether or not write checking was previously disabled for that partition.

id*csf*[a-h]: write check enabled
Write checking for the partition has been enabled via the *DKIOCWCHK* ioctl() function, possibly by **dkctl**(8). See **dkio**(4S) and **dkctl**(8), This message is issued whether or not write checking was previously enabled for that partition. Write checking, when enabled, causes the driver to perform a read after every write to the partition to verify that the write succeeded. If the read fails, the write will be retried.

IPI DIAGNOSTICS

The following diagnostics come directly from controller status. The messages may occur in combinations determined by the controller. In most cases, more information about the conditions under which the status is issued will be found in the controller manual.

In the examples shown, the message for the response parameter is shown first, then the message for a single bit in the response parameter. In practice, there will be a response parameter message followed by one or more messages for the individual bits, all on the same line.

The messages will be preceded by the disk or controller number. The message descriptions refer to the appropriate disk or controller as the addressee, since it was addressed in the command or response.

Normally, these messages are not printed when the condition described arises. They will only be printed if the condition is "interesting." Certain conditions print all of the messages associated with a response, even though the individual messages might not be especially interesting.

The description of the message meaning is excerpted from the IPI-3 standards document. Refer to it for a more complete explanation.

Message/Microcode Exception. Message.
The slave has included a message within Extended Substatus for the master.

Message/Microcode Exception. Failure Message.
The slave encountered a failure condition which resulted in the identification of a FRU (Field Replaceable Unit).

Message/Microcode Exception. Port Disable Pending.
The addressee has received a manual or programmed Port Disable command that will take effect when the Disable conditions are met.

Message/Microcode Exception. Port Response.
A port has executed a Port Response command.

Intervention Required. Not P-Available.
The selected addressee is not powered on or is not installed.

Intervention Required. Not Ready.
The selected addressee cannot execute its intended functions. The addressee's Not Ready condition may be cleared by operator intervention.

Intervention Required. Not P-Available Transition.
This is presented by the controller to advise the driver that a facility has become Not P-Available since the time that a command addressed to it was accepted. Note: if the transition had occurred before the command packet was accepted, the status would have been Not P-Available.

Intervention Required. Not Ready Transition.
This is presented by the controller to advise the driver that a facility has dropped ready since the time that a command addressed to it was accepted. Note: if the transition had occurred before the command packet was accepted, the status would have been Not Ready.

Intervention Required. Attribute Table may be corrupted.
The controller has encountered a condition under which it is possible that the Attributes table has been corrupted, and it is not prepared to continue operation without master intervention.

Intervention Required. Busy.
The command cannot be executed because the addressee has been Busy for a time determined by the Facility Timeout Value specified in Attributes.

Alternate Port Exception. Priority Reserve Issued.
The addressee has been instructed to release allegiance to this port because of a Priority Reserve from an alternate port.

Alternate Port Exception. Attributes Updated.
An Attributes command has been issued from an alternate port which has changed the addressee's attributes.

Alternate Port Exception. Initialization Completed.
The addressee has completed an initialization procedure which may have affected this port, and was originated by a Reset from an alternate port.

Alternate Port Exception. Format Completed.
The addressee has completed a **FORMAT** command from an alternate port.

Alternate Port Exception. Facility switched to other port.
The slave has determined that the facility is switched to another port.

Machine Exception. No longer busy.
The addressee is notifying the master that it is no longer busy.

Machine Exception. P-Available.

This is presented asynchronously by the slave to advise the master that a facility which was previously Not P-Available has become P-Available.

Machine Exception. Ready.
This is presented asynchronously by the slave to advise the master that a facility which was not previously ready has become Ready.

Machine Exception. Operation Timeout.
There has been a failure condition in the addressee which has been detected by an internal timeout mechanism.

Machine Exception. Physical Interface Check.
The slave detected a check condition on the Physical Interface — for example, an invalid sequence generated by the "state machine" or parity error on the bus(es).

Machine Exception. Slave Initiated Reset.
An internal condition caused the slave to initiate a reset; the master shall assume all outstanding commands and buffer contents are either lost or suspect.

Machine Exception. Environmental Error.
Some condition internal or external to the addressee has been detected which may cause failure condition(s) — for example, a temperature sensor alert.

Machine Exception. Power Fail Alert.
The addressee has detected an impending power failure condition.

Machine Exception. Data Check (on Raw Data).
The master has requested raw data and the addressee has detected a data error.

Machine Exception. Uncorrectable Data Check.
The slave detected a data error which has persisted after the slave has exhausted any possible recovery actions. On write operations, the malfunction may have caused invalid data to be recorded.

Machine Exception. Fatal Error.
The addressee detected an internal machine error that precludes execution or continuation of the current command.

Machine Exception. Hardware Write Protected.
An attempt was made to write on a drive that was protected against writing by something physical — for example, a switch on the drive.

Machine Exception. Queue Full.
The command queue for the addressee is full.

Machine Exception. Command Failure.
The command in execution encountered a condition which caused it to complete correctly but unsuccessfully; for example, a **COMPARE** of two files detected a discrepancy.

Machine Exception. Read Access Violation.
An attempt was made to read on an addressee which had been protected using Access Permits.

Machine Exception. Write Access Violation.
An attempt was made to write on an addressee which had been protected using Access Permits.

Machine Exception. Data Overrun.
This can occur during direct data transfer, or if the slave has a buffer which was not adequate, and the buffer overran during a read or a write operation.

Machine Exception. Reallocation space exhausted.
Space required for reallocation of data due to media defects is not available; that is, all space assigned for that purpose has been exhausted.

Machine Exception. Unexpected Master Action.

The slave has encountered an unexpected action by the master; for example, the Master Status at the Physical Interface does not correlate to the anticipated status, no status was expected and some was presented by the master, the master did not respond with Data I/O or Control of Bus following a Transfer Notification packet.

Machine Exception. Error Log Full.
The Error Log capacity has been exceeded.

Machine Exception. Defect Directory Full.
The Defect Directory capacity has been exceeded and no more blocks can be re-allocated.

Command Exception. Invalid Packet Length.
The packet length is invalid; for example, the length of the parameter list plus the basic packet does not equal the packet length.

Command Exception. Invalid Command Reference Number.
The Command Reference Number is invalid; for example, it duplicates one in a command that is currently active.

Command Exception. Invalid Slave Address.
The Slave Address in the command packet is invalid; for example, it does not match the selected slave's address.

Command Exception. Invalid Facility Address.
The Facility Address in the command packet is invalid.

Command Exception. Invalid Selection Address.
The facility selected at the Physical Interface does not match the facility address supplied in the command packet.

Command Exception. Invalid Opcode.
The command packet contained an invalid or unsupported Opcode.

Command Exception. Invalid Modifier.
The Modifier was invalid or is not supported for the Opcode specified.

Command Exception. Invalid Extent.
The Data Address (for example, the block number) plus the Count specified in an Extent parameter is not valid for the addressee.

Command Exception. Out of Context.
The slave has encountered a situation in which it cannot process the command because it considers it out of context — for example, a **RESUME** command without a previous **COPY.**

Command Exception. Invalid Parameter.
One or more of the parameters in the command packet was invalid.

Command Exception. Missing Parameter.
One or more of the parameters required in the command is not present.

Command Exception. Reserved Value nonzero.
A reserved value defined by the standard does not contain zero.

Command Exception. Invalid Combination.
The addressee has detected that two valid but mutually exclusive options have been selected by the master.

Command Aborted. Command Aborted.
The command this response packet is related to was **ABORT**ed by the master.

Command Aborted. Command Sequence Terminated.
Command Sequencing was terminated because this command failed to complete successfully.

Command Aborted. Unexecuted Command from Terminated Sequence.

The command related to this response packet was not executed but was terminated because a prior command which was sequenced to it failed to complete successfully.

Command Aborted. Command Chain Terminated.
Command Chaining was terminated because this command failed to complete successfully.

Command Aborted. Unexecuted Command from Terminated Chain.
The command related to this response packet was not executed but was terminated because a prior command which was chained to it failed to complete successfully.

Command Aborted. Command Order Terminated.
Command Ordering was terminated because this command failed to complete successfully.

Command Aborted. Unexecuted Command from Terminated Order.
The command related to this response packet was not executed but was terminated because a prior command which was ordered to it failed to complete successfully.

Conditional Success. Logging Data Appended.
The slave has appended information in this response which the slave is advising the master is relevant to be logged.

Conditional Success. Abort Received – no Command Active.
An **ABORT** command was issued to an addressee in the L-Available condition but the referenced command could not be found.

Conditional Success. Abort Received – Status Pending.
An **ABORT** command was issued to an addressee which has the response status for the referenced command pending; for example, the command has been completed.

Conditional Success. Abort Received – Not Operational.
An **ABORT** command was issued to a facility which is Not Operational.

Conditional Success. Anticipated Error.
The addressee has detected a condition which may result in future error conditions; for example, on disc seek retries were needed.

Conditional Success. Anticipated Data Error.
The addressee has detected a condition which may result in future data errors; for example, successive retries were needed for reading disk data.

Conditional Success. Re-allocation Required.
The addressee has detected a data error condition which requires reallocation action — for example, an unrecoverable read error.

Conditional Success. Re-allocation Discontinuity.
The slave has automatically reallocated a block which contained a data error and the reallocated data is no longer in close proximity to the blocks previously contiguous to it.

Conditional Success. Defect Directory Threshold Exceeded.
The threshold within the addressee's Defect Directory has been exceeded, indicating that there is a limited number of entries remaining for adding more defects.

Conditional Success. Error Retry Performed.
The addressee has completed the command but error retry had to be invoked. Note: Error Retry does not include actions associated with data transfer.

Conditional Success. Data Retry Performed.
The addressee has completed the command but data retry had to be invoked — for example, a physical re-read. Data Retry includes all actions associated with the transfer of data.

Conditional Success. Motion Retry Performed.
The addressee has completed the command but motion retry had to be invoked.

Conditional Success. Data Correction Performed.

The addressee has completed the command but data correction had to be applied.

Conditional Success. Soft Error.
The slave detected an internal machine error that did not preclude execution or continuation of the current command.

Conditional Success. Release of Unreserved Addressee.
The addressee has received a release command for which there is no reservation.

Conditional Success. Request Diagnostic Control Command.
As a result of executing a diagnostic command which provided more information than can be returned by a Response, the addressee is requesting that the master issue a Diagnostic Read Command.

Conditional Success. Error Log Request.
The master is requested to capture the contents of the Error Log (which contains manufacturer dependent information) because the threshold has been exceeded.

Conditional Success. Statistics Update Requested.
There has been a change in meaningful statistics during the execution of this command, and the master is requested to update its Statistics Table (if any).

Conditional Success. Parameter Update Requested.
There has been a change in meaningful device parameters during the execution of this command, and the master is requested to update its Statistics Table (if any).

Conditional Success. Asynchronous Event Occurrence.
An asynchronous event has occurred which may be described further in Extended Status.

Conditional Success. Master Terminated Transfer.
The previous Information Transfer which had a Master Termination Parameter, was terminated by the master.

Incomplete. Command May be Resumed.
This status is used to advise the master that an otherwise successful command did not complete. The incomplete command remains on the slave's queue, and its Command Reference Number shall remain valid, until the command is resumed or aborted.

Incomplete. Response Packet Truncated.
The maximum Information Transfer Length specified by the attributes was exceeded by the response packet, which was truncated at that size, and the response is considered complete by the slave.

Incomplete. Data Length Difference.
The addressee has not transferred all the information specified in a transfer command.

NAME

ie – Intel 10 Mb/s Ethernet interface

CONFIG — SUN-4 SYSTEM

device ie0 at obio ? csr 0xf6000000 priority 3
device ie1 at vme24d16 ? csr 0xe88000 priority 3 vector ieintr 0x75
device ie2 at vme24d16 ? csr 0x31ff02 priority 3 vector ieintr 0x76
device ie3 at vme24d16 ? csr 0x35ff02 priority 3 vector ieintr 0x77
device ie4 at vme24d16 ? csr 0x2dff02 priority 3 vector ieintr 0x7c

ie4 is supported on the SPARCsystems 300 and 400 only.

CONFIG — SUN-3x SYSTEM

device ie0 at obio ? csr 0x65000000 priority 3
device ie1 at vme24d16 ? csr 0xe88000 priority 3 vector ieintr 0x75
device ie2 at vme24d32 ? csr 0x31ff02 priority 3 vector ieintr 0x76
device ie3 at vme24d32 ? csr 0x35ff02 priority 3 vector ieintr 0x77

CONFIG — SUN-3 SYSTEM

device ie0 at obio ? csr 0xc0000 priority 3
device ie1 at vme24d16 ? csr 0xe88000 priority 3 vector ieintr 0x75
device ie2 at vme24d32 ? csr 0x31ff02 priority 3 vector ieintr 0x76
device ie3 at vme24d32 ? csr 0x35ff02 priority 3 vector ieintr 0x77

CONFIG — SUN-3E SYSTEM

device ie0 at vme24d16 ? csr 0x31ff02 priority 3 vector ieintr 0x74

CONFIG — SUN386i SYSTEM

device ie0 at obmem ? csr 0xD0000000 irq 21 priority 3

DESCRIPTION

The **ie** interface provides access to a 10 Mb/s Ethernet network through a controller using the Intel 82586 LAN Coprocessor chip. For a general description of network interfaces see **if**(4N).

ie0 specifies a CPU-board-resident interface, except on a Sun-3E where **ie0** is the Sun-3/E Ethernet expansion board. **ie1** specifies a Multibus Intel Ethernet interface for use with a VME adapter. **ie2** and **ie3** specify SunNet Ethernet/VME Controllers, also known as a Sun-3/E Ethernet expansion boards.

SEE ALSO

if(4N), **le**(4S)

DIAGNOSTICS

There are too many driver messages to list them all individually here. Some of the more common messages and their meanings follow.

ie%d: Ethernet jammed
Network activity has become so intense that sixteen successive transmission attempts failed, and the 82586 gave up on the current packet. Another possible cause of this message is a noise source somewhere in the network, such as a loose transceiver connection.

ie%d: no carrier
The 82586 has lost input to its carrier detect pin while trying to transmit a packet, causing the packet to be dropped. Possible causes include an open circuit somewhere in the network and noise on the carrier detect line from the transceiver.

ie%d: lost interrupt: resetting
The driver and 82586 chip have lost synchronization with each other. The driver recovers by resetting itself and the chip.

ie%d: iebark reset
The 82586 failed to complete a watchdog timeout command in the allotted time. The driver recovers by resetting itself and the chip.

ie%d: WARNING: requeuing

The driver has run out of resources while getting a packet ready to transmit. The packet is put back on the output queue for retransmission after more resources become available.

ie%d: panic: scb overwritten

The driver has discovered that memory that should remain unchanged after initialization has become corrupted. This error usually is a symptom of a bad 82586 chip.

ie%d: giant packet

Provided that all stations on the Ethernet are operating according to the Ethernet specification, this error "should never happen," since the driver allocates its receive buffers to be large enough to hold packets of the largest permitted size. The most likely cause of this message is that some other station on the net is transmitting packets whose lengths exceed the maximum permitted for Ethernet.

BUGS

Raw sockets should receive ICMP error packets relating to the protocol; currently such packets are simply discarded.

Users of higher-level protocols such as TCP and UDP should be able to see received IP options.

NAME

ipi – IPI driver

CONFIG — SUN-4, SPARCsystem 600MP SERIES

controller ipi *cpu-type* **at nexus ?**

DESCRIPTION

The controller line above declares the pseudo-bus IPI, providing a connection between IPI device drivers and IPI channel drivers (also known as host adaptor drivers). Declaring the bus pulls in this middle layer of the IPI drivers, so the IPI bus declaration should be omitted if there are no IPI drivers. See **id**(4S) and **is**(4S). The *cpu-type* number should be the same as that used in the **obio** declaration.

FILES

This driver supports only an interface to other drivers, and therefore does not have any special file access to user programs.

SEE ALSO

id(4S), **is**(4S)

DIAGNOSTICS

ipi_alloc *csf*: **invalid IPI address**

A device driver called **ipi_alloc( )** for a device on a non-existent channel or for an illegal slave address.

ipi_async: ipi *csf* **slave number out of range**

A channel driver called **ipi_async( )** with a response packet containing an illegal slave address.

ipi_setup *csf*: **invalid IPI address**

An IPI device driver used an incorrect IPI address. The channel, slave, or facility number was out of range. This could be a configuration problem.

ipi *csf*: **channel not configured**

This message is printed when a channel driver makes an invalid call to the IPI driver.

ipi_async: IPI channel *c* **not configured**

The channel driver made an invalid call to **ipi_async( )**.

ipi_channel: No memory to add channel *c*

Dynamic memory allocation failed.

ipi_channel: no memory to alloc channel *c*

Dynamic memory allocation failed.

ipi_channel: channel *c* **already defined**

A channel driver tried to add a channel which was previously assigned to a channel driver. This could be caused by a configuration problem.

ipi_free_chan: channel *c* **not allocated**

A channel driver attempted to delete a channel which had never been added.

ipi channel *c* **deleted**

A channel driver deleted the channel because the channel device failed to respond.

ipi: assign_refnum: q *addr* **already has refnum** *r*

A command reference number is being assigned to a command that already had a reference number. This indicates a driver problem.

free_refnum: q *addr* **refnum** *r* **lookup** *l-addr*

The driver was freeing an IPI request at address *addr*, and found that the reference number it used, *r*, was assigned to another request at address *l-addr*.

ipi_lookup: found wrong command for refnum

ipi_lookup: looking for *r*, **found** *fr* **q** *addr*

The driver was looking for a command with a particular reference number, *r*, but the lookup table

found one with a different reference number, *fr*. These messages are always received together.

ipi *csf*: **missing interrupt. refnum** *r*

The driver detected that a command took longer than the device driver expected it to take. This message will be followed by other messages from the device driver.

NAME

is – IPI channel driver for Sun IPI string controllers

CONFIG — SUN-4, SPARCsystem 600MP SERIES

```
controller   isc0   at vme32d32 ? csr 0x01080000 priority 2 vector isintr 0x4c
channel      is0    at isc0 ipi_addr 0x00000  # channel 0 slave 0
controller   isc1   at vme32d32 ? csr 0x01080400 priority 2 vector isintr 0x4d
channel      is1    at isc1 ipi_addr 0x00100  # channel 0 slave 1
controller   isc2   at vme32d32 ? csr 0x01080800 priority 2 vector isintr 0x4e
channel      is2    at isc2 ipi_addr 0x00200  # channel 0 slave 2
controller   isc3   at vme32d32 ? csr 0x01080c00 priority 2 vector isintr 0x4f
channel      is3    at isc3 ipi_addr 0x00300  # channel 0 slave 3
controller   isc4   at vme32d32 ? csr 0x01081000 priority 2 vector isintr 0x50
channel      is4    at isc4 ipi_addr 0x00400  # channel 0 slave 4
```

DESCRIPTION

The first four **controller** and **channel** lines given in the synopsis section above correspond to the first, second, third, and fourth ISP-80 IPI disk controllers in a Sun system; the fifth controller and channel pair are currently available on SPARCsystem 600MP series machines only. These controllers are treated as integrated channel/disk-controller cards, in that there are two drivers that manage them. This driver, the **is** driver, presents a generic IPI-3 interface to the **id** driver, which handles the controller as if it were a channel-attached IPI-3 controller. See **id**(4S).

The **csr** value gives the VME address of the controller. The **ipi_addr** field gives the 3-byte channel, slave, and facility address supported by the controller; the facility address portion is ignored and should be specified as zero.

SEE ALSO

id(4S)

DIAGNOSTICS

isc*n*: channel reset timed out
The probe routine, called during system initialization, determined that the controller reset, initiated by the boot, had not completed in the allotted time. The controller cannot be used.

isc*n*: bad IPI address *addr*
The specified address contained a slave address larger than 7.

isc*n*: interrupt vector not specified in config
The configuration did not specify an interrupt vector for the controller.

is*n*: refnum lookup on success failed. refnum *refnum*
A completion interrupt occurred for a command which was unknown to the controller. The interrupt is ignored.

is*n*: response too short. len *l* **min** *m* **response** *r*
The response packet presented in an interrupt was too short to contain a valid header. The response is printed, but otherwise the interrupt is ignored.

is*n*: response too long. max *m* **len** *l* **truncating**
The response packet is longer than the maximum, *m*, for this controller,

is*n*: is_reset_slave: resetting slave
The driver is resetting the controller under direction from the **id** disk driver. This occurs during recovery from timed out requests, also known as missing interrupts.

is*n*: is_reset_slave: slave reset not enabled
The **id** driver attempted to reset the controller to recovery from an error, but the driver has been set to disable such resets.

is*n*: ctlr reset timed out

The reset of the controller took longer than expected and has presumably failed.

isc*n*: response reg *r* not same as packet refnum *ref*
After reading a response packet, the response register was read and did not match the command reference number in the response packet. This could indicate a problem with the driver or the controller.

isc*n*: error status *s* response *r*
The controller presented error status code *s* and a response register value of *r*. These will be interpreted by messages that follow.

isc*n*: ctlr fault *f* - bus error on cmd refnum *r*
The controller reported a bus error while fetching the command.

isc*n*: ctlr fault *f* - timeout on cmd refnum *r*
The controller reported a VME timeout while fetching the command.

isc*n*: ctlr fault *f* - invalid command reg write on cmd refnum *r*
The controller indicates that the command register was written after the controller has panicked.

isc*n*: ctlr fault *f* - unknown fault code on cmd refnum *r*
The controller has presented an unknown fault code in its status.

NAME

kb – Sun keyboard STREAMS module

CONFIG

pseudo-device kb*number*

SYNOPSIS

```
#include <sys/types.h>
#include <sys/stream.h>
#include <sys/stropts.h>
#include <sundev/vuid_event.h>
#include <sundev/kbio.h>
#include <sundev/kbd.h>

ioctl(fd, I_PUSH, "kb");
```

DESCRIPTION

The **kb** STREAMS module processes byte streams generated by Sun keyboards attached to a CPU serial or parallel port. Definitions for altering keyboard translation, and reading events from the keyboard, are in **<sundev/kbio.h>** and **<sundev/kbd.h>**. *number* specifies the maximum number of keyboards supported by the system.

kb recognizes which keys have been typed using a set of tables for each known type of keyboard. Each translation table is an array of 128 16-bit words (**unsigned shorts**). If an entry in the table is less than 0x100, it is treated as an ISO 8859/1 character. Higher values indicate special characters that invoke more complicated actions.

Keyboard Translation Mode

The keyboard can be in one of the following translation modes:

TR_NONE	Keyboard translation is turned off and up/down key codes are reported.
TR_ASCII	ISO 8859/1 codes are reported.
TR_EVENT	**firm_events** are reported (see *SunView Programmer's Guide*).
TR_UNTRANS_EVENT	**firm_events** containing unencoded keystation codes are reported for all input events within the window system.

Keyboard Translation-Table Entries

All instances of the **kb** module share seven translation tables used to convert raw keystation codes to event values. The tables are:

Unshifted	Used when a key is depressed and no shifts are in effect.
Shifted	Used when a key is depressed and a Shift key is being held down.
Caps Lock	Used when a key is depressed and Caps Lock is in effect.
Alt Graph	Used when a key is depressed and the Alt Graph key is being held down.
Num Lock	Used when a key is depressed and Num Lock is in effect.
Controlled	Used when a key is depressed and the Control key is being held down (regardless of whether a Shift key or the Alt Graph is being held down, or whether Caps Lock or Num Lock is in effect).
Key Up	Used when a key is released.

Each key on the keyboard has a "key station" code which is a number from 0 to 127. This number is used as an index into the translation table that is currently in effect. If the corresponding entry in that translation table is a value from 0 to 255, this value is treated as an ISO 8859/1 character, and that character is the result of the translation.

NAME

le – LANCE 10Mb/s Ethernet interface

CONFIG — SUN-4/3xx, SUN-4/xx, SS6xxMP SYSTEMS

device le0 at obio ? csr 0xf9000000 priority 3

CONFIG — OPEN BOOT PROM systems

device-driver le
device-driver lebuf
device-driver ledma

DESCRIPTION

The **le** interface provides access to a 10 Mb/s Ethernet network through a controller using the AMD LANCE (Local Area Network Controller for Ethernet) Am7990 chip. For a general description of network interfaces see **if**(4N). The **lebuf** and **ledma** device drivers, when present, support hardware-specific buffering and DMA for each **le** interface.

DIAGNOSTICS

le%d: out of mbufs: output packet dropped
The driver has run out of memory to use to buffer packets on output. The packet being transmitted at the time of occurrence is lost. This error is usually symptomatic of trouble elsewhere in the kernel.

le%d: out of tmds - packet dropped
The driver has run out of transmit message descriptors needed to queue a packet to the LANCE and discarded the packet. This error usually indicates that the upper protocol layers are generating packets faster than the driver can transmit them on the ethernet and is usually self-correcting.

le%d: trailer error
An incoming packet claimed to have a trailing header but did not.

le%d: RINT with buffer owned by chip
The LANCE has generated a receive interrupt but the driver finds no packets have been received. This causes the driver to reinitialize the chip.

le%d: runt packet
An incoming packet's size was below the Ethernet minimum transmission size.

le%d: Receive: overflow error
Indicates that the receiver has lost all or part of an incoming packet due to an inability to store the packet in a memory buffer before the LANCE internal silo overflowed.

le%d: Receive: BUFF set in rmd
This error "should never happen," as it occurs only in conjunction with a LANCE feature that the driver does not use.

le%d: Receive: framming error
The driver has received a packet containing a noninteger multiple of eight bits and there was a CRC error.

le%d: Receive: crc error
The driver has received a packet with an incorrect checksum field.

le%d: Receive: STP in rmd cleared
The driver has received a packet that straddles multiple receive buffers and therefore consumes more than one of the LANCE chip's receive descriptors. Provided that all stations on the Ethernet are operating according to the Ethernet specification, this error "should never happen," since the driver allocates its receive buffers to be large enough to hold packets of the largest permitted size. Most likely, some other station on the net is transmitting packets whose lengths exceed the maximum permitted for Ethernet.

le%d: Receive: ENP in rmd cleared
The driver has received a packet that straddles multiple receive buffers and therefore consumes more than one of the LANCE chip's receive descriptors. Provided that all stations on the Ethernet are operating according to the Ethernet specification, this error "should never happen," since the driver allocates its receive buffers to be large enough to hold packets of the largest permitted size. The most likely cause of the message is that some other station on the net is transmitting packets whose lengths exceed the maximum permitted for Ethernet.

le%d: Transmit: BUFF set in tmd
Excessive bus contention has prevented the LANCE chip from gathering packet contents quickly enough to sustain the packet's transmission over the Ethernet. The affected packet is lost.

le%d: Transmit underflow
The transmitter has truncated a packet due to data late from memory.

le%d: Transmit late collision – Net problem?
A packet collision has occurred after the channel's slot time has elapsed. This error usually indicates faulty hardware elsewhere on the net.

le%d: No carrier – transceiver cable problem?
The LANCE chip has lost input to its carrier detect pin while trying to transmit a packet.

le%d: Transmit retried more than 16 times – net jammed
Network activity has become so intense that sixteen successive transmission attempts failed, the LANCE chip gave up on the current packet.

le%d: missed packet
The driver has dropped an incoming packet because it had no buffer space for it.

le%d: Babble error – sent a packet longer than the maximum length
While transmitting a packet, the LANCE chip has noticed that the packet's length exceeds the maximum allowed for Ethernet (1519 bytes). This error indicates a kernel bug.

le%d: Memory Error!
The LANCE chip timed out while trying to acquire the bus for a DVMA transfer.

le%d: Reception stopped
Because of some other error, the receive section of the LANCE chip shut down and had to be reinitialized.

le%d: Transmission stopped
Because of some other error, the transmit section of the LANCE chip shut down and had to be reinitialized.

NAME

mcp, alm – Sun MCP Multiprotocol Communications Processor/ALM-2 Asynchronous Line Multiplexer

CONFIG — SUN-3, SUN-3x, SUN-4, SPARCsystem 600MP SERIES

MCP

device mcp0 at vme32d32 ? csr 0x1000000 flags 0x1ffff priority 4 vector mcpintr 0x8b
device mcp1 at vme32d32 ? csr 0x1010000 flags 0x1ffff priority 4 vector mcpintr 0x8a
device mcp2 at vme32d32 ? csr 0x1020000 flags 0x1ffff priority 4 vector mcpintr 0x89
device mcp3 at vme32d32 ? csr 0x1030000 flags 0x1ffff priority 4 vector mcpintr 0x88
device mcp4 at vme32d32 ? csr 0x2000000 flags 0x1ffff priority 4 vector mcpintr 0xa0
device mcp5 at vme32d32 ? csr 0x2010000 flags 0x1ffff priority 4 vector mcpintr 0xa1
device mcp6 at vme32d32 ? csr 0x2020000 flags 0x1ffff priority 4 vector mcpintr 0xa2
device mcp7 at vme32d32 ? csr 0x2030000 flags 0x1ffff priority 4 vector mcpintr 0xa3

ALM-2

pseudo-device mcpa128

SYNOPSIS

#include <fcntl.h>
#include <sys/termios.h>
open("/dev/tty*xy***", mode);**
open("/dev/ttyd*n***", mode);**
open("/dev/cua*n***", mode);**

DESCRIPTION (MCP)

The Sun MCP (Multiprotocol Communications Processor) supports up to four synchronous serial lines in conjunction with SunLink™ Multiple Communication Protocol products.

DESCRIPTION (ALM-2)

The Sun ALM-2 Asynchronous Line Multiplexer provides 16 asynchronous serial communication lines with modem control and one Centronics-compatible parallel printer port.

Each port supports those **termio**(4) device control functions specified by flags in the **c_cflag** word of the **termios** structure and by the **IGNBRK**, **IGNPAR**, **PARMRK**, or **INPCK** flags in the **c_iflag** word of the **termios** structure are performed by the **mcp** driver. All other **termio**(4) functions must be performed by STREAMS modules pushed atop the driver; when a device is opened, the **ldterm**(4M) and **ttcompat**(4M) STREAMS modules are automatically pushed on top of the stream, providing the standard **termio**(4) interface.

Bit *i* of **flags** may be specified to say that a line is not properly connected, and that the line *i* should be treated as hard-wired with carrier always present. Thus, specifying **flags 0x0004** in the specification of **mcp0** would treat line **/dev/ttyh2** in this way.

Minor device numbers in the range 0 – 127 correspond directly to the normal tty lines and are named **/dev/tty***XY*, where *X* represents the physical board as one of the characters **h**, **i**, **j**, **k**, **l**, **m**, **n**, or **o**, and *Y* is the line number on the board as a single hexadecimal digit. Thus the first line on the first board is **/dev/ttyh0**, and the sixteenth line on the third board is **/dev/ttyjf**.

To allow a single tty line to be connected to a modem and used for both incoming and outgoing calls, a special feature, controlled by the minor device number, has been added. Minor device numbers in the range 128 – 255 correspond to the same physical lines as those above (that is, the same line as the minor device number minus 128).

A dial-in line has a minor device in the range 0 – 127 and is conventionally renamed **/dev/ttyd***n*, where *n* is a number indicating which dial-in line it is (so that **/dev/ttyd0** is the first dial-in line), and the dial-out line corresponding to that dial-in line has a minor device number 128 greater than the minor device number of the dial-in line and is conventionally named **/dev/cua***n*, where *n* is the number of the dial-in line.

The **/dev/cua***n* lines are special in that they can be opened even when there is no carrier on the line. Once a **/dev/cua***n* line is opened, the corresponding tty line cannot be opened until the **/dev/cua***n* line is closed; a blocking open will wait until the **/dev/cua***n* line is closed (which will drop Data Terminal Ready, after which Carrier Detect will usually drop as well) and carrier is detected again, and a non-blocking open will return an error. Also, if the **/dev/ttyd***n* line has been opened successfully (usually only when carrier is recognized on the modem) the corresponding **/dev/cua***n* line cannot be opened. This allows a modem to be attached, for example, to **/dev/ttyd0** (renamed from **/dev/ttyh0**) and used for dialin (by enabling the line for login in **/etc/ttytab**) and also used for dialout (by **tip**(1C) or **uucp**(1C)) as **/dev/cua0** when no one is logged in on the line. Note: the bit in the **flags** word in the configuration file (see above) must be zero for this line, which enables hardware carrier detection.

IOCTLS

The standard set of **termio ioctl( )** calls are supported by the ALM-2.

If the **CRTSCTS** flag in the **c_cflag** is set, output will be generated only if CTS is high; if CTS is low, output will be frozen. If the **CRTSCTS** flag is clear, the state of CTS has no effect. Breaks can be generated by the **TCSBRK**, **TIOCSBRK**, and **TIOCCBRK ioctl( )** calls. The modem control lines **TIOCM_CAR**, **TIOCM_CTS**, **TIOCM_RTS**, and **TIOCM_DTR** are provided.

The input and output line speeds may be set to any of the speeds supported by **termio**. The speeds cannot be set independently; when the output speed is set, the input speed is set to the same speed.

ERRORS

An **open( )** on a **/dev/tty*** or a **/dev/cu*** device will fail if:

ENXIO	The unit being opened does not exist.
EBUSY	The dial-out device is being opened and the dial-in device is already open, or the dial-in device is being opened with a no-delay open and the dial-out device is already open.
EBUSY	The unit has been marked as exclusive-use by another process with a **TIOCEXCL ioctl( )** call.
EINTR	The open was interrupted by the delivery of a signal.

DESCRIPTION (PARALLEL PORT)

The parallel port is Centronics-compatible and is suitable for most common parallel printers. Devices attached to this interface are normally handled by the line printer spooling system, and should not be accessed directly by the user.

The printer devices reside on a separate major device number from the serial devices. Minor device numbers in the range 0 – 7 access the printer, and the recommended naming is **/dev/mcpp[0-7]**.

IOCTLS

Various control flags and status bits may be fetched and set on an MCP printer port. The following flags and status bits are supported; they are defined in **sundev/mcpcmd.h**:

MCPRIGNSLCT	0x02	set if interface ignoring SLCT– on open
MCPRDIAG	0x04	set if printer port is in self-test mode
MCPRVMEINT	0x08	set if VME bus interrupts enabled
MCPRINTPE	0x10	print message when out of paper
MCPRINTSLCT	0x20	print message when printer offline
MCPRPE	0x40	set if device ready, cleared if device out of paper
MCPRSLCT	0x80	set if device online (Centronics SLCT asserted)

The flags **MCPRINTSLCT, MCPRINTPE,** and **MCPRDIAG** may be changed; the other bits are status bits and may not be changed.

The **ioctl( )** calls supported by MCP printer ports are listed below.

MCPIOGPR	The argument is a pointer to an **unsigned char**. The printer flags and status bits are stored in the **unsigned char** pointed to by the argument.

MCPIOSPR The argument is a pointer to an **unsigned char**. The printer flags are set from the **unsigned char** pointed to by the argument.

ERRORS

Normally, the interface only reports the status of the device when attempting an **open**(2V) call. An **open**( ) on a **/dev/mcpp*** device will fail if:

ENXIO The unit being opened does not exist.

EIO The device is offline or out of paper.

Bit 17 of the configuration **flags** may be specified to say that the interface should ignore Centronics SLCT– and RDY/PE– when attempting to open the device, but this is normally useful only for configuration and troubleshooting: if the SLCT– and RDY lines are not asserted during an actual data transfer (as with a **write**(2V) call), no data is transferred.

FILES

/dev/mcpp[0-7]	parallel printer port
/dev/tty[h-o][0-9a-f]	hardwired tty lines
/dev/ttyd[0-9a-f]	dialin tty lines
/dev/cua[0-9a-f]	dialout tty lines

SEE ALSO

tip(1C), **uucp**(1C), **mti**(4S), **termio**(4), **ldterm**(4M), **ttcompat**(4M), **zs**(4S)

DIAGNOSTICS

Most of these diagnostics ''should never happen''; their occurrence usually indicates problems elsewhere in the system as well.

mcpa*n*: silo overflow.
More than *n* characters (*n* very large) have been received by the **mcp** hardware without being read by the software.

port *n* supports RS449 interface**
Probably an incorrect jumper configuration. Consult the hardware manual.

mcp port *n* receive buffer error
The **mcp** encountered an error concerning the synchronous receive buffer.

Printer on mcpp*n* is out of paper
Printer on mcpp*n* paper ok
Printer on mcpp*n* is offline
Printer on mcpp*n* online
Assorted printer diagnostics, if enabled as discussed above.

BUGS

Note: pin 4 is used for hardware flow control on ALM–2 ports 0 through 3. These two pins should *not* be tied together on the ALM end.

NAME

mem, kmem, zero, vme16d16, vme24d16, vme32d16, vme16d32, vme24d32, vme32d32, eeprom, atbus, sbus – main memory and bus I/O space

CONFIG

None; included with standard system.

DESCRIPTION

These devices are special files that map memory and bus I/O space. They may be read, written, seeked, and (except for **kmem**) memory-mapped. See **read**(2V), **write**(2V), **mmap**(2), and **directory**(3V).

All Systems

mem is a special file that is an image of the physical memory of the computer. It may be used, for example, to examine (and even to patch) the system.

kmem is a special file that is an image of the kernel virtual memory of the system.

zero is a special file which is a source of private zero pages.

eeprom is a special file that is an image of the EEPROM or NVRAM.

Sun-3 and Sun-4 Systems VMEbus

vme16d16 (also known as **vme16**) is a special file that is an image of VMEbus 16-bit addresses with 16-bit data. **vme16** address space extends from 0 to 64K.

vme24d16 (also known as **vme24**) is a special file that is an image of VMEbus 24-bit addresses with 16-bit data. **vme24** address space extends from 0 to 16 megabytes. The VME 16-bit address space overlaps the top 64K of the 24-bit address space.

vme32d16 is a special file that is an image of VMEbus 32-bit addresses with 16-bit data.

vme16d32 is a special file that is an image of VMEbus 16-bit addresses with 32-bit data.

vme24d32 is a special file that is an image of VMEbus 24-bit addresses with 32-bit data.

vme32d32 (also known as **vme32**) is a special file that is an image of VMEbus 32-bit addresses with 32-bit data. **vme32** address space extends from 0 to 4 gigabytes. The VME 24-bit address space overlaps the top 16 megabytes of the 32-bit address space.

Desktop SPARCsystems

The **sbus** is represented by a series of entries each of which is an image of a single **sbus** slot. The entries are named **sbus***n*, where *n* is the slot number in hexadecimal. The number of **sbus** slots and the address range within each slot may vary between implementations.

SPARCsystem 600MP Series

SPARCsystem 600MP series systems have both **sbus** and VMEbus as described above.

Sun386i Systems

atbus is a special file that is an image of the AT bus space. It extends from 0 to 16 megabytes.

FILES

/dev/mem
/dev/kmem
/dev/zero
/dev/vme16d16
/dev/vme16
/dev/vme24d16
/dev/vme24
/dev/vme32d16
/dev/vme16d32
/dev/vme24d32
/dev/vme32d32
/dev/vme32

/dev/eeprom
/dev/atbus
/dev/sbus[0-3]

SEE ALSO

mmap(2), **read**(2V), **write**(2V), **directory**(3V)

NAME

mouse – Sun mouse

CONFIG

None; included in standard system.

DESCRIPTION

The **mouse** indirect device provides access to the Sun Workstation mouse. When opened, it redirects operations to the standard mouse device for the workstation (attached either to a CPU serial or parallel port), and pushes the **ms**(4M) and **ttcompat**(4M) STREAMS modules on top of that device.

FILES

/dev/mouse

SEE ALSO

ms(4M), **ttcompat**(4M), **win**(4S), **zs**(4S)

NAME

mtio – general magnetic tape interface

SYNOPSIS

#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/mtio.h>

DESCRIPTION

1/2", 1/4" and 8 mm magnetic tape drives all share the same general character device interface.

There are two types of tape records: data records and end-of-file (EOF) records. EOF records are also known as tape marks and file marks. A record is separated by interrecord (or tape) gaps on a tape.

End-of-recorded-media (EOM) is indicated by two EOF marks on 1/2" tape; by one on 1/4" and 8 mm cartridge tapes.

1/2" Reel Tape

Data bytes are recorded in parallel onto the 9–track tape. The number of bytes in a physical record varies between 1 and 65535 bytes.

The recording formats available (check specific tape drive) are 800 BPI, 1600 BPI, and 6250 BPI, and data compression. Actual storage capacity is a function of the recording format and the length of the tape reel. For example, using a 2400 foot tape, 20 MB can be stored using 800 BPI, 40 MB using 1600 BPI, 140 MB using 6250 BPI, or up to 700 MB using data compression.

1/4" Cartridge Tape

Data is recorded serially onto 1/4" cartridge tape. The number of bytes per record is determined by the physical record size of the device. The I/O request size must be a multiple of the physical record size of the device. For QIC–11, QIC–24, and QIC–150 tape drives the block size is 512 bytes.

The records are recorded on tracks in a serpentine motion. As one track is completed, the drive switches to the next and begins writing in the opposite direction, eliminating the wasted motion of rewinding. Each file, including the last, ends with one file mark.

Storage capacity is based on the number of tracks the drive is capable of recording. For example, 4–track drives can only record 20 MB of data on a 450 foot tape; 9–track drives can record up to 45 MB of data on a tape of the same length. QIC–11 is the only tape format available for 4–track tape drives. In contrast, 9–track tape drives can use either QIC–24 or QIC–11. Storage capacity is not appreciably affected by using either format. QIC–24 is preferable to QIC–11 because it records a reference signal to mark the position of the first track on the tape, and each block has a unique block number.

The QIC–150 tape drives require DC–6150 (or equivalent) tape cartridges for writing. However, they can read other tape cartridges in QIC–11, QIC–24, QIC–120, or QIC–150 tape formats.

8 mm Cartridge Tape

Data is recorded serially onto 8 mm helical scan cartridge tape. The number of bytes in a physical record varies between 1 and 65535 bytes.

Read Operation

read(2V) reads the next record on the tape. The record size is passed back as the number of bytes read, provided it is no greater than the number requested. When a tape mark is read, a zero byte count is returned; another read will fetch the first record of the next tape file. Two successive reads returning zero byte counts indicate the EOM. No further reading should be performed past the EOM.

Fixed-length I/O tape devices require the number of bytes read to be a multiple of the physical record size. For example, 1/4" cartridge tape devices only read multiples of 512 bytes. If the blocking factor is greater than 64512 bytes (minphys limit), fixed-length I/O tape devices read multiple records.

Tape devices which support variable-length I/O operations, such as 1/2" and 8mm tape, may read a range of 1 to 65535 bytes. If the record size exceeds 65535 bytes, the driver reads multiple records to satisfy the request. These multiple records are limited to 65534 bytes.

Write Operation

write(2V) writes the next record on the tape. The record has the same length as the given buffer.

Writing is allowed on 1/4" tape at either the beginning of tape or after the last written file on the tape.

Writing is not so restricted on 1/2" and 8 mm cartridge tape. Care should be used when appending files onto 1/2" reel tape devices, since an extra file mark is appended after the last file to mark the EOM. This extra file mark must be overwritten to prevent the creation of a null file. To facilitate write append operations, a space to the EOM ioctl is provided. Care should be taken when overwriting records; the erase head is just forward of the write head and any following records will also be erased.

Fixed-length I/O tape devices require the number of bytes written to be a multiple of the physical record size. For example, 1/4" cartridge tape devices only write multiples of 512 bytes. Fixed-length I/O tape devices write multiple records if the blocking factor is greater than 64512 bytes (minphys limit). These multiple writes are limited to 64512 bytes. For example, if a write request is issued for 65536 bytes using a 1/4" cartridge tape, two writes are issued; the first for 64512 bytes and the second for 1024 bytes.

Tape devices which support variable-length I/O operations, such as 1/2" and 8mm tape, may write a range of 1 to 65535 bytes. If the record size exceeds 65535 bytes, the driver writes multiple records to satisfy the request. These multiple records are limited to 65534 bytes. As an example, if a write request for 65540 bytes is issued using 1/2" reel tape, two records are written; one for 65534 bytes followed by one for 6 bytes.

EOT handling on write is different among the various devices; see the appropriate device manual page. Reading past EOT is transparent to the user.

Seeks are ignored in tape I/O.

Close Operation

Magnetic tapes are rewound when closed, except when the "no-rewind" devices have been specified. The names of no-rewind device files use the letter **n** as the beginning of the final component. The no-rewind version of **/dev/rmt0** is **/dev/nrmt0**.

If data was written, a file mark is automatically written by the driver upon close. If the rewinding device was specified, the tape will be rewound after the file mark is written. If the user wrote a file mark prior to closing, then no file mark is written upon close. If a file positioning ioctl, like rewind, is issued after writing, a file mark is written before repositioning the tape.

Note: for 1/2" reel tape devices, two file marks are written to mark the EOM before rewinding or performing a file positioning ioctl. If the user wrote a file mark before closing a 1/2" reel tape device, the driver will always write a file mark before closing to insure that the end of recorded media is marked properly. If the non-rewinding **xt** device was specified, two file marks are written and the tape is left positioned between the two so that the second one is overwritten on a subsequent **open**(2V) and **write**(2V). For performance reasons, some **st** drivers postpone writing the second tape mark until just before a file positioning ioctl is issued (for example, rewind). This means that the user must not manually rewind the tape because the tape will be missing the second tape mark which marks EOM.

If no data was written and the driver was opened for WRITE-ONLY access, a file mark is written thus creating a null file.

Ioctls

Not all devices support all ioctls. The driver returns an ENOTTY error on unsupported ioctls.

The following structure definitions for magnetic tape ioctl commands are from **<sys/mtio.h>**:

The minor device byte structure looks as follows:

7	6	5	4	3	2	1	0
Reserved	Unit # High Bit*	Reserved	Density Select	Density Select	No rewind on Close	Unit # Lower 2 Bits	

```
/*
 * Layout of minor device byte:
 */
#define MTUNIT(dev)          (((minor(dev) & 0x40) >> 4) + (minor(dev) & 0x3))
#define MT_NOREWIND          (1 <<2)
#define MT_DENSITY_MASK      (3 <<3)
#define MT_DENSITY1          (0 <<3)     /* Lowest density/format */
#define MT_DENSITY2          (1 <<3)
#define MT_DENSITY3          (2 <<3)
#define MT_DENSITY4          (3 <<3)     /* Highest density/format */
#define MTMINOR(unit)        (((unit & 0x04) << 4) + (unit & 0x3))
```

*NOTE that bit 6 of the minor device byte is always 0 for 4.1-based systems, since 4.1 supports a maximum of 4 SCSI tape drives. However, 4.1 PSR A-based systems support a maximum of 8 SCSI tape drives; thus this bit is 1 for *st4 – st7*. Note also that both 4.1 and 4.1 PSR A ship with the 4.1 version of */usr/include/sys/mtio.h*. The additional bit with MTUNIT and MTMINOR macros is defined in the 4.1 PSR A version of *mtio.h*, found in */sys/sys/mtio.h* on machines running 4.1 PSR A.

```
/* structure for MTIOCTOP – magnetic tape operation command */
struct    mtop      {
          short     mt_op;     /* operation */
          daddr_t   mt_count;  /* number of operations */
};
```

The following ioctls are supported:

MTWEOF	write an end-of-file record
MTFSF	forward space over file mark
MTBSF	backward space over file mark (1/2", 8 mm only)
MTFSR	forward space to inter-record gap
MTBSR	backward space to inter-record gap
MTREW	rewind
MTOFFL	rewind and take the drive offline
MTNOP	no operation, sets status only
MTRETEN	retension the tape (cartridge tape only)
MTERASE	erase the entire tape and rewind
MTEOM	position to EOM
MTNBSF	backward space file to beginning of file

```
/* structure for MTIOCGET – magnetic tape get status command */
struct    mtget {
          short     mt_type;          /* type of magtape device */
/* the following two registers are device dependent */
          short     mt_dsreg;         /* "drive status" register */
          short     mt_erreg;         /* "error" register */
```

```
/* optional error info. */
        daddr_t     mt_resid;           /* residual count */
        daddr_t     mt_fileno;          /* file number of current position */
        daddr_t     mt_blkno;           /* block number of current position */
        u_short mt_flags;
        short   mt_bf;                          /* optimum blocking factor */
};
```

When spacing forward over a record (either data or EOF), the tape head is positioned in the tape gap between the record just skipped and the next record. When spacing forward over file marks (EOF records), the tape head is positioned in the tape gap between the next EOF record and the record that follows it.

When spacing backward over a record (either data or EOF), the tape head is positioned in the tape gap immediately preceding the tape record where the tape head is currently positioned. When spacing backward over file marks (EOF records), the tape head is positioned in the tape gap preceding the EOF. Thus the next read would fetch the EOF.

Note, the following features are unique to the **st** driver: record skipping does not go past a file mark; file skipping does not go past the EOM. Both the **st** and **xt** drivers stop upon encountering EOF during a record skipping command, but leave the tape positioned differently. For example, after an MTFSR <huge number> command the **st** driver leaves the tape positioned *before* the EOF. After the same command, the **xt** driver leaves the tapes positioned *after* the EOF. Consequently on the next read, the **xt** driver fetches the first record of the next file whereas the **st** driver fetches the EOF. A related **st** feature is that EOFs remain pending until the tape is closed. For example, a program which first reads all the records of a file up to and including the EOF and then performs an MTFSF command will leave the tape positioned just after that same EOF, rather than skipping the next file.

The MTNBSF and MTFSF operations are inverses. Thus, an MTFSF "–1" is equivalent to an MTNBSF "1". An MTNBSF "0" is the same as MTFSF "0"; both position the tape device to the beginning of the current file.

MTBSF moves the tape backwards by file marks. The tape position will end on the beginning of tape side of the desired file mark.

MTBSR and MTFSR operations perform much like space file operations, except that they move by records instead of files. Variable-length I/O devices (1/2" reel, for example) space actual records; fixed-length I/O devices space physical records (blocks). 1/4" cartridge tape, for example, spaces 512 byte physical records. The status ioctl residual count contains the number of files or records not skipped.

MTOFFL rewinds and, if appropriate, takes the device offline by unloading the tape. The tape must be inserted before the tape device can be used again.

MTRETEN The retension ioctl only applies to 1/4" cartridge tape devices. It is used to restore tape tension improving the tape's soft error rate after extensive start-stop operations or long-term storage.

MTERASE rewinds the tape, erases it completely, and returns to the beginning of tape.

MTEOM positions the tape at a location just after the last file written on the tape. For 1/4" cartridge and 8 mm tape, this is after the last file mark on the tape. For 1/2" reel tape, this is just after the first file mark but before the second (and last) file mark on the tape. Additional files can then be appended onto the tape from that point.

Note the difference between MTBSF (backspace over file mark) and MTNBSF (backspace file to beginning of file). The former moves the tape backward until it crosses an EOF mark, leaving the tape positioned *before* the file mark. The latter leaves the tape positioned *after* the file mark. Hence, "MTNBSF n" is equivalent to "MTBSF (n+1)" followed by "MTFSF 1". 1/4 " cartridge tape devices do not support MTBSF.

The MTIOCGET get status ioctl call returns the drive id (*mt_type*), sense key error (*mt_erreg*), file number (*mt_fileno*), optimum blocking factor (*mt_bf*) and record number (*mt_blkno*) of the last error. The residual count (*mt_resid*) is set to the number of bytes not transferred or files/records not spaced. The flags word (*mt_flags*) contains information such as whether the device is SCSI, whether it is a reel device and whether the device supports absolute file positioning.

EXAMPLES

Suppose you have written 3 files to the non-rewinding 1/2" tape device, **/dev/nrmt0**, and that you want to go back and **dd**(1) the second file off the tape. The commands to do this are:

```
mt –f /dev/nrmt0 bsf 3
mt –f /dev/nrmt0 fsf 1
dd if=/dev/nrmt0
```

To accomplish the same tape positioning in a C program, followed by a get status ioctl:

```
struct mtop mt_command;
struct mtget mt_status;

mt_command.mt_op = MTBSF;
mt_command.mt_count = 3;
ioctl(fd, MTIOCTOP, &mt_command);
mt_command.mt_op = MTFSF;
mt_command.mt_count = 1;
ioctl(fd, MTIOCTOP, &mt_command);
ioctl(fd, MTIOCGET, (char *)&mt_status);
```

or

```
struct mtop mt_command;
struct mtget mt_status;

mt_command.mt_op = MTNBSF;
mt_command.mt_count = 2;
ioctl(fd, MTIOCTOP, &mt_command);
ioctl(fd, MTIOCGET, (char *)&mt_status);
```

FILES

/dev/rmt*
/dev/rst*
/dev/rar*
/dev/nrmt*
/dev/nrst*
/dev/nrar*

SEE ALSO

dd(1), **mt**(1), **tar**(1), **read**(2V), **write**(2V), **ar**(4S), **st**(4S), **tm**(4S), **xt**(4S)

1/4 Inch Tape Drive Tutorial

WARNINGS

Avoid the use of device files **/dev/rmt4** and **/dev/rmt12**, as they are going away in a future release.

NAME

nfs, NFS – network file system

CONFIG

options NFS

DESCRIPTION

The Network File System, or NFS, allows a client workstation to perform transparent file access over the network. Using it, a client workstation can operate on files that reside on a variety of servers, server architectures and across a variety of operating systems. Client file access calls are converted to NFS protocol requests, and are sent to the server system over the network. The server receives the request, performs the actual file system operation, and sends a response back to the client.

The Network File System operates in a stateless fashion using remote procedure (RPC) calls built on top of external data representation (XDR) protocol. These protocols are documented in *Network Interfaces Programmer's Guide* The RPC protocol provides for version and authentication parameters to be exchanged for security over the network.

A server can grant access to a specific filesystem to certain clients by adding an entry for that filesystem to the server's **/etc/exports** file and running **exportfs**(8).

A client gains access to that filesystem with the **mount**(2V) system call, which requests a file handle for the filesystem itself. Once the filesystem is mounted by the client, the server issues a file handle to the client for each file (or directory) the client accesses or creates. If the file is somehow removed on the server side, the file handle becomes stale (dissociated with a known file).

A server may also be a client with respect to filesystems it has mounted over the network, but its clients cannot gain access to those filesystems. Instead, the client must mount a filesystem directly from the server on which it resides.

The user ID and group ID mappings must be the same between client and server. However, the server maps uid 0 (the super-user) to uid –2 before performing access checks for a client. This inhibits super-user privileges on remote filesystems. This may be changed by use of the "anon" export option. See **exportfs**(8).

NFS-related routines and structure definitions are described in *Network Interfaces Programmer's Guide*.

ERRORS

Generally physical disk I/O errors detected at the server are returned to the client for action. If the server is down or inaccessible, the client will see the console message:

NFS server *host* **not responding still trying.**

Depending on whether the file system has been mounted "hard" or "soft" (see **mount**(8)), the client will either continue (forever) to resend the request until it receives an acknowledgement from the server, or return an error to user-level. For hard mounts, this means the server can crash or power down and come back up without any special action required by the client. If the "intr" mount option was not specified, a client process requesting I/O will block and remain insensitive to signals, sleeping inside the kernel at **PRIBIO** until the request is satisfied.

FILES

/etc/exports

SEE ALSO

mount(2V), **exports**(5), **fstab**(5), **fstab**(5), **exportfs**(8), **mount**(8), **nfsd**(8), **sticky**(8)

Network Interfaces Programmer's Guide

NAME

null – data sink

CONFIG

None; included with standard system.

SYNOPSIS

#include <fcntl.h>

open("/dev/null", *mode*);

DESCRIPTION

Data written on the **null** special file is discarded.

Reads from the **null** special file always return an end-of-file indication.

FILES

/dev/null

NAME

openprom – PROM monitor configuration interface

CONFIG

pseudo-device openeepr

SYNOPSIS

#include <fcntl.h>
#include <sys/types.h>
#include <sundev/openpromio.h>
open("/dev/openprom", mode);

AVAILABILITY

Desktop SPARCsystems and SPARCsystem 600MP Series only.

DESCRIPTION

As with other Sun systems, configuration options are stored in an EEPROM or NVRAM on desktop SPARCsystems and SPARCsystem 600MP series machines. However, unlike other Sun systems, the encoding of these options is private to the PROM monitor. The **openprom** device provides an interface to the PROM monitor allowing a user program to query and set these configuration options through the use of **ioctl**(2) requests. These requests are defined in **<sundev/openpromio.h>**:

```
struct openpromio {
        u_int   oprom_size;         /* real size of following array */
        char    oprom_array[1];           /* For property names and values
                                  /* NB: Adjacent, Null terminated */
};
#define OPROMMAXPARAM   1024        /* max size of array */

/*
 * Note that all OPROM ioctl codes are type void. Since the amount
 * of data copied in/out may (and does) vary, the openprom driver
 * handles the copyin/copyout itself.
 */
#define OPROMGETOPT         _IO(O, 1)
#define OPROMNXTOPT         _IO(O, 3)
#define OPROMSETOPT2        _IO(O, 4)
```

For all **ioctl( )** requests, the third parameter is a pointer to a '**struct openpromio**'. All property names and values are null-terminated strings; the value of a numeric option is its ASCII representation.

IOCTLS

The **OPROMGETOPT** ioctl takes the null-terminated name of a property in the *oprom_array* and returns its null-terminated value (overlaying its name). *oprom_size* should be set to the size of *oprom_array*; on return it will contain the size of the returned value. If the named property does not exist, or if there is not enough space to hold its value, then *oprom_size* will be set to zero. See BUGS below.

The **OPROMSETOPT2** ioctl takes two adjacent strings in *oprom_array*: the null-terminated property name followed by the null-terminated value.

The **OPROMNXTOPT** ioctl is used to retrieve properties sequentially. The null-terminated name of a property is placed into *oprom_array* and on return it is replaced with the null-terminated name of the next property in the sequence, with *oprom_size* set to its length. A null string on input means return the name of the first property; an *oprom_size* of zero on output means there are no more properties.

ERRORS

EINVAL	The size value was invalid, or (for **OPROMSETOPT**) the property does not exist.
ENOMEM	The kernel could not allocate space to copy the user's structure

FILES

/dev/openprom	PROM monitor configuration interface

SEE ALSO

mem(4S), **eeprom**(8S), **monitor**(8S), **openboot**(8S)

BUGS

There should be separate return values for non-existent properties as opposed to not enough space for the value.

The driver should be more consistent in its treatment of errors and edge conditions.

NAME

PCFS – MS-DOS formatted filesystem

CONFIG

options PCFS

AVAILABILITY

Available only on Sun-3/80, and Desktop SPARCsystems with internal floppy drives.

DESCRIPTION

PCFS is a filesystem type that allows users direct access to files on MS-DOS formatted disks from within the SunOS operating system. Once mounted, a PCFS filesystem provides standard SunOS file operations and semantics. That is, users can create, delete, read, write files on an MS-DOS formatted disk. They can also create/delete directories and list files in a directory.

PCFS filesystems are mounted either with the command:

mount –t pcfs *device-special directory-name*

or

mount /pcfs

if the following line

/dev/fd0 /pcfs pcfs rw,noauto 0 0

is in your **/etc/fstab.**

File and directories created through PCFS have to comply with the MS-DOS file name convention, which is of the form *filename*[*.ext*], where *filename* consists of one through eight upper-case characters, while the optional *ext* consists of one through three upper-case characters. PCFS converts all the lower-case characters in a file name to upper-case, and chops off any extra characters in *filename* or *ext*. When displaying file names, PCFS only shows them in lower-case.

One can use either the MS-DOS **FORMAT** command, or

fdformat –d

command in the SunOS system to format a diskette in MS-DOS format.

EXAMPLES

If you copy a file

financial.data

from a UNIX filesystem to a PCFS filesystem, it will show up as

FINANCIA.DAT

on the MS-DOS disk.

The following file names

.login

test.sh.orig

data+

are considered illegal in MS-DOS, therefore can not be created through **PCFS**.

NOTES

The following are all the legal characters that are allowed in file names or extentions in PCFS:

0-9, a-z, A-Z, and $#&@!%()-{}<>‘_\^~|’

Since **SunOS** and **DOS** operating systems use different character sets, and have different requirements for the text file format, one can use

dos2unix

or

unix2dos

command to convert files between them.

PCFS offers a convenient transportation vehicle for files between Sun Workstations and PC's. Since the MS-DOS disk format was designed for use under DOS, it is quite inefficient to operate under the SunOS system. Therefore, it should not be used as the format for a regular local storage. You should use **ufs** for local storage within the SunOS system.

FILES

/usr/etc/mount_pcfs

SEE ALSO

dos(1), **dos2unix**(1), **eject**(1), **fd**(4), **fdformat(1), unix2dos**(1), **fstab**(5), **mount**(8)

DIAGNOSTICS

mount(8) will fail and produce the following message:

mount_pcfs: /dev/fd0 on /pcfs type pcfs: Invalid argument

if the floppy in the **fd0** drive is not in MS-DOS format. For example, if you try to mount a **ufs** formatted floppy as a PCFS filesystem.

mount_pcfs: /dev/fd0 on /pcfs type pcfs: No such device

if configuration option **options PCFS** is missing from the kernel.

WARNINGS

It is not recommended to physically eject an MS-DOS floppy while the device is still mounted as a PCFS filesystem.

Since PCFS truncates any extra characters in file names and extentions like MS-DOS, be careful when copying files from a UNIX filesystem to a PCFS filesystem. For instance, the following two files

test.data1 test.data2

in a UNIX filesystem will get copied to the same file

TEST.DAT

in PCFS.

BUGS

PCFS should handle the disk change condition like MS-DOS, so that the user does not need to unmount the filesystem to change floppies. PCFS is currently not NFS mountable. Trying to mount a PCFS filesystem through NFS will fail with an EACCES error.

NAME

pp – Centronics-compatible parallel printer port

CONFIG — Sun386i SYSTEMS

device pp0 at obio ? csr 0x378 irq 15 priority 2

CONFIG — SUN-3x SYSTEMS

device pp0 at obio ? csr 0x6f000000 priority 1

This synopsis line should be used to generate a kernel for Sun-3/80 systems only.

AVAILABILITY

Sun386i and Sun-3/80 systems only.

DESCRIPTION

This device driver provides an interface to the Sun386i and Sun-3/80 systems' on-board Centronics-compatible parallel printer port. It supports most standard PC printers with Centronics interfaces.

FILES

/dev/pp0

DIAGNOSTICS

pp*: printer not online

pp*: printer out of paper

NAME

sd – driver for SCSI disk devices

CONFIG — SUN-3, SUN-3x, and SUN-4 SYSTEMS

controller si0 at vme24d16 ? csr 0x200000 priority 2 vector siintr 0x40
controller si0 at obio ? csr 0x140000 priority 2
disk sd0 at si0 drive 0 flags 0
disk sd1 at si0 drive 1 flags 0
disk sd2 at si0 drive 8 flags 0
disk sd3 at si0 drive 9 flags 0
disk sd4 at si0 drive 16 flags 0
disk sd6 at si0 drive 24 flags 0

controller sc0 at vme24d16 ? csr 0x200000 priority 2 vector scintr 0x40
disk sd0 at sc0 drive 0 flags 0
disk sd1 at sc0 drive 1 flags 0
disk sd2 at sc0 drive 8 flags 0
disk sd3 at sc0 drive 9 flags 0
disk sd4 at sc0 drive 16 flags 0
disk sd6 at sc0 drive 24 flags 0

The first two **controller** lines above specify the first and second SCSI host adapters for Sun-3, Sun-3x, and Sun-4 VME systems. The third **controller** line specifies the first and only SCSI host adapter on Sun-3/50 and Sun-3/60 systems.

The four lines following the **controller** specification lines define the available **disk** devices, **sd0** – **sd6**.

The **flags** field is used to specify the SCSI device type to the host adapter. **flags** must be set to 0 to identify **disk** devices.

The **drive** value is calculated using the formula:

$8 * target + lun$

where *target* is the SCSI target, and *lun* is the SCSI logical unit number.

The next configuration block, following **si0** and **si1** above, describes the configuration for the older **sc0** host adapter. It uses the same configuration description as the **si0** host adapter.

CONFIG — SPARCsystem 330 and SUN-3/80 SYSTEMS

controller sm0 at obio ? csr 0xfa000000 priority 2
disk sd0 at sm0 drive 0 flags 0
disk sd1 at sm0 drive 1 flags 0
disk sd2 at sm0 drive 8 flags 0
disk sd3 at sm0 drive 9 flags 0
disk sd4 at sm0 drive 16 flags 0
disk sd6 at sm0 drive 24 flags 0

The SPARCsystem 330 and Sun-3/80 use an on-board SCSI host adapter, **sm0**. It follows the same rules as described above for the Sun-3, Sun-3x, and Sun-4 section.

CONFIG — SUN-4/110 SYSTEM

controller sw0 at obio 2 csr 0xa000000 priority 2
disk sd0 at sw0 drive 0 flags 0
disk sd1 at sw0 drive 1 flags 0
disk sd2 at sw0 drive 8 flags 0
disk sd3 at sw0 drive 9 flags 0
disk sd4 at sw0 drive 16 flags 0
disk sd6 at sw0 drive 24 flags 0

The Sun-4/110 uses an on-board SCSI host adapter, **sw0**. It follows the same rules as described above for the Sun-3, Sun-3x, and Sun-4 section.

CONFIG — SUN-3/E SYSTEM

controller se0 at vme24d16 ? csr 0x300000 priority 2 vector se_intr 0x40
disk sd0 at se0 drive 0 flags 0
disk sd1 at se0 drive 1 flags 0
disk sd2 at se0 drive 8 flags 0
disk sd3 at se0 drive 9 flags 0

The Sun-3/E uses a VME-based SCSI host adapter, **se0**. It follows the same rules as described above for the Sun-3, Sun-3x, and Sun-4 section.

CONFIG — Sun386i

controller wds0 at obmem ? csr 0xFB000000 dmachan 7 irq 16 priority 2
disk sd0 at wds0 drive 0 flags 0
disk sd1 at wds0 drive 8 flags 0
disk sd2 at wds0 drive 16 flags 0

The Sun386i configuration follows the same rules described above for the Sun-3, Sun-3x, and Sun-4 section.

CONFIG — Desktop SPARCsystems

device-driver esp
scsibus0 at esp
disk sd0 at scsibus0 target 3 lun 0
disk sd1 at scsibus0 target 1 lun 0
disk sd2 at scsibus0 target 2 lun 0
disk sd3 at scsibus0 target 0 lun 0

The Desktop SPARCsystem configuration files specify a device driver (**esp**), and a SCSI bus attached to that device driver, and then disks on that SCSI bus at the SCSI Target and Logical Unit addresses specified.

CONFIG — SPARCsystem 600MP SERIES

device-driver esp
scsibus0 at esp
disk sd0 at scsibus0 target 3 lun 0
disk sd1 at scsibus0 target 1 lun 0
disk sd2 at scsibus0 target 2 lun 0
disk sd3 at scsibus0 target 0 lun 0
disk sd16 at scsibus0 target 5 lun 0

scsibus1 at esp
disk sd4 at scsibus1 target 3 lun 0
disk sd5 at scsibus1 target 1 lun 0
disk sd6 at scsibus1 target 2 lun 0
disk sd7 at scsibus1 target 0 lun 0

scsibus2 at esp
disk sd8 at scsibus2 target 3 lun 0
disk sd9 at scsibus2 target 1 lun 0
disk sd10 at scsibus2 target 2 lun 0
disk sd11 at scsibus2 target 0 lun 0

scsibus3 at esp
disk sd12 at scsibus3 target 3 lun 0
disk sd13 at scsibus3 target 1 lun 0
disk sd14 at scsibus3 target 2 lun 0

disk sd15 at scsibus3 target 0 lun 0

scsibus4 at esp
disk sd16 at scsibus4 target 3 lun 0
disk sd17 at scsibus4 target 1 lun 0
disk sd18 at scsibus4 target 2 lun 0
disk sd19 at scsibus4 target 0 lun 0

SPARCsystem 600MP series configuration files specify a device driver (**esp**), SCSI buses attached to that device driver, and then disks on those SCSI buses at the SCSI Target and Logical Unit addresses specified.

DESCRIPTION

Files with minor device numbers 0 through 7 refer to various portions of drive 0. The standard device names begin with "**sd**" followed by the drive number and then a letter a-h for partitions 0-7 respectively. The character **?** stands here for a drive number in the range 0-20.

The block-files access the disk using the system's normal buffering mechanism and are read and written without regard to physical disk records. There is also a "raw" interface that provides for direct transmission between the disk and the user's read or write buffer. A single read or write call usually results in one I/O operation; raw I/O is therefore considerably more efficient when many bytes are transmitted. The names of the raw files conventionally begin with an extra '**r**.'

I/O requests (such as **lseek (2V)**) to the SCSI disk must have an offset that is a multiple of 512 bytes (DEV_BSIZE), or the driver returns an EINVAL error. If the transfer length is not a multiple of 512 bytes, the transfer count is rounded up by the driver.

Disk Support

This driver handles the Adaptec ACB-4000 disk controller for ST-506 drives, the Emulex MD21 disk controller for ESDI drives, and embedded, CCS-compatible SCSI disk drives.

On Sun386i and Desktop SPARCsystems, this driver supports the CDC Wren III half-height, and Wren IV full-height SCSI disk drives. The SPARCsystem 600MP series also supports the Seagate Elite SCSI drive.

The type of disk drive is determined using the SCSI inquiry command and reading the volume label stored on block 0 of the drive. The volume label describes the disk geometry and partitioning; it must be present or the disk cannot be mounted by the system.

The **sd?a** partition is normally used for the root file system on a disk, the **sd?b** partition as a paging area (e.g. swap), and the **sd?c** partition for pack-pack copying. **sd?c** normally maps the entire disk and may also be used as the mount point for secondary disks in the system. The rest of the disk is normally the **sd?g** partition. For the primary disk, the user file system is located here.

FILES

/dev/sd[0-20][a-h]	block files
/dev/rsd[0-20][a-h]	raw files

SEE ALSO

dkio(4S), **directory**(3V), **lseek**(2V), **read**(2V), **write**(2V)

Product Specification for Wren IV SCSI Model 94171
Product Specification for Wren III SCSI Model 94161
Product Specification for Wren III SCSI Model 94211
Emulex MD21 Disk Controller Programmer Reference Manual
Adaptec ACB-4000 Disk Controller OEM Manual

DIAGNOSTICS

sd?: sdtimer: I/O request timeout

A tape I/O operation has taken too long to complete. A device or host adapter failure may have occurred.

sd?: sdtimer: can't abort request
The driver is unable to find the request in the disconnect queue to notify the device driver that it has failed.

sd?: no space for inquiry data
sd?: no space for disk label
The driver was unable to get enough space for temporary storage. The driver is unable to open the disk device.

sd?: <%s>
The driver has found a SCSI disk device and opened it for the first time. The disk label is displayed to notify the user.

sd?: SCSI bus failure
A host adapter error was detected. The system may need to be rebooted.

sd?: single sector I/O failed
The driver attempted to recover from a transfer by writing each sector, one at a time, and failed. The disk needs to be reformatted to map out the new defect causing this error.

sd?: retry failed
sd?: rezero failed
A disk operation failed. The driver first tries to recover by retrying the command; if that fails, the driver rezeros the heads to cylinder 0 and repeats the retries. A failure of either the retry or rezero operations results in these warning messages; the error recovery operation continues until the retry count is exhausted. At that time a hard error is posted.

sd?: request sense failed
The driver was attempting to determine the cause of an I/O failure and was unable to get more information. This implies that the disk device may have failed.

sd?: warning, abs. block %d has failed %d times
The driver is warning the user that the specified block has failed repeatedly.

sd?: block %d needs mapping
sd?: reassigning defective abs. block %d
The specified block has failed repeatedly and may soon become an unrecoverable failure. If the driver does not map out the specified block automatically, it is recommended that the user correct the problem.

sd?: reassign block failed
The driver attempted to map out a block having excessive soft errors and failed. The user needs to run format and repair the disk.

sd?%c: cmd how blk %d (rel. blk %d)
sense key(0x%x): %s, error code(0x%x): %s
An I/O operation (**cmd**), encountered an error condition at absolute block (**blk %d**), partition (**sd?%c:**), or relative block (**rel. block%d**). The error recovery operation (**how**) indicates whether it *retry*'ed, *restored*, or *failed*. The **sense key** and **error code** of the error are displayed for diagnostic purposes. The absolute **blk** of the error is used for mapping out the defective block. The **rel. blk** is the block (sector) in error, relative to the beginning of the partition involved. This is useful for using **icheck**(8) to repair a damaged file structure on the disk.

Additional SPARCsystem Diagnostics
The diagnostics for desktop SPARCsystems and the SPARCsystem 600MP series are much like those above. The following diagnostics are unique to these systems:

sd?: SCSI transport failed: reason 'xxxx': {retrying|giving up}
The host adapter has failed to transport a command to the target for the reason stated. The

driver will either retry the command or, ultimately, give up.

sd?: disk not responding to selection
The target disk isn't responding. You may have accidently kicked a power cord loose.

sd?: disk ok
The target disk is now responding again.

sd?: disk offline
The driver has decided that the target disk is no longer there.

BUGS

These disk drivers assume that you don't have removable media drives, and also that in order to operate normally, a valid Sun disk label must be in sector zero.

A logical block size of 512 bytes is assumed (and enforced on desktop SPARCsystems).

NAME

sockio – ioctls that operate directly on sockets

SYNOPSIS

#include <sys/sockio.h>

DESCRIPTION

The IOCTL's listed in this manual page apply directly to sockets, independent of any underlying protocol. Note: the **setsockopt** system call (see **getsockopt**(2)) is the primary method for operating on sockets as such, rather than on the underlying protocol or network interface. **ioctl**s for a specific network interface or protocol are documented in the manual page for that interface or protocol.

SIOCSPGRP The argument is a pointer to an **int**. Set the process-group ID that will subsequently receive **SIGIO** or **SIGURG** signals for the socket referred to by the descriptor passed to **ioctl** to the value of that **int**.

SIOCGPGRP The argument is a pointer to an **int**. Set the value of that **int** to the process-group ID that is receiving **SIGIO** or **SIGURG** signals for the socket referred to by the descriptor passed to **ioctl**.

SIOCCATMARK The argument is a pointer to an **int**. Set the value of that **int** to 1 if the read pointer for the socket referred to by the descriptor passed to **ioctl** points to a mark in the data stream for an out-of-band message, and to 0 if it does not point to a mark.

SEE ALSO

ioctl(2), **getsockopt**(2), **filio**(4)

NAME

sr – driver for CDROM SCSI controller

CONFIG — Desktop SPARCsystems

disk sr0 at scsibus0 target 6 lun 0

CONFIG — SPARCsystem 600MP SERIES

disk sr0 at scsibus0 target 6 lun 0
disk sr1 at scsibus0 target 5 lun 0
disk sr2 at scsibus0 target 1 lun 0
disk sr3 at scsibus0 target 0 lun 0
disk sr4 at scsibus1 target 6 lun 0
disk sr5 at scsibus3 target 6 lun 0

CONFIG — SUN-4/330 SYSTEMS

disk sr0 at sm0 drive 060 flags 2

CONFIG — SUN-4 SYSTEMS

disk sr0 at sc0 drive 060 flags 2
disk sr0 at si0 drive 060 flags 2

DESCRIPTION

CDROM is a removable read-only direct-access device connected to the system's SCSI bus. **CDROM** drives are designed to work with any disc that meets the Sony-Philips "red-book" or "yellow-book" documents. They can read **CDROM** data discs, digital audio discs (Audio CD's) or combined-mode discs (that is, some tracks are audio, some tracks are data). A **CDROM** disc is single-sided containing approximately 540 megabytes of data or 74 minutes of audio.

The first **CDROM** drive controller in a system is set up as SCSI target 6, with logical unit number 0. Addressing for additional controllers, when supported, is given above. For the first **CDROM** drive in a system, device names are **/dev/sr0** for block device and **/dev/rsr0** for character device. Additional drives, when supported, are designated **/dev/sr1** for block device and **/dev/rsr1** for character device, and so forth.

The device driver supports **open**(2V), **read**(2V), **close**(2V) function calls through its block device and character device interface. In addition, it supports ioctl function calls through the character device interface. When the device is first opened, the **CDROM** drive's eject button will be disabled (which prevents the manual removal of the disc) until the last **close**(2V) is called.

CDROM Drive Support

This driver supports the SONY CDU-8012 **CDROM** drive controller and other **CDROM** drives which have the same SCSI command set as the SONY CDU-8012. The type of **CDROM** drive is determined using the SCSI inquiry command.

There is no volume label stored on the **CDROM**. The disc geometry and paritioning information is always the same. The minor device number is always 0. If the **CDROM** is in ISO 9660 or **High Sierra Disk** format, it can be mounted as a file system.

FILES

/dev/sr[0-5]	block files
/dev/rsr[0-5]	raw files

SEE ALSO

cdromio(4S), **fstab**(5), **mount**(8)

NAME

st – driver for SCSI tape devices

CONFIG — SUN-3, SUN-3/400, SUN-4, SPARCsystem 400 SERIES

controller si0 at vme24d16 ? csr 0x200000 priority 2 vector siintr 0x40
controller si1 at vme24d16 ? csr 0x204000 priority 2 vector siintr 0x41
controller si0 at obio ? csr 0x140000 priority 2
tape st0 at si0 drive 040 flags 1
tape st1 at si0 drive 050 flags 1
tape st2 at si0 drive 030 flags 1
tape st3 at si0 drive 020 flags 1
tape st4 at si1 drive 040 flags 1
tape st5 at si1 drive 050 flags 1
tape st6 at si1 drive 030 flags 1
tape st7 at si1 drive 020 flags 1

controller sc0 at vme24d16 ? csr 0x200000 priority 2 vector scintr 0x40
tape st0 at sc0 drive 040 flags 1
tape st1 at sc0 drive 050 flags 1

The first two **controller** lines above specify the first and second SCSI host adapters for Sun-3, Sun-3/400, Sun-4, and SPARCsystem 400 VME systems. The third **controller** line specifies the first and only SCSI host adapter on Sun-3/50 and Sun-3/60 systems.

Following the **controller** specification lines are eight lines which define the available **tape** devices, **st0–st7**. The first four **tape** devices, **st0–st3** are on the first **controller**, **si0**. The next four **tape** devices, **st4–st7** are on the second **controller**, **si1**. For small configurations, only the first controller, **si0**, is used. For larger configurations, a second controller, **si1**, is added.

The **flags** field is used to specify the SCSI device type to the host adapter. The **flags** field must be set to 1 to identify **tape** devices.

The **drive** value is calculated using the formula: SCSI bus, to select a specific target and logical unit number. The value to enter in the **drive** field of the configuration line is calculated using the formula:

$8 * target + lun$

where *target* is the SCSI target, and *lun* is the SCSI logical unit number. Note the drive number is displayed in octal and not decimal as was done in earlier releases. For boot proms, this unit number must be specified in hex rather than decimal; for example, **st4** is unit 20, not 32 (decimal) or 040 (octal).

The next configuration block, following **si0** and **si1** above, describes the older **sc0** host adapter configuration. It follows the same configuration description as the **si0** host adapter.

CONFIG — SPARCsystem 300 SERIES

controller sm0 at obio ? csr 0x66000000 priority 2
controller si1 at vme24d16 ? csr 0x204000 priority 2 vector siintr 0x41
tape st0 at sm0 drive 040 flags 1
tape st1 at sm0 drive 050 flags 1
tape st2 at sm0 drive 030 flags 1
tape st3 at sm0 drive 020 flags 1
tape st4 at si1 drive 040 flags 1
tape st5 at si1 drive 050 flags 1
tape st6 at si1 drive 030 flags 1
tape st7 at si1 drive 020 flags 1

For small configurations, only the on-board controller, **sm0**, is used. For larger configurations, a second controller, **si1**, is added. This configuration follows the same rules described above in the first CONFIG section.

CONFIG — Desktop SPARCsystems

scsibus0 at esp
scsibus1 at esp
tape st0 at scsibus0 target 4 lun 0
tape st1 at scsibus0 target 5 lun 0
tape st2 at scsibus1 target 4 lun 0
tape st3 at scsibus1 target 5 lun 0

The desktop SPARCsystems configuration files specify a device driver, **esp**, and the SCSI buses attached to that device driver. The first and second tape devices (**st0** and **st1**) are attached on **scsibus0**, and the second and third tape devices (**st2** and **st3**) are attached on **scsibus1**. These devices are attached at the Target and Logical Unit addresses specified.

CONFIG — SPARCsystem 600MP SERIES

scsibus0 at esp
tape st0 at scsibus0 target 4 lun 0
tape st1 at scsibus0 target 5 lun 0
tape st2 at scsibus0 target 1 lun 0
tape st3 at scsibus0 target 0 lun 0

scsibus1 at esp
tape st4 at scsibus1 target 4 lun 0
tape st5 at scsibus1 target 5 lun 0

scsibus2 at esp

scsibus3 at esp
tape st6 at scsibus3 target 4 lun 0
tape st7 at scsibus3 target 5 lun 0

scsibus4 at esp

Like the desktop SPARCsystems, the SPARCsystem 600MP series configuration files specify a device driver, **esp**, and the SCSI buses attached to that device driver. The first through fourth tape devices (**st0**, **st1**, **st2**, and **st3**) are attached on **scsibus0**, the fifth and sixth tape devices (**st4** and **st5**) are attached on **scsibus1**, and the seventh and eighth tape devices (**st6** and **st7**) are attached on **scsibus3**. These devices are attached at the Target and Logical Unit addresses specified.

CONFIG — SUN–3/80 SYSTEMS

controller sm0 at obio ? csr 0x66000000 priority 2
tape st0 at sm0 drive 040 flags 1
tape st1 at sm0 drive 050 flags 1
tape st3 at sm0 drive 030 flags 1
tape st4 at sm0 drive 020 flags 1

Sun-3/80 systems use an on-board SCSI host adapter, **sm0**, which follows the same rules described above in the first CONFIG section.

CONFIG — SUN–4/110 SYSTEMS

controller sw0 at obio 2 csr 0xa000000 priority 2
tape st0 at sw0 drive 040 flags 1
tape st1 at sw0 drive 050 flags 1

The Sun-4/110 uses an on-board SCSI host adapter, **sw0**, which follows the rules described above in the first CONFIG section.

CONFIG — SUN–3/E SYSTEMS

controller se0 at vme24d16 ? csr 0x300000 priority 2 vector se_intr 0x40
tape st0 at se0 drive 040 flags 1
tape st1 at se0 drive 050 flags 1

The Sun-3/E uses a VME-based SCSI host adapter, **se0**, which follows the rules described above for the first CONFIG section.

CONFIG — Sun386i SYSTEMS

controller wds0 at obmem ? csr 0xFB000000 dmachan 7 irq 16 priority 2
tape st0 at wds0 drive 32 flags 1

The Sun386*i* configuration follows the rules described above in the first CONFIG section.

DESCRIPTION

The **st** device driver is an interface to various SCSI tape devices. Supported 1/4" cartridge devices include the Archive Viper QIC–150 streaming tape drive, the Emulex MT–02 tape controller, and the Sysgen SC4000 tape controller (except on desktop SPARCsystems and SPARCsystem 600MP series machines). Supported 1/2" and 8mm devices include the HP–88780 1/2" tape drive and the Exabyte EXB-8200/8500 8mm cartridge tape subsystem. **st** provides a standard interface to these various devices; see **mtio**(4) for details.

The driver can be opened with either rewind on close (**/dev/rst***) or no rewind on close (**/dev/nrst***) options. A maximum of four tape formats per device are supported (see FILES below). The tape format is specified using the device name. The four rewind on close formats for **st0**, for example, are **/dev/rst0**, **/dev/rst8**, **/dev/rst16**, and **/dev/rst24**.

Read Operation

If the driver is opened for reading in a different format than the tape is written in, the driver overrides the user selected format. For example, if a 1/4" cartridge tape is written in QIC-24 format and opened for reading in QIC–11, the driver will detect a read failure on the first read and automatically switch to QIC–24 to recover the data.

Note: if the **/dev/*st[0–7]** format is used, no indication is given that the driver has overridden the user selected format. Other formats issue a warning message to inform the user of an overridden format selection. Some devices automatically perform this function and do not require driver support (1/2" reel and QIC–150 tape drives, for example).

Write Operation

Writing from the beginning of tape is performed in the user-specified format. The original tape format is used for appending onto previously written tapes. A warning message is issued if the driver has to override the user-specified format.

EOT Handling

The Emulex and Sysgen drives have only a physical end of tape (PEOT); thus it is not possible to write past EOT. All other drives have a logical end of tape (LEOT) before PEOT to guarantee flushing the data onto the tape. The amount of storage between LEOT and PEOT varies from less than a megabyte to about 20 megabytes depending on the tape drive. Further writing, except for trailer records, is inhibited to prevent running off the end of the reel.

If EOT is encountered while writing an Emulex or Sysgen tape, no error is reported but the number of bytes transferred is zero and no further writing is allowed. On all other drives, the current transfer is completed, returning the number of bytes written. The next write will return zero, at which time it is possible to write trailer records after first writing an EOF.

The Desktop SPARCsystems tape driver differs in EOT handling and supports writing trailer records without first writing an EOF. The first write that encounters EOT will return a short count or zero. If a short count is returned, then the next write will return zero. After a zero count is returned, the next write returns a full count or short count. A following write will return zero again.

It is important that the number and size of trailer records be kept as small as possible to prevent data loss. For this reason, this practice is not recommended.

Reading past logical EOT is transparent to the user. Reading is only stopped by reading EOF's. For 1/2" reel devices, it is possible to read off the end of the reel if you read past the two file marks which mark the end of recorded media. All other devices have safeguards to eliminate this problem.

Ioctls

The behavior of SCSI tape positioning ioctls is the same across all devices which support them. However, not all devices support all ioctls. The driver returns an ENOTTY error on unsupported ioctls.

The retension ioctl only applies to 1/4" cartridge tape devices. It is used to restore tape tension, thus improving the tape's soft error rate after extensive start-stop operations or long-term storage.

Note: the error status is reset by the **MTIOCGET** get status ioctl call or the next read, write, or other ioctl operation. If no error has occurred (sense key is zero), the current file and record position is returned.

ERRORS

EACCES	The driver is opened for write access and the tape is write protected. For writing with QIC–150 tape drives, this error is also reported if the wrong tape media is used for writing.
EBUSY	The tape driver is in use by another process. Only one process can use the tape drive at a time. The driver will allow a grace period of two minutes for the other process to finish before reporting this error.
EINVAL	The number of bytes read or written is not a multiple of the physical record size (fixed-length tape devices only).
EIO	During opening, the tape device is not ready because either no tape is in the drive, or the drive is not on-line. Once open, this error is returned if the requested I/O transfer could not be completed.
ENOTTY	This indicates that the tape device does not support the requested ioctl function.
ENXIO	During opening, the tape device does not exist.
EPERM	Another system has reserved the tape drive for its use. The tape drive cannot be used until the other system releases it.

FILES

For 1/2" reel tape devices (HP-88780):

/dev/rst[0–7]	800 BPI density
/dev/rst[8–15]	1600 BPI density
/dev/rst[16–23]	6250 BPI density
/dev/rst[24–31]	data compression
/dev/nrst[0–7]	non–rewinding 800 BPI density
/dev/nrst[8–15]	non–rewinding 1600 BPI density
/dev/nrst[16–23]	non–rewinding 6250 density
/dev/nrst[24–31]	non–rewinding data compression

For helical-scan tape devices (Exabyte):

/dev/rst[0–7]	Standard EXB-8200 (2GB) Format
/dev/rst[8–15]	EXB-8500 (5GB) Format
/dev/rst[16–23]	Compressed Format (EXB-8500 only)
/dev/rst[24–31]	Compressed Format (EXB-8500 only)
/dev/nrst[0–7]	non-rewinding Standard EXB-8200 (2GB)Format
/dev/nrst[8–15]	non-rewinding EXB-8500 (5GB) Format
/dev/nrst[16–23]	non-rewinding Compressed Format (EXB-8500 only)
/dev/nrst[24–31]	non-rewinding Compressed Format (EXB-8500 only)

For QIC–150 tape devices (Archive Viper):

/dev/rst[0–7]	QIC–150 Format
/dev/rst[8–15]	QIC–150 Format
/dev/rst[16–23]	QIC–150 Format
/dev/rst[24–31]	QIC–150 Format
/dev/nrst[0–7]	non-rewinding QIC–150 Format
/dev/nrst[8–15]	non-rewinding QIC–150 Format
/dev/nrst[16–23]	non-rewinding QIC–150 Format
/dev/nrst[24–31]	non-rewinding QIC–150 Format

Note: The drive will automatically read both QIC–11 and QIC–24 formats too.

For QIC–24 tape devices (Emulex MT–02 and Sysgen SC4000):

/dev/rst[0–7]	QIC–11 Format
/dev/rst[8–15]	QIC–24 Format
/dev/rst[16–23]	QIC–24 Format
/dev/rst[24–31]	QIC–24 Format
/dev/nrst[0–7]	non-rewinding QIC–11 Format
/dev/nrst[8–15]	non-rewinding QIC–24 Format
/dev/nrst[16–23]	non-rewinding QIC–24 Format
/dev/nrst[24–31]	non-rewinding QIC–24 Format

Note: QIC–24 is the preferred format for all systems except Sun-2's. For Sun-2 systems, QIC–11 is preferred.

Note: Only four tape devices were supported in SunOS4.1. Thus only the first four device names above can be used if you are running this release. Releases following 4.1 PSR A (inclusive) support all device names.

SEE ALSO

cpio(1), **mt**(1), **tar**(1), **mtio**(4), **dump**(8), **restore**(8)

DIAGNOSTICS

st?: I/O request timeout
A tape I/O operation has taken too long to complete. A device or host adapter failure may have occurred.

st?: warning, unknown tape drive found
The driver does not recognize the tape device. Only the default tape density is used; block size is set to the value specified by the tape drive.

st?: write protected
The tape is write protected.

st?: wrong tape media for writing, use DC6150 tape (or equivalent)
For QIC–150 tape drives, this indicates that the user is trying to write on a DC–300XL (or equivalent) tape. Only DC–6150 (or equivalent) tapes can be used for writing.
Note: DC–6150 was formerly known as DC–600XTD.

st?: warning, rewinding tape
The driver is rewinding tape in order to set the tape format.

st?: warning, using alternate tape format
The driver is overriding the user-selected tape format and using the previously used format.

st?: warning, tape rewound
For Sysgen tape controllers, the tape may be rewound as a result of getting sense data.

st?: format change failed
The specified tape density or format is not supported by the tape drive. This is detected by the tape drive rejecting the mode select command to change the tape density/format.

st?: warning, The tape may be wearing out or the head may need cleaning.

The number of allowable records re-read or re-written (e.g. soft errors) has exceeded the specified limit for the tape media. This indicates the tape media is wearing out. No data loss has occurred on the tape yet. This message is an early warning that continued usage will result in unrecoverable errors. Continued usage of this tape is not recommended.

This error message can also be issued if the tape heads are dirty. Tape oxide buildup on the head interferes with reading and writing data. If the head has not been cleaned recently, it should be cleaned and the tape retested to determine the source of this error.

If this error message is persistent over different known good tapes, this is indicative of a tape drive hardware problem.

st?: read retries= count (rate %), file= %d, block= %d

st?: write retries= count (rate %), file= %d, block= %d

These messages display the measured soft error rate of the tape at the time the driver is closed. The **count** indicates the number of soft errors. The number of soft errors as a percentage of total blocks transferred is **rate**. The **rate** should remain relatively constant across the tape. Sudden jumps indicate worn spots on the tape. The retry **count** is reset every time the tape is rewound.

Ballpark upper limits for read retries is 3%. For write retries, the upper limit is 10%. The exact specifications depend on the tape device in use.

st?: <cmd> failed

st? error: sense key(0x%x): %s error, code(0x%x): %s

An error has occurred. The first line identifies the command name, **<cmd>**. The second line reports the sense key error code and message string. In addition, the error code and message string may be reported depending on whether this extension is supported by the tape drive. This information is intended for diagnostic purposes.

st?: stread: not modulo %d block size

st?: stwrite: not modulo %d block size

The read or write request size must be a multiple of the **%d** physical block size. Typically, this means a read or write of less than 512 bytes was requested. Note: the block size can vary with the tape drive in use.

st?: file positioning error

st?: block positioning error

The driver was unable to position the tape to the desired file or block (record). This is probably caused by a damaged tape.

st?: tape synchronization lost

The user hit CTRL–C while a tape operation was in progress, causing the tape position to be lost. The tape will be rewound on the next open to resynchronize. Note: other signals may also have a similar effect.

st?: no iopb space for buffer

The driver could not allocate space for its internal working buffer because the kernel is out of resources; too many devices were configured into the system.

Additional SPARCsystem Diagnostics

The diagnostics for desktop SPARCsystems and the < SPARCsystem 600 series are much like those described above. The following diagnostics are unique to these systems:

st?: SCSI transport failed: reason 'xxxx': {retrying|giving up}

The host adapter has failed to transport a command to the target for the reason stated. The driver will either retry the command or, ultimately, give up.

BUGS

Foreign tape devices which do not return a BUSY status during tape loading prevent user commands from being held until the device is ready. The user must delay issuing any tape operations until the tape device is ready. This is not a problem for Sun-supplied tape devices.

Foreign tape devices which do not report a blank check error at the end of recorded media cause file positioning operations to fail. Some tape drives, for example, mistakenly report media error instead of blank check error.

Systems using the older **sc0** host adapter or the Sysgen SC4000 tape controller prevent disk I/O over the SCSI bus while the tape is in use (during a rewind, for example). This problem is caused by the fact that they do not support disconnect/reconnect to free the SCSI bus. Newer tape devices, like the Emulex MT–02, and host adapters, like **si0**, eliminate this problem.

Some older Sysgen drives support only the QIC–11 format. This can be determined using **mt**(1).

WARNINGS

The Sysgen SC4000 tape controller is being obsoleted.

NAME

tmpfs – memory based filesystem

CONFIG

options TMPFS

SYNOPSIS

#include <sys/mount.h>
mount ("tmpfs", dir, M_NEWTYPE | flags, args);

DESCRIPTION

tmpfs is a memory based filesystem which uses kernel resources relating to the VM system and page cache as a filesystem. Once mounted, a **tmpfs** filesystem provides standard file operations and semantics. **tmpfs** is so named because files and directories are not preserved across reboot or unmounts, all files residing on a **tmpfs** filesystem that is unmounted will be lost.

tmpfs filesystems are mounted either with the command:

mount –t tmp swap *directory-name*

or by placing the line

swap *directory-name* **tmp rw 0 0**

in your **/etc/fstab** file and using the **mount**(8) command as normal. The **/etc/rc.local** file contains commands to mount a **tmpfs** filesystem on **/tmp** at multi-user startup time but is by default commented out. To mount a **tmpfs** filesystem on **/tmp** (maximizing possible performance improvements), add the above line to **/etc/fstab** and uncomment the following line in **/etc/rc.local**:

#mount /tmp

tmpfs is designed as a performance enhancement which is achieved by cacheing the writes to files residing on a **tmpfs** filesystem. Performance improvements are most noticeable when a large number of short lived files are written and accessed on a **tmpfs** filesystem. Large compilations with **tmpfs** mounted on **/tmp** are a good example of this.

Users of **tmpfs** should be aware of some tradeoffs involved in mounting a **tmpfs** filesystem. The resources used by **tmpfs** are the same as those used when commands are executed (for example, swap space allocation). This means that a large sized or number of **tmpfs** files can affect the amount of space left over for programs to execute. Likewise, programs requiring large amounts of memory use up the space available to **tmpfs**. Users running into these constraints (for example, running out of space on **tmpfs**) can allocate more swap space by using the **swapon**(8) command.

Normal filesystem writes are scheduled to be written to a permanent storage medium along with all control information associated with the file (for example, modification time, file permissions). **tmpfs** control information resides only in memory and never needs to be written to permanent storage. File data remains in core until memory demands are sufficient to cause pages associated with **tmpfs** to be reused at which time they are copied out to swap.

SEE ALSO

df(1V), **mount**(2V), **umount**(2V), **fstab**(5), **mount**(8), **swapon**(8)

System Services Overview,
System and Network Administration

NOTES

swapon to a **tmpfs** file is not supported. File and record locking is not supported.

df(1V) output is of limited accuracy since a **tmpfs** filesystem size is not static and the space available to **tmpfs** is dependent on the swap space demands of the entire system.

DIAGNOSTICS

If **tmpfs** runs out of space, one of the following messages will be printed to the console.

directory: **file system full, anon reservation exceeded**

directory: **file system full, anon allocation exceeded**

A page could not be allocated while writing to a file. This can occur if **tmpfs** is attempting to write more than it is allowed, or if currently executing programs are using a lot of memory. To make more space available, remove unneccessary files, exit from some programs, or allocate more swap space using **swapon**(8).

directory: **file system full, kmem_alloc failure**

tmpfs ran out of physical memory while attempting to create a new file or directory. Remove unneccesary files or directories or install more physical memory.

WARNINGS

A **tmpfs** filesystem should *not* be mounted on **/var/tmp**, this directory is used by **vi**(1) for preserved files. Files and directories on a **tmpfs** filesystem are not preserved across reboots or unmounts. Command scripts or programs which count on this will not work as expected.

NAME

vx – Sun Visualization Accelerator and Multiprocessor Visualization Accelerator

CONFIG – SUN-4 SYSTEMS

device vx0 at vme32d32 3 csr 0x30000000 priority 4
vector vxintr 0xaa cgvxintr 0xab
device vx0 at vme32d32 4 csr 0x30000000 priority 4
vector vxintr 0xaa cgvxintr 0xab

The first line should be used to generate a kernel for Sun-4/370 and Sun-4/330 systems. The second line should be used to generate a kernel for Sun-4/470 systems.

AVAILABILITY

VX/MVX can only be used in Sun-4 VME-bus packages with 3 or more full size (9U) slots. An MVX can only be used in combination with a VX.

DESCRIPTION

The **vx** interface supports the optional VX/MVX Visualization Accelerator. The VX and MVX are visualization accelerators that provide high-speed computing power for such techniques as image processing, volume rendering, 3-D interactive graphics, and high quality rendering. Each of the devices, the VX and the MVX, is a full size (9U) VME board.

The **VX** is a single-board accelerator with a high-performance processing node, 4Mbytes of program/data memory, and an additional 16 Mbytes of image/data memory. It contains two frame buffers; a low-end graphics accelerator designed to enhance vector and polygon drawing performance, and a full color frame buffer accelerated by the processing node.

The low-end graphics accelerator is nearly identical to the **cgsix** or **GX** device. It has a 2 MByte 8-bit color frame buffer and provides the standard frame buffer interface as defined in **fbio**(4S). As with the **cgsix,** this part of the **VX** has registers and memory that may be mapped with **mmap**(2), using the offsets defined in **<sundev/vxreg.h>**.

The full color frame buffer is 16 MBytes of 32-bit image memory. See the *VX/MVX Programming with the Graphics and Visualization Library* for detailed information on accessing the full color frame buffer and the processing node.

The **MVX** is a multiprocessor accelerator with four parallel processing nodes. Each node is identical to the processing node on the **VX** board and each has 4 Mbytes of program/data memory.

Programs can be downloaded for execution on the VX/MVX directly, they can be executed by the host processor, or the host processor and the VX/MVX engine can be used in combination. See the *VX/MVX Programming with the C Developer's Kit* for detailed information on accessing the VX/MVX from the host. This manual also describes the C compiler, the programming tools, and the support libraries for the VX/MVX.

SEE ALSO

mmap(2), fbio(4S), cgsix(4)

Software Installation for the VX and MVX Visualization Accelerators

Hardware Installation for the VX and MVX Visualization Accelerators

VX/MVX Programming with the C Developer's Kit

VX/MVX Programming with the Graphics and Visualization Library

NAME

zero – source of zeroes

SYNOPSIS

None; included with standard system.

DESCRIPTION

A zero special file is a source of zeroed unnamed memory.

Reads from a zero special file always return a buffer full of zeroes. The file is of infinite length.

Writes to a zero special file are always successful, but the data written is ignored.

Mapping a zero special file creates a zero-initialized unnamed memory object of a length equal to the length of the mapping and rounded up to the nearest page size as returned by **getpagesize**(2). Multiple processes can share such a zero special file object provided a common ancestor mapped the object **MAP_SHARED**.

FILES

/dev/zero

SEE ALSO

fork(2V), **getpagesize**(2), **mmap**(2)

NAME

zs – Zilog 8530 SCC serial communications driver

CONFIG — SUN-3 SYSTEM

device zs0 at obio ? csr 0x20000 flags 3 priority 3
device zs1 at obio ? csr 0x00000 flags 0x103 priority 3

CONFIG — SUN-3x SYSTEM

device zs0 at obio ? csr 0x62002000 flags 3 priority 3
device zs1 at obio ? csr 0x62000000 flags 0x103 priority 3

CONFIG — SUN-4 SYSTEM

device zs1 at obio ? csr 0xf0000000 flags 0x103 priority 3
device zs2 at obio 3 csr 0xe0000000 flags 3 priority 3

CONFIG — Desktop SPARCsystems, SPARCsystem 600MP SERIES

device-driver zs

CONFIG — Sun386i SYSTEM

device zs0 at obmem ? csr 0xFC000000 flags 3 irq 9 priority 6
device zs1 at obmem ? csr 0xA0000020 flags 0x103 irq 9 priority 6

SYNOPSIS

#include <fcntl.h>
#include <sys/termios.h>
open("/dev/tty*n*", mode);
open("/dev/ttyd*n*", mode);
open("/dev/cua*n*", mode);

DESCRIPTION

The Zilog 8530 provides 2 serial communication ports with full modem control in asynchronous mode. Each port supports **termio**(4) device control functions specified by flags in the **c_cflag** word of the **termios** structure, and the functions specified by the **IGNBRK**, **IGNPAR**, **PARMRK**, and **INPCK** flags of the **c_ciflag** word of the **termios** structure. These functions are performed directly by the **zs** driver. All other **termio**(4) functions must be performed by STREAMS modules pushed atop the driver. When a device is opened, the **ldterm**(4M) and **ttcompat**(4M) STREAMS modules are automatically pushed on top of the stream, providing the standard **termio**(4) interface.

Of the synopsis lines above, the line for **zs0** specifies the serial I/O port(s) provided by the CPU board, the line for **zs1** specifies the Video Board ports (which are used for keyboard and mouse), the lines for **zs2** and **zs3** specify the first and second ports on the first SCSI board in a system, and those for **zs4** and **zs5** specify the first and second ports provided by the second SCSI board in a system, respectively.

Bit *i* of **flags** may be specified to say that a line is not properly connected, and that the line *i* should be treated as hard-wired with carrier always present. Thus specifying **flags 0x2** in the specification of **zs0** would treat line **/dev/ttyb** in this way.

Minor device numbers in the range 0 – 11 correspond directly to the normal tty lines and are named **/dev/ttya** and **/dev/ttyb** for the two serial ports on the CPU board and **/dev/ttys***n* for the ports on the SCSI boards; *n* is 0 or 1 for the ports on the first SCSI board, and 2 or 3 for the ports on the second SCSI board.

To allow a single tty line to be connected to a modem and used for both incoming and outgoing calls, a special feature, controlled by the minor device number, has been added. Minor device numbers in the range 128 – 139 correspond to the same physical lines as those above (that is, the same line as the minor device number minus 128).

A dial-in line has a minor device in the range 0 – 11 and is conventionally renamed **/dev/ttyd***n*, where *n* is a number indicating which dial-in line it is (so that **/dev/ttyd0** is the first dial-in line), and the dial-out line corresponding to that dial-in line has a minor device number 128 greater than the minor device number of the dial-in line and is conventionally named **/dev/cua***n*, where *n* is the number of the dial-in line.

The **/dev/cua***n* lines are special in that they can be opened even when there is no carrier on the line. Once a **/dev/cua***n* line is opened, the corresponding tty line cannot be opened until the **/dev/cua***n* line is closed; a blocking open will wait until the **/dev/cua***n* line is closed (which will drop Data Terminal Ready, after which Carrier Detect will usually drop as well) and carrier is detected again, and a non-blocking open will return an error. Also, if the **/dev/ttyd***n* line has been opened successfully (usually only when carrier is recognized on the modem) the corresponding **/dev/cua***n* line can not be opened. This allows a modem to be attached, for example, to **/dev/ttyd0** (renamed from **/dev/ttya**) and used for dial-in (by enabling the line for login in **/etc/ttytab**) and also used for dial-out (by **tip**(1C) or **uucp**(1C)) as **/dev/cua0** when no one is logged in on the line. Note: the bit in the **flags** word in the configuration file (see above) must be zero for this line, which enables hardware carrier detection.

IOCTLS

The standard set of **termio ioctl( )** calls are supported by **zs**.

If the **CRTSCTS** flag in the **c_cflag** is set, output will be generated only if CTS is high; if CTS is low, output will be frozen. If the **CRTSCTS** flag is clear, the state of CTS has no effect. Breaks can be generated by the **TCSBRK**, **TIOCSBRK**, and **TIOCCBRK ioctl( )** calls. The modem control lines **TIOCM_CAR**, **TIOCM_CTS**, **TIOCM_RTS**, and **TIOCM_DTR** are provided.

The input and output line speeds may be set to any of the speeds supported by **termio**. The speeds cannot be set independently; when the output speed is set, the input speed is set to the same speed.

ERRORS

An **open( )** will fail if:

ENXIO	The unit being opened does not exist.
EBUSY	The dial-out device is being opened and the dial-in device is already open, or the dial-in device is being opened with a no-delay open and the dial-out device is already open.
EBUSY	The unit has been marked as exclusive-use by another process with a **TIOCEXCL ioctl( )** call.
EINTR	The open was interrupted by the delivery of a signal.

FILES

/dev/tty{a,b,s[0-3]}	hardwired tty lines
/dev/ttyd[0-9a-f]	dial-in tty lines
/dev/cua[0-9a-f]	dial-out tty lines

SEE ALSO

tip(1C), **uucp**(1C), **mcp**(4S), **mti**(4S), **termio**(4), **ldterm**(4M), **ttcompat**(4M), **ttysoftcar**(8)

DIAGNOSTICS

zs*n c* **: silo overflow.**

The 8530 character input silo overflowed before it could be serviced.

zs*n c* **: ring buffer overflow.**

The driver's character input ring buffer overflowed before it could be serviced.

NAME

intro – file formats used or read by various programs

DESCRIPTION

This section describes formats of files used by various programs.

A 5V section number means one or more of the following:

- The man page documents System V formats only.
- The man page documents default SunOS formats, and System V formats as they differ from the default formats. These System V differences are presented under **SYSTEM V** section headers.
- The man page documents formats compliant with *IEEE Std 1003.1-1988* (POSIX.1).

LIST OF FILE FORMATS

Name	Appears on Page	Description
acct	**acct**(5)	execution accounting file
addresses	**aliases**(5)	addresses and aliases for sendmail
aliases	**aliases**(5)	addresses and aliases for sendmail
a.out	**a.out**(5)	assembler and link editor output format
ar	**ar**(5)	archive (library) file format
audit_control	**audit_control**(5)	control information for system audit daemon
audit_data	**audit_data**(5)	current information on audit daemon
audit.log	**audit.log**(5)	the security audit trail file
auto.home	**auto.home**(5)	automount map for home directories
auto.vol	**auto.vol**(5)	automount map for volumes
bar	**bar**(5)	tape archive file format
boards.pc	**boards.pc**(5)	AT- and XT-compatible boards for DOS windows
bootparams	**bootparams**(5)	boot parameter data base
bootservers	**bootservers**(5)	NIS bootservers file
coff	**coff**(5)	common assembler and link editor output
core	**core**(5)	format of memory image file
cpio	**cpio**(5)	format of cpio archive
crontab	**crontab**(5)	table of times to run periodic jobs
dir	**dir**(5)	format of directories
dump	**dump**(5)	incremental dump format
dumpdates	**dump**(5)	incremental dump format
environ	**environ**(5V)	user environment
ethers	**ethers**(5)	Ethernet address to hostname database or NIS domain
exports	**exports**(5)	directories to export to NFS clients
ext_ports	**ext_ports**(5)	external ports file for network printers, terminals, and modems
fbtab	**fbtab**(5)	framebuffer table
fcntl	**fcntl**(5)	file control options
filetype	**filetype**(5)	DeskSet binding database file
forward	**aliases**(5)	addresses and aliases for sendmail
fs	**fs**(5)	format of a 4.2 (ufs) file system volume
fspec	**fspec**(5)	format specification in text files
fstab	**fstab**(5)	static filesystem mounting table, mounted filesystems table
ftpusers	**ftpusers**(5)	list of users prohibited by FTP
gettytab	**gettytab**(5)	terminal configuration data base
group	**group**(5)	group file
group.adjunct	**group.adjunct**(5)	group security data file
help	**help**(5)	help file format
help_viewer	**help_viewer**(5)	help viewer file format
hosts	**hosts**(5)	host name data base

hosts.equiv	**hosts.equiv**(5)	trusted hosts by system and by user
indent.pro	**indent.pro**(5)	default options for indent
inetd.conf	**inetd.conf**(5)	Internet servers database
inode	**fs**(5)	format of a 4.2 (ufs) file system volume
internat	**internat**(5)	key mapping table for internationalization
ipalloc.netrange	**ipalloc.netrange**(5)	range of addresses to allocate
keytables	**keytables**(5)	keyboard table descriptions for loadkeys and dumpkeys
lastlog	**utmp**(5V)	login records
link	**link**(5)	link editor interfaces
locale	**locale**(5)	locale database
magic	**magic**(5)	file command's magic number file
mtab	**fstab**(5)	static filesystem mounting table, mounted filesystems table
mtab	**mtab**(5)	mounted file system table
netgroup	**netgroup**(5)	list of network groups
netmasks	**netmasks**(5)	network mask data base
netrc	**netrc**(5)	file for ftp remote login data
networks	**networks**(5)	network name data base
orgrc	**orgrc**(5)	organizer configuration and initialization file
passwd	**passwd**(5)	password file
passwd.adjunct	**passwd.adjunct**(5)	user security data file
phones	**phones**(5)	remote host phone number data base
plot	**plot**(5)	graphics interface
pnp.sysnames	**pnp.sysnames**(5)	file used to allocate system names
policies	**policies**(5)	network administration policies
printcap	**printcap**(5)	printer capability data base
proto	**proto**(5)	prototype job file for at
protocols	**protocols**(5)	protocol name data base
publickey	**publickey**(5)	public key database
queuedefs	**queuedefs**(5)	queue description file for at, batch, and cron
rasterfile	**rasterfile**(5)	Sun's file format for raster images
remote	**remote**(5)	remote host description file
resolv.conf	**resolv.conf**(5)	configuration file for domain name system resolver
rfmaster	**rfmaster**(5)	Remote File Sharing name server master file
rgb	**rgb**(5)	available colors (by name) for coloredit
rhosts	**hosts.equiv**(5)	trusted hosts by system and by user
rmtab	**rmtab**(5)	remote mounted file system table
rootmenu	**rootmenu**(5)	root menu specification for SunView
rpc	**rpc**(5)	rpc program number data base
sccsfile	**sccsfile**(5)	format of an SCCS history file
services	**services**(5)	Internet services and aliases
setup.pc	**setup.pc**(5)	master configuration file for DOS
sm	**sm**(5)	in.statd directory and file structures
sm	**statmon**(5)	statd directories and file structures
sm.bak	**sm**(5)	in.statd directory and file structures
sm.bak	**statmon**(5)	statd directories and file structures
sm.state	**sm**(5)	in.statd directory and file structures
state	**statmon**(5)	statd directories and file structures
sunview	**sunview**(5)	initialization file for SunView
svdtab	**svdtab**(5)	SunView device table
syslog.conf	**syslog.conf**(5)	configuration file for syslogd system log daemon
systems	**systems**(5)	NIS systems file
tar	**tar**(5)	tape archive file format
termcap	**termcap**(5)	terminal capability data base

term	**term**(5)	terminal driving tables for nroff
term	**term**(5V)	format of compiled term file
terminfo	**terminfo**(5V)	terminal capability data base
toc	**toc**(5)	table of contents of optional clusters
translate	**translate**(5)	input and output files for system message translation
ttys	**ttytab**(5)	terminal initialization data
ttytab	**ttytab**(5)	terminal initialization data
types	**types**(5)	primitive system data types
tzfile	**tzfile**(5)	time zone information
ugid_alloc.range	**ugid_alloc.range**(5)	range of user IDs and group IDs to allocate
updaters	**updaters**(5)	configuration file for NIS updating
utmp	**utmp**(5V)	login records
uuencode	**uuencode**(5)	format of an encoded uuencode file
vfont	**vfont**(5)	font formats
vgrindefs	**vgrindefs**(5)	vgrind's language definition data base
wtmp	**utmp**(5V)	login records
xtab	**exports**(5)	directories to export to NFS clients
ypaliases	**ypaliases**(5)	NIS aliases for sendmail
ypfiles	**ypfiles**(5)	NIS database and directory structure
ypgroup	**ypgroup**(5)	NIS group file
yppasswd	**yppasswd**(5)	NIS password file
ypprintcap	**ypprintcap**(5)	NIS printer capability database

NAME

a.out – assembler and link editor output format

SYNOPSIS

```
#include <a.out.h>
#include <stab.h>
#include <nlist.h>
```

AVAILABILITY

Sun-2, Sun-3, and Sun-4 systems only. For Sun386i systems refer to **coff**(5).

DESCRIPTION

a.out is the output format of the assembler **as**(1) and the link editor **ld**(1). The link editor makes **a.out** executable files.

A file in **a.out** format consists of: a header, the program text, program data, text and data relocation information, a symbol table, and a string table (in that order). In the header, the sizes of each section are given in bytes. The last three sections may be absent if the program was loaded with the **–s** option of **ld** or if the symbols and relocation have been removed by **strip**(1).

The machine type in the header indicates the type of hardware on which the object code can be executed. Sun-2 code runs on Sun-3 systems, but not vice versa. Program files predating release 3.0 are recognized by a machine type of '**0**'. Sun-4 code may not be run on Sun-2 or Sun-3, nor vice versa.

Header

The header consists of a **exec** structure. The **exec** structure has the form:

```
struct exec {
        unsigned char   a_dynamic:1;    /* has a __DYNAMIC */
        unsigned char   a_toolversion:7; /* version of toolset used to create this file */
        unsigned char   a_machtype;     /* machine type */
        unsigned short  a_magic;        /* magic number */
        unsigned long   a_text;         /* size of text segment */
        unsigned long   a_data;         /* size of initialized data */
        unsigned long   a_bss;          /* size of uninitialized data */
        unsigned long   a_syms;         /* size of symbol table */
        unsigned long   a_entry;        /* entry point */
        unsigned long   a_trsize;       /* size of text relocation */
        unsigned long   a_drsize;       /* size of data relocation */
};
```

The members of the structure are:

a_dynamic **1** if the **a.out** file is dynamically linked or is a shared object, **0** otherwise.

a_toolversion The version number of the toolset (**as**, **ld**, etc.) used to create the file.

a_machtype One of the following:

- **0** pre-3.0 executable image
- **M_68010** executable image using only MC68010 instructions that can run on Sun-2 or Sun-3 systems.
- **M_68020** executable image using MC68020 instructions that can run only on Sun-3 systems.
- **M_SPARC** executable image using SPARC instructions that can run only on Sun-4 systems.

a_magic One of the following:

- **OMAGIC** An text executable image which is not to be write-protected, so the data segment is immediately contiguous with the text segment.

NAME

ethers – Ethernet address to hostname database or NIS domain

DESCRIPTION

The **ethers** file contains information regarding the known (48 bit) Ethernet addresses of hosts on the Internet. For each host on an Ethernet, a single line should be present with the following information:

Ethernet-address official-host-name

Items are separated by any number of blanks and/or TAB characters. A '#' indicates the beginning of a comment extending to the end of line.

The standard form for Ethernet addresses is ''*x*:*x*:*x*:*x*:*x*:*x*'' where *x* is a hexadecimal number between 0 and ff, representing one byte. The address bytes are always in network order. Host names may contain any printable character other than a SPACE, TAB, NEWLINE, or comment character. It is intended that host names in the **ethers** file correspond to the host names in the **hosts**(5) file.

The **ether_line**( ) routine from the Ethernet address manipulation library, **ethers**(3N) may be used to scan lines of the **ethers** file.

FILES

/etc/ethers

SEE ALSO

ethers(3N), **hosts**(5)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

exports, xtab – directories to export to NFS clients

SYNOPSIS

/etc/exports

/etc/xtab

DESCRIPTION

The **/etc/exports** file contains entries for directories that can be exported to NFS clients. This file is read automatically by the **exportfs**(8) command. If you change this file, you must run **exportfs**(8) for the changes to affect the daemon's operation.

Only when this file is present at boot time does the **rc.local** script execute **exportfs**(8) and start the NFS file-system daemon, **nfsd**(8).

The **/etc/xtab** file contains entries for directories that are *currently* exported. This file should only be accessed by programs using **getexportent( )** (see **exportent**(3)). Use the **–u** option of **exportfs** to remove entries from this file.

An entry for a directory consists of a line of the following form:

directory *–option*[**,** *option*] . . .

directory is the pathname of a directory (or file).

option is one of the following:

ro Export the directory read-only. If not specified, the directory is exported read-write.

rw=*hostnames***[:***hostname***]...**
Export the directory read-mostly. Read-mostly means read-only to most machines, but read-write to those specified. If not specified, the directory is exported read-write to all.

anon=*uid*
If a request comes from an unknown user, use *uid* as the effective user ID. Note: root users (uid 0) are always considered "unknown" by the NFS server, unless they are included in the "root" option below. The default value for this option is the UID of the user "nobody". If the user "nobody" does not exist then the value 65534 is used. Setting the value of "anon" to 65535 disables anonymous access. Note: by default secure NFS accepts insecure requests as anonymous, and those wishing for extra security can disable this feature by setting "anon" to 65534.

root=*hostnames***[:***hostname***]...**
Give root access only to the root users from a specified *hostname*. The default is for no hosts to be granted root access.

access=*client***[:***client***]...**
Give mount access to each *client* listed. A *client* can either be a hostname, or a netgroup (see **netgroup**(5)). Each *client* in the list is first checked for in the **/etc/hosts** database, and then the **/etc/netgroups** database. The default value allows any machine to mount the given directory.

secure Require clients to use a more secure protocol when accessing the directory.

A '**#**' (pound-sign) anywhere in the file indicates a comment that extends to the end of the line.

EXAMPLE

```
/usr          -access=clients              # export to my clients
/usr/local                                 # export to the world
/usr2         -access=hermes:zip:tutorial  # export to only these machines
/usr/sun      -root=hermes:zip             # give root access only to these
/usr/new      -anon=0                      # give all machines root access
/usr/bin      -ro                          # export read-only to everyone
/usr/stuff    -access=zip,anon=-3,ro       # several options on one line
```

FILES

/etc/exports
/etc/xtab
/etc/hosts
/etc/netgroup
rc.local

SEE ALSO

exportent(3), **hosts**(5), **netgroup**(5), **exportfs**(8), **nfsd**(8)

WARNINGS

You cannot export either a parent directory or a subdirectory of an exported directory that is *within the same filesystem*. It would be illegal, for instance, to export both **/usr** and **/usr/local** if both directories resided on the same disk partition.

NAME

ext_ports – external ports file for network printers, terminals, and modems

SYNOPSIS

/etc/ext_ports

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

The **ext_ports** external ports file is an ASCII file in the **/etc** directory on the Network Information Service (NIS) master server. **ext_ports** is used only by SNAP, and contains basic information about each printer, terminal, and modem on the network. This file contains a one-line entry for each device, and each field *must* be separated by a TAB character:

system:port type status baud model name #comment

system names the system to which the device is attached. This field contains only lower case and numeric characters, must start with a lower case character, and must not be longer than 32 characters.

port names the port in **/dev** on the *system*: **ttya** for the Sun386i serial port, **pp0** for the parallel port, and **ttym0** and **ttym1** for ports on an AT bus serial card.

type **printer**, **terminal**, or **modem**.

status indicates the device status. For terminals and printers, this can be **on** or **off**. An **off** status means the device is disabled from access by the SunOS operating system, but can still be accessed by DOS. For modems, this can be **in** to enable dialin, **out** to enable dialout, **in_out** to enable dialin and dialout, or **off**. An **off** status means the device is disabled from access by the SunOS operating system, but it can still be accessed by DOS.

baud is the baud rate.

model indicates the manufacturer or kind of device. For printers, this can be **epson**, **hp**, or **text**, for Epson and compatibles, HP Laserjet and compatibles, or for text-only printers. For terminals, this can be **vt100** or **wyse-50** for DEC VT-100 and compatibles or for Wyse WY-50 and compatibles. For modems, this can be **hayes** for Hayes and compatibles.

name is only used for unique naming of printers on the network. Up to 16 characters can be entered. This field is blank for terminals and modems — simply insert a TAB character.

#comment

can contain anything you want, up to a maximum of 96 characters.

EXAMPLE

In this example of an **ext_ports** file, the system vulcan has an epson printer attached to its parallel port, and a Wyse-50 terminal attached to its serial port, but with logins currently disabled. The system android has a VT100 attached to its serial port, with logins enabled. The system polaris has a **hayes** modem set for dialing out on an installed AT bus serial card.

```
vulcan:pp0      printer   on      9600   epson      lp   #Engineering lab
android:ttya    terminal  on      9600   vt100           #Reception
vulcan:ttya     terminal  off     9600   wyse-50         #Engineering lab
polaris:ttym0   modem     in_out  2400   hayes           #QA lab
```

FILES

/etc/ext_ports

NAME

fcntl – file control options

SYNOPSIS

#include <fcntl.h>

DESCRIPTION

The **fcntl**(2V) function provides for control over open files. This include file describes *requests* and *arguments* to **fcntl** and **open**(2V) as shown below:

```
/*              @ (#)fcntl.h 1.2 83/12/08 SMI; from UCB 4.2 83/09/25*/
/*
* Flag values accessible to open(2V) and fcntl(2)
* (The first three can only be set by open)
*/
#define         O_RDONLY                0
#define         O_WRONLY                1
#define         O_RDWR                  2
#define         O_NDELAY                FNDELAY         /* Non-blocking I/O */
#define         O_APPEND                FAPPEND         /* append (writes guaranteed at the end) */
#ifndef         F_DUPFD
/* fcntl(2) requests */
#define         F_DUPFD                 0               /* Duplicate fildes */
#define         F_GETFD                 1               /* Get fildes flags */
#define         F_SETFD                 2               /* Set fildes flags */
#define         F_GETFL                 3               /* Get file flags */
#define         F_SETFL                 4               /* Set file flags */
#define         F_GETOWN                5               /* Get owner */
#define         F_SETOWN                6               /* Set owner */
/* flags for F_GETFL, F_SETFL— copied from <sys/file.h> */
#define         FNDELAY                                 00004/* non-blocking reads */
#define         FAPPEND                                 00010/* append on each write */
#define         FASYNC                                  00100/* signal pgrp when data ready */
#endif
```

SEE ALSO

fcntl(2V), **open**(2V)

NAME

filetype – DeskSet binding database file

SYNOPSIS

/etc/filetype

DESCRIPTION

The **filetype** file is a database file used by both the OpenWindows and SunView DeskSet tools.

The file describes bindings between types of files. That is, it contains entries defining what type of action should be done on a type of file. Specifically, it binds an icon and color to a file type, and defines what method is used to invoke and print a given filetype.

Files may be described in one of two ways: by a name pattern (such as ***.ras**) or by their magic number (as defined in the **/etc/magic** file).

/etc/filetype is the system default database file created by running the **install_deskset** script (for the SunView DeskSet), or **install_filemgr** script (for the OpenWindows DeskSet).

$HOME/.filetype is the user's own personal binding database file, which can be created and modified using the binder tool.

The DeskSet will first use the user's personal **$HOME/.filetype** file if it is found; else it will use the system defaults defined in **/etc/filetype**. These two files are mutually exclusive; entries are not shared between them.

These files are not meant to be modified by hand; the DeskSet tool binder should always be used to modify them. Remember to start the binder as root if you wish to modify the system's defaults in **/etc/filetype**.

FILES

/etc/magic
/etc/filetype
$HOME/.filetype

NOTES

These files will only be used for SunView DeskSet 1.0 & 1.1 and OpenWindows DeskSet 2.0. A new way of storing and retrieving binding information will be used in future OpenWindows DeskSet tools.

NAME

fs, inode – format of a 4.2 (ufs) file system volume

SYNOPSIS

#include <sys/types.h>
#include <ufs/fs.h>
#include <ufs/inode.h>

DESCRIPTION

Standard 4.2 (ufs) file system storage volumes have a common format for certain vital information. Every such volume is divided into a certain number of blocks. The block size is a parameter of the file system. Sectors 0 to 15 contain primary and secondary bootstrapping programs.

The actual file system begins at sector 16 with the *super-block*. The layout of the super block is defined by the include file **<ufs/fs.h>**

Each disk drive contains some number of file systems. A file system consists of a number of cylinder groups. Each cylinder group contains inodes and data.

A file system is described by its super-block, which in turn describes the cylinder groups. The super-block is critical data and is replicated in each cylinder group to protect against catastrophic loss. This is done at file system creation time and the critical super-block data does not change, so the copies need not be referenced further unless disaster strikes.

Addresses stored in inodes are capable of addressing fragments of "blocks." File system blocks of at most size **MAXBSIZE** can be optionally broken into 2, 4, or 8 pieces, each of which is addressable; these pieces may be **DEV_BSIZE**, or some multiple of a **DEV_BSIZE** unit.

Large files consist of exclusively large data blocks. To avoid undue wasted disk space, the last data block of a small file is allocated as only as many fragments of a large block as are necessary. The file system format retains only a single pointer to such a fragment, which is a piece of a single large block that has been divided. The size of such a fragment is determinable from information in the inode, using the '**blksize(fs, ip, lbn)**' macro.

The file system records space availability at the fragment level; to determine block availability, aligned fragments are examined.

The root inode is the root of the file system. Inode 0 cannot be used for normal purposes and historically bad blocks were linked to inode 1, thus the root inode is 2 (inode 1 is no longer used for this purpose, however numerous dump tapes make this assumption, so we are stuck with it). The *lost+found* directory is given the next available inode when it is initially created by **mkfs**(8).

fs_minfree gives the minimum acceptable percentage of file system blocks which may be free. If the freelist drops below this level only the super-user may continue to allocate blocks. This may be set to 0 if no reserve of free blocks is deemed necessary, however severe performance degradations will be observed if the file system is run at greater than 90% full; thus the default value of **fs_minfree** is 10%.

Empirically the best trade-off between block fragmentation and overall disk utilization at a loading of 90% comes with a fragmentation of 4, thus the default fragment size is a fourth of the block size.

Cylinder group related limits: Each cylinder keeps track of the availability of blocks at different rotational positions, so that sequential blocks can be laid out with minimum rotational latency. **fs_nrpos** is the number of rotational positions which are distinguished. With the default **fs_nrpos** of 8 the resolution of the summary information is 2ms for a typical 3600 rpm drive.

fs_rotdelay gives the minimum number of milliseconds to initiate another disk transfer on the same cylinder. It is used in determining the rotationally optimal layout for disk blocks within a file; the default value for **fs_rotdelay** varies from drive to drive, see **tunefs**(8).

fs_maxcontig gives the maximum number of blocks, belonging to one file, that will be allocated contiguously before inserting a rotational delay.

Each file system has a statically allocated number of inodes. An inode is allocated for each **NBPI** bytes of disk space. The inode allocation strategy is extremely conservative.

MINBSIZE is the smallest allowable block size. With a **MINBSIZE** of 4096 it is possible to create files of size 2^32 with only two levels of indirection. **MINBSIZE** must be big enough to hold a cylinder group block, thus changes to **(struct cg)** must keep its size within **MINBSIZE.** Note: super blocks are never more than size **SBSIZE.**

The path name on which the file system is mounted is maintained in **fs_fsmnt**. **MAXMNTLEN** defines the amount of space allocated in the super block for this name. The limit on the amount of summary information per file system is defined by **MAXCSBUFS.** It is currently parameterized for a maximum of two million cylinders.

Per cylinder group information is summarized in blocks allocated from the first cylinder group's data blocks. These blocks are read in from **fs_csaddr** (size **fs_cssize**) in addition to the super block.

Note: sizeof **(struct csum)** must be a power of two in order for the **fs_cs** macro to work.

inode: The inode is the focus of all file activity in the file system. There is a unique inode allocated for each active file, each current directory, each mounted-on file, text file, and the root. An inode is "named" by its device/i-number pair. For further information, see the include file **<ufs/inode.h>**.

SEE ALSO

mkfs(8), **newfs**(8), and **tunefs**(8).

NAME

fspec – format specification in text files

DESCRIPTION

It is sometimes convenient to maintain text files on the operating system with non-standard tab stop settings, (that is, tab stops that are not set at every eighth column). Such files must generally be converted to a standard format, frequently by replacing all TAB characters with the appropriate number of SPACE characters, before they can be processed by operating system commands. A format specification occurring in the first line of a text file specifies how TAB characters are to be expanded in the remainder of the file.

A format specification consists of a sequence of parameters separated by blanks and surrounded by the brackets **<:** and **:>**. Each parameter consists of a keyletter, possibly followed immediately by a value. The following parameters are recognized:

t *tabs* The **t** parameter specifies the tab stop settings for the file. The value of *tabs* must be one of the following:

- A list of column numbers separated by commas, indicating tab stops set at the specified columns;
- A '–' followed immediately by an integer *n*, indicating tab stops set at intervals of *n* columns, that is, at 1+*n*, 1+2*n*, and so on;
- A '–' followed by the name of a ''canned'' tab stop specification.

Up to 40 numbers are allowed in a comma-separated list of tab stop settings. If any number (except the first one) is preceded by a plus sign, it is taken as an increment to be added to the previous value. Thus, the formats **t1, 10, 20, 30** and **t1, 10, +10, +10** are considered identical.

Standard tab stops are specified by **t–8**, or equivalently, **t1, 9, 17, 25,** etc. This is the tab stop setting that most operating system utilities assume, and is the most likely setting to be found at a terminal. The specification **t–0** specifies no tab stops at all.

The ''canned'' tab stops specifications that are recognized are as follows:

a 1, 10, 16, 36, 72
Assembler, IBM S/370, first format

a2 1, 10, 16, 40, 72
Assembler, IBM S/370, second format

c 1, 8, 12, 16, 20, 55
COBOL, normal format

c2 1, 6, 10, 14, 49
COBOL compact format (columns 1-6 omitted). Using this code, the first typed character corresponds to card column 7, one space gets you to column 8, and a TAB reaches column 12. Files using this tab stop setup should include a format specification as follows:

<:t–c2 m6 s66 d:>

c3 1, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62, 67
COBOL compact format (columns 1-6 omitted), with more tab stops than **c2**. This is the recommended format for COBOL. The appropriate format specification is:

<:t–c3 m6 s66 d:>

f 1, 7, 11, 15, 19, 23
FORTRAN

p 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61
PL/I

s 1, 10, 55
SNOBOL

u 1, 12, 20, 44
UNIVAC 1100 Assembler

s *size* The **s** parameter specifies a maximum line size. The value of **size** must be an integer. Size checking is performed after TAB characters have been expanded, but before the margin is prepended.

m *margin*
The **m** parameter specifies a number of SPACE characters to be prepended to each line. The value of *margin* must be an integer.

d The **d** parameter takes no value. Its presence indicates that the line containing the format specification is to be deleted from the converted file.

e The **e** parameter takes no value. Its presence indicates that the current format is to prevail only until another format specification is encountered in the file.

Default values, which are assumed for parameters not supplied, are **t–8** and **m0**. If the **s** parameter is not specified, no size checking is performed. If the first line of a file does not contain a format specification, the above defaults are assumed for the entire file. The following is an example of a line containing a format specification:

*** <:t5,10,15 s72:> ***

If a format specification can be disguised as a comment, it is not necessary to code the **d** parameter.

SEE ALSO

ed(1), **tabs**(1V)

NAME

fstab, mtab – static file system mounting table, mounted file systems table

SYNOPSIS

/etc/fstab

/etc/mtab

DESCRIPTION

The **/etc/fstab** file contains entries for file systems and disk partitions to mount using the **mount**(8) command, which is normally invoked by the **rc.boot** script at boot time. This file is used by various utilities that mount, unmount, check the consistency of, dump, and restore file systems. It is also used by the system itself when locating the swap partition.

The **/etc/mtab** file contains entries for file systems *currently* mounted, and is read by programs using the routines described in **getmntent**(3). **umount**(8) removes entries from this file; **mount** adds entries to this file.

Each entry consists of a line of the form:

filesystem directory type options freq pass

filesystem is the pathname of a block-special device, the name of a remote file system in *host*:*pathname* form, or the name of a "swap file" made with **mkfile**(8).

directory is the pathname of the directory on which to mount the file system.

type is the file system type, which can be one of:

- **4.2** to mount a block-special device
- **lo** to loopback-mount a file system
- **nfs** to mount an exported NFS file system
- **swap** to indicate a swap partition
- **ignore** to have the **mount** command ignore the current entry (good for noting disk partitions that are not being used)
- **rfs** to mount an RFS file system
- **tmp** file system in virtual memory
- **hsfs** to mount an ISO 9660 Standard or High Sierra Standard with Rock Ridge extensions CD-ROM file system

options contains a comma-separated list (no spaces) of mounting options, some of which can be applied to all types of file systems, and others which only apply to specific types.

4.2 options:

quota | noquota Disk quotas are enforced or not enforced. The default is **noquota**.

nfs options:

- **bg | fg** If the first attempt fails, retry in the background, or, in the foreground.
- **noquota** Prevent **quota**(1) from checking whether the user is over quota on this file system; if the file system has quotas enabled on the server, quotas will still be checked for operations on this file system.
- **retry=***n* The number of times to retry the mount operation.
- **rsize=***n* Set the read buffer size to *n* bytes.
- **wsize=***n* Set the write buffer size to *n* bytes.
- **timeo=***n* Set the NFS timeout to *n* tenths of a second.
- **retrans=***n* The number of NFS retransmissions.
- **port=***n* The server IP port number.
- **soft | hard** Return an error if the server does not respond, or continue the retry request until the server responds.

intr Allow keyboard interrupts on hard mounts.
secure Use a more secure protocol for NFS transactions.
acregmin=*n*
Hold cached attributes for at least *n* seconds after file modification.
acregmax=*n*
Hold cached attributes for no more than *n* seconds after file modification.
acdirmin=*n*
Hold cached attributes for at least *n* seconds after directory update.
acdirmax=*n*
Hold cached attributes for no more than *n* seconds after directory update.
actimeo=*n*
Set *min* and *max* times for regular files and directories to *n* seconds.
noac Suppress attribute caching.

Regular defaults are:
fg,retry=10000,timeo=7,retrans=3,port=NFS_PORT,hard,
acregmin=3,acregmax=60,acdirmin=30,acdirmax=60

actimeo has no default; it sets **acregmin**, **acregmax**, **acdirmin** and **acdirmax**

Defaults for **rsize** and **wsize** are set internally by the system kernel.

rfs options:
bg | fg If the first attempt fails, retry in the background, or, in the foreground.
retry=*n* The number of times to retry the mount operation.

Defaults are the same as for NFS.

hsfs options:
norrip Disable processing of Rock Ridge extensions for the file system.

Common options:

ro | rw mount either read-only or read-write
suid | nosuid
setuid execution allowed or disallowed
grpid Create files with BSD semantics for propagation of the group ID. With this option, files inherit the group ID of the directory in which they are created, regardless of the directory's setgid bit.
noauto Do not mount this file system automatically (using '**mount –a**').

freq is the interval (in days) between dumps.

pass indicates whether **fsck**(8) should check the partition. File systems with *pass* 0 are not checked. When preening the file systems in **/etc/fstab**, **fsck**(8) automatically overlaps file system checks by simultaneously running one process per disk. If run in "force" mode (**–f**), **fsck** checks file systems with pass 1 sequentially, then overlaps the remainder of the file systems checks. In general, only the root (/) and /usr file systems need to be checked in pass 1, with others checked in the second pass.

A hash-sign (**#**) as the first character indicates a comment line which is ignored by routines that read this file. The order of records in **/etc/fstab** is important because **fsck**, **mount**, and **umount** process the file sequentially; an entry for a file system must appear *after* the entry for any file system it is to be mounted on top of.

EXAMPLES

In this example, two partitions on the local disk are **4.2** mounted. Several **/export** directories are loopback mounted to appear in the traditional file system locations on the local system. The **/home/user** directory is hard mounted read-write over the NFS, along with additional swap space in the form of a mounted swap file (see *System and Network Administration* for details on adding swap space):

```
/dev/xy0a / 4.2 rw,noquota 1 1
/dev/xy0b /usr 4.2 rw,noquota 1 1
/export/tmp/localhost /tmp lo rw 0 0
/export/var/localhost /var lo rw 0 0
/export/cluster/sun386.sunos4.0.1 /usr/cluster lo rw 0 0
/export/local/sun386 /usr/local lo rw 0 0
example:/home/user /home/user nfs rw,hard,fg 0 0
/export/swap/myswap swap swap rw 0 0
```

FILES

/etc/fstab
/etc/mtab

SEE ALSO

swapon(2), **getmntent**(3), **lofs**(4S), **fsck**(8), **mkfile**(8), **mount**(8), **quotacheck**(8), **quotaon**(8), **swapon**(8)

System and Network Administration

NAME

rgb – available colors (by name) for coloredit

SYNOPSIS

.rgb

$HOME/.rgb

/usr/lib/.rgb

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

.rgb is an ASCII file containing consecutive lines terminated by newlines. Each line starts with three integers, each in the range 0-255. These integers are the RGB equivalent for the color named on the same line. At least one tab character delimits the last integer from the name field. The coloreditor searches for this file, first in the current directory; next, in the users home directory; and finally, in **/usr/lib**. The user can add to or delete from the **.rgb** file that he or she has access to, thus changing the available color table for subsequent invocations of coloredit.

EXAMPLES

The following is an example of a **.rgb** file.

0 0 0	**Black**
0 0 255	**Blue**
95 159 159	**Cadet Blue**
66 66 111	**Cornflower Blue**
107 35 142	**Dark Slate Blue**

SEE ALSO

coloredit(1)

NAME

rmtab – remote mounted file system table

SYNOPSIS

/etc/rmtab

DESCRIPTION

rmtab resides in the **/etc** directory, and contains a table of filesystems that are remotely mounted by NFS clients. This file is maintained by **rpc.mountd**(8), the mount daemon. The data in this file should only be obtained from **rpc.mountd**(8) using the **MOUTPROC_DUMP** remote procedure call.

The file contains a line of information for each remotely mounted filesystem. There are a number of lines of the form:

hostname:*fsname*

The mount daemon adds an entry for any client that successfully executes a mount request and deletes the appropriate entries for an unmount request.

Lines beginning with a '#' are commented out. These lines are removed from the file by **rpc.mountd**(8) when it first starts up. Stale entries may accumulate for clients that crash without sending an unmount request.

FILES

/etc/rmtab

SEE ALSO

rpc.mountd(8), showmount(8)

NAME

rootmenu – root menu specification for SunView

SYNOPSIS

~/.rootmenu
/usr/lib/.rootmenu

DESCRIPTION

If a **.rootmenu** file is present in a user's home directory, it specifies the SunView menu, the menu that appears when the user clicks and holds the right mouse button in the background of the SunView desktop. If a **.rootmenu** file is not present in the user's home directory, **/usr/lib/.rootmenu** specifies the SunView menu.

Each line of a **.rootmenu** file has the format:

menu item *command*

menu item can be a character string, or an icon file delimited by angle brackets:

<icon-filename>

If *menu item* is a character string with embedded spaces, it must be enclosed by double quotes ('"').

command can be a command line to be executed when the menu item is selected, or one of the following reserved-word commands:

EXIT	Exit **sunview** (requires confirmation).
REFRESH	Redraw the entire screen.
MENU	This menu item is a pull-right item with a submenu. If a full pathname follows the **MENU** command, the submenu contents are taken from that file. Otherwise, all the lines between a **MENU** command and a matching **END** command are added to the submenu.
END	Mark the end of a nested submenu. The left side of this line should match the left side of a line with a **MENU** command.

If *command* is not one of the reserved-word commands, it is treated as a command line, although no shell interpretation is done.

Lines beginning with a '**#**' character are considered comments and are ignored.

If a user's **.rootmenu** file is modified, the SunView menu immediately reflects the changes.

See **sunview**(1) for more details about **.rootmenu**.

EXAMPLES

The following is a sample **.rootmenu** file:

```
#
#       sample root menu
#
"Lock Screen"           lockscreen
Tools   MENU
        Perfmeter               perfmeter
        Calculator              calc
        Mailtool                mailtool
Tools   END
"ShellTool"             shelltool
"CommandTool"           cmdtool
"Console"               cmdtool -C
#"MailTool"             mailtool
"TextEditor"            textedit
```

NAME

intro – introduction to system maintenance and operation commands

DESCRIPTION

This section contains information related to system bootstrapping, operation and maintenance. It describes all the server processes and daemons that run on the system, as well as standalone (PROM monitor) programs.

An 8V section number means one or more of the following:

- The man page documents System V behavior only.
- The man page documents default SunOS behavior, and System V behavior as it differs from the default behavior. These System V differences are presented under **SYSTEM V** section headers.
- The man page documents behavior compliant with *IEEE Std 1003.1-1988* (POSIX.1).

Disk formatting and labeling is done by **format**(8S). Bootstrapping of the system is described in **boot**(8S), **openboot**(8S) and **init**(8). The standard set of commands run by the system when it boots is described in **rc**(8). Related commands include those that check the consistency of file systems, **fsck**(8); those that mount and unmount file systems, **mount**(8); add swap devices, **swapon**(8); force completion of outstanding file system I/O, **sync**(2); shutdown or reboot a running system **shutdown**(8), **halt**(8), and **reboot**(8); and, set the time on a machine from the time on another machine **rdate**(8C).

Creation of file systems is discussed in **mkfs**(8) and **newfs**(8). File system performance parameters can be adjusted with **tunefs**(8). File system backups and restores are described in **dump**(8) and **restore**(8).

Procedures for adding new users to a system are described in **adduser**(8), using **vipw**(8) to lock the password file during editing. **panic**(8S) which describes what happens when the system crashes, **savecore**(8) which can be used to analyze system crash dumps. Occasionally useful as adjuncts to the **fsck**(8) file system repair program are **clri**(8), **dcheck**(8), **icheck**(8), and **ncheck**(8).

Configuring a new version of the kernel requires using the program **config**(8); major system bootstraps often require the use of **mkproto**(8). New devices are added to the **/dev** directory (once device drivers are configured into the system) using **makedev**(8) and **mknod**(8). The **installboot**(8S) command can be used to install freshly compiled programs. The **catman**(8) command preformats the on-line manual pages.

Resource accounting is enabled by the **accton** command, and summarized by **sa**(8). Login time accounting is performed by **ac**(8). Disk quotas are managed using **quot**(8), **quotacheck**(8), **quotaon**(8), and **repquota**(8).

A number of servers and daemon processes are described in this section. The **update**(8) daemon forces delayed file system I/O to occur and **cron**(8) runs periodic events (such as removing temporary files from the disk periodically). The **syslogd**(8) daemon maintains the system error log. The **init**(8) process is the initial process created when the system boots. It manages the reboot process and creates the initial login prompts on the various system terminals, using **getty**(8). The Internet super-server **inetd**(8C) invokes all other internet servers as needed. These servers include the remote shell servers **rshd**(8C) and **rexecd**(8C), the remote login server **rlogind**(8C), the FTP and TELNET daemons **ftpd**(8C), and **telnetd**(8C), the TFTP daemon **tftpd**(8C), and the mail arrival notification daemon **comsat**(8C). Other network daemons include the 'load average/who is logged in' daemon **rwhod**(8C), the routing daemon **routed**(8C), and the mail daemon **sendmail**(8).

If network protocols are being debugged, then the protocol debugging trace program **trpt**(8C) is often useful. Remote magnetic tape access is provided by **rsh** and **rmt**(8C). Remote line printer access is provided by **lpd**(8), and control over the various print queues is provided by **lpc**(8). Printer cost-accounting is done through **pac**(8).

Network host tables may be gotten from the ARPA NIC using **gettable**(8C) and converted to UNIX-system-usable format using **htable**(8).

RPC and NFS daemons

RPC and NFS daemons include:

portmap	used by RPC based services.
ypbind	used by the Network Information Service (NIS) to locate the NIS server. Note: the Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.
biod	used by NFS clients to read ahead to, and write behind from, network file systems.
nfsd	the NFS server process that responds to NFS requests on NFS server machines.
ypserv	the NIS server, typically run on each NFS server.
rstatd	the server counterpart of the remote speedometer tools.
mountd	the **mount** server that runs on NFS server machines and responds to requests by other machines to mount file systems.
rwalld	used for broadcasting messages over the network.

LIST OF MAINTENANCE COMMANDS

Name	Appears on Page	Description
ac	**ac**(8)	login accounting
acctcms	**acctcms**(8)	command summary from per-process accounting records
acctcon1	**acctcon**(8)	connect-time accounting
acctcon2	**acctcon**(8)	connect-time accounting
acctdisk	**acct**(8)	miscellaneous accounting commands
acctdusg	**acct**(8)	miscellaneous accounting commands
acctmerg	**acctmerg**(8)	merge or add total accounting files
accton	**acct**(8)	miscellaneous accounting commands
accton	**sa**(8)	system accounting
acctprc1	**acctprc**(8)	process accounting
acctprc2	**acctprc**(8)	process accounting
acctwtmp	**acct**(8)	miscellaneous accounting commands
adbgen	**adbgen**(8)	generate adb script
add_client	**add_client**(8)	create a diskless network bootable NFS client on a server
add_services	**add_services**(8)	provide software installation services for any architecture
add_user	**add_user**(8)	shellscript to add a user account to a machine
adduser	**adduser**(8)	procedure for adding new users
adv	**adv**(8)	advertise a directory for remote access with RFS
analyze	**old-analyze**(8)	postmortem system crash analyzer
arp	**arp**(8C)	address resolution display and control
audit	**audit**(8)	audit trail maintenance
auditd	**auditd**(8)	audit daemon
audit_warn	**audit_warn**(8)	audit daemon warning script
automount	**automount**(8)	automatically mount NFS file systems
biod	**nfsd**(8)	NFS daemons
boot	**boot**(8S)	start the system kernel or a standalone program
bootparamd	**bootparamd**(8)	boot parameter server
C2conv	**c2conv**(8)	convert system to or from C2 security
C2unconv	**c2conv**(8)	convert system to or from C2 security
captoinfo	**captoinfo**(8V)	convert a termcap description into a terminfo description
catman	**catman**(8)	create the cat files for the manual
change_login	**change_login**(8)	control screen blanking and choice of login utility
chargefee	**acctsh**(8)	shell procedures for accounting
check4	**set4**(8)	check the virtual address space limit flag in a module
chown	**chown**(8)	change owner
chroot	**chroot**(8)	change root directory for a command
chrtbl	**chrtbl**(8)	generate character classification table

ckpacct	**acctsh**(8)	shell procedures for accounting
client	**client**(8)	add or remove diskless Sun386i systems
clri	**clri**(8)	clear inode
colldef	**colldef**(8)	convert collation sequence source definition
comsat	**comsat**(8C)	biff server
config	**config**(8)	build system configuration files
copy_home	**copy_home**(8)	fetch default startup files for new home directories
crash	**crash**(8)	examine system images
cron	**cron**(8)	clock daemon
dbconfig	**dbconfig**(8)	initializes the dial box
dcheck	**dcheck**(8)	file system directory consistency check
devinfo	**devinfo**(8S)	print out system device information
devnm	**devnm**(8V)	device name
diskusg	**diskusg**(8)	generate disk accounting data by user
dkctl	**dkctl**(8)	control special disk operations
dkinfo	**dkinfo**(8)	report information about a disk's geometry and partitioning
dmesg	**dmesg**(8)	collect system diagnostic messages to form error log
dname	**dname**(8)	print RFS domain and network names
dodisk	**acctsh**(8)	shell procedures for accounting
dorfs	**dorfs**(8)	initialize, start and stop RFS automatically
dump	**dump**(8)	incremental file system dump
dumpfs	**dumpfs**(8)	dump file system information
edquota	**edquota**(8)	edit user quotas
eeprom	**eeprom**(8S)	EEPROM display and load utility
etherd	**etherd**(8C)	Ethernet statistics server
etherfind	**etherfind**(8C)	find packets on Ethernet
exportfs	**exportfs**(8)	export and unexport directories to NFS clients
extract_files	**extract_files**(8)	extract files from release media
extract_unbundled	**extract_unbundled**(8)	extract and execute unbundled-product installation scripts
fastboot	**fastboot**(8)	reboot/halt the system without checking the disks
fasthalt	**fastboot**(8)	reboot/halt the system without checking the disks
fingerd	**fingerd**(8C)	remote user information server
format	**format**(8S)	disk partitioning and maintenance utility
fpa_download	**fpa_download**(8)	download to the Floating Point Accelerator
fparel	**fparel**(8)	Sun FPA online reliability tests
fpaversion	**fpaversion**(8)	print FPA version, load microcode
fpurel	**fpurel**(8)	perform tests the Sun Floating Point Co-processor
fpuversion4	**fpuversion4**(8)	print the Sun-4 FPU version
fsck	**fsck**(8)	file system consistency check and interactive repair
fsirand	**fsirand**(8)	install random inode generation numbers
ftpd	**ftpd**(8C)	TCP/IP Internet File Transfer Protocol server
fumount	**fumount**(8)	force unmount of an advertised RFS resource
fusage	**fusage**(8)	RFS disk access profiler
fuser	**fuser**(8)	identify processes using a file or file structure
fwtmp	**fwtmp**(8)	manipulate connect accounting records
gencat	**installtxt**(8)	create a message archive
gettable	**gettable**(8C)	get DARPA Internet format host table from a host
getty	**getty**(8)	set terminal mode
gid_allocd	**uid_allocd**(8C)	UID and GID allocator daemons
gpconfig	**gpconfig**(8)	initialize the Graphics Processor
grpck	**grpck**(8V)	check group database entries
gtconfig	**gtconfig**(8)	initialize the GT graphics accelerator and download microcode
gt_lpconfig	**gt_lpconfig**(8)	configure GT lightpen parameters

gxtest	**gxtest**(8S)	stand alone test for the Sun video graphics board
halt	**halt**(8)	stop the processor
hostconfig	**hostconfig**(8C)	configure a system's host parameters
hostrfs	**hostrfs**(8)	Convert IP addresses to RFS format
htable	**htable**(8)	convert DoD Internet format host table
icheck	**icheck**(8)	file system storage consistency check
idload	**idload**(8)	RFS user and group mapping
ifconfig	**ifconfig**(8C)	configure network interface parameters
imemtest	**imemtest**(8S)	stand alone memory test
in.comsat	**comsat**(8C)	biff server
inetd	**inetd**(8C)	Internet services daemon
in.fingerd	**fingerd**(8C)	remote user information server
infocmp	**infocmp**(8V)	compare or print out terminfo descriptions
in.ftpd	**ftpd**(8C)	TCP/IP Internet File Transfer Protocol server
init	**init**(8)	process control initialization
in.named	**named**(8C)	Internet domain name server
in.rexecd	**rexecd**(8C)	remote execution server
in.rlogind	**rlogind**(8C)	remote login server
in.routed	**routed**(8C)	network routing daemon
in.rshd	**rshd**(8C)	remote shell server
in.rwhod	**rwhod**(8C)	system status server
installboot	**installboot**(8S)	install bootblocks in a disk partition
install_small_kernel	**install_small_kernel**(8)	install a small, pre-configured kernel
installtxt	**installtxt**(8)	create a message archive
in.talkd	**talkd**(8C)	server for talk program
in.telnetd	**telnetd**(8C)	TCP/IP TELNET protocol server
in.tftpd	**tftpd**(8C)	TCP/IP Trivial File Transfer Protocol server
in.tnamed	**tnamed**(8C)	TCP/IP Trivial name server
intr	**intr**(8)	allow a command to be interruptible
iostat	**iostat**(8)	report I/O statistics
ipallocd	**ipallocd**(8C)	Ethernet-to-IP address allocator
kadb	**kadb**(8S)	adb-like kernel and standalone-program debugger
keyenvoy	**keyenvoy**(8C)	talk to keyserver
keyserv	**keyserv**(8C)	server for storing public and private keys
kgmon	**kgmon**(8)	generate a dump of the operating system's profile buffers
lastlogin	**acctsh**(8)	shell procedures for accounting
ldconfig	**ldconfig**(8)	link-editor configuration
link	**link**(8V)	exercise link and unlink system calls
list_files	**list_files**(8)	list files from release media
listen	**nlsadmin**(8)	network listener service administration for RFS
lockd	**lockd**(8C)	network lock daemon
logintool	**logintool**(8)	graphic login interface
lpc	**lpc**(8)	line printer control program
lpd	**lpd**(8)	printer daemon
mailstats	**mailstats**(8)	print statistics collected by sendmail
makedbm	**makedbm**(8)	make a NIS ndbm file
MAKEDEV	**makedev**(8)	make system special files
makekey	**makekey**(8)	generate encryption key
mc68881version	**mc68881version**(8)	print the MC68881 mask number and approximate clock rate
mconnect	**mconnect**(8)	connect to SMTP mail server socket
mkfile	**mkfile**(8)	create a file
mkfs	**mkfs**(8)	construct a file system
mknod	**mknod**(8)	build special file

mkproto	**mkproto**(8)	construct a prototype file system
modload	**modload**(8)	load a module
modstat	**modstat**(8)	display status of loadable modules
modunload	**modunload**(8)	unload a module
monacct	**acctsh**(8)	shell procedures for accounting
monitor	**monitor**(8S)	system ROM monitor
mountd	**mountd**(8C)	NFS mount request server
mount	**mount**(8)	mount and unmount file systems
mount_tfs	**mount_tfs**(8)	mount and dismount TFS filesystems
named	**named**(8C)	Internet domain name server
ncheck	**ncheck**(8)	generate names from i-numbers
ndbootd	**ndbootd**(8C)	ND boot block server
netconfig	**netconfig**(8C)	PNP boot service
netstat	**netstat**(8C)	show network status
newaliases	**newaliases**(8)	rebuild the data base for the mail aliases file
newfs	**newfs**(8)	create a new file system
newkey	**newkey**(8)	create a new key in the publickey database
nfsd	**nfsd**(8)	NFS daemons
nfsstat	**nfsstat**(8C)	Network File System statistics
nlsadmin	**nlsadmin**(8)	network listener service administration for RFS
nslookup	**nslookup**(8C)	query domain name servers interactively
nsquery	**nsquery**(8)	RFS name server query
nulladm	**acctsh**(8)	shell procedures for accounting
old-analyze	**old-analyze**(8)	postmortem system crash analyzer
openboot	**openboot**(8S)	start the system kernel or a standalone program
pac	**pac**(8)	printer/plotter accounting information
panic	**panic**(8S)	what happens when the system crashes
ping	**ping**(8C)	send ICMP ECHO_REQUEST packets to network hosts
pnpboot	**pnpboot**(8C)	pnp diskless boot service
pnpd	**pnpd**(8C)	PNP daemon
pnp.s386	**pnpboot**(8C)	pnp diskless boot service
portmap	**portmap**(8C)	TCP/IP port to RPC program number mapper
praudit	**praudit**(8)	print contents of an audit trail file
prctmp	**acctsh**(8)	shell procedures for accounting
prdaily	**acctsh**(8)	shell procedures for accounting
prtacct	**acctsh**(8)	shell procedures for accounting
pstat	**pstat**(8)	print system facts
pwck	**pwck**(8V)	check password database entries
pwdauthd	**pwdauthd**(8C)	server for authenticating passwords
quotacheck	**quotacheck**(8)	file system quota consistency checker
quotaoff	**quotaon**(8)	turn file system quotas on and off
quotaon	**quotaon**(8)	turn file system quotas on and off
quot	**quot**(8)	summarize file system ownership
rarpd	**rarpd**(8C)	TCP/IP Reverse Address Resolution Protocol server
rc	**rc**(8)	command scripts for auto-reboot and daemons
rc.boot	**rc**(8)	command scripts for auto-reboot and daemons
rc.local	**rc**(8)	command scripts for auto-reboot and daemons
rdate	**rdate**(8C)	set system date from a remote host
rdump	**dump**(8)	incremental file system dump
reboot	**reboot**(8)	restart the operating system
renice	**renice**(8)	alter nice value of running processes
repquota	**repquota**(8)	summarize quotas for a file system
restore	**restore**(8)	incremental file system restore

rexd	**rexd**(8C)	RPC-based remote execution server
rexecd	**rexecd**(8C)	remote execution server
rfadmin	**rfadmin**(8)	RFS domain administration
rfpasswd	**rfpasswd**(8)	change RFS host password
rfstart	**rfstart**(8)	start RFS
rfstop	**rfstop**(8)	stop the RFS environment
rfuadmin	**rfuadmin**(8)	RFS notification shell script
rfudaemon	**rfudaemon**(8)	Remote File Sharing daemon
rlogind	**rlogind**(8C)	remote login server
rmail	**rmail**(8C)	handle remote mail received via uucp
rm_client	**rm_client**(8)	remove an NFS client
rmntstat	**rmntstat**(8)	display RFS mounted resource information
rmt	**rmt**(8C)	remote magtape protocol module
route	**route**(8C)	manually manipulate the routing tables
routed	**routed**(8C)	network routing daemon
rpc.etherd	**etherd**(8C)	Ethernet statistics server
rpcinfo	**rpcinfo**(8C)	report RPC information
rpc.lockd	**lockd**(8C)	network lock daemon
rpc.mountd	**mountd**(8C)	NFS mount request server
rpc.rexd	**rexd**(8C)	RPC-based remote execution server
rpc.rquotad	**rquotad**(8C)	remote quota server
rpc.rstatd	**rstatd**(8C)	kernel statistics server
rpc.rusersd	**rusersd**(8C)	network username server
rpc.rwalld	**rwalld**(8C)	network rwall server
rpc.sprayd	**sprayd**(8C)	spray server
rpc.statd	**statd**(8C)	network status monitor
rpc.yppasswdd	**yppasswdd**(8C)	server for modifying NIS password file
rpc.ypupdated	**ypupdated**(8C)	server for changing NIS information
rquotad	**rquotad**(8C)	remote quota server
rrestore	**restore**(8)	incremental file system restore
rshd	**rshd**(8C)	remote shell server
rstatd	**rstatd**(8C)	kernel statistics server
runacct	**acctsh**(8)	shell procedures for accounting
runacct	**runacct**(8)	run daily accounting
rusage	**rusage**(8)	print resource usage for a command
rusersd	**rusersd**(8C)	network username server
rwalld	**rwalld**(8C)	network rwall server
rwhod	**rwhod**(8C)	system status server
sa	**sa**(8)	system accounting
savecore	**savecore**(8)	save a core dump of the operating system
sendmail	**sendmail**(8)	send mail over the internet
set4	**set4**(8)	set the virtual address space limit flag in a module
setsid	**setsid**(8V)	set process to session leader
showfhd	**showfhd**(8C)	showfh daemon run on the NFS servers
showfh	**showfh**(8C)	print full pathname of file from the NFS file handle
showmount	**showmount**(8)	show all remote mounts
showrev	**showrev**(8)	show machine and software revision information
shutacct	**acctsh**(8)	shell procedures for accounting
shutdown	**shutdown**(8)	close down the system at a given time
skyversion	**skyversion**(8)	print the SKYFFP board microcode version number
sprayd	**sprayd**(8C)	spray server
spray	**spray**(8C)	spray packets
start_applic	**start_applic**(8)	generic application startup procedures

startup	**acctsh**(8)	shell procedures for accounting
statd	**statd**(8C)	network status monitor
sticky	**sticky**(8)	mark files for special treatment
sundiag	**sundiag**(8)	system diagnostics
suninstall	**suninstall**(8)	install and upgrade the SunOS operating system
swapon	**swapon**(8)	specify additional device for paging and swapping
syslogd	**syslogd**(8)	log system messages
sys-unconfig	**sys-unconfig**(8)	undo a system's configuration
talkd	**talkd**(8C)	server for talk program
telnetd	**telnetd**(8C)	TCP/IP TELNET protocol server
tfsd	**tfsd**(8)	TFS daemon
tftpd	**tftpd**(8C)	TCP/IP Trivial File Transfer Protocol server
tic	**tic**(8V)	terminfo compiler
tnamed	**tnamed**(8C)	TCP/IP Trivial name server
trpt	**trpt**(8C)	transliterate protocol trace
ttysoftcar	**ttysoftcar**(8)	enable/disable carrier detect
tunefs	**tunefs**(8)	tune up an existing file system
turnacct	**acctsh**(8)	shell procedures for accounting
tzsetup	**tzsetup**(8)	set up old-style time zone information in the kernel
uid_allocd	**uid_allocd**(8C)	UID and GID allocator daemons
umount	**mount**(8)	mount and unmount file systems
umount_tfs	**mount_tfs**(8)	mount and dismount TFS filesystems
unadv	**unadv**(8)	unadvertise a Remote File Sharing resource
unconfigure	**unconfigure**(8)	reset the network configuration for a Sun386i system
unixname2bootname	**unixname2bootname**(8)	convert SunOS device name to boot device name
unlink	**link**(8V)	exercise link and unlink system calls
unset4	**set4**(8)	unset the virtual address space limit flag in a module
update	**update**(8)	periodically update the super block
user_agentd	**user_agentd**(8C)	user agent daemon
uucheck	**uucheck**(8C)	check the UUCP directories and Permissions file
uucico	**uucico**(8C)	file transport program for the UUCP system
uuclean	**uuclean**(8C)	uucp spool directory clean-up
uucleanup	**uucleanup**(8C)	UUCP spool directory clean-up
uucpd	**uucpd**(8C)	UUCP server
uusched	**uusched**(8C)	the scheduler for the UUCP file transport program
uuxqt	**uuxqt**(8C)	execute remote command requests
vipw	**vipw**(8)	edit the password file
vmstat	**vmstat**(8)	report virtual memory statistics
wtmpfix	**fwtmp**(8)	manipulate connect accounting records
ypbatchupd	**ypbatchupd**(8C)	NIS batch update daemon
ypbind	**ypserv**(8)	NIS server and binder processes
ypinit	**ypinit**(8)	build and install NIS database
ypmake	**ypmake**(8)	rebuild NIS database
yppasswdd	**yppasswdd**(8C)	server for modifying NIS password file
yppoll	**yppoll**(8)	version of NIS map at NIS server
yppush	**yppush**(8)	force propagation of changed NIS map
ypserv	**ypserv**(8)	NIS server and binder processes
ypset	**ypset**(8)	point ypbind at a particular server
ypsync	**ypsync**(8)	collect most up-to-date NIS maps
ypupdated	**ypupdated**(8C)	server for changing NIS information
ypxfr	**ypxfr**(8)	transfer NIS map from NIS server to here
zdump	**zdump**(8)	time zone dumper
zic	**zic**(8)	time zone compiler

NAME

ac – login accounting

SYNOPSIS

/usr/etc/ac [**–w** *wtmp*] [**–p**] [**–d**] [*username*] ...

DESCRIPTION

ac produces a printout giving connect time for each user who has logged in during the life of the current *wtmp* file. A total is also produced.

The accounting file **/var/adm/wtmp** is maintained by **init**(8) and **login**(1). Neither of these programs creates the file, so if it does not exist no connect-time accounting is done. To start accounting, it should be created with length 0. On the other hand if the file is left undisturbed it will grow without bound, so periodically any information desired should be collected and the file truncated.

OPTIONS

–w *wtmp*
: Specify an alternate *wtmp* file.

–p Print individual totals; without this option, only totals are printed.

–d Printout for each midnight to midnight period. Any *people* will limit the printout to only the specified login names. If no *wtmp* file is given, **/var/adm/wtmp** is used.

FILES

/var/adm/wtmp

SEE ALSO

login(1), **utmp**(5V), **init**(8), **sa**(8)

SEE ALSO

acctcom(1), **acct**(2V), **acct**(5), **utmp**(5V), **acct**(8), **acctcms**(8), **acctcon**(8), **acctmerg**(8), **acctprc**(8), **cron**(8), **diskusg(8), fwtmp**(8), **runacct**(8)

System and Network Administration

NAME

adbgen – generate adb script

SYNOPSIS

/usr/lib/adb/adbgen *filename* **.adb** ...

DESCRIPTION

adbgen makes it possible to write **adb**(1) scripts that do not contain hard-coded dependencies on structure member offsets. The input to **adbgen** is a file named *filename* **.adb** which contains **adbgen** header information, then a null line, then the name of a structure, and finally an **adb** script. **adbgen** only deals with one structure per file; all member names are assumed to be in this structure. The output of **adbgen** is an **adb** script in *filename*. **adbgen** operates by generating a C program which determines structure member offsets and sizes, which in turn generates the **adb** script.

The header lines, up to the null line, are copied verbatim into the generated C program. Typically these include C **#include** statements to include the header files containing the relevant structure declarations.

The **adb** script part may contain any valid **adb** commands (see **adb**(1)), and may also contain **adbgen** requests, each enclosed in { }s. Request types are:

- Print a structure member. The request form is {*member*,*format*}. *member* is a member name of the *structure* given earlier, and *format* is any valid **adb** format request. For example, to print the **p_pid** field of the *proc* structure as a decimal number, you would write **{p_pid,d}**.
- Reference a structure member. The request form is {*****member,*base*}. *member* is the member name whose value is desired, and *base* is an **adb** register name which contains the base address of the structure. For example, to get the **p_pid** field of the *proc* structure, you would get the proc structure address in an **adb** register, say **<f**, and write **{*p_pid,<f}**.
- Tell **adbgen** that the offset is okay. The request form is **{OFFSETOK}**. This is useful after invoking another **adb** script which moves the **adb** *dot*.
- Get the size of the *structure*. The request form is **{SIZEOF}**. **adbgen** replaces this request with the size of the structure. This is useful in incrementing a pointer to step through an array of structures.
- Calculate an arbitrary C expression. The request form is **{EXPR,***expression***}**. **adbgen** replaces this request with the value of the expression. This is useful when more than one structure is involved in the script.
- Get the offset to the end of the structure. The request form is **{END}**. This is useful at the end of the structure to get **adb** to align the *dot* for printing the next structure member.

adbgen keeps track of the movement of the **adb** *dot* and emits **adb** code to move forward or backward as necessary before printing any structure member in a script. **adbgen**'s model of the behavior of **adb**'s *dot* is simple: it is assumed that the first line of the script is of the form *struct_address/adb text* and that subsequent lines are of the form *+/adb text*. The **adb** *dot* then moves in a sane fashion. **adbgen** does not check the script to ensure that these limitations are met. **adbgen** also checks the size of the structure member against the size of the **adb** format code and warns you if they are not equal.

EXAMPLES

If there were an include file **x.h** which contained:

```
struct x {
        char    *x_cp;
        char    x_c;
        int     x_i;
};
```

Then an **adbgen** file (call it **script.adb**) to print it would be:

```
#include "x.h"
x
./"x_cp"16t"x_c"8t"x_i"n{x_cp,X}{x_c,C}{x_i,D}
```

After running **adbgen** the output file **script** would contain:

16t"x_c"8t"x_i"nXC+D"" ./"x_cp"16t"x_c"8t"x_i"nXC+D

To invoke the script you would type:

x$<script

FILES

/usr/lib/adb/* **adb** scripts for debugging the kernel

SEE ALSO

adb(1), **kadb**(8S)

Debugging Tools Manual

BUGS

adb syntax is ugly; there should be a higher level interface for generating scripts.

Structure members which are bit fields cannot be handled because C will not give the address of a bit field. The address is needed to determine the offset.

DIAGNOSTICS

Warnings about structure member sizes not equal to **adb** format items and complaints about badly formatted requests. The C compiler complains if you reference a structure member that does not exist. It also complains about & before array names; these complaints may be ignored.

NAME

add_client – create a diskless network bootable NFS client on a server

SYNOPSIS

/usr/etc/install/add_client [**–inpv**] [**–a** *kernel-arch*] [**–e** *exec-path*] [**–f** *share-path*] [**–h** *home-path*]
[**–k** *kvm-path*] [**–m** *mail-path*] [**–r** *root-path*] [**–s** *swap-path*] [**–t** *term-type*]
[**–y** *yptype*] [**–z** *swapsize*] *client* ...

/usr/etc/install/add_client –i | –p [**–nv**] [**–a** *kernel-arch*] [**–e** *exec-path*] [**–f** *share-path*]
[**–h** *home-path*] [**–k** *kvm-path*] [**–m** *mail-path*] [**–r** *root-path*] [**–s** *swap-path*]
[**–t** *term-type*] [**–y** *yptype*] [**–z** *swapsize*] [*client* ...]

DESCRIPTION

add_client adds an NFS client to a server. It can only be run by the super-user.

A default standard layout is used to set up the client's environment, but most pathnames can be overridden with the appropriate option, or menu field change.

Before you can add a client, you must first make sure that the Internet and Ethernet addresses for *client* are listed in the Network Information Service (NIS) hosts database (if the server is running the NIS service), or in the server's **/etc/hosts** and **/etc/ethers** databases, respectively. If **add_client** cannot find the client entry in the hosts database it aborts the operation. If there is no client entry in the **/etc/ethers** database, **add_client** issues a warning to update this file while adding the client.

The default root and swap partitions are **/export/root/***client* and **/export/swap/***client*, respectively.

add_client updates the **/etc/bootparams** file on the server but not the bootparams database in the NIS service (if used).

If the server is not running as an NIS master, **add_client** issues a warning to indicate that the database is out of date and the NIS master should be updated.

add_client updates the server's **/etc/exports** file to allow client's root access to each client's root file system. It also exports each client's swap file accordingly. Note: the system administrator should verify that the **/etc/exports** file contains correct information, and that file systems are exported to the correct users and groups. Refer to **exportfs**(8) for details on exporting file systems.

If the **–i** or **–p** option is not specified, at least one *client* argument must be supplied on the command line.

OPTIONS

–i Interactive. Bring up a full-screen menu interface to **add_client**.

–n Print the working parameters and exit without doing anything. This is used to verify what parameters **add_client** will use before actually doing anything.

–p Display a short version of all client information, If *client*s are specified on the command line, only display information for those clients. When combined with the **–v** option, a long version of client information is displayed.

–v Verbose. Report information about the client as steps are performed.

–a *kernel-arch* Specify the client kernel architecture (for instance, **sun3**, **sun4**, **sun4c**...). **add_client** prompts for the kernel architecture when unable to determine the correct value.

–e *exec-path* Set the pathname of the directory in which the executables for the architecture specified by **–a**. The client mounts **/export/exec/***arch.rel* as **/usr**. See WARNINGS.

–f *share-path* Set the pathname of the share directory, which is normally a link to **/usr/share**.

–h *home-path* Set the pathname of the directory for the client's home. The default is **/home/***servername*.

–k *kvm-path* Set the pathname of the directory containing the client's kernel executables. See WARNINGS.

–m *mail-path* Set the pathname of the client's mail directory. The default is **/var/spool/mail**.

–r *root-path* Set the pathname of parent directory for client root directories; *root/client* is the pathname of the client's root directory. The default is **/export/root/***client-name*.

–s *swap-path* Set the pathname of parent directory for client swap files; *swap/client* is the pathname of the client's swap file. The default is **/export/swap/***client-name*.

–t *term-type* Set the terminal type of the client's console.

–y *yptype* Indicate the type of NIS server or if client is to be an NIS client; it can be **client** or **none**. The **none** argument results in the NIS service being disabled on the client. The default is **client**.

–z *swapsize* Reserve *swapsize* bytes for the client's swap file. **swapsize** can be flagged as kilobytes, blocks, or megabytes, with the **k**, **b**, or **m** suffixes, respectively. The default is 16Mb, and bytes are used when no units are specified.

FILES

/etc/bootparams
/etc/ethers
/etc/exports
/etc/hosts
/export/exec/proto.root.*release*
architecture independent base for the client root file system
/tftpboot.*client-ipaddr* link to **/tftpboot/boot.***arch*

SEE ALSO

add_services(8), **bootparamd(8), exportfs**(8), **ndbootd**(8C), **rm_client**(8), **suninstall**(8)

Installing the SunOS System Software

DIAGNOSTICS

add_client: must be super-user
You must be root to use **add_client**.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

WARNINGS

The **–e** *exec-path* and the **–k** *kvm-path* options should not be used since the correct paths are determined when the adding the client's architecture service. See **add_services**(8).

NAME

add_services – provide software installation services for any architecture

SYNOPSIS

/usr/etc/install/add_services

DESCRIPTION

add_services is a menu-based program to setup a system as a server and/or to add additional software categories or other architecture releases. It is used to provide support to diskless clients, dataless clients, or just to act as a file server. **add_services** can only be run by the super-user.

add_services updates the **/etc/exports** file (see **exports**(5) and **exportfs**(8)) to export the necessary file systems to become a file server. After running **add_services**, the system administrator should verify this file to make sure that the new services have been exported to the correct groups.

FILES

/etc/hosts	hosts database, host must be in this database or in the Network Information Service (NIS) hosts map
/etc/exports	database of exported file systems, service related directories must be exported
/tftpboot	add_services sets up this directory in order to provide boot service to clients

SEE ALSO

exports(5), **add_client**(8), **exportfs**(8), **rm_client**(8), **suninstall**(8)

Installing the SunOS System Software

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

add_user – shell script to add a user account to a machine

SYNOPSIS

/usr/etc/install/add_user *uname uid gid fullname homedir shell*

DESCRIPTION

The **add_user** shell script creates a new user account. It is purposely limited in its abilities, and should be considered as a working example of how to add a user account to a machine. System administrators may wish to use this script as the basis for their own tools to create new accounts.

The **adduser**(8) manual page provides a more detailed description of the steps involved in adding a user to a machine, including those steps automated by **add_user**. Also refer to **adduser**(8) for information about user accounts involving the Network Information Service (NIS).

When run, **add_user** first checks that the following conditions are satisfied.

The command must be run by root.

A valid number of arguments must be suplied.

uid and *gid* are between 10 and 60000.

shell exists and is executable.

homedir does *not* already exist.

The parent directory where the home directory will be created exists.

The filesystem where the home directory will be created is a local filesystem.

uname and *uid* do not already exist in the **/etc/passwd** file.

When all of the preceding checks are satisfied, **add_user** performs the following steps to create a user account.

Creates an entry in the **/etc/passwd** file.

Creates the home directory.

Changes the ownership and group of the home directory to that of the new user.

Copies the default user startup files (**.login .cshrc .rootmenu .sunview**) to the user's home directory, and changes ownership and group of these files to that of the user's.

EXAMPLES

The following example shows the input required to add the user "Lynn Long" to the current system. Note that **/usr/etc/install** is not a usual path for user executables, only administrators should have this in their path.

example# /usr/etc/install/add_user lynnl 1099 10 "Lynn Long" /home/example/lynnl /bin/csh

This adds the user "Lynn Long" to the system, creating an entry in the **/etc/passwd** file that looks like:

lynnl::1099:10:Lynn Long:/home/example/lynnl:/bin/csh

It also creates the directory **/home/example/lynnl** and copies the default user startup files into it.

FILES

/usr/lib/Login	the default **.login** file placed in the user's home directory
/usr/lib/Cshrc	the default **.cshrc** file placed in the user's home directory
/usr/lib/.sunview	the default **.sunview** file placed in the user's home directory
/usr/lib/.rootmenu	the default **.rootmenu** file placed in the user's home directory
/etc/passwd	the system password file

SEE ALSO

csh(1), **passwd**(1), **sunview**(1), **adduser**(8)

DIAGNOSTICS

The numerous error messages that **add_user** may issue are intended to be self-explanatory.

WARNINGS

Do not attempt to run **add_user** on a system that has the *C2 Security* software installed on it.

NOTES

After successfully running **add_user**, either the system administrator (running as root) or the user should immediately create a password for the account, using the **passwd**(1) command, to deter unauthorized use of the new account.

BUGS

add_user does not check to see if the *gid* exists in the **/etc/group** file on the machine. Thus the user's group may not have a name associated with it.

Since each direct map entry results in a separate mount for the mount daemon such maps should be kept short. Entries added to a direct map will have no effect until the automounter is restarted.

Entries in both direct and indirect maps can be modified at any time. The new information will be used when **automount** next uses the map entry to do a mount. **automount** does not cache map entries.

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

BUGS

The **bg** mount option is not recognized by the automounter.

Since **automount** is single-threaded, any request that is delayed by a slow or non-responding NFS server will delay all subsequent automatic mount requests until it completes.

Programs that read **/etc/mtab** and then touch files that reside under automatic mount points will introduce further entries to the file.

Automatically-mounted file systems are mounted with type **ignore**; they do not appear in the output of either **mount**(8), or **df**(1V).

NAME

boot – start the system kernel or a standalone program

SYNOPSIS

SUN-3, SUN-4 SYSTEMS

>b [*device* [(*c*, *u*, *p*)]] [*filename*] [**–av**] *boot-flags*
>b *tapedev* ([*c*, *u*, *p*])**install** [*filenum*] *diskdev*([*c*],[*u*],[*p*])[**vmunix**[*boot-flags*]]
>b?
>b!

SPARCstation 1 SYSTEMS

>b [*device* [(*c*, *u*, *p*)]] [*filename*] *boot-flags*

SPARCstation 2 SYSTEMS, SPARCsystem 600MP SERIES

>b [*device-specifier*] [*filename*] *boot-flags*

DESCRIPTION

The boot program is started by the PROM monitor and loads the kernel, or another executable program, into memory.

The following applies to all Sun systems *except* Desktop SPARCsystems and SPARCsystem 600MP series. See **openboot**(8S) for those systems.

The form **b**...**install** loads the miniroot from a SunOS distribution tape or CDROM into the designated partition of *diskdev* (commonly **1**, the swap partition), and then boots the mini-root kernel in preparation for a software installation.

The form **b?** displays all boot devices and their device arguments.

The form **b!** boots, but does not perform a RESET.

USAGE

Booting Standalone

When booting standalone, the boot program (**/boot**) is brought in by the PROM from the file system. This program contains drivers for all devices.

Booting a Sun-3 System Over the Network

When booting over the network, the Sun-3 system PROM obtains a version of the boot program from a server using the Trivial File Transfer Protocol (TFTP). The client broadcasts a RARP request containing its Ethernet address. A server responds with the client's Internet address. The client then sends a TFTP request for its boot program to that server (or if that fails, it broadcasts the request). The filename requested (unqualified — not a pathname) is the hexadecimal, uppercase representation of the client's Internet address; for example:

Using IP Address 192.9.1.17 = C0090111

When the Sun server receives the request, it looks in the directory **/tftpboot** for *filename*. That file is typically a symbolic link to the client's boot program, normally **boot.sun3**, in the same directory. The server invokes the TFTP server, **tftpd**(8C), to transfer the file to the client.

When the file is successfully read in by the client, the boot program jumps to the load-point and loads **vmunix** (or a standalone program). In order to do this, the boot program makes a broadcast RARP request to find the client's IP address, and then makes a second broadcast request to a **bootparamd**(8) bootparams daemon, for information necessary to boot the client. The bootparams daemon obtains this information either from a local **/etc/bootparams** database file, or from a Network Information Service (NIS) map. The boot program sends two requests to the bootparams daemon — the first, **whoami**, to obtain its hostname, and the second, **getfile**, to obtain the name of the client's server and the pathname of the client's root partition.

The boot program then performs a **mount**(8) operation to mount the client's root partition, after which it can read in and execute any program within that partition by pathname (including a symbolic link to another file within that same partition). Typically, it reads in the file **/vmunix**. If the program is not read in

successfully, **boot** responds with a short diagnostic message.

Booting Other Sun Systems Over the Network

Other Sun systems boot over the network in a similar fashion. However, the filename requested from a server must have a suffix that reflects the *kernel* architecture of the machine being booted. For these systems, the requested filename has the form:

ip-address.arch

where *ip-address* is the machine's Internet Protocol (IP) address in hex, and *arch* is a suffix representing its kernel architecture. Only Sun-3 systems may omit the *arch* suffix. These filenames are restricted to 14 characters for compatibility with UNIX System V and other operating systems. Therefore, the architecture suffix is limited to 5 characters; it must be in upper case. At present, the following suffixes are recognized: **SUN3** for Sun-3 systems, **SUN3X** for Sun-3x systems, **SUN4** for Sun-4 systems, **S386** for Sun386i systems, and **PCNFS** for **PC-NFS. arch**(1) may be used to determine the kernel architecture of a machine.

System Startup

Once the system is loaded and running, the kernel performs some internal housekeeping, configures its device drivers, and allocates its internal tables and buffers. The kernel then starts process number 1 to run **init**(8), which performs file system housekeeping, starts system daemons, initializes the system console, and begins multiuser operation. Some of these activities are omitted when **init** is invoked with certain *boot-flags*. These are typically entered as arguments to the boot command and passed along by the kernel to **init**.

OPTIONS

device One of:

le	Lance Ethernet
ie	Intel Ethernet (Sun-3, Sun-4 systems only)
sd	SCSI disk, CDROM
st	SCSI 1/4" or 1/2" tape
fd	Diskette (Sun386i, Sun-3/80 systems and Desktop SPARCsystems only)
id	IPI disk (Sun-4 systems and SPARCsystem 600MP series only)
mt	Tape Master 9-track 1/2" tape (Sun-3, Sun-4 systems only)
xd	Xylogics 7053 disk (Sun-3, Sun-4 systems only)
xt	Xylogics 1/2" tape (Sun-3, Sun-4 systems only)
xy	Xylogics 440/450 disk (Sun-3, Sun-4 systems only)

diskdev Disk or CDROM; one of **sd**, **id**, **xy** from the above device set.

tapedev Tape; one of **st**, **xt**, **mt** from the above device set.

c Controller number, **0** if there is only one controller for the indicated type of device.

u Unit number, **0** if there is only one driver.

p Partition number when booting off a disk, or tape file number when booting from a tape. Defaults to **0**.

filename Name of a standalone program in the selected partition, such as **stand/diag** or **vmunix**. Note: *filename* is relative to the root of the selected device and partition. It never begins with a '*/*' (slash). If *filename* is not given, the boot program uses a default value (normally **vmunix**). This is stored in the **vmunix** variable in the **boot** executable file supplied by Sun, but can be patched to indicate another standalone program loaded using **adb**(1).

filenum Filenumber to load from *tapedev* to *diskdev*. Defaults to **3**, which is the location of the miniroot on current SunOS distribution tapes.

–a Prompt interactively for the device and name of the file to boot. For more information on how to boot from a specific device, refer to *Installing the SunOS System Software.*

–v Verbose. Print more detailed information to assist in diagnosing diskless booting problems.

boot-flags The boot program passes all *boot-flags* to the kernel or standalone program. They are typically arguments to that program or, as with those listed below, arguments to progams that it invokes.

–b Pass the **–b** flag through the kernel to **init**(8) to skip execution of the **/etc/rc.local** script.

–h Halt after loading the system.

–s Pass the **–s** flag through the kernel to **init**(8) for single-user operation.

–i *initname*
Pass the **–i** *initname* to the kernel to tell it to run *initname* as the first program rather than the default **/sbin/init**.

FILES

/boot	standalone boot program
/tftpboot/*address*	symbolic link to the boot program for the client whose Internet address, in uppercase hexadecimal, is *address*
/tftpboot/boot.sun3	Sun-3 first stage boot program
/tftpboot/boot.sun4	Sun-4 first stage boot program
/usr/etc/in.tftpd	TFTP server
/usr/kvm/mdec/installboot	program to install boot blocks from a remote host
/vmunix	kernel file that is booted by default
/usr/kvm/boot	
/etc/bootparams	file defining root and swap paths for clients

SEE ALSO

adb(1), **arch**(1), **tftp**(1C), **bootparamd**(8), **init**(8), **kadb**(8S), **monitor**(8S), **mount**(8), **ndbootd**(8C), **openboot**(8S), **rc**(8), **reboot**(8), **tftpd**(8C)

Installing the SunOS System Software
System and Network Administration

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

clri – clear inode

SYNOPSIS

/usr/etc/clri *filesystem i-number...*

DESCRIPTION

Note: **clri** has been superseded for normal file system repair work by **fsck**(8).

clri writes zeros on the inodes with the decimal *i-numbers* on the *filesystem*. After **clri,** any blocks in the affected file will show up as ''missing'' in an **icheck**(8) of the *filesystem.*

Read and write permission is required on the specified file system device. The inode becomes allocatable.

The primary purpose of this routine is to remove a file which for some reason appears in no directory. If it is used to zap an inode which does appear in a directory, care should be taken to track down the entry and remove it. Otherwise, when the inode is reallocated to some new file, the old entry will still point to that file. At that point removing the old entry will destroy the new file. The new entry will again point to an unallocated inode, so the whole cycle is likely to be repeated again and again.

SEE ALSO

icheck(8) **fsck**(8)

BUGS

If the file is open, **clri** is likely to be ineffective.

NAME

colldef – convert collation sequence source definition

SYNOPSIS

/usr/etc/colldef *filename*

DESCRIPTION

colldef converts a collation sequence source definition into a format usable by the **strxfrm( )** and **strcoll**(3) functions. It is used to define the many ways in which strings can be ordered and collated. **strxfrm( )** transforms its first argument and places the result in its second argument. The transformed string is such that it can be correctly ordered with other transformed strings by using **strcmp( )**, **strncmp( )**, or **memcmp( )** (see **string**(3) and **memory**(3)). **strcoll**(3) transforms its arguments and does a comparison.

colldef reads the collation sequence source definition from the standard input and stores the converted definition in *filename*. The output file produced contains the database with collating sequence information in a form usable by system commands and routines.

The collation sequence definition specifies a set of collating elements and the rules defining how strings containing these should be ordered. This is most useful for different language definitions.

The **colldef** command can support languages whose mapping and collating sequences can be described by the following cases:

- Ordering of single characters within the codeset. For example, in Swedish, **V** is sorted after **U**, before **X** and with **W** (**V** and **W** are considered identical as far as sorting is concerned).
- Equivalence class definition. A collection of characters is defined to have the same primary sorting value.
- Ordering of "double characters" in the collation sequence. For example, in Spanish, **ch** and **ll** are collated after **c** and **l**, respectively.
- Ordering of a single character as if it consists of two characters. For example, in German, the "sharp s", **ß**, is sorted as **ss**. This is a special instance of the next case below.
- Substitution of one character with a character string, that is, a one-to-many mapping. In the example above, the character **ß** is replaced with **ss** during sorting.
- Ignoring certain characters in the codeset during collation. For example, if '–' is not specified in the specification table, then the strings **re–locate** and **relocate** are equal.
- Null character mapping. A character is mapped to a null collating element, and is ignored in sorting sequences.
- Secondary ordering between characters. In the case where two characters are sorted together in the collation sequence, (for example, they have the same "primary" ordering), there is sometimes a secondary ordering that is used if two strings are identical except for characters that have the same primary ordering. For example, in French, the letters **e** and **è** have the same primary ordering but **e** comes before **è** in the secondary ordering. Thus the word **lever** would be ordered before **lèver**, but **lèver** would be sorted before **levitate**. Note: if **e** came before **è** in the primary ordering, then **lèver** would be sorted after **levitate**.

USAGE

The specification file can consist of three statements: **charmap**, **substitute**, and **order**. Of these, only the **order** statement is required. When **charmap** or **substitute** is supplied, these statements must be ordered as above. Any statements after the **order** statement are ignored.

Lines in the specification file beginning with a **#** are treated as comments and are ignored. Blank lines are also ignored.

charmap *charmapfile*

charmap defines where a mapping of the character and collating element symbols to the actual character encoding can be found. The *charmapfile* filename cannot be a keyword (for example, **substitute**, **order**, or **with**) or special symbols (for example, **...**, **;**, **<**, **>**, or **,**).

The format of *charmapfile* is shown below. Symbol names are separated from their values by TAB or SPACE characters. *symbol-value* can be specified in a hexadecimal (**\x***??*) or octal (*****???*) representation, and can be only one character in length.

symbol-name1 *symbol-value1*
symbol-name2 *symbol-value2*
...

The following sample *charmapfile* maps the symbol names, **c**, **h**, **H**, and **A-grave**, to their respective symbol values.

c **\x63**
h **\x68**
H **\110**
A-grave **\300**

The symbol names defined in *charmapfile* can be used in **order** statements by enclosing the symbol name in angle brackets, **<***symbol-name***>**. For example,

order (a, <A-grave>);b;<c>;...;<h>;<H>;i;...;z

This statement is equivalent to,

order (a, À);b;c;...;h;H;i;...;z

Symbol names cannot be specified in **substitute** fields. Symbol names also cannot be combined with any other representation, such as, **<c>h**, **c<h>**, **<c>\x68**, or **<c><h>.** Symbol names can be used with primary and secondary ordering as in the following example.

order a;b;c;(<c>,<h>);d;...;z;
A;...;G;{H,<H>};I;...;Z

The **charmap** statement is optional.

substitute *char* **with** *repl*

The **substitute** statement substitutes the character *char* with the string *repl*.

The simple use of the **substitute** statement mentioned above substituted a single character with two characters, as with the substitution of **β** with **ss** in German.

substitute "β" with "ss"

This statement can also be used to specify characters to be ignored by mapping them to the null string.

substitute "m" with ""

This is convenient for simplifying **order** statements. When used with the statement below, the lower-case **m** is ignored — even though it is implicitly included in the **order** statement.

order a;...;z

Without the null string mapping statement above, this would be specified as,

order a;...;l;n;...;z

The **substitute** statement is optional.

order *order_list*

order_list is a list of symbols, separated by semicolons, that defines the collating sequence. The special symbol, **...**, specifies, in a short-hand form, symbols that are sequential in machine code order. The following example specifies the list of lower-case letters.

order a;b;c;d;...;x;y;z

Of course, this could be further compressed to just **a;...;z**.

A symbol can be up to two characters in length and can be represented in any one of the following ways:

- The symbol itself (for example, **a** for the lower-case letter **a**).
- In octal representation (for example, **\141** for the letter **a**).
- In hexadecimal representation (for example, **\x61** for the letter **a**).
- The symbol name as defined in the **charmap** file.

Any combination of these may be used as well.

The backslash character, \, is used for continuation. In this case, no characters are permitted after the backslash character.

Symbols enclosed in parentheses are assigned the same primary ordering but different secondary ordering. Symbols enclosed in curly brackets are assigned only the same primary ordering. For example,

order a;b;c;ch;d;(e,è);f;...;z;
{1,2,3,4,5,6,7,8,9};A;...;Z

In the above example, **e** and **è** are assigned the same primary ordering and different secondary ordering, and digits 1 through 9 are assigned the same primary ordering and no secondary ordering. Note that the ellipses cannot be specified within curly brackets. Only primary ordering is assigned to the remaining symbols. Notice how double letters can be specified in the collating sequence (letter **ch** comes between **c** and **d**).

If a character is not included in the **order** statement it is excluded from the ordering and will be ignored during sorting.

EXAMPLES

The following example shows the collation specification required to support a hypothetical telephone book sorting sequence.

The sorting sequence is defined by the following rules:

- Upper and lower case letters must be sorted together, but upper case letters have precedence over lower case letters.
- All special characters and punctuation should be ignored.
- Digits must be sorted as their alphabetic counterparts (for example, **0** as **zero**, **1** as **one**).
- The **CH**, **Ch**, **ch** combinations must be collated between **c** and **D**.
- **V** and **W**, **v** and **w** must be collated together.

The input specification file for this example contains:

```
substitute "0" with "zero"
substitute "1" with "one"
substitute "2" with "two"
substitute "3" with "three"
substitute "4" with "four"
substitute "5" with "five"
substitute "6" with "six"
substitute "7" with "seven"
substitute "8" with "eight"
substitute "9" with "nine"

order A;a;B;b;C;c;CH;Ch;ch;D;d;E;e;F;f;\
    G;g;H;h:I;i;J;j;K;k;L;l;M;m;N;n;O;o;P;p;\
    Q;q;R;r;S;s;T;t;U;u;{V,W};{v,w};X;x;Y;y;Z;z
```

EXIT STATUS

colldef exits with the following values:

0 No errors were found and the output was successfully created.

>0 Errors were found.

FILES

/etc/locale/LC_COLLATE/*locale*
standard private location for collation orders under the *locale* locale

/usr/share/lib/locale/LC_COLLATE/*locale*
standard shared location for collation orders under the *locale* locale

SEE ALSO

memory(3), **strcoll**(3), **string**(3)
System Services Overview

NAME

comsat, in.comsat – biff server

SYNOPSIS

/usr/etc/in.comsat

DESCRIPTION

comsat is the server process which listens for reports of incoming mail and notifies users who have requested to be told when mail arrives. It is invoked as needed by **inetd**(8C), and times out if inactive for a few minutes.

comsat listens on a datagram port associated with the **biff**(1) service specification (see **services**(5)) for one line messages of the form

user@mailbox-offset

If the *user* specified is logged in to the system and the associated terminal has the owner execute bit turned on (by a '**biff y**'), the *offset* is used as a seek offset into the appropriate mailbox file and the first 7 lines or 560 characters of the message are printed on the user's terminal. Lines which appear to be part of the message header other than the **From**, **To**, **Date**, or **Subject** lines are not printed when displaying the message.

FILES

/etc/utmp	to find out who's logged on and on what terminals

SEE ALSO

biff(1), **services**(5), **inetd**(8C)

BUGS

The message header filtering is prone to error.

The notification should appear in a separate window so it does not mess up the screen.

NAME

config – build system configuration files

SYNOPSIS

/usr/etc/config [**–fgnp**] [**–o** *obj_dir*] *config_file*

DESCRIPTION

config does the preparation necessary for building a new system kernel with **make**(1). The *config_file* named on the command line describes the kernel to be made in terms of options you want in your system, size of tables, and device drivers to be included. When you run **config**, it uses several input files located in the current directory (typically the **conf** subdirectory of the system source including your *config_file*). The format of this file is described below.

If the directory named **../***config_file* does not exist, **config** will create one. One of **config**'s output files is a makefile which you use with **make**(1) to build your system.

config must be run from the **conf** subdirectory of the system source (in a typical Sun environment, from **/usr/share/sys/sun{ 3, 3x, 4, 4c, 4m }/conf**):

example# /usr/etc/config config_file
Doing a "make depend"
example# cd ../config_file
example# make
... lots of output...

While **config** is running watch for any errors. Never use a kernel which **config** has complained about; the results are unpredictable. If **config** completes successfully, you can change directory to the **../***config_file* directory, where it has placed the new makefile, and use **make** to build a kernel. The output files placed in this directory include **ioconf.c**, which contains a description of I/O devices attached to the system; **mbglue.s**, which contains short assembly language routines used for vectored interrupts, a makefile, which is used by **make** to build the system; a set of header files (*device_name*.**h**) which contain the number of various devices that may be compiled into the system; and a set of swap configuration files which contain definitions for the disk areas to be used for the root file system, swapping, and system dumps.

Now you can install your new kernel and try it out.

OPTIONS

–f Set up the makefile for fast builds. This is done by building a **vmunix.o** file which includes all the **.o** files which have no source. This reduces the number of files which have to be **stat**ed during a system build. This is done by prelinking all the files for which no source exists into another file which is then linked in place of all these files when the kernel is made. This makefile is faster because it does not **stat** the object files during the build.

–g Get the current version of a missing source file from its SCCS history, if possible.

–n Do not do the '**make depend**'. Normally **config** will do the '**make depend**' automatically. If this option is used **config** will print '**Don't forget to do a "make depend"**' before completing as a reminder.

–p Configure the system for profiling (see **kgmon**(8) and **gprof**(1)). This option is only available for systems with full source releases.

–o *obj_dir*
Use **../***obj_dir* instead of **../OBJ** as the directory to find the object files when the corresponding source file is not present in order to generate the files necessary to compile and link your kernel.

USAGE

Input Grammar

In the following descriptions, a number can be a decimal integer, a whole octal number or a whole hexadecimal number. Hex and octal numbers are specified to **config** in the same way they are specified to the C compiler, a number starting with **0x** is a hex number and a number starting with just a **0** is an octal number.

Comments are begin with a # character, and end at the next NEWLINE. Lines beginning with TAB characters are considered continuations of the previous line. Lines of the configuration file can be one of two basic types. First, there are lines which describe general things about your system:

machine "*type*"
This is system is to run on the machine type specified. Only one machine type can appear in the config file. The legal *type*s for a Sun system are **sun3**, **sun3x**, **sun4**, **sun4c**, **sun4m**, and **sun386**. Note: the double quotes around *type* are part of the syntax, and must be included.

cpu "*type*"
This system is to run on the CPU type specified. More than one CPU type can appear in the config file.

ident *name*
Give the system identifier — a name for the machine or machines that run this kernel. Note: *name* must be enclosed in double quotes if it contains both letters and digits. Also, note that if *name* is **GENERIC**, you need not include the '**options GENERIC**' clause in order to specify '**swap generic**'.

maxusers *number*
The maximum expected number of simultaneously active user on this system is *number*. This number is used to size several system data structures.

options *optlist*
Compile the listed options into the system. Options in this list are separated by commas. A line of the form:

options FUNNY , HAHA

yields

–DFUNNY –DHAHA

to the C compiler. An option may be given a value, by following its name with = (equal sign) then the value enclosed in (double) quotes. None of the standard options use such a value.

In addition, options can be used to bring in additional files if the option is listed in the **files** files. All options should be listed in upper case. In this case, no corresponding *option*.**h** will be created as it would be using the corresponding *pseudo-device* method.

config *sysname config_clauses* . . .
Generate a system with name *sysname* and configuration as specified in *config-clauses*. The *sysname* is used to name the resultant binary image and per-system swap configuration files. The *config_clauses* indicate the location for the root file system, one or more disk partitions for swapping and paging, and a disk partition to which system dumps should be made. All but the root device specification may be omitted; **config** will assign default values as described below.

root A root device specification is of the form:

root on *xy0d*

If a specific partition is omitted — for example, if only

root on xy0

is specified — the '**a**' partition is assumed. When a generic system is being built, no root specification should be given; the root device will be defined at boot time by prompting the console.

swap To specify a primary swap partition, use a clause of the form:

swap on *partition*

Swapping areas may be almost any size. Partitions used for swapping are sized at boot time by the system; to override dynamic sizing of a swap area the number of sectors in

the swap area can be specified in the config file. For example,

swap on *xy0b* **size 99999**

would configure a swap partition with 99999 sectors. If **swap generic** or no *partition* is specified with **on**, partition b on the root device is used. For dataless clients, use:

swap on type nfs

dumps The location to which system dumps are sent may be specified with a clause of the form:

dumps on *xy1*

If no dump device is specified, the first swap partition specified is used. If a device is specified without a particular partition, the '**b**' partition is assumed. If a generic configuration is to be built, no dump device should be specified; the dump device will be assigned to the swap device dynamically configured at boot time. Dumps are placed at the end of the partition specified. Their size and location is recorded in global kernel variables *dumpsize* and *dumplo*, respectively, for use by **savecore**(8).

Device names specified in configuration clauses are mapped to block device major numbers with the file **devices.***machine*, where *machine* is the machine type previously specified in the configuration file. If a device name to block device major number mapping must be overridden, a device specification may be given in the form:

major *x* **minor** *y*

The second group of lines in the configuration file describe which devices your system has and what they are connected to (for example, an IPI Channel Adaptor on the VMEbus). These lines have the following format:

dev_type dev_name **at** *con_dev more_info*

dev_type is either **controller**, **disk**, **tape**, **device**, **device-driver**, or **pseudo-device**. These types have the following meanings:

controller A disk or tape controller.

disk or **tape** Devices connected to a controller.

device Something "attached" to the main system bus, like a cartridge tape interface.

device-driver This declares support for a device of name *dev_name*. For most devices on desktop SPARCsystems, this is all that is required. See '**Desktop SPARCsystem Input Grammar**' below for details.

pseudo-device A software subsystem or driver treated like a device driver, but without any associated hardware. Current examples are the pseudo-tty driver and various network subsystems. For pseudo-devices, **more_info** may be specified as an integer, that gives the value of the symbol defined in the header file created for that device, and is generally used to indicate the number of instances of the pseudo-device to create.

dev_name is the standard device name and unit number (if the device is not a **pseudo-device**) of the device you are specifying. For example, **idc0** is the *dev_name* for the first IPI disk controller in a system; **st0** names the first SCSI tape controller.

con_dev is what the device you are specifying is connected to. It is either **nexus?**, a bus type, or a controller. There are several bus types which are used by **config** and the kernel.

The possible bus types are:

obmem	On board memory
obio	On board io
vme16d16 (vme16)	16 bit VMEbus/ 16 bit data
vme24d16 (vme24)	24 bit VMEbus/ 16 bit data
vme32d16	32 bit VMEbus/ 16 bit data
vme16d32	16 bit VMEbus/ 32 bit data
vme24d32	24 bit VMEbus/ 32 bit data
vme32d32 (vme32)	32 bit VMEbus/ 32 bit data
ipi	IPI pseudo bus (**sun4** system only)

All of these bus types are declared to be connected to **nexus**. The devices are hung off these buses. If the bus is wildcarded, then the autoconfiguation code will determine if it is appropriate to probe for the device on the machine that it is running on. If the bus is numbered, then the autoconfiguation code will only look for that device on machine type **N**. In general, the VMEbus bus types are always wildcarded.

more_info is a sequence of the following:

csr *address* Specify the address of the **csr** (command and status registers) for a device. The **csr** addresses specified for the device are the addresses within the bus type specified.

The **csr** address must be specified for all controllers, and for all devices connected to a main system bus.

drive *number* For a disk or tape, specify which drive this is.

flags *number* These flags are made available to the device driver, and are usually read at system initialization time.

priority *level* For devices which interrupt, specify the interrupt level at which the device operates.

vector *intr number* [*intr number* . . .]
For devices which use vectored interrupts on VMEbus systems, *intr* specify the vectored interrupt routine and *number* the corresponding vector to be used (0x40-0xFF).

A **?** may be substituted for a number in two places and the system will figure out what to fill in for the **?** when it boots. You can put question marks on a *con_dev* (for example, at virtual '?'), or on a drive number (for example, drive '?'). This allows redundancy, as a single system can be built which will boot on different hardware configurations.

The easiest way to understand **config** files it to look at a working one and modify it to suit your system. Good examples are provided in *Installing the SunOS System Software.*

Desktop SPARCsystem Input Grammar

Desktop SPARCsystems' usage is a good deal simpler than what is described above, due primarily to information provided by the PROM monitor that obviates the specific descriptions of **csr** and **vector** values. There is no need to declare a **nexus**, or a **controller**: all primary controllers and main I/O units are simply described by the **device-driver** keyword. That is, a complete specification of all **UART** controllers (see **zs**(4S)) for a desktop SPARCsystem is done by declaring:

device-driver zs

An additional keyword has been introduced for desktop SPARCsystems to describe SCSI disks and tapes that may be resident on the system: **scsibus**. Its usage is:

scsibus*N* **at** *device-driver*

which declares that there exists a SCSI bus supported by a device-driver previously declared.

After specifying that there is a SCSI bus, you then can specify disks and tapes that may be connected to this SCSI bus. For example, the declaration

disk sd0 at scsibus0 target 3 lun 0

states that there may be a disk (in this example, **sd0**) attached to **scsibus0**, at SCSI Target ID **3**, SCSI Logical unit **0**.

SPARCsystem 600MP Series Input Grammar

SPARCsystem 600MP series machines have a combination of both the common input grammar (described above in '**Input Grammar**'), and the desktop SPARCsystem grammar (described above in '**Desktop SPARCsystem Input Grammar**'). For VMEbus devices, such as IPI, the grammar is similar to the common grammar. For sbus devices, such as SCSI, the grammar is similar to that of desktop SPARCsystems.

FILES

Files in **/usr/share/sys/sun{ 3, 3x, 4, 4c, 4m }/conf** which may be useful for developing the *config_file* used by **config** are:

GENERIC	The generic configuration file for a given Sun system. This file contains all possible device description lines for the particular architecture.
GENERIC_SMALL	A reduced generic configuration file for certain vanilla configurations. Check the comments at the top of the file for specific architectures and devices supported.
DL*model*, **SDST***model*, etc.	Many of the architectures supply template configuration files trimmed to support only certain devices or environments (like diskless clients, standalone systems with SCSI disks and tape, and so on). Comments at the top of these files specify models and devices supported.
README	File describing how to make a new kernel.

As shipped from Sun, the files used by **/usr/etc/config** as input are in the **/usr/include/sys/conf** directory:

config_file	System-specific configuration file
Makefile.src	Generic prototype makefile for Sun–[34] systems
files	List of common files required to build a basic kernel
devices	Name to major device mapping file for Sun–[34] systems

/usr/etc/config places its output files in the **../***config_file* directory:

mbglue.s	Short assembly language routines used for vectored interrupts
ioconf.c	Describes I/O devices attached to the system
makefile	Used with **make**(1) to build the system
*device_name***.h**	a set of header files (various *device_name*'s) containing devices which can be compiled into the system

SEE ALSO

gprof(1), **make**(1), **zs**(4S), **kgmon**(8), **savecore**(8), **swapon**(8)

The SYNOPSIS portion of each device entry in Section 4 of this manual.

Installing the SunOS System Software
System and Network Administration

NAME

copy_home – fetch default startup files for new home directories

SYNOPSIS

/home/*groupname***/copy_home /home/***groupname* **/home/***username*

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

Whenever **snap**(1) is used to add a new user account, the **copy_home** script in the selected primary group's home directory is executed to copy the default files to the new user's home directory, and also perform any additional custom setup.

copy_home copies default environment files, such as **.cshrc**, **.login**, and **.orgrc**, from a group's **defaults** directory to a new user's home directory. It is started by **user_agentd**(8) when **snap**(1) is used to create new home directories on a Sun386i home directory server.

Every new group created by **snap**(1) has a home directory, which can be accessed using **/home/***groupname*. **user_agentd**(8) copies the contents of the Sun386i's default group, **/home/users**, into the home directory of the new group. This includes the **Welcome.txt** file, the **copy_home** script, and the **defaults** directory . **copy_home** can be modified to customize the default setup environment for new users in the group.

SEE ALSO

snap(1), **user_agentd**(8)

Sun386i SNAP Administration
Sun386i Advanced Administration

NAME

dcheck – file system directory consistency check

SYNOPSIS

/usr/etc/dcheck [**–i** *numbers*] [*filesystem*]

DESCRIPTION

Note: **dcheck** has been superseded for normal consistency checking by **fsck**(8).

dcheck reads the directories in a file system and compares the link-count in each inode with the number of directory entries by which it is referenced. If the file system is not specified, **dcheck** checks a set of default file systems.

dcheck is fastest if the raw version of the special file is used, since the i-list is read in large chunks.

OPTIONS

–i *numbers*

numbers is a list of i-numbers; when one of those i-numbers turns up in a directory, the number, the i-number of the directory, and the name of the entry are reported.

FILES

Default file systems vary with installation.

SEE ALSO

fs(5), **fsck**(8), **clri**(8), **icheck**(8), **ncheck**(8)

DIAGNOSTICS

When a file turns up for which the link-count and the number of directory entries disagree, the relevant facts are reported. Allocated files which have 0 link-count and no entries are also listed. The only dangerous situation occurs when there are more entries than links; if entries are removed, so the link-count drops to 0, the remaining entries point to thin air. They should be removed. When there are more links than entries, or there is an allocated file with neither links nor entries, some disk space may be lost but the situation will not degenerate.

BUGS

Since **dcheck** is inherently two-pass in nature, extraneous diagnostics may be produced if applied to active file systems.

Inode numbers less than 2 are invalid.

NAME

devinfo – print out system device information

SYNOPSIS

/usr/etc/devinfo [**–pv**] [**–f** *promdevice*]

AVAILABILITY

This program is available for desktop SPARCsystems and for the SPARCsystem 600MP series only.

DESCRIPTION

devinfo displays the devices that the system knows about. The output will state the name of the device, its unit number, and whether a system device driver has claimed it. Since the internal system representation of this information is an *n*–ary tree, indentation is used to denote a parent-child relationship, and devices reported at the same indentation level are considered sibling devices.

OPTIONS

–f With the **–p** option, use the argument *promdevice* as the monitor PROM device, instead of the default */dev/openprom*.

–p Report the *monitor's* view of the attached devices, instead of the kernel's view.

–v Report hardware specifications such as register addresses and interrupt priorities for each device.

EXAMPLE

The following example displays the format of **devinfo** output:

```
example% devinfo
Node 'Sun 4/60', unit #0 (no driver)
        Node 'options', unit #0 (no driver)
        Node 'zs', unit #0
        Node 'zs', unit #1
        Node 'fd', unit #0
        Node 'audio', unit #0
        Node 'sbus', unit #0
                Node 'dma', unit #0
                Node 'esp', unit #0
                        Node 'st', unit #1 (no driver)
                        Node 'st', unit #0
                        Node 'sd', unit #3
                        Node 'sd', unit #2
                        Node 'sd', unit #1
                        Node 'sd', unit #0
                Node 'le', unit #0
                Node 'bwtwo', unit #0
        Node 'auxiliary-io', unit #0
        Node 'interrupt-enable', unit #0
        Node 'memory-error', unit #0
        Node 'counter-timer', unit #0
        Node 'eeprom', unit #0
```

FILES

/dev/kmem	to get kernel device information
/dev/openprom	to get the monitor's device information

NAME

dorfs – initialize, start and stop RFS automatically

SYNOPSIS

dorfs init *domain netspec* [*address*]
dorfs start [**–v**]
dorfs stop

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing the SunOS System Software* for information on how to install optional software.

DESCRIPTION

dorfs sets up necessary environment to run Remote File Sharing (RFS). You can also use it to start or stop RFS automatically, after its environment is initialized. The environment only needs to be set up once and **/usr/nserve/rfmaster** must exist before the environment is initialized. Descriptions of **/usr/nserve/rfmaster** are in **rfmaster**(5). You must be the super-user to run this command.

USAGE

Subcommands

init *domain netspec*[*address*]

domain is the name of the RFS domain. *netspec* is the name of a device file in the **/dev** directory which represents the streams-based transport provider on which RFS will run. Currently, **tcp** is the only accepted value for this field. *address* is the optional **tcp** port number on which the listener will listen. If unspecified, it defaults to **0x1450**. This subcommand only needs to be run once to initialize the environment. You do not need to rerun **dorfs** with the **init** argument, unless you want to change *netspec*. **/usr/nserve/rfmaster** must exists before you run this command to initialize the environment. To reinitialize the environment, you need to remove **/usr/nserve/***domain*, **/usr/nserve/***netspec*, **/var/net/nls/***netspec*/*address* and **/var/net/nls/***netspec***/dbf** beforehand.

start [**–v**]

Start RFS automatically. It also automatically advertises resources that are stored in **/etc/rstab** and mounts RFS resources that are stored in **/etc/fstab**.

–v Verify clients on mounts (see '**rfstart –v**').

stop

Takes down RFS by forced unmounting of all advertised resources, umounting all remotely mounted resources, executing **rfstop**, and stopping **listener**.

FILES

/etc/advtab
/etc/rstab
/var/net/nls/tcp/addr
/var/net/nls/tcp/dbf
/usr/nserve/domain
/usr/nserve/netspec
/usr/nserve/rfmaster

SEE ALSO

rfmaster(5), **dname**(8), **fumount**(8), **mount**(8), **nlsadmin**(8), **rfstart**(8), **rfstop**(8)

NAME

dump, rdump – incremental file system dump

SYNOPSIS

/usr/etc/dump [*options* [*arguments*]] *filesystem*
/usr/etc/dump [*options* [*arguments*]] *filename* ...

/usr/etc/rdump [*options* [*arguments*]] *filesystem*
/usr/etc/rdump [*options* [*arguments*]] *filename* ...

DESCRIPTION

dump backs up all files in *filesystem*, or files changed after a certain date, or a specified set of files and directories, to magnetic tape, diskettes, or files. *options* is a string that specifies **dump** options, as shown below. Any *arguments* supplied for specific options are given as subsequent words on the command line, in the same order as that of the *options* listed.

If **dump** is called as **rdump,** the dump device defaults to **dumphost:/dev/rmt8**.

If no *options* are given, the default is **9u**.

dump is normally used to back up a complete filesystem. To restrict the dump to a specified set of files and directories on one filesystem, list their names on the command line. In this mode the dump level is set to **0** and the **u** option is ignored.

OPTIONS

0–9 The "dump level." All files in the *filesystem* that have been modified since the last **dump** at a lower dump level are copied to the volume. For instance, if you did a "level **2**" dump on Monday, followed by a "level **4**" dump on Tuesday, a subsequent "level **3**" dump on Wednesday would contain all files modified or added since the "level **2**" (Monday) backup. A "level **0**" dump copies the entire filesystem to the dump volume.

a *archive-file*
Create a dump table-of-contents archive in the specified file, *archive-file*. This file can be used by **restore**(8) to determine whether a file is present on a dump tape, and if so, on which volume it resides. For further information on the use of a dump archive file, see **restore**(8).

b *factor* Blocking factor. Specify the blocking factor for tape writes. The default is 20 blocks per write. Note: the blocking factor is specified in terms of 512 bytes blocks, for compatibility with **tar**(1). The default blocking factor for tapes of density 6250 BPI and greater is 64. The default blocking factor for cartridge tapes (**c** option specified) is 126. The highest blocking factor available with most tape drives is 126.

c Cartridge. Use a cartridge instead of the standard half-inch reel. This sets the density to 1000 BPI, the blocking factor to 126, and the length to 425 feet. This option also sets the "inter-record gap" to the appropriate length. When cartridge tapes are used, and this option is *not* specified, **dump** will slightly miscompute the size of the tape. If the **b**, **d**, **s** or **t** options are specified with this option, their values will override the defaults set by this option.

d *bpi* Tape density. The density of the tape, expressed in BPI, is taken from *bpi.* This is used to keep a running tab on the amount of tape used per reel. Default densities are:

1/2" tape	1600 BPI
1/4" cartridge	1000 BPI
2.3-Gbyte 8mm tape	54,000 BPI

Unless a higher density is specified explicitly, **dump** uses its default density — even if the tape drive is capable of higher-density operation (for instance, 6250 BPI). Note: the density specified should correspond to the density of the tape device being used, or **dump** will not be able to handle end-of-tape properly. The **d** option is not compatible with the **D** option.

D Diskette. Specify diskette as the dump media.

f *dump-file*

Dump file. Use *dump-file* as the file to dump to, instead of **/dev/rmt8.** If *dump-file* is specified as '–', dump to the standard output. If the file name argument is of the form *machine:device*, dump to a remote machine. Since **dump** is normally run by *root,* the name of the local machine must appear in the **.rhosts** file of the remote machine. If the file name argument is of the form *user@machine:device*, **dump** will attempt to execute as the specified user on the remote machine. The specified user must have a **.rhosts** file on the remote machine that allows root from the local machine. If **dump** is called as **rdump,** the dump device defaults to **dumphost:/dev/rmt8**. To direct the output to a desired remote machine, set up an alias for **dumphost** in the file **/etc/hosts**.

n Notify. When this option is specified, if **dump** requires attention, it sends a terminal message (similar to **wall**(1)) to all operators in the "operator" group.

s *size* Specify the *size* of the volume being dumped to. When the specified size is reached, **dump** waits for you to change the volume. **dump** interprets the specified size as the length in feet for tapes, and cartridges and as the number of 1024 byte blocks for diskettes. The following are defaults:

1/2" tape	2300 feet
60-Mbyte 1/4" cartridge	425 feet
150-Mbyte 1/4" cartridge	700 feet
2.3-Gbyte 8mm	6000 feet
diskette	1422 blocks (Corresponds to a 1.44-Mbyte diskette, with one cylinder reserved for bad block information.)

t *tracks* Specify the number of tracks for a cartridge tape. The **t** option is not compatible with the **D** option. The following are defaults:

60-Mbyte 1/4" cartridge (Sun2 only)	4 tracks
60-Mbyte 1/4" cartridge (all other platforms)	9 tracks
150-Mbyte 1/4" cartridge	18 tracks

u Update the dump record. Add an entry to the file **/etc/dumpdates,** for each filesystem successfully dumped that includes the filesystem name, date, and dump level. This file can be edited by the super-user.

v After writing each volume of the dump, the media is rewound and is verified against the filesystem being dumped. If any discrepancies are found, dump will respond as if a write error had occurred; the operator will be asked to mount new media, and dump will attempt to rewrite the volume. Note that *any* change to the filesystem, even the update of the access time on a file will cause the verification to fail. Thus, the verify option can only be used on a quiescent filesystem.

w List the filesystems that need backing up. This information is gleaned from the files **/etc/dumpdates** and **/etc/fstab**. When the **w** option is used, all other options are ignored. After reporting, **dump** exits immediately.

W Like **w**, but includes all filesystems that appear in **/etc/dumpdates,** along with information about their most recent dump dates and levels. Filesystems that need backing up are highlighted.

FILES

/dev/rmt8	default unit to dump to
dumphost:/dev/rmt8	default remote unit to dump to if called as **rdump**
/dev/rst*	Sun386i cartridge tape dump device
/dev/rfd0a	Sun386i 1.44 megabyte 3.5-inch high density diskette drive dump device
/dev/rfdl0a	Sun386i 720 kilobyte 3.5-inch low density diskette drive dump device
/dev/rfd0c	Sun386i 1.44 megabyte 3.5-inch high density diskette drive dump device
/dev/rfdl0c	Sun386i 720 kilobyte 3.5-inch low density diskette drive dump device
/etc/dumpdates	dump date record
/etc/fstab	dump table: file systems and frequency
/etc/group	to find group *operator*
/etc/hosts	

SEE ALSO

bar(1), **fdformat**(1), **tar**(1), **wall**(1), **dump**(5), **fstab**(5), **restore**(8), **shutdown**(8)

DIAGNOSTICS

While running, **dump** emits many verbose messages.

Exit Codes

0	Normal exit.
1	Startup errors encountered.
3	Abort – no checkpoint attempted.

BUGS

Fewer than 32 read errors on the file system are ignored.

Each reel requires a new process, so parent processes for reels already written just hang around until the entire tape is written.

It is recommended that incremental dumps also be performed with the system running in single-user mode.

dump does not support multi-file multi-volume tapes.

EXAMPLES

Here are some examples of arguments which produce satisfactory results on a number of typical tape drives. Note that individual options can be in any order; however, the position of each following argument depends on the relative position of each option.

60-MByte cartridge (Sun2 only):	dump cdst 1000 425 4
60-MByte cartridge:	dump cdst 1000 425 9
150-MByte cartridge:	dump cdst 1000 700 18
1/2" tape:	dump dsb 1600 2300 126
2.3-GByte 8mm tape:	dump dsb 54000 6000 126

To make a full dump of a root filesystem on sd3, on a 150-MByte cartridge tape st0, use:

```
dump 0cdstfu 1000 700 18 /dev/rst0 /dev/sd3a
```

To make and verify an incremental dump at level 5 of the usr partition of sd3, on a 1/2" reel tape st1:

```
dump 5dsbfuv 1600 2300 126 /dev/rst1 /dev/sd3g
```

To make a full backup of the entire disk sd3, on a 2.3-GByte 8mm tape st2, use:

```
dump 0dsbfu 54000 6000 126 /dev/rst2 /dev/sd3c
```

NOTES

Operator Intervention

dump requires operator intervention on these conditions: end of volume, end of dump, volume write error, volume open error or disk read error (if there are more than a threshold of 32). In addition to alerting all operators implied by the **n** option, **dump** interacts with the operator on **dump**'s control terminal at times when **dump** can no longer proceed, or if something is grossly wrong. All questions **dump** poses *must* be answered by typing **yes** or **no**, as appropriate.

Since backing up a disk can involve a lot of time and effort, **dump** checkpoints at the start of each volume. If writing that volume fails for some reason, **dump** will, with operator permission, restart itself from the checkpoint after a defective volume has been replaced.

dump reports periodically, and in verbose fashion. Each report includes estimates of the percentage of the dump completed and how long it will take to complete the dump. The estimated time is given as *hours*:*minutes*.

Suggested Dump Schedule

It is vital to perform full, "level **0**", dumps at regular intervals. When performing a full dump, bring the machine down to single-user mode using **shutdown**(8). While preparing for a full dump, it is a good idea to clean the tape drive and heads.

Incremental dumps allow for convenient backup and recovery on a more frequent basis of active files, with a minimum of media and time. However there are some tradeoffs. First, the interval between backups should be kept to a minimum (once a day at least). To guard against data loss as a result of a media failure (a rare, but possible occurrence), it is a good idea to capture active files on (at least) two sets of dump volumes. Another consideration is the desire to keep unnecessary duplication of files to a minimum to save both operator time and media storage. A third consideration is the ease with which a particular backed-up version of a file can be located and restored. The following four-week schedule offers a reasonable trade-off between these goals.

	Sun	*Mon*	*Tue*	*Wed*	*Thu*	*Fri*
Week 1:	**Full**	**5**	**5**	**5**	**5**	**3**
Week 2:		**5**	**5**	**5**	**5**	**3**
Week 3:		**5**	**5**	**5**	**5**	**3**
Week 4:		**5**	**5**	**5**	**5**	**3**

Although the Tuesday — Friday incrementals contain "extra copies" of files from Monday, this scheme assures that any file modified during the week can be recovered from the previous day's incremental dump.

Process Priority of dump

dump uses multiple processes to allow it to read from the disk and write to the media concurrently. Due to the way it synchronizes between these processes, any attempt to run dump with a **nice** (process priority) of '–5' or better will likely make **dump** run *slower* instead of faster.

NAME

eeprom – EEPROM display and load utility

SYNOPSIS

SUN-3, SUN-4 SYSTEMS

eeprom [–] [**–c**] [**–i**] [**–f** *device*] [*field*[=*value*] . . .]

Desktop SPARCsystems, SPARCsystem 600MP SERIES

eeprom [–] [**–f** *device*] [*field*[=*value*] . . .]

DESCRIPTION

eeprom displays or changes the values of fields in the EEPROM. It processes fields in the order given. When processing a *field* accompanied by a *value*, **eeprom** makes the indicated alteration to the EEPROM; otherwise it displays the *field*'s value. When given no field specifiers, **eeprom** displays the values of all EEPROM fields. A '–' flag specifies that fields and values are to be read from the standard input (one *field* or *field*=*value* per line).

Only the super-user may alter the EEPROM contents.

eeprom verifies the EEPROM checksums and complains if they are incorrect; if the **–i** flag is specified, erroneous checksums are ignored. If the **–c** flag is specified, all incorrect checksums are recomputed and corrected in the EEPROM.

OPTIONS

–c	Correct bad checksums. (Ignored on Desktop SPARCsystems and SPARCsystem 600MP series.)
–i	Ignore bad checksums. (Ignored on Desktop SPARCsystems and SPARCsystem 600MP series.)
–f *device*	Use *device* as the EEPROM device.

FIELDS and VALUES

SUN-3, SUN-4 SYSTEMS

hwupdate	a valid date (including **today** and **now**)
memsize	8 bit integer (megabytes of memory on machine)
memtest	8 bit integer (megabytes of memory to test)
scrsize	**1024x1024**, **1152x900**, **1600x1280**, or **1440x1440**
watchdog_reboot	**true** or **false**
default_boot	**true** or **false**
bootdev	*charchar*(*hex-int*,*hex-int*,*hex-int*) (with *char* a character, and *hex-int* a hexadecimal integer.)
kbdtype	8 bit integer (0 for all Sun keyboards)
keyclick	**true** or **false**
console	**b&w** or **ttya** or **ttyb** or **color**
custom_logo	**true** or **false**
banner	banner string
diagdev	%c%c (%x,%x,%x) — diagnostic boot device
diagpath	diagnostic boot path
ttya_no_rtsdtr	**true** or **false**
ttyb_no_rtsdtr	**true** or **false**
ttya_use_baud	**true** or **false**
ttyb_use_baud	**true** or **false**
ttya_baud	baud rate (16-bit decimal integer)
ttyb_baud	baud rate (16-bit decimal integer)
columns	number of columns on screen (8-bit integer)
rows	number of rows on screen (8-bit integer)
secure	**none**, **command**, or **full**. If **secure=none** the PROM monitor runs in the non-secure mode. In this mode all PROM monitor commands are allowed with no

password required. If **secure=command** the PROM monitor is in the command secure mode. In this mode, only the **b** (boot) command with no parameters and the **c** (continue) command with no parameters may be entered without a password being required. Any other command requires that the PROM monitor password be entered. If **secure=full** the PROM monitor is in the fully secure mode. In this mode, only the **c** (continue) command with no parameters may be entered without a password being required. Entry of any other command requires that the PROM monitor password be entered. Note: the system will not auto-reboot in fully secure mode. The PROM monitor password must be entered before the boot process will take place. When changing the security mode from non-secure to either command secure or fully secure, **eeprom** prompts for the entry and re-entry of a new PROM password as in the **passwd**(1) command. Changing from one secure mode to the other secure mode, or to the non-secure mode does not prompt for a password. Changing to non-secure mode erases the password.

bad_login number of bad login tries (16-bit unsigned integer, 0 if **reset**). The field **bad_login** maintains the count of bad login tries. It may be reset to zero (0) by specifying **bad_login=reset**.

password PROM monitor password (8-bytes). The content of the **password** field is never displayed to any user. If the security mode is not **none**, the super-user may change the PROM monitor password by entering:

example# eeprom password=

eeprom prompts for a new password to be entered and re-entered.

Desktop SPARCsystems, SPARCsystem 600MP SERIES

hardware-revision 7 chars (for example, **30Mar88**)

selftest-#megs 32 bit decimal integer (megabytes of memory to test)

watchdog-reboot? **true** or **false**; **true** to reboot after watchdog reset

boot-from A string specifying boot string (for example, **le( )vmunix**); defaults to **vmunix**. (SPARCstation 1 systems only)

keyboard-click? **true** or **false**; **true** to enable clicking of keys on each keystroke

input-device A string specifying one of **keyboard**, **ttya**, or **ttyb**; if the specified device is unavailable, **ttya** is used for both input and output *only* if input-device specified the keyboard *and* output-device specified the screen.

output-device A string specifying one of **screen**, **ttya**, or **ttyb**; if the specified device is unavailable, **ttya** is used for *both* input and output *only* if input-device specified the keyboard *and* output-device specified the screen.

oem-banner? **true** or **false**; **true** to use custom banner string instead of Sun banner

oem-banner 80 chars for custom banner string

oem-logo? **true** or **false**; **true** to display custom logo instead of Sun logo

oem-logo Name of file (in **iconedit** format) containing custom logo.

boot-from-diag 80 chars specifying diag boot string (for example, **sd( )dexec**); defaults to **le( )vmunix**. (SPARCstation 1 systems only)

ttya-mode 16 chars to specify 5 comma-separated fields of configuration information (for example, **1200,8,1,n,–**); defaults to **9600,8,1,n,–**.

Fields, in left-to-right order, are:

baud rate:	110, 300, 1200, 4800, 9600...
data bits:	5, 6, 7, 8
parity:	n(none), e(even), o(odd), m(mark), s(space)
stop bits:	1, 1.5, 2
handshake:	–(none), h(hardware:rts/cts), s(software:xon/xoff)

ttyb-mode 16 chars to specify 5 comma-separated fields of configuration information (for example, **1200,7,1,n,s**); defaults to **9600,8,1,n,–**.

Fields, in left-to-right order, are:

baud rate: 110, 300, 1200, 4800, 9600...
data bits: 5, 6, 7, 8
stop bits: 1, 1.5, 2
parity: n(none), e(even), o(odd), m(mark), s(space)
handshake: –(none), h(hardware:rts/cts), s(software:xon/xoff)

ttyb-rts-dtr-off **true** or **false**. Defaults to **false**.

ttya-rts-dtr-off **true** or **false**. Defaults to **false**.

ttya-ignore-cd **true** or **false**. Defaults to **true**.

ttyb-ignore-cd **true** or **false**; **true** to ignore the **CARRIER DETECT** line. Defaults to **true**.

screen-#rows number of rows on output device; defaults to 34 (for some devices actual values used may be less)

screen-#columns number of columns on output device; defaults to 80 (for some devices actual values used may be less)

auto-boot? **true** or **false**; **true** to boot on power-on

scsi-initiator-id An integer between 0 and 7 that specifies the SCSI initiator ID of the onboard SCSI host adapter.

sd-targets An array of 8 integers that map SCSI disk unit numbers to SCSI target numbers. The unit number is used to index into this string. The default settings are **31204567**, which means that unit 0 maps to target 3, unit 1 maps to target 1, and so on. (SPARCstation 1 systems only)

st-targets An array of 8 integers that map SCSI tape unit numbers to SCSI target numbers. The unit number is used to index into this string. The default settings are **45670123**, which means that unit 0 maps to target 4, unit 1 maps to target 5, and so on. (SPARCstation 1 systems only)

sunmon-compat? **true** or **false**. Defaults to **true**.

security-mode **none**, **command**, or **full**. See above, the **secure** field.

security-password password associated with **security-mode**. The content of the **security-password** field is never displayed to any user. If the security mode is not **none**, the super-user may change the PROM monitor password by entering:

example# eeprom security-password=

eeprom prompts for a new password to be entered and re-entered.

security-#badlogins number of bad login tries (16-bit unsigned integer, 0 if **reset**). The field **security-#badlogins** maintains the count of bad login tries. It may be reset to zero (0) by specifying **security-#badlogins=reset**.

sbus-probe-list Defaults to 0123.

fcode-debug? **true** or **false**. Defaults to **false**.

last-hardware-update Date the CPU board was manufactured or upgraded to the latest hardware revision. The format is a human-readable date string, such as **23May89**.

testarea Defaults to 0.

mfg-switch? **true** or **false**. Defaults to **false**.

diag-switch? **true** or **false**. Defaults to **true**.

boot-file Default boot file and arguments when **diag-switch?** is **false**; defaults to **vmunix**. (SPARCstation 2, SPARCsystem 600MP series only)

boot-device Default boot device specifier when **diag-switch?** is **false**; defaults to **disk**. (SPARCstation 2, SPARCsystem 600MP series only)

diag-file Default boot file and arguments when **diag-switch?** is **true**; defaults to **vmunix** (SPARCstation 2, SPARCsystem 600MP series only)

diag-device Default boot device specifier when **diag-switch?** is **true**; defaults to **net**. (SPARCstation 2, SPARCsystem 600MP series only)

local-mac-address? **true** or **false**. Defaults to **false**. (SPARCstation 2, SPARCsystem 600MP series only)

nvramrc Contents of NVRAMRC. (SPARCstation 2, SPARCsystem 600MP series only)

use-nvramrc?	**true** or **false**; **true** to execute commands in NVRAMRC during PROM start-up; defults to **false**. (SPARCstation 2, SPARCsystem 600MP series only)
vme-ipi-probe-list	Defaults to 0123. (SPARCsystem 600MP series only)
skip-vme-loopback?	**true** or **false**. **true** to skip some of the VME loopback tests on POST. Defaults to **false**. (SPARCsystem 600MP series only)

FILES

SUN-3, SUN-4 SYSTEMS

/dev/eeprom

Desktop SPARCsystems, SPARCsystem 600MP SERIES

/dev/openprom

SEE ALSO

passwd(1)

PROM User's Manual
Open Boot PROM 2.0 Toolkit User's Guide

NAME

etherd, rpc.etherd – Ethernet statistics server

SYNOPSIS

/usr/etc/rpc.etherd *interface*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing the SunOS System Software* for information on how to install optional software.

DESCRIPTION

etherd is a server which puts *interface* into promiscuous mode, and keeps summary statistics of all the packets received on that interface. It responds to RPC requests for the summary. You must be root to run **etherd**.

interface is a networking interface such as **ie0**, **ie1**, **ec0**, **ec1** and **le0**.

traffic(1C) displays the information obtained from **etherd** in graphical form.

SEE ALSO

traffic(1C)

NAME

etherfind – find packets on Ethernet

SYNOPSIS

etherfind [**–d**] [**–n**] [**–p**] [**–r**] [**–t**] [**–u**] [**–v**] [**–x**] [**–c** *count*] [**–i** *interface*] [**–l** *length*] *expression*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing the SunOS System Software* for information on how to install optional software.

DESCRIPTION

etherfind prints out the information about packets on the ethernet that match the boolean *expression*. The short display, without the **–v** option, displays only the destination and src (with port numbers). When an Internet packet is fragmented into more than one ethernet packet, all fragments except the first are marked with an asterisk. With the **–v** option, the display is much more verbose, giving a trace that is suitable for analyzing many network problems. You must be root to invoke **etherfind**.

OPTIONS

–d Print the number of dropped packets. Not necessarily reliable.

–n Do not convert host addresses and port numbers to names.

–p Normally, the selected interface is put into promiscuous mode, so that **etherfind** has access to all packets on the ethernet. However, when the **–p** flag is used, the interface will not go promiscuous.

–r RPC mode: treat each packet as an RPC message, printing the program and procedure numbers. Routing packets are also more fully decoded using this option, and Network Information Service (NIS) and NFS requests have their arguments printed.

–t Timestamps: precede each packet listing with a time value in seconds and hundredths of seconds since the first packet.

–u Make the output line buffered.

–v Verbose mode: print out some of the fields of TCP and UDP packets.

–x Dump the packet in hex, in addition to the line printed for each packet by default. Use the **–l** option to limit this printout.

–c *count*

Exit after receiving *count* packets. This is sometimes useful for dumping a sample of ethernet traffic to a file for later analysis.

–i *interface*

etherfind listens on *interface*. The program **netstat**(8C) when invoked with the **–i** flag lists all the interfaces that a machine has.

–l *length*

Use with the **–x** option to limit the number of bytes printed out.

expression

The syntax of *expression* is similar to that used by **find**(1). Here are the allowable primaries.

dst *destination*

True if the destination field of the packet is *destination*, which may be either an address or a name.

src *source*

True if the source field of the packet is *source*, which may be either an address or a name.

host *name*
True if either the source or the destination of the packet is *name*.

between *host1 host2*
True if either the source of the packet is *host1* and the destination *host2*, or the source is *host2* and the destination *host1*.

dstnet *destination*
True if the destination field of the packet has a network part of *destination*, which may be either an address or a name.

srcnet *source*
True if the source field of the packet has a network part of *source*, which may be either an address or a name.

srcport *port*
True if the packet has a source port value of *port*. This will check the source port value of either UDP or TCP packets (see **tcp**(4P)), and **udp**(4P)). The *port* can be a number or a name used in **/etc/services**.

dstport *port*
True if the packet has a destination port value of *port*. The *port* can be a number or a name.

less *length*
True if the packet has a length less than or equal to *length*.

greater *length*
True if the packet has a length greater than or equal to *length*.

–proto *protocol*
True if the packet is an IP packet (see **ip**(4P)) of protocol type *protocol*. *Protocol* can be a number or one of the names **icmp**, **udp**, **nd**, or **tcp**.

byte *byte op value*
True if byte number *byte* of the packet is in relation *op* to *value*. Legal values for *op* are **+**, **<**, **>**, **&**, and **|**. Thus **4=6** is true if the fourth byte of the packet has the value 6, and **20&0xf** is true if byte twenty has one of its four low order bits nonzero.

broadcast
True if the packet is a broadcast packet.

arp True if the packet is an ARP packet (see **arp**(4P)).

rarp True if the packet is a rarp packet.

–ip True if the packet is an IP packet.

–decnet
True if the packet is a DECNET packet.

–apple True if the packet is an AppleTalk protocol packet.

The primaries may be combined using the following operators (in order of decreasing precedence):

A parenthesized group of primaries and operators (parentheses are special to the Shell and must be escaped).

The negation of a primary (‘**not**’ is the unary *not* operator).

Concatenation of primaries (the *and* operation is implied by the juxtaposition of two primaries, or can be specified with '**and**').

Alternation of primaries ('**or**' is the *or* operator).

EXAMPLE

To find all packets arriving at or departing from the host **sundown**, or that are ICMP packets:

example% etherfind host sundown or proto icmp

SEE ALSO

find(1), **traffic**(1C), **arp**(4P), **ip**(4P), **nit**(4P) **tcp**(4P), **udp**(4P), **netstat**(8C)

BUGS

The syntax is painful.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

exportfs – export and unexport directories to NFS clients

SYNOPSIS

/usr/etc/exportfs [**–aiuv**] [**–o** *options*] [*pathname*]

DESCRIPTION

exportfs makes a local directory or filename available for mounting over the network by NFS clients. It is normally invoked at boot time by the **/etc/rc.local** script, and uses information contained in the **/etc/exports** file to export *pathname* (which must be specified as a full pathname). The super-user can run **exportfs** at any time to alter the list or characteristics of exported directories and filenames. Directories and files that are currently exported are listed in the file **/etc/xtab**.

With no options or arguments, **exportfs** prints out the list of directories and filenames currently exported.

OPTIONS

–a All. Export all pathnames listed in **/etc/exports**, or if **–u** is specified, unexport all of the currently exported pathnames.

–i Ignore the options in **/etc/exports.** Normally, **exportfs** will consult **/etc/exports** for the options associated with the exported pathname.

–u Unexport the indicated pathnames.

–v Verbose. Print each directory or filename as it is exported or unexported.

–o *options*

Specify a comma-separated list of optional characteristics for the pathname being exported. *options* can be selected from among:

ro Export the pathname read-only. If not specified, the pathname is exported read-write.

rw=*hostname*[**:***hostname*]...

Export the pathname read-mostly. Read-mostly means exported read-only to most machines, but read-write to those specified. If not specified, the pathname is exported read-write to all.

anon=*uid*

If a request comes from an unknown user, use UID as the effective user ID. Note: root users (UID 0) are always considered "unknown" by the NFS server, unless they are included in the **root** option below. The default value for this option is the UID of the user "nobody". If the user "nobody" does not exist then the value 65534 is used. Setting the value of "anon" to 65535 disables anonymous access. Note: by default secure NFS accepts insecure requests as anonymous, and those wishing for extra security can disable this feature by setting "anon" to 65535.

root=*hostname*[**:***hostname*]...

Give root access only to the root users from a specified *hostname*. The default is for no hosts to be granted root access.

access=*client*[**:***client*]...

Give mount access to each *client* listed. A *client* can either be a hostname, or a netgroup (see **netgroup**(5)). Each *client* in the list is first checked for in the **/etc/hosts** database, and then the **/etc/netgroup** database. The default value allows any machine to mount the given directory.

secure Require clients to use a more secure protocol when accessing the directory.

FILES

/etc/exports	static export information
/etc/xtab	current state of exported pathnames
/etc/netgroup	

SEE ALSO

exports(5), **netgroup**(5), **showmount**(8)

WARNINGS

You cannot export a directory that is either a parent- or a sub-directory of one that is currently exported and *within the same filesystem*. It would be illegal, for example, to export both **/usr** and **/usr/local** if both directories resided in the same disk partition.

NAME

extract_files, list_files – extract/list files from a release media

SYNOPSIS

/usr/etc/install/extract_files *devname category* [**–a** *arch*] [**–r** *release*] [**–f** *file* ...]

/usr/etc/install/list_files *devname category* [**–a** *arch*] [**–r** *release*]

DESCRIPTION

Given the name of a *category*, and optionally, a file (or a list of files) and/or a kernel architecture type, **extract_files** extracts all files contained in *category* from *devname*, or optionally extracts selected file(s) from *files* in *category* from *devname*. Note: **extract_files** does not pre-compute the disk size that needs to hold the extracted data.

list_files performs the same function as **extract_files** except that files are not extracted, instead, they are listed as contents of a given *category*.

The **–a** and **–r** options are provided for CD devices, which can contain multiple releases for multiple architecture types. Users normally do not need to specify these values, they default to the current running system.

devname is the media name on which files are kept. Examples are **st**, **sr**, **mt**... etc.

category is a suninstall category name which can be found by running **/usr/etc/install/toc_xlat**.

OPTIONS

–a *arch* Specify the kernel architecture type such as **sun4**, **sun4c**, ... etc. Defaults to the current system's kernel architecture.

–r *release* Specify a software release such as **4_1**, **4_1_1**, ... etc. Defaults to the current system's software release.

–f *file* Specify a file or a list of file to extract.

EXAMPLES

The following examples show some common uses of **extract_files** and **list_files.**

```
example# cd /usr
example# /usr/etc/install/extract_files st0 Text –f ./bin/addbib ./bin/eqn
```

Extracts from tape device **st0** the files **./bin/addbib** and **./bin/eqn** from the category **Text**.

```
example# cd /usr
example# /usr/etc/install/extract_files st0 Text –f ./bin/checkeq
```

Extracts from tape device **st0** the file **./bin/checkeq** from the category **Text**.

```
example# /usr/etc/install/list_files sr0 Text –a sun4c
```

Lists all files in category **Text** of **sun4c** kernel architecture type from CD device **sr0**.

```
example# /usr/etc/install/list_files sr0 TLI –a sun4c –r 4_1_1
```

Lists all files in category **TLI** of **sun4c** kernel architecture type from CD device **sr0**, under the **4_1_1** software release.

BUGS

extract_files and **list_files** do not support floppy disc device.

NAME

extract_patch – extract and execute patch files from installation tapes

SYNOPSIS

extract_patch [**–d***device* [**–r***remote-host*]] [**–p***patch-name*] [**–DEFAULT**]

DESCRIPTION

extract_patch extracts a patch from a release tape onto the current system. If no options are specifed, it prompts for input as to the patch name, tape device, or remote hostname from which to the software is to be installed. If the named patch cannot be found, a list of valid patches are printed.

If the named patch is found then the patch is extracted from the tape onto the system. If there is a **README** file in the extracted contents then the user is given a chance to view it. If there is a patch installation program the user is given a chance to run it.

Patches must appear in the tape's table of contents, and must have a name that starts with "Patch_".

OPTIONS

–d*device*

Install from the indicated tape drive, such as **st0**, or **mt0**.

–r*remote-host*

Install from the device given in the **–d** option on the indicated remote host.

–p*patch-name*

Specifes the name of the patch to extract.

–DEFAULT

Execute the installation script using all default values. Otherwise the installation script prompts for any optional values.

SEE ALSO

extract_unbundled(8)

NAME

extract_unbundled – extract and execute unbundled-product installation scripts

SYNOPSIS

extract_unbundled [**–d***device* [**–r***remote-host*]] [**–DEFAULT**]

DESCRIPTION

extract_unbundled extracts and executes the installation scripts from release tapes and diskettes for Sun unbundled software products. If no options are specifed, it prompts for input as to the tape device, or remote hostname from which to the software is to be installed. For information about installing a specific product, refer to the installation manual that accompanies that product.

OPTIONS

–d*device*

Install from the indicated tape drive, such as **st0 mt0**, or **fd0c**.

–r*remote_host*

Install from the device given in the **–d** option on the indicated remote host.

–DEFAULT

Execute the installation script using all default values. Otherwise the installation script prompts for any optional values.

BUGS

The **–r** option cannot be used with floppy diskettes.

NAME

fastboot, fasthalt – reboot/halt the system while disabling disk checking

SYNOPSIS

/usr/etc/fastboot [*boot-options*]

/usr/etc/fasthalt [*halt-options*]

DESCRIPTION

fasthalt halts the system (like **halt**(8)) and **fastboot** reboots the system (like **reboot**(8)). Both these programs disable subsequent checking of the file systems on the next reboot. This is done by creating a file **/fastboot**. The system startup script **/etc/rc** looks for this file and, if present, skips the normal invocation of **fsck**(8).

FILES

/usr/etc/fastboot
/etc/rc
/fastboot

SEE ALSO

fsck(8), **halt**(8), **init**(8), **rc**(8), **reboot**(8)

NAME

fpuversion4 – print the Sun-4 FPU version

SYNOPSIS

/usr/etc/fpuversion4

AVAILABILITY

Sun-4 systems only.

DESCRIPTION

fpuversion4 reads the **%fsr** register to determine the FPU version installed on a Sun-4. The printed version field contains a value in the range 0-7; by SPARC convention 7 indicates that no FPU is installed, so floating-point instructions are always emulated in the kernel.

NAME

fsck – file system consistency check and interactive repair

SYNOPSIS

/usr/etc/fsck –p [**–f**] [**–w**] [**–l** *number*] [*filesystem* **...**]

/usr/etc/fsck [**–b** *block#*] [**–w**] [**–y**] [**–n**] [**–c**] [*filesystem* **...**]

DESCRIPTION

fsck is a program that checks and repairs file system consistency. It can operate in two modes, "preen" and interactive. "Preen" is a non-interactive mode which only repairs a subset of file system inconsistencies. The interactive mode allows users to audit and repair any inconsistencies.

"Preen" Mode

With the **–p** option, **fsck** audits and automatically repairs ("preens") inconsistencies on a set of file systems. If a list of file systems is specified on the command line, **fsck** sequentially checks each one; otherwise, **fsck** reads the table **/etc/fstab** to determine the file systems to check. It then inspects disks in parallel, taking advantage of I/O overlap to check the file systems quickly. The number of disks checked in parallel can be limited using the **–l** option. This helps systems that do not have sufficient memory to run enough **fsck**s to check all disks in parallel. Preen mode is normally used in the **/etc/rc** script during automatic reboot.

Each file system's super block **clean flag** is examined and only those file systems not marked **clean** or **stable** are checked. A message identifying the name of the device and its file system's state is printed if the clean flag indicates that checking is not necessary. If the **–f** (force) option is in effect, **fsck** checks the file system regardless of the state of its clean flag.

Only partitions marked in **/etc/fstab** with a file system type of "4.2" and a non-zero pass number are checked. If the force option is in effect, file systems with pass number 1 (typically **/**, **/usr**, and **/usr/kvm**) are checked one at a time. When pass 1 completes, all remaining file systems are checked, running one process per disk drive. If the force option is not in effect (the default case), all file systems with non-zero pass numbers are checked in as parallel a manner as possible.

fsck corrects innocuous inconsistencies such as: unreferenced inodes, too-large link counts in inodes, missing blocks in the free list, blocks appearing in the free list and also in files, or incorrect counts in the super block, automatically. It displays a message for each inconsistency corrected that identifies the nature of, and file system on which, the correction is to take place. After successfully correcting a file system, **fsck** sets the file system's super block **clean flag** to **stable,** prints the number of files on that file system, the number of used and free blocks, and the percentage of fragmentation.

If **fsck** encounters other inconsistencies that it cannot fix automatically, it does not change the state of the super block **clean flag** and exits with an abnormal return status (and the reboot fails). A list of file systems containing such uncorrectable inconsistencies is printed just before **fsck** exits.

If sent a QUIT signal while preening the file systems listed in **/etc/fstab**, **fsck** finishes the file system checks, then exit with an abnormal return status and the automatic reboot fails. This is useful when you wish to finish the file system checks, but do not want the machine to come up multiuser.

Interactive Mode

Without the **–p** option, **fsck** audits and interactively repairs inconsistent conditions on file systems. File systems are checked regardless of the state of their **clean flag**. In this case, **fsck** asks for confirmation before attempting any corrections. Inconsistencies other than those mentioned above can often result in some loss of data. The amount and severity of data lost can be determined from the diagnostic output.

The default action for each correction is to wait for the operator to respond either **yes** or **no**. If the operator does not have write permission on the file system, **fsck** defaults to a **–n** (no corrections) action.

If no file systems are given to **fsck** then a default list of file systems is read from the file **/etc/fstab**.

Inconsistencies checked in order are as follows:

- Blocks claimed by more than one inode or the free list.
- Blocks claimed by an inode or the free list outside the range of the file system.
- Incorrect link counts.
- Incorrect directory sizes.
- Bad inode format.
- Blocks not accounted for anywhere.
- Directory checks, file pointing to unallocated inode, inode number out of range.
- Super Block checks: more blocks for inodes than there are in the file system.
- Bad free block list format.
- Total free block and/or free inode count incorrect.
- Clean flag state.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the **lost+found** directory. The name assigned is the inode number. If the **lost+found** directory does not exist, it is created. If there is insufficient space its size is increased.

A file system may be specified by giving the name of the cooked or raw device on which it resides, or by giving the name of its mount point. If the latter is given, **fsck** finds the name of the device on which the file system resides by looking in **/etc/fstab**.

OPTIONS

–b Use the block specified immediately after the flag as the super block for the file system. Block 32 is always an alternate super block.

–f Force checking of file systems regardless of the state of their super block **clean flag .**

–l *number*
Limits the *number* of **fsck**s that are run concurrently in preen mode. (See **–p**.)

–w Check writable file systems only.

–y Assume a **yes** response to all questions asked by **fsck**; this should be used with extreme caution, as it is a free license to continue, even after severe problems are encountered.

-n Assume a **no** response to all questions asked by **fsck**; do not open the file system for writing.

–p Audit and automatically repair inconsistencies on file systems whose state is either **active** or **unknown**. If no other options are specified, **fsck** inspects file systems listed in **/etc/fstab** in parallel, simultaneously checking one file system per disk. If file systems are specified on the command line, inspection is sequential.

–c If the file system is in the old (static table) format, convert it to the new (dynamic table) format. If the file system is in the new format, convert it to the old format provided the old format can support the file system configuration. In interactive mode, **fsck** lists the direction the conversion is to be made and asks whether the conversion should be done. If a negative answer is given, no further operations are done on the filesystem. In preen mode, the direction of the conversion is listed and done if possible without user interaction. Conversion in preen mode is best used when all the file systems are being converted at once. The format of a file system can be determined from the first line of output from **dumpfs**(8)

FILES

/etc/fstab	default list of file systems to check
/etc/mtab	list of mounted file systems

EXIT STATUS

0 Either no errors detected or all errors were corrected.

2 A QUIT signal was caught while preening the file systems listed in **/etc/fstab**; abort the reboot procedure.

4 Errors on the root or a mounted file system were corrected. The system must be rebooted.

8 Some uncorrected errors exist on one or more of the file systems checked, there was a syntax error, or some other operational error occurred.

12 An INTERRUPT signal was caught during processing.

DIAGNOSTICS

The diagnostics produced by **fsck** are fully enumerated and explained in the *System and Network Administration Manual* with the following additions.

1. After checking a file system in non-preening mode, the question:

   CLEAN FLAG IN SUPERBLOCK IS WRONG; FIX?

   is asked if the file system's clean state is wrong. A "yes" response instructs **fsck** to reset this state to **active** if there are inconsistencies, or to **stable** if there are no uncorrected inconsistencies. A "no" response instructs **fsck** to leave the existing state unchanged.

2. When **fsck** is run in preen mode, the file systems that need additional attention are normally scrolled off the screen. With the enhanced version of **fsck** these file systems listed as **fsck** exits. This provides the user with a list of the the file systems that require attention. An example follows:

   .
   .
   .
   /dev/rsd6a: UNEXPECTED INCONSISTENCY; RUN fsck MANUALLY.

   THE FOLLOWING FILE SYSTEMS HAD AN UNEXPECTED INCONSISTENCY:
   /dev/rsd6a (/sd6a), /dev/rsd6b (/sd6b)
   Reboot failed...help!

CAVEAT

Because super block consistency checks are not made when the -b option is used, it is recommended that -b be augumented with the -n option to verify fsck actions. Once you are satisfied that the fsck actions are correct, then remove the -n.

SEE ALSO

fs(5), **fstab**(5), **dumpfs**(8), **newfs**(8), **mkfs**(8), **panic**(8S), **reboot**(8), **rexecd**(8C), **ypserv**(8)

System and Network Administration

NAME

fsirand – install random inode generation numbers

SYNOPSIS

fsirand [**–p**] *special*

DESCRIPTION

fsirand installs random inode generation numbers on all the inodes on device *special,* and also installs a filesystem ID in the superblock. This helps increase the security of filesystems exported by NFS.

fsirand must be used only on an unmounted filesystem that has been checked with **fsck**(8). The only exception is that it can be used on the root filesystem in single-user mode, if the system is immediately re-booted afterwords.

OPTIONS

–p Print out the generation numbers for all the inodes, but do not change the generation numbers.

SEE ALSO

fsck(8)

NAME

getty – set terminal mode

SYNOPSIS

/usr/etc/getty [*type* [*tty*]]

Sun386i SYSTEM SYNOPSIS

/usr/etc/getty [**–n**] [*type* [*tty*]]

DESCRIPTION

getty, which is invoked by **init**(8), opens and initializes a tty line, reads a login name, and invokes **login**(1).

The *tty* argument is the name of the character-special file in **/dev** that corresponds to the terminal. If there is no *tty* argument, or the argument is '–', the tty line is assumed to be opened as file descriptor **0**.

The *type* argument, if supplied, is used as an index into the **gettytab**(5) database—to determine the characteristics of the line. If this argument is absent, or if there is no such entry, the default entry is used. If there is no **/etc/gettytab** file, a set of system-supplied defaults is used.

When the indicated entry is located, **getty** clears the terminal screen, prints a banner heading, and prompts for a login name. Usually, either the banner or the login prompt includes the system's hostname.

Next, **getty** prompts for a login and reads the login name, one character at a time. When it receives a null character (which is assumed to be the result pressing the BREAK , or "interrupt" key), **getty** switches to the entry **gettytab** entry named in the **nx** field. It reinitializes the line to the new characteristics, and then prompts for a login once again. This mechanism typically is used to cycle through a set of line speeds (baud rates) for each terminal line. For instance, a rotary dialup might have entries for the speeds: 300, 1200, 150, and 110 baud, with each **nx** field pointing to the next one in succession.

The user terminates login input line with a NEWLINE or RETURN character. The latter is preferable; it sets up the proper treatment of RETURN characters (see **tty**(4)). **getty** checks to see if the terminal has only upper-case alphabetical characters. If all alphabetical characters in the login name are in upper case, the system maps them along with all subsequent upper-case input characters to lower-case internally; they are displayed in upper case for the benefit of the terminal. To force recognition of an upper-case character, the shell allows them to be quoted (typically by preceding each with a backslash, '\').

Finally, **getty** calls **login**(1) with the login name as an argument.

getty can be set to time out after a certain interval; this hangs up dial-up lines if the login name is not entered in time.

Sun386i SYSTEM DESCRIPTION

For Sun386i system, the value of *type* is the constant **Sun**, for the console frame buffer.

Sun386i SYSTEM OPTIONS

–n invoke the full screen login program **logintool**(8), and optionally the "New User Accounts" feature. May only be used on a frame buffer. Unless removed from the console entry in */etc/ttytab*, this option is in effect by default.

FILES

/etc/gettytab

SEE ALSO

login(1), **ioctl**(2), **tty**(4), **fbtab**(5), **gettytab**(5), **svdtab**(5), **ttytab**(5), **init**(8), **logintool**(8)

DIAGNOSTICS

ttyxx: **No such device or address.**

ttyxx: **No such file or directory.**

A terminal which is turned on in the **ttys** file cannot be opened, likely because the requisite lines are either not configured into the system, the associated device was not attached during boot-time system configuration, or the special file in **/dev** does not exist.

NAME

gpconfig – initialize the VME Graphics Processor or SBus GS graphics accelerator

SYOPNSIS

/usr/etc/gpconfig [**–u** *ucode-filename*] *gpunit* [[**–b**] [**–f**] [**–F**] *fbunit*...]

DESCRIPTION

gpconfig binds a VME color frame buffer (either the **cgtwo**(4S) or **cgnine**(4S)) to the VME GP (Graphics Processor), then loads and starts the appropriate microcode in the GP.

gpconfig also loads and starts the appropriate microcode in the SBus GS **cgtwelve**(4S) integrated accelerator and frame buffer.

For example, the command line:

/usr/etc/gpconfig gpone0 cgtwo0

binds the VME **cgtwo0** frame buffer board to the VME **gpone0** Graphics Processor. The device **/dev/gpone0a** then refers to the combination of **gpone** and **cgtwo0**.

All VME frame buffer boards bound to a VME GP device must be configured to the same width and height.

The standard version of the file **/etc/rc.local** contains the following **gpconfig** command line:

/usr/etc/gpconfig –f –b

For VME systems, this binds **gpone0** and either **cgtwo0** or **cgnine0** as **gpone0a**, causes **gpone0a** to use the specified frame buffer if it is present, redirects **/dev/fb** to be **/dev/gpone0a** and loads and starts the appropriate microcode.

For SBus systems, this loads and starts the appropriate microcode.

If another configuration is desired, edit the command line in **/etc/rc.local**.

It is inadvisable to run the **gpconfig** command while the configured device is being used; unpredictable results may occur. If it is necessary to change the frame buffer bindings to the GP (or to stop using the GP altogether), bring the system down gently, boot single user, edit the **gpconfig** line in the **/etc/rc.local** file, and bring the system back up multi-user.

gpconfig tries to create the device special files under the **/dev** directory when they are absent.

For VME systems, the default *gpunit* is **/dev/gpone0**, and the default frame buffer is **/dev/cgtwo0** or **/dev/cgnine0**, whichever is present. The appropriate microcode file will be loaded based on the version of the GP in the system. The **cgnine**(4S) frame buffer can be bound only to the GP2 device.

For SBus systems, the default *fbunit* is **/dev/cgtwelve0**. There is no *gpunit*. A *gpunit* specification on the command line is ignored.

OPTIONS

–b Configure the GP or GP+ to use the specified frame buffer, or to use the first frame buffer found, if none is specified. Only one GP device, two GP+ devices, and four GP2 devices may be bound to a frame buffer at a single time. Note: this option has no effect on the SBus GS **cgtwelve**(4S) device.

–f Redirect **/dev/fb** to the device formed by binding *gpunit* with *fbunit*. Only the last **–f** option in the command line takes effect. Note: this option has no effect on the SBus GS **cgtwelve**(4S) device.

–F Return the **/dev/fb** to the default device in effect before **gpconfig** was run.

–u *microcode-file*
Load the specified file as the alternate microcode to the GP or GS.

FILES

/dev/fb	default frame buffer
/dev/cgtwelve0	SBus 24-bit frame buffer and graphics accelerator
/dev/cgtwo0	VME 8-bit color frame buffer

/dev/cgnine0	VME 24-bit color frame buffer
/dev/gpone0[abcd]	VME GP graphics accelerator
/usr/lib/gs.ucode	GS microcode file
/usr/lib/gp2.ucode	GP2 microcode file
/usr/lib/gp1cg2.1024.ucode	GP1 microcode file
/usr/lib/gp1cg2.1152.ucode	GP1 microcode file
/etc/rc.local	local boot script

SEE ALSO

cgnine(4S), **cgtwelve**(4S), **cgtwo**(4S), **gpone**(4S)

NAME

grpck – check group database entries

SYNOPSIS

/usr/etc/grpck [*filename*]

AVAILABILITY

This command is available with the *System V* software installation option. Refer to *Installing the SunOS System Software* for information on how to install optional software.

DESCRIPTION

grpck checks that a file in **group**(5) does not contain any errors; it checks the **/etc/group** file by default.

FILES

/etc/group

DIAGNOSTICS

Too many/few fields
An entry in the group file does not have the proper number of fields.

No group name
The group name field of an entry is empty.

Bad character(s) in group name
The group name in an entry contains characters other than lower-case letters and digits.

Invalid GID
The group ID field in an entry is not numeric or is greater than 65535.

Null login name
A login name in the list of login names in an entry is null.

Login name not found in password file
A login name in the list of login names in an entry is not in the password file.

First char in group name not lower case alpha
The group name in an entry does not begin witha lower-case letter.

Group name too long
The group name in an entry has more than 8 characters.

SEE ALSO

groups(1), **group**(5), **passwd**(5)

NAME

gt_lpconfig – Configure the GT lightpen device parameters.

SYNOPSIS

/usr/etc/gt_lpconfig

DESCRIPTION

It is a window based tool for setting up the calibration and smoothening parameters in the lightpen driver. A help function provided to guide the user with the actual specifics of this tool.

This tool should be run only after the GT has been initialized through gtconfig.

OPTIONS

NONE

FILES

/dev/gt0	GT Graphics Accelerator file
/dev/lightpen	Lightpen file

SEE ALSO

gt(4S), **gtconfig**(8),

NAME

gtconfig – initialize the GT Graphics Accelerator and download microcode.

SYNOPSIS

/usr/etc/gtconfig [**–d** *device-filename*] [**–f** *filename*] [**–s0** *filename*] [**–s1** *filename*] [**–I** *microcode-directory*] [**–E** *bit-code*] [**–g|G** *gamma-value*] [**–degamma8|DEGAMMA8** *on|off* [**–i**] [**–m|M** *monitor-type*] [**–c** *1-14*] [**–w** *2-9*] [**–v**]

DESCRIPTION

gtconfig initializes the GT Graphics Accelerator and downloads microcode from the host. It is normally run as a part of **/etc/rc.local** to download GT microcode files and to complete GT initialization.

The standard version of the file **/etc/rc.local** contains the following **gtconfig** command line:

/usr/etc/gtconfig

If another configuration is desired, edit the command line in **/etc/rc.local**.

It is inadvisable to run the **gtconfig** command while the configured device is being used; unpredictable results may occur. If it is necessary to change the **gtconfig** command, bring the system down gently, boot single user, edit the **gtconfig** line in the **/etc/rc.local** file, and bring the system back up multi-user.

Filenames may be either relative or absolute pathnames. Relative pathnames are prepended with the path specified by **-I**, or the default path **/usr/lib**.

OPTIONS

–d *device-filename*
Specifies the GT special file. The default is **/dev/gt0**.

–f *filename*
Specifies the Front End microcode file. The default is **gt.ucode**.

–s0 *filename*
Specifies the Setup Processor 0 microcode file. The default is **gt.c30.ucode**.

–s1 *filename*
Specifies the Setup Processor 1 microcode file. The default is **gt.c31.ucode**.

–I *microcode-directory*
Specifies the directory containing microcode files. The default is **/usr/lib**.

–E *bit-code*
Initializes the cursor enable plane to contain all zeros or all ones. The default is **0**.

–g *gamma-value*
Specifies the gamma correction value. The default is 2.22.

–G *gamma-value*
Loads the gamma correction table only; does not initialize the GT or download microcode.

–degamma8 *on|off*
Specifies automatic inverse gamma correction of 8-bit indexed color maps. This allows color-maps with built-in gamma correction to work properly on GT. The default is **on**.

–DEGAMMA8 *on|off*
Like degamma8, but only specifies automatic inverse gamma correction; does not otherwise initialize GT or download microcode.

–i Initialize the GT backend.

–m *monitor-type*
Specifies the monitor type of 1280_76, 1280_67, or stereo. The default is 1280_67.

–M *monitor-type*
Sets the monitor type only; does not initialize the GT or download microcode.

–c *1-14* Sets the OpenWindows server CLUT quota. The default is 8.

–w *2-9* Sets the OpenWindows server WID plane quota. The default is 5.

–v Sets verbose mode.

FILES

/dev/gt0	GT Graphics Accelerator file
gt.ucode	GT Front End microcode file
gt.c30.ucode	Setup Processor 0 microcode file
gt.c31.ucode	Setup Processor 1 microcode file
/usr/lib	directory that normally contains the microcode files
/etc/rc.local	local boot script

SEE ALSO

gt(4S), **fbio**(4S), **mmap**(2),

NAME

gxtest – stand alone test for the Sun video graphics board

SYNOPSIS

b /stand/gxtest

DESCRIPTION

gxtest runs stand alone, not under control of the operating system. With the PROM resident monitor in control of the system, type the command:

> b /stand/gxtest

and the monitor boots the video test program into memory. **gxtest** is completely self-explanatory and runs under its own steam. It reports any errors it finds on the screen.

NAME

halt – stop the processor

SYNOPSIS

/usr/etc/halt [**–nqy**]

DESCRIPTION

halt writes out any information pending to the disks and then stops the processor.

halt normally logs the system shutdown to the system log daemon, **syslogd**(8), and places a shutdown record in the login accounting file **/var/adm/wtmp**. These actions are inhibited if the **–n** or **–q** options are present.

OPTIONS

–n Prevent the *sync* before stopping.

–q Do a quick halt. No graceful shutdown is attempted.

–y Halt the system, even from a dialup terminal.

FILES

/var/adm/wtmp login accounting file

SEE ALSO

reboot(8), **shutdown**(8), **syslogd**(8)

NAME

hostconfig – configure a system's host parameters

SYNOPSIS

/usr/etc/hostconfig –p *protocol* [**–d**] [**–v**] [**–n**] [**–i** *interface*] [**–f** *hostname*]

DESCRIPTION

The **hostconfig** program uses a network protocol to acquire a machine's ''host parameters'' and then sets these parameters on the system. The program selects which protocol to use based on the argument to the required **–p** flag. Different protocols may set different host parameters. Currently, two protocols are defined but only one protocol is supported.

OPTIONS

–p bootparams	Use the ''whoami'' call of the Sun RPC ''bootparams'' protocol. This sets the system's *hostname*, *domainname*, and *default IP router* parameters.
–p bootp	Use the BOOTP protocol [not currently supported].
–d	Enable ''debug'' output.
–v	Enable verbose output.
–n	Run the network protocol, but do not set the acquired parameters into the system.
–i *interface*	Use only the named network interface to run the protocol.
–f *hostname*	Run the protocol as if this machine were named *hostname*.

EXAMPLES

To configure a machine's host parameters using the ''bootparams whoami'' protocol, getting a verbose output:

hostconfig –p bootparams –v

To see what parameters would be set using the ''bootparams whoami'' protocol:

hostconfig –p bootparams –n –v

SEE ALSO

hostname(1), **domainname**(1), **bootparam**(3R), **route**(8C)

NAME

hostrfs – convert IP addresses to RFS format

SYNOPSIS

hostrfs *hostname* [*portnum*]

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing the SunOS System Software* for information on how to install optional software.

DESCRIPTION

hostrfs converts IP addresses to a format suitable for use by Remote File Sharing (RFS). It takes a hostname and an optional portnumber and produces an address in the following format:

\x<AF-INET><*portnum*>**<***IP-address***>0000000000000000**

Each field given above is a hex ASCII representation. The **AF_INET** field is the address family which always has the value **0002**. *portnum* is the two-byte TCP port number; if not specified on the command line it defaults to **1450**. *IP-address* is the IP address of the *hostname* given on the command line followed by 16 trailing zeroes.

The output of this command may be directly used as the network address field for the address of an RFS name server in the **rfmaster**(5) file. It may also be used as input to the **nlsadmin (8)** command to initialize the addresses on which the **listener** program listens for service requests.

EXAMPLES

The output of

example% hostrfs wopr

is

\0002145081903509000000000000000000

The output of the command can be used to initialize the network address on which the RFS **listener** program listens for remote service requests, for example:

example# nlsadmin –l 'hostrfs wopr' tcp

SEE ALSO

rfmaster(5), **nlsadmin**(8)

System and Network Administration

FILES

/etc/passwd
/etc/group
/usr/nserve/auth.info/*domain*/*host*/**[user | group]**
/usr/nserve/auth.info/vid.rules
/usr/nserve/auth.info/gid.rules

SEE ALSO

mount(8)

NAME

ifconfig – configure network interface parameters

SYNOPSIS

/usr/etc/ifconfig *interface* [*address_family*] [*address* [*dest_address*]] [**netmask** *mask*]
[**broadcast** *address*] [**up**] [**down**] [**trailers**] [**–trailers**] [**arp**] [**–arp**] [**private**]
[**–private**] [**metric** *n*] [**auto-revarp**]

/usr/etc/ifconfig *interface* [*protocol_family*]

DESCRIPTION

ifconfig is used to assign an address to a network interface and/or to configure network interface parameters. **ifconfig** must be used at boot time to define the network address of each interface present on a machine; it may also be used at a later time to redefine an interface's address or other operating parameters. Used without options, **ifconfig** displays the current configuration for a network interface. If a protocol family is specified, **ifconfig** will report only the details specific to that protocol family. Only the super-user may modify the configuration of a network interface.

The *interface* parameter is a string of the form *nameunit*, for example **le0** or **ie1**. Three special interface names, **–a**, **–ad** and **–au**, are reserved and refer to all or a subset of the interfaces in the system. If one of these interface names is given, the commands following it are applied to all of the interfaces that match:

–a Apply the commands to all interfaces in the system.

–ad Apply the commands to all ''down'' interfaces in the system.

–au Apply the commands to all ''up'' interfaces in the system.

Since an interface may receive transmissions in differing protocols, each of which may require separate naming schemes, the parameters and addresses are interpreted according to the rules of some address family, specified by the *address_family* parameter. The address families currently supported are **ether** and **inet**. If no address family is specified, **inet** is assumed.

For the TCP/IP family (**inet**), the address is either a host name present in the host name data base (see **hosts**(5)) or in the Network Information Service (NIS) map **hosts**, or a TCP/IP address expressed in the Internet standard "dot notation". Typically, an Internet address specified in dot notation will consist of your system's network number and the machine's unique host number. A typical Internet address is **192.9.200.44**, where **192.9.200** is the network number and **44** is the machine's host number.

For the **ether** address family, the address is an Ethernet address represented as *x*:*x*:*x*:*x*:*x*:*x* where *x* is a hexadecimal number between 0 and ff. Only the super-user may use the **ether** address family.

If the *dest_address* parameter is supplied in addition to the *address* parameter, it specifies the address of the correspondent on the other end of a point to point link.

OPTIONS

up Mark an interface "up". This happens automatically when setting the first address on an interface. The **up** option enables an interface after an **ifconfig down**, reinitializing the hardware.

down Mark an interface "down". When an interface is marked "down", the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.

trailers This flag used to cause a non-standard encapsulation of inet packets on certain link levels. Sun drivers no longer use this flag, but it is ignored for compatibility.

–trailers Disable the use of a "trailer" link level encapsulation.

arp Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between TCP/IP addresses and 10Mb/s Ethernet addresses.

–arp Disable the use of the Address Resolution Protocol.

private Tells the **in.routed** routing daemon (see **routed**(8C)) that the interface should not be advertised.

–private Specify unadvertised interfaces.

auto-revarp Use the Reverse Address Resolution Protocol (RARP) to automatically acquire an address for this interface. Available beginning with SunOS 4.1.1 Rev B.

metric *n* Set the routing metric of the interface to *n*, default 0. The routing metric is used by the routing protocol (**routed**(8C)). Higher metrics have the effect of making a route less favorable; metrics are counted as addition hops to the destination network or host.

netmask *mask* (**inet** only) Specify how much of the address to reserve for subdividing networks into sub-networks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation address, or with a pseudo-network name listed in the network table **networks**(5). The mask contains 1's for the bit positions in the 32-bit address which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion. If a '**+**' (plus sign) is given for the netmask value, then the network number is looked up in the NIS **netmasks.byaddr** map (or in the **/etc/netmasks**) file if not running the NIS service.

broadcast *address*
(**inet** only) Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 0's. A **+** (plus sign) given for the broadcast value causes the broadcast address to be reset to a default appropriate for the (possibly new) address and netmask. Note that the arguments of **ifconfig** are interpreted left to right, and therefore

ifconfig –a netmask + broadcast +

and

ifconfig –a broadcast + netmask +

may result in different values being assigned for the interfaces' broadcast addresses.

EXAMPLES

If your workstation is not attached to an Ethernet, the **ie0** interface should be marked "down" as follows:

ifconfig ie0 down

To print out the addressing information for each interface, use

ifconfig –a

To reset each interface's broadcast address after the netmasks have been correctly set, use

ifconfig –a broadcast +

FILES

/dev/nit
/etc/netmasks

SEE ALSO

intro(3), **ethers**(3N), **arp**(4P), **hosts**(5), **netmasks**(5), **networks**(5) **netstat**(8C), **rc**(8), **routed**(8C)

DIAGNOSTICS

Messages indicating the specified interface does not exist, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

NOTES

The network information service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

imemtest – stand alone memory test

SYNOPSIS

b /stand/imemtest

DESCRIPTION

imemtest runs stand alone, not under control of the operating system. With the PROM resident monitor in control of the system, type the command:

> b /stand/imemtest

and the monitor boots the memory test program into memory. **imemtest** is completely self-explanatory. It prompts for all start and end addresses, and after that it runs under its own steam. It reports any errors it finds on the screen.

NAME

installboot – install bootblocks in a disk partition

SYNOPSIS

/usr/kvm/mdec/installboot [**–hltv**] *bootfile protobootblk bootdevice*

DESCRIPTION

The **boot**(8S) program is loaded from disk by bootblock code which resides in the bootblock area of a disk partition. In order for the bootblock code to read the boot program (usually **/boot**) it is necessary for it to know the block numbers occupied by the boot program.

installboot plugs the block numbers of the boot program into a table in the bootblock code, and writes the modified bootblock code onto the disk. Note: **installboot** must be run every time the boot program is reinstalled, since in general, the block list of the boot program will change each time it is written.

bootfile is the name of the boot program, usually **/boot**. *protobootblk* is the name of the bootblock code into which the block numbers of the boot program are to be inserted. Sun distributes a number of prototype boot blocks for different Sun-supported devices in **/usr/kvm/mdec**. The prototype bootblock file must have an **a.out**(5) header, but it will be written out to the device with the header removed. See '**SPARCstation 1 Systems Only**' below for exceptions. *bootdevice* is the name of the disk device onto which the bootblock code is to be installed.

You can see how **installboot** works by making the destination a regular file instead of a device, and examining the result with **od**(1V).

SPARCstation 1 Systems Only

The SPARCstation 1 system (or any machine of kernel architecture **sun4c**) uses bootblocks which are regular **a.out**(5) executables. This means the header is left intact when the bootblock is transferred to the disk. This is primarily used to distinguish bootable floppy disks: since the header contains a machine type code, a machine of one architecture can avoid attempting to boot a floppy which contains bootblocks for a different architecture.

installboot distributed with the **sun4c** executables leaves the header on the bootblock automatically; this behavior may be duplicated by other versions of **installboot** by using the **–h** flag described below.

OPTIONS

–h Leave the **a.out** header on the bootblock when installed on disk.

–l Print out the list of block numbers of the boot program.

–t Test. Display various internal test messages.

–v Verbose. Display detailed information about the size of the boot program, etc.

EXAMPLE

To install the bootblocks onto the root partition on a Xylogics disk:

```
example% cd /usr/kvm/mdec
example% installboot –vlt /boot bootxy /dev/rxy0a
```

For an SD disk, you would use **bootsd** and **/dev/rsd0a**, respectively, in place of **bootxy** and **/dev/rxy0a**.

SEE ALSO

od(1V), **a.out**(5), **boot**(8S), **bootparamd**(8), **init**(8), **kadb**(8S), **monitor**(8S), **ndbootd**(8C), **rc**(8), **reboot**(8)

System and Network Administration
Installing the SunOS System Software

NAME

install_small_kernel – install a small, pre-configured kernel

SYNOPSIS

/usr/etc/install/install_small_kernel [*hostname*] . . .

DESCRIPTION

install_small_kernel is a script that installs a small, pre-configured kernel, **GENERIC_SMALL** on a host. This kernel supports approximately four users, and is only available for the following configurations:

Sun-3/50 and Sun-3/60 systems with up to 2 SCSI disks, 1 SCSI tape
Sun-3/80 systems with up to 4 SCSI disks, 1 SCSI tape
Sun-4/110 systems with up to 2 SCSI disks, 1 SCSI tape
SPARCsystem 330 systems with up to 4 SCSI disks, 1 SCSI tape
SPARCstation 1, 1+, and 2 systems with up to 8 SCSI disks, 4 SCSI tapes, 2 CD-ROM drives, 1 floppy disk

If *hostname* is a server that does not fit any of the above configurations, **install_small_kernel** can be used to install the small kernel on its clients.

If no hostnames are specified, **install_small_kernel** cycles through all the clients configured for a server to determine the small kernel installs to be made. If the '**small_kernel**' flag in the client file, **/etc/install/client.***hostname* is set to '**yes**', that client will not be processed. To force re-installation of a small kernel on any clients, simply call **install_small_kernel** with the appropriate client names.

install_small_kernel prompts for confirmation before actually doing the install on any host.

install_small_kernel is executable from the miniroot, as well as single-user and multi-user modes. It supports standalone and server configuration in all cases, but dataless systems are supported in multi-user mode only. This script is restricted to the super-user.

FILES

/usr/sys/sun*arch***/conf/GENERIC_SMALL**
kernel configuration file for *arch* **/usr/install/client.***hostname*

SEE ALSO

add_client(8), **add_services**(8), **rm_client**(8). **suninstall**(8)

System and Network Administration

NAME

iostat – report I/O statistics

SYNOPSIS

iostat [**–cdDIt**] [**–l** *n*] [*disk* ...] [*interval* [*count*]]

DESCRIPTION

iostat can iteratively report terminal and disk I/O activity, as well as CPU utilization. The first report is for all time since a reboot and each subsequent report is for the prior interval only.

In order to compute this information, the kernel maintains a number of counters. For each disk, seeks and data transfer completions and number of words transferred are counted; for terminals collectively, the number of input and output characters are counted. Also, at each clock tick, the state of each disk is examined and a tally is made if the disk is active. The kernel also provides approximate transfer rates of the devices.

OPTIONS

iostat's activity class options default to **tdc** (terminal, disk, and CPU). If any activity class options are specified, the default is completely overridden. Therefore, if only **–d** is specified, neither terminal nor CPU statistics will be reported. The last disk option specified (either **–d** or **–D**) is the only one that is used.

–c Report the percentage of time the system has spent in user mode, in user mode running low priority processes, see **nice**(1), in system mode, and idling.

–d For each disk, report the number of kilobytes transferred per second, the number of transfers per second, and the milliseconds per average seek (see BUGS below).

–D For each disk, report the reads per second, writes per second, and percentage disk utilization.

–I Report the counts in each interval, rather than reporting rates.

–t Report the number of characters read and written to terminals.

–l *n* Limit the number of disks included in the report to *n*; the disk limit defaults to 4. Note: disks explicitly requested (see *disk* below) are not subject to this disk limit.

disk Explicitly specify the disks to be reported; in addition to any explicit disks, any active disks up to the disk limit (see **–l** above) will also be reported.

interval Report once each *interval* seconds.

count Only print *count* reports.

FILES

/dev/kmem
/vmunix

SEE ALSO

vmstat(8)

BUGS

Milliseconds per average seek is an approximation based on the disk (not the controller) transfer rate. Therefore, the seek time will be over-estimated in systems with slower controllers.

iostat only provides statistics on the first ten disk drives, all others are ignored.

NAME

ipallocd – Ethernet-to-IP address allocator

SYNOPSIS

/usr/etc/rpc.ipallocd

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

ipallocd is a daemon that determines or temporarily allocates IP addresses within a network segment. The service is only available on the system which is home to the address authority for the network segment, currently the Network Information Service (NIS) master of the **hosts.byaddr** map although the service is not tied to the NIS service. It has complete knowledge of the hosts listed in the NIS service, and, if the system is running the name server, of any hosts listed in internet domain tables automatically accessed on that host through the standard library **gethostent**(3N) call.

This protocol uses DES authentication (the Sun Secure RPC protocol) to restrict access to this function. The only clients privileged to allocate addresses are those whose net IDs are in the networks group. For machine IDs, the machine must be an NIS server.

The daemon uses permanent entries in the **/etc/ethers** and **/etc/hosts** files when they exist and are usable. In other cases, such as when a system is new to the network, **ipallocd** enters a temporary mapping in a local cache. Entries in the cache are removed when there have been no references to a given entry in the last hour. This cache survives system crashes so that IP addresses remain consistent.

The daemon also provides corresponding IP address to name mapping.

If the file **/etc/ipalloc.netrange** exists, **ipallocd** refuses to allocate addresses on networks not listed in the **netrange** file, or for which no free address is available.

FILES

/etc/ipalloc.cache	temporary cache
/etc/ipalloc.netrange	optional file to allocate network addresses

SEE ALSO

ipalloc(3R), **pnp**(3R), **ipalloc.netrange**(5), **ipallocd**(8C), **netconfig**(8C), **pnpboot**(8C), **rarpd**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

mkfs – construct a file system

SYNOPSIS

/usr/etc/mkfs [**–N**] *special size* [*nsect*] [*ntrack*] [*blksize*] [*fragsize*] [*ncpg*] [*minfree*] [*rps*] [*nbpi*] [*opt*] [*apc*] [*rot*] [*nrpos*] [*maxcontig*]

DESCRIPTION

Note: file systems are normally created with the **newfs**(8) command.

mkfs constructs a file system by writing on the special file *special* unless the **–N** flag has been specified. *special* must be specified as a raw device and disk partition. For example, to create a file system on **sd0**, specify **/dev/rsd0[a-h]**, where **a-h** is the disk partition.

The numeric *size* specifies the number of sectors in the file system. **mkfs** builds a file system with a root directory and a lost+found directory (see **fsck**(8)). The number of inodes is calculated as a function of the file system size. No boot program is initialized by **mkfs** (see **newfs**(8)).

You must be super-user to use this command.

OPTIONS

–N Print out the file system parameters without actually creating the file system.

The following arguments allow fine tune control over the parameters of the file system.

nsect The number of sectors per track on the disk. The default is **32**.

ntrack The number of tracks per cylinder on the disk. The default is **16**.

blksize The primary block size for files on the file system. It must be a power of two, currently selected from **4096** or **8192** (the default).

fragsize The fragment size for files on the file system. The *fragsize* represents the smallest amount of disk space that will be allocated to a file. It must be a power of two currently selected from the range **512** to **8192**. The default is **1024**.

ncpg The number of disk cylinders per cylinder group. The default is **16**.

minfree The minimum percentage of free disk space allowed. Once the file system capacity reaches this threshold, only the super-user is allowed to allocate disk blocks. The default value is **10%**.

rps The rotational speed of the disk, in revolutions per second. The default is **60**.

nbpi The number of bytes for which one inode block is allocated. This parameter is currently set at **one** inode block for every 2048 bytes.

opt Space or time optimization preference; **s** specifies optimization for space, **t** specifies optimization for time. The default is **t**.

apc The number of alternates per cylinder (SCSI devices only). The default is **0**.

rot The expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file. The default is **1**.

Note: in earlier releases **mkfs** tried to guess what the right value for this parameter by querying the controller type. Since **mkfs** is a more primitive interface, this query has been moved into **newfs**. **mkfs** now does exactly what you tell it to do.

nrpos The number of distinguished rotational positions. The default is **8**.

maxcontig

The maximum number of blocks that will be allocated contiguously before inserting a rotational delay. The default is **1**.

Note: This parameter also controls clustering. Regardless of the value of *rotdelay*, clustering is enabled only when *maxcontig* is greater than 1. Clustering allows higher I/O rates for sequential I/O and is described in **tunefs**(8).

Users with special demands for their file systems are referred to the paper cited below for a discussion of the tradeoffs in using different configurations.

SEE ALSO

dir(5), **fs**(5), **fsck**(8), **newfs**(8), **tunefs**(8)

System and Network Administration
McKusick, Joy, Leffler; *A Fast File System for UNIX*

NOTES

newfs(8) is preferred for most routine uses.

NAME

modload – load a kernel module

SYNOPSIS

modload *filename* [**–d**] [**–v**] [**–sym**] [**–A** *vmunix_file*] [**–conf** *config_file*] [**–entry** *entry_point*] [**–exec** *exec_file*] [**–o** *output_file*]

DESCRIPTION

modload loads a loadable module into a running system. The input file *filename* is an object file (**.o** file).

OPTIONS

–d Debug. Used to debug **modload** itself.

–v Verbose. Print comments on the loading process.

–sym Preserve symbols for use by **kadb**(8S).

–A *vmunix_file*
Specify the file that is passd to the linker to resolve module references to kernel symbols. The default is **/vmunix**. The symbol file must be for the currently running kernel or the module is likely to crash the system.

–conf *config_file*
Use this configuration file to configure the loadable driver being loaded. The commands in this file are the same as those that the **config**(8) program recognizes. There are two additional commands recognized, **blockmajor** and **charmajor**. See the *Writing Device Drivers* for information on these commands.

–entry *entry_point*
Specify the module entry point. This is passed by **modload** to **ld**(1) when the module is linked. The default module entry point name is '*xxx* **init**'.

–exec *exec_file*
Specify the name of a shell script or executable image file that will be executed if the module is successfully loaded. It is always passed the module id (in decimal) and module type (in hexadecimal) as the first two arguments. Module types are listed in **/usr/include/sun/vddrv.h**. For loadable drivers, the third and fourth arguments are the block major and character major numbers respectively. For a loadable system call, the third argument is the system call number.

–o *output_file*
Specify the name of the output file that is produced by the linker. If this option is omitted, then the output file name is *filename* without the '**.o**'.

BUGS

On Sun-3 machines, the **config**(8) program generates assembly language wrappers to provide the proper linkage to device interrupt handlers. **modload** does not generate these wrappers; on interrupt, control passes directly to the loadable driver's interrupt handler. Consequently, the driver must provide its own wrapper. See the ioconf.c file generated by **config**(8) for examples of wrappers.

SEE ALSO

ld(1), **config**(8), **kadb**(8S), **modunload**(8), **modstat**(8)

NAME

modstat – display status of loaded kernel modules

SYNOPSIS

modstat [**–id** *module_id*]

DESCRIPTION

modstat displays the status of the loaded modules.

OPTIONS

–id *module_id* Display the status of only this module.

SEE ALSO

modload(8), **modunload**(8)

NAME

modunload – unload a module

SYNOPSIS

modunload –id *module_id* [**–exec exec_file**]

DESCRIPTION

modunload unloads a loadable module from a running system. The *module_id* is the ID of the module as shown by **modstat**(8).

OPTIONS

–exec *exec_file*

This is the name of a shell script or executable image file that will be executed before the module is unloaded. It is always passed the module ID and module type as the first two arguments. For loadable drivers, the third and fourth arguments are the block major and character major numbers respectively. For a loadable system call, the third argument is the system call number.

SEE ALSO

modload(8), **modstat**(8)

NAME

monitor – system ROM monitor

SYNOPSIS

L1–A

BREAK

DESCRIPTION

The CPU board of the Sun workstation contains an EPROM (or set of EPROMs), called the *monitor*, that controls the system during startup. The monitor tests the system before attempting to boot the operating system. If you interrupt the boot procedure by holding down **L1** while typing **a** or **A** on the workstation keyboard (or **BREAK** if the console is a dumb terminal), the monitor issues the prompt:

>

and accepts commands interactively.

On a Desktop SPARCsystem or a SPARCsystem 600MP, the message

Type b (boot), c (continue), or n (new command mode)

will be displayed prior to the monitor prompt '**>**'.

USAGE

Modes

The monitor supports three security modes (non-secure, command secure, and fully secure) and an authentication password. Access to monitor commands is controlled by these security modes. In **non-secure** mode all monitor commands are allowed. In **command secure** mode, only the **b**(boot) command with no arguments and the **c**(continue) command with no arguments may be entered without supplying the authentication password. In **fully secure** mode, only the **c**(continue) command with no arguments may be entered without supplying the authentication password. Note: The system will not auto-reboot in fully secure mode. The authentication password must be entered before booting will take place.

Commands

Note: the following commands are available on all Sun systems *except* Desktop SPARCsystems and SPARCsystem 600MP series; only the **b**, **c**, and **n** commands listed below are available on those systems.

+|– Increment or decrement the current address and display the contents of the new location.

^C *source destination n*
(caret-C) Copy, byte-by-byte, a block of length *n* from the *source* address to the *destination* address.

^I *program* (caret-I) Display the compilation date and location of *program*.

^T *virtual_address*
(caret-T) Display the physical address to which *virtual_address* is mapped.

a [*n*] [*action*]... (Sun-3 systems only)
Open **A**–register (cpu address register) *n*, and perform indicated actions. The number *n* can be any value from **0** to **7**, inclusive. The default value is **0**. A hexadecimal *action* argument assigns the value you supply to the register *n*. A non-hex *action* terminates command input.

b [*device* [(*c*,*u*,*p*)]] [*pathname*] [*arguments_list*] (SPARCstation 1 systems only)
See **openboot**(8S) for details.

b [*device-specifier*] [*pathname*] [*arguments-list*] (SPARCstation 2 systems, SPARCsystem 600MP series only)
See **openboot**(8s) for details.

b [**!**] [*device* [(*c*,*u*,*p*)]] [*pathname*] [*arguments_list*]

b[**?**] Reset appropriate parts of the system and bootstrap a program. Note: the '**b!**' and '**b?**' commands are not available on Desktop SPARCsystems and SPARCsystem 600MP series,

see **openboot**(8S). A '**!**' (preceding the *device* argument) prevents the system reset from occurring. Programs can be loaded from various devices (such as a disk, tape, or Ethernet). '**b**' with no arguments will cause a default boot, either from a disk, or from an Ethernet controller. '**b?**' displays all boot devices and their *device* arguments, where *device* is one of:

le Lance Ethernet
ie Intel Ethernet (Sun-3, Sun-4 systems only)
sd SCSI disk, CDROM
st SCSI 1/4" or 1/2" tape
fd Diskette (Sun386i, Sun-3/80 systems, Desktop SPARCsystems only)
id IPI disk (Sun-4 systems, SPARCsystem 600MP series only)
mt Tape Master 9-track 1/2" tape (Sun-3, Sun-4 systems only)
xd Xylogics 7053 disk (Sun-3, Sun-4 systems only)
xt Xylogics 1/2" tape (Sun-3, Sun-4 systems only)
xy Xylogics 440/450 disk (Sun-3, Sun-4 systems only)

c A controller number (**0** if only one controller),

u A unit number (**0** if only one driver), and

p A partition.

pathname
A pathname for a program such as **/stand/diag**. **/vmunix** is the default.

arguments_list
A list of up to seven arguments to pass to the program being booted.

c (Desktop SPARCsystems, SPARCsystem 600MP series only);
c [*virtual_address*] (Sun-3, Sun-4 and Sun386i systems only)
Resume execution of a program. Desktop SPARCsystems and SPARCsystem 600MP series do not accept a virtual address; the current PC is assumed. See the *Open PROM Toolkit User's Guide* or *Open PROM 2.0 Toolkit User's Guide* for details on changing this value. When given, *virtual_address* is the address at which execution will resume. The default is the current PC (EIP on Sun386i systems). Registers are restored to the values shown by the **a**, **d**, and **r** commands (for Sun-3 systems), or by the **d** and **r** commands (for Sun-4 systems), or by the **d** command (for Sun386i systems).

d [*window_number*] (Sun-4 systems only)
Display (dump) the state of the processor. The processor state is observable only after:

- An unexpected trap was encountered.
- A user program dropped into the monitor (by calling *abortent*).
- The user manually entered the monitor by typing **L1–A** or **BREAK**.

The display consists of the following:

- The special registers: PSR, PC, nPC, TBR, WIM, and Y
- Eight global registers, and
- 24 window registers (8 *in*, 8 *local*, and 8 *out*), corresponding to one of the 7 available windows. If a Floating-Point Unit is on board, its status register along with its 32 floating-point registers are also shown.

window_number
Display the indicated *window_number*, which can be any value between 0 and 6, inclusive. If no window is specified and the PSR's current window pointer contains a valid window number, registers from the window that was active just prior to entry into the monitor are displayed. Otherwise, registers from window 0 are displayed.

d (Sun386i systems only)

Display (dump) the state of the processor. This display consists of the registers, listed below:

Processor Registers:	EAX, ECX, EDX, ESI, EDI, ESP, EBP, EFLAGS, EIP
Segment Registers:	ES, CS, SS, DS, FS, GS
Memory Management Registers:	GDTR, LDTR, IDTR, TR
Control Registers:	CR0, CR2, CR3
Debug Registers:	DR0, DR1 , DR2 , DR3, DR6, DR7
Test Registers:	TR6, TR7

The processor's state is observable only after an unexpected trap, a user program has "dropped" into the monitor (by calling monitor function *abortent*) or the user has manually "broken " into the monitor (by typing **L1–A** on the Workstation console, or **BREAK** on the dumb terminal's keyboard.

d [*n*] [*action*] . . . (Sun-3 systems only)

Open **D**–register (cpu data register) *n*, and perform indicated actions. The number *n* can be any value from **0** to **7**, inclusive. The default is **0**. See the **a** command for a description of *action*.

e [*virtual_address*] [*action*] . . .

Open the 16-bit word at *virtual_address* (default zero). On Sun-3, and Sun-4 systems, the address is interpreted in the address space defined by the **s** command. See the **a** command for a description of *action*.

f *virtual_address1 virtual_address2 pattern* [*size*] (Sun-3 and Sun-4 systems only)

Fill the bytes, words, or long words from *virtual_address1* (lower) to *virtual_address2* (higher) with the constant, *pattern*. The *size* argument can take one of the following values

b byte format (the default)
w word format
l long word format

For example, the following command fills the address block from 0x1000 to 0x2000 with the word pattern, 0xABCD:

f 1000 2000 ABCD W

g [*vector*] [*argument*]
g [*virtual_address*] [*argument*]

Goto (jump to) a predetermined or default routine (first form), or to a user-specified routine (second form). The value of *argument* is passed to the routine. If the *vector* or *virtual_address* argument is omitted, the value in the PC is used as the address to jump to.

To set up a predetermined routine to jump to, a user program must, prior to executing the monitor's **g** command, set the variable ***romp->v_vector_cmd** to be equal to the virtual address of the desired routine. Predetermined routines need not necessarily return control to the monitor.

The default routine, defined by the monitor, prints the user-supplied *vector* according to the format supplied in *argument*. This format can be one of:

%x hexadecimal
%d decimal

g0 (Sun-3, and Sun-4 only)

When the monitor is running as a result of the system being interrupted, force a panic and produce a crash dump.

g4

When the monitor is running as a result of the system being interrupted, force a kernel stack trace.

h (Sun-3 and Sun-4 and Sun386i systems)
Display the help menu for monitor commands and their descriptions. To return to the monitor's basic command level, press ESCAPE or **q** before pressing RETURN.

i [*cache_data_offset*] [*action*] . . . (Sun-3/200 series and Sun-4 systems only)
Modify cache data RAM command. Display and/or modify one or more of the cache data addresses. See the **a** command for a description of *action*.

j [*cache_tag_offset*] [*action*] . . . (Sun-3/200 series and Sun-4 systems only)
Modify cache tag RAM command. Display and/or modify the contents of one or more of the cache tag addresses. See the **a** command for a description of *action*.

k [*reset_level*]
Reset the system. If *reset_level* is:

- **0** CPU reset only (Sun-3 systems). Reset VMEbus, interrupt registers, video monitor (Sun-4 systems). This is the default. Reset video (Sun386i systems).
- **1** Software reset.
- **2** Power-on reset. Resets and clears the memory. Runs the EPROM-based diagnostic self test, which can take several minutes, depending upon how much memory is being tested.

kb Display the system banner.

l [*virtual_address*] [*action*] . . .
Open the long word (32 bit) at memory address *virtual_address* (default zero). On Sun-3 and Sun-4 systems, the address is interpreted in the address space defined by the **s** command (below). See the **a** command for a description of *action*.

m [*virtual_address*] [*action*] . . .
Open the segment map entry that maps *virtual_address* (default zero). On Sun-3 and Sun-4 systems, the address is interpreted in the address space defined by the **s** command. Not supported on Sun386i. See the **a** command for a description of *action*.

n (Desktop SPARCsystems, SPARCsystem 600MP series only)
Enter the new command mode. Type '**old-mode**' to return to the old command mode. See the *Open PROM Toolkit User's Guide* for a complete list of commands available on a SPARCstation 1 systems; See the *Open Boot PROM 2.0 Toolkit User's Guide* for a complete list of commands available on SPARCstation 2 and SPARCsystem 600MP series.

nd (Sun386i systems only)
ne
ni Disable, enable, or invalidate the cache, respectively.

o [*virtual_address*] [action*]* . . .
Open the byte location specified by *virtual_address* (default zero). On Sun-3 and Sun-4 systems, the address is interpreted in the address space defined by the **s** command. See the **a** command for a description of *action*.

p [*virtual_address*] [*action*] . . .
Open the page map entry that maps *virtual_address* (default zero) in the address space defined by the **s** command. See the **a** command for a description of *action*.

p [*port_address*] [[*nonhex_char* [*hex_value*] | *hex_value*] ...] (Sun386i systems only)
Display or modify the contents of one or more port I/O addresses in byte mode. Each port address is treated as an 8-bit unit. The optional *port_address* argument, which is a 16-bit quantity, specifies the initial port I/O address. See the **e** command for argument descriptions.

q [*eeprom_offset*] [*action*]... (Sun-3 and Sun-4 systems only)

Open the EEPROM *eeprom_offset* (default zero) in the EEPROM address space. All addresses are referenced from the beginning or base of the EEPROM in physical address space, and a limit check is performed to insure that no address beyond the EEPROM physical space is accessed. On Sun386i systems, open the NVRAM *nvram_offset* (default zero). This command is used to display or modify configuration parameters, such as: the amount of memory to test during self test, whether to display a standard or custom banner, if a serial port (A or B) is to be the system console, etc. See the **a** command for a description of *action*.

r [*reg_name*] [[*nonhex_char* [*hex_value*] | *hex_value*] ...] (Sun386i systems only)

Display or modify one or more of the processor registers. If *reg_name* is specified (2 or 3 characters from the above list), that register is displayed first. The default is **EAX**. See note on register availability under the command **d** (for Sun386i systems). See the **e** command for argument descriptions.

s [*step_count*] (Sun386i systems only)

Single step the execution of the interrupted program. The *step_count* argument specifies the number of single steps to execute before displaying the monitor prompt. The default is 1.

r [*register_number*] [*action*]... (Sun-3 systems only)

Display and/or modify the register indicated. *register_number* can be one of:

CA 68020 Cache Address Register
CC 68020 Cache Control Register
CX 68020 System and User Context
DF Destination Function code
IS 68020 Interrupt Stack Pointer
MS 68020 Master Stack Pointer
PC Program Counter
SC 68010 System Context
SF Source Function code
SR Status Register
SS 68010 Supervisor Stack Pointer
UC 68010 User Context
US User Stack Pointer
VB Vector Base

Alterations to these registers (except **SC** and **UC**) do not take effect until the next **c** command is executed. See the **a** command for a description of *action*.

r [*register_number*] (Sun-4 systems only)
r [*register_type*]
r [*w window_number*]

Display and/or modify one or more of the IU or FPU registers.

A hexadecimal *register_number* can be one of:

0x00—0x0f window(0,i0)—window(0,i7), window(0,i0)—window(0,i7)
0x16—0x1f window(1,i0)—window(1,i7), window(1,i0)—window(1,i7)
0x20—0x2f window(2,i0)—window(2,i7), window(2,i0)—window(2,i7)
0x30—0x3f window(3,i0)—window(3,i7), window(3,i0)—window(3,i7)
0x40—0x4f window(4,i0)—window(4,i7), window(4,i0)—window(4,i7)
0x50—0x5f window(5,i0)—window(5,i7), window(5,i0)—window(5,i7)
0x60—0x6f window(6,i0)—window(6,i7), window(6,i0)—window(6,i7)
0x70—0x77 g0, g1, g2, g3, g4, g5, g6, g7
0x78—0x7d PSR, PC, nPC, WIM, TBR, Y
0x7e—0x9e FSR, f0—f31

Register numbers can only be displayed after an unexpected trap, a user program

has entered the monitor using the *abortent* function, or the user has entered the monitor by manually typing **L1–A** or **BREAK**.

If a *register_type* is given, the first register of the indicated type is displayed. *register_type* can be one of:

f floating-point
g global
s special

If **w** and a *window_number* (**0—6**) are given, the first *in*-register within the indicated window is displayed. If *window_number* is omitted, the window that was active just prior to entering the monitor is used. If the PSR's current window pointer is invalid, window 0 is used.

s [*code*] (Sun-3 systems only)

Set or query the address space to be used by subsequent memory access commands. *code* is one of:

0 undefined
1 user data space
2 user program space
3 user control space
4 undefined
5 supervisor data space
6 supervisor program space
7 supervisor control space

If *code* is omitted, **s** displays the current address space.

s [*asi*] (Sun-4 systems only)

Set or display the Address Space Identifier. With no argument, **s** displays the current Address Space Identifier. The *asi* value can be one of:

0x2 control space
0x3 segment table
0x4 Page table
0x8 user instruction
0x9 supervisor instruction
0xa user data
0xb supervisor data
0xc flush segment
0xd flush page
0xe flush context
0xf cache data

t [*program*] (Sun-3 systems only)

Trace the indicated standalone *program*. Works only with programs that do not affect interrupt vectors.

u [*echo*]
u [*port*] [*options*] [*baud_rate*]
u [**u**] [*virtual_address*]

With no arguments, display the current I/O device characteristics including: current input device, current output device, baud rates for serial ports A and B, an input-to-output echo indicator, and virtual addresses of mapped UART devices. With arguments, set or configure the current I/O device. With the **u** argument (**uu**...), set the I/O device to be the *virtual_address* of a UART device currently mapped.

echo Can be either **e** to enable input to be echoed to the output device, or

ne, to indicate that input is not echoed.

port Assign the indicated *port* to be the current I/O device. *port* can be one of:

- **a** serial port A
- **b** serial port B (except on Sun386i systems)
- **k** the workstation keyboard
- **s** the workstation screen

baud_rate Any legal baud rate.

options can be any combination of:

- **i** input
- **o** output
- **u** UART
- **e** echo input to output
- **ne** do not echo input
- **r** reset indicated serial port (**a** and **b** ports only)

If either **a** or **b** is supplied, and no *options* are given, the serial port is assigned for both input and output. If **k** is supplied with no options, it is assigned for input only. If **s** is supplied with no options, it is assigned for output only.

v *virtual_address1 virtual_address2* [*size*] (Sun-3 and Sun-4 systems only)

Display the contents of *virtual_address1* (lower) *virtual_address2* (higher) in the format specified by *size*:

- **b** byte format (the default)
- **w** word format
- **l** long word format

Enter return to pause for viewing; enter another return character to resume the display. To terminate the display at any time, press the space bar.

For example, the following command displays the contents of virtual address space from address 0x1000 to 0x2000 in word format:

v 1000 2000 W

w [*virtual_address*] [*argument*] (Sun-3 and Sun-4 systems only)

Set the execution vector to a predetermined or default routine. Pass *virtual_address* and *argument* to that routine.

To set up a predetermined routine to jump to, a user program must, prior to executing the monitor's **w** command, set the variable ***romp->v_vector_cmd** to be equal to the virtual address of the desired routine. Predetermined routines need not necessarily return control to the monitor.

The default routine, defined by the monitor, prints the user-supplied *vector* according to the format supplied in *argument*. This format can be one of:

- **%x** hexadecimal
- **%d** decimal

x (Sun-3 and Sun-4 systems only)

Display a menu of extended tests. These diagnostics permit additional testing of such things as the I/O port connectors, video memory, workstation memory and keyboard, and boot device paths.

y c *context_number* (Sun-4 systems only)
y p|**s** *context_number virtual_address*

Flush the indicated context, context page, or context segment.

c flush context *context_number*
p flush the page beginning at *virtual_address* within context *context_number*
s flush the segment beginning at *virtual_address* within context *context_number*

z [*number*] [*breakpoint_virtual_address* [*type*] [*len*]] (Sun386i systems only)

Set or reset breakpoints for debugging. With no arguments, this command displays the existing breakpoints. The *number* argument is a values from 0 to 3, corresponding to the processor debug registers, **DR0** to **DR3**, respectively. Up to 4 distinct breakpoints can be specified. If *number* is not specified then the monitor chooses a breakpoint number. The *breakpoint_virtual_address* argument specifies the breakpoint address. The *type* argument can be one of:

x Instruction Execution breakpoint (the default)
m for Data Write only breakpoint
r Data Reads and Writes only breakpoint.

The *len* argument can be one of: '**b**', '**w**', or '**l**', corresponding to the breakpoint field length of byte, word, or long-word, respectively. The default is '**b**'. Since the breakpoints are set in the on-chip registers, an instruction breakpoint can be placed in ROM code or in code shared by several tasks. If the *number* argument is specified but not *breakpoint_virtual_address*, the corresponding breakpoint is reset.

z [*virtual_address*] (Sun-3 systems only)

Set a breakpoint at *virtual_address* in the address space selected by the **s** command.

FILES

/vmunix

SEE ALSO

boot(8S), **eeprom**(8S), **openboot**(8S)
Open PROM Toolkit User's Guide
Open PROM 2.0 Toolkit User's Guide

NAME

mount, umount – mount and unmount file systems

SYNOPSIS

/usr/etc/mount [**–p**]
/usr/etc/mount –a [**fnv**] [**–t** *type*]
/usr/etc/mount [**–fnrv**] [**–t** *type*] [**–o** *options*] *filesystem directory*
/usr/etc/mount [**–vfn**] [**–o** *options*] *filesystem* | *directory*
/usr/etc/mount –d [**fnvr**] [**–o** *options*] *RFS-resource* | *directory*

/usr/etc/umount [**–t** *type*] [**–h** *host*]
/usr/etc/umount –a [**v**]
/usr/etc/umount [**–v**] *filesystem* | *directory* ...
/usr/etc/umount [**–d**] *RFS-resource* | *directory*

DESCRIPTION

mount attaches a named *filesystem* to the file system hierarchy at the pathname location *directory*, which must already exist. If *directory* has any contents prior to the **mount** operation, these remain hidden until the *filesystem* is once again unmounted. If *filesystem* is of the form *host***:***pathname*, it is assumed to be an NFS file system (type **nfs**).

umount unmounts a currently mounted file system, which can be specified either as a *directory* or a *filesystem*.

mount and **umount** maintain a table of mounted file systems in **/etc/mtab**, described in **fstab**(5). If invoked without an argument, **mount** displays the contents of this table. If invoked with either a *filesystem* or *directory* only, **mount** searches the file **/etc/fstab** for a matching entry, and mounts the file system indicated in that entry on the indicated directory.

mount also allows the creation of new, virtual file systems using **loopback mounts**. Loopback file systems provide access to existing files using alternate pathnames. Once a virtual file system is created, other file systems can be mounted within it without affecting the original file system. File systems that are subsequently mounted onto the original file system, however, **are** visible to the virtual file system, unless or until the corresponding mount point in the virtual file system is covered by a file system mounted there.

Recursive traversal of loopback mount points is not allowed; after the loopback mount of **/tmp/newroot**, the file **/tmp/newroot/tmp/newroot** does not contain yet another file system hierarchy. Rather, it appears just as **/tmp/newroot** did before the loopback mount was performed (say, as an empty directory).

The standard RC files first perform **4.2** mounts, then **nfs** mounts, during booting. On Sun386i systems, **lo** (loopback) mounts are performed just after **4.2** mounts. **/etc/fstab** files depending on alternate mount orders at boot time will fail to work as expected. Manual modification of **/etc/rc.local** will be needed to make such mount orders work.

See **lofs**(4S) and **fstab**(5) for more information and WARNINGS about loopback mounts.

OPTIONS

mount

–p Print the list of mounted file systems in a format suitable for use in **/etc/fstab**.

–a All. Attempt to mount all the file systems described in **/etc/fstab**. If a *type* argument is specified with **–t**, mount all file systems of that type. Using **–a**, **mount** builds a dependency tree of mount points in **/etc/fstab**. **mount** will correctly mount these file systems regardless of their order in **/etc/fstab** (except loopback mounts; see WARNINGS below).

–f Fake an **/etc/mtab** entry, but do not actually mount any file systems.

–n Mount the file system without making an entry in **/etc/mtab**.

–v Verbose. Display a message indicating each file system being mounted.

−t *type* Specify a file system type. The accepted types are **4.2**, **nfs**, **rfs**, **lo**, **hsfs**, and **tmp**. See **fstab**(5) for a description of **4.2**, **hsfs**, and **nfs**; see **lofs**(4S) for a description of **lo**; and see **tmpfs**(4) for a description of **tmp**. See *System and Network Administration* for details on **rfs**.

−r Mount the specified file system read-only, even if the entry in **/etc/fstab** specifies that it is to be mounted read-write.

Physically write-protected and magnetic-tape file systems must be mounted read-only. Otherwise errors occur when the system attempts to update access times, even if no write operation is attempted.

−d Mount an RFS file system. This option provides compatibility with the System V, Release 3 syntax for RFS mounts. Alternatively, the equivalent Sun syntax, **−t rfs**, may be used.

−o *options*

Specify file system *options*, a comma-separated list of words from the list below. Some options are valid for all file system types, while others apply to a specific type only.

options valid on *all* file systems:

rw | **ro**	Read/write or read-only.
suid | **nosuid**	Setuid execution allowed or disallowed.
grpid	Create files with BSD semantics for the propagation of the group ID. Under this option, files inherit the GID of the directory in which they are created, regardless of the directory's set-GID bit.
noauto	Do not mount this file system that is currently mounted read-only. If the file system is not currently mounted, an error results.
remount	If the file system is currently mounted, and if the entry in **/etc/fstab** specifies that it is to be mounted read-write or **rw** was specified along with **remount**, remount the file system making it read-write. If the entry in **/etc/fstab** specifies that it is to be mounted read-only and **rw** was not specified, the file system is not remounted. If the file system is currently mounted read-write, specifying **ro** along with **remount** results in an error. If the file system is not currently mounted, an error results.

The default is '**rw, suid**'.

options specific to **4.2** file systems:

quota | **noquota**	Usage limits are enforced, or are not enforced. The default is **noquota**.

options specific to **nfs** (NFS) file systems:

bg | **fg**	If the first attempt fails, retry in the background, or, in the foreground.
noquota	Prevent **quota**(1) from checking whether the user is over quota on this file system; if the file system has quotas enabled on the server, quotas will still be checked for operations on this file system.
retry=*n*	The number of times to retry the mount operation.
rsize=*n*	Set the read buffer size to *n* bytes.
wsize=*n*	Set the write buffer size to *n* bytes.
timeo=*n*	Set the NFS timeout to *n* tenths of a second.
retrans=*n*	The number of NFS retransmissions.
port=*n*	The server IP port number.
soft | **hard**	Return an error if the server does not respond, or continue the retry request until the server responds.
intr	Allow keyboard interrupts on hard mounts.
secure	Use a more secure protocol for NFS transactions.
posix	Request POSIX.1 semantics for the file system. Requires a mount version 2 **mountd**(8C) on the server.

acregmin=*n* Hold cached attributes for at least *n* seconds after file modification.
acregmax=*n* Hold cached attributes for no more than *n* seconds after file modification.
acdirmin=*n* Hold cached attributes for at least *n* seconds after directory update.
acdirmax=*n* Hold cached attributes for no more than *n* seconds after directory update.
actimeo=*n* Set *min* and *max* times for regular files and directories to *n* seconds.
nocto Suppress fresh attributes when opening a file.
noac Suppress attribute and name (lookup) caching.

Regular defaults are:

fg,retry=10000,timeo=7,retrans=3,port=NFS_PORT,hard,
acregmin=3,acregmax=60,acdirmin=30,acdirmax=60

actimeo has no default; it sets **acregmin**, **acregmax**, **acdirmin** and **acdirmax**

Defaults for **rsize** and **wsize** are set internally by the system kernel.

options specific to **rfs** (**RFS**) file systems:

bg | fg If the first attempt fails, retry in the background, or, in the foreground.
retry=*n* The number of times to retry the mount operation.

Defaults are the same as for NFS.

options specific to **hsfs** (**HSFS**) file systems:

norrip Disable processing of Rock Ridge extensions for the file system.

umount

–h *host* Unmount all file systems listed in **/etc/mtab** that are remote-mounted from *host*.

–t *type* Unmount all file systems listed in **/etc/mtab** that are of a given *type*.

–a Unmount all file systems currently mounted (as listed in **/etc/mtab**).

–v Verbose. Display a message indicating each file system being unmounted.

–d Unmount an RFS file system. This option provides compatibility with the System V, Release 3 syntax for unmounting an RFS file system.

NFS FILESYSTEMS

Background vs. Foreground

Filesystems mounted with the **bg** option indicate that **mount** is to retry in the background if the server's mount daemon (mountd(**8C**)) does not respond. **mount** retries the request up to the count specified in the **retry=***n* option. Once the file system is mounted, each NFS request made in the kernel waits **timeo=***n* tenths of a second for a response. If no response arrives, the time-out is multiplied by **2** and the request is retransmitted. When the number of retransmissions has reached the number specified in the **retrans=***n* option, a file system mounted with the **soft** option returns an error on the request; one mounted with the **hard** option prints a warning message and continues to retry the request.

Read-Write vs. Read-Only

File systems that are mounted **rw** (read-write) should use the **hard** option.

Interrupting Processes With Pending NFS Requests

The **intr** option allows keyboard interrupts to kill a process that is hung while waiting for a response on a hard-mounted file system.

Quotas

Quota checking on NFS file systems is performed by the server, not the client; if the file system has the **quota** option on the server, quota checking is performed for both local requests and NFS requests. When a user logs in, **login**(1) runs the **quota**(1) program to check whether the user is over their quota on any of the file systems mounted on the machine. This check is performed for NFS file systems by an RPC call to the **rquotad**(8C) server on the machine from which the file system is mounted. This can be time-consuming, especially if the remote machine is down. If the **noquota** option is specified for an NFS file system, **quota** will not check whether the user is over their quota on that file system, which can speed up the process of logging in. This does *not* disable quota checking for operations on that file system; it merely disables reporting whether the user is over quota on that file system.

Secure Filesystems

The **secure** option must be given if the server requires secure mounting for the file system.

File Attributes

The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be flushed. If the file is modified before the flush time, then the flush time is extended by the time since the last modification (under the assumption that files that changed recently are likely to change soon). There is a minimum and maximum flush time extension for regular files and for directories. Setting **actimeo=***n* extends flush time by *n* seconds for both regular files and directories.

SYSTEM V COMPATIBILITY

System V File-Creation Semantics

Ordinarily, when a file is created its GID is set to the effective GID of the calling process. This behavior may be overridden on a per-directory basis, by setting the set-GID bit of the parent directory; in this case, the GID is set to the GID of the parent directory (see **open**(2V) and **mkdir**(2V)). Files created on file systems that are mounted with the **grpid** option will obey BSD semantics; that is, the GID is unconditionally inherited from that of the parent directory.

EXAMPLES

To mount a local disk:

mount /dev/xy0g /usr

To fake an entry for **nd** root:

mount –ft 4.2 /dev/nd0 /

To mount all 4.2 file systems:

mount –at 4.2

To mount a remote file system:

mount –t nfs serv:/usr/src /usr/src

To mount a remote file system:

mount serv:/usr/src /usr/src

To hard mount a remote file system:

mount –o hard serv:/usr/src /usr/src

To mount an RFS remote file system, retrying in the background on failure:

mount –d –o bg SRC /usr/src

To mount an RFS remote file system read-only:

mount –t rfs –r SRC /usr/src

To save current mount state:

mount –p > /etc/fstab

Note: this is not recommended when running the automounter, see **automount**(8).

To loopback mount file systems:

mount –t lo /export/tmp/localhost /tmp
mount –t lo /export/var/localhost /var lo
mount –t lo /export/cluster/sun386.sunos4.0.1 /usr/cluster
mount –t lo /export/local/sun386 /usr/local

FILES

/etc/mtab	table of mounted file systems
/etc/fstab	table of file systems mounted at boot

WARNINGS

mount does not understand the mount order dependencies involved in loopback mounting. Loopback mounts may be dependent on two mounts having been previously performed, while **nfs** and **4.2** mounts are dependent only on a single previous mount. As a rule of thumb, place loopback mounts at the end of the **/etc/fstab** file. See **lofs**(4S) for a complete description.

SEE ALSO

mkdir(2V), **mount**(2V), **open**(2V), **unmount**(2V), **lofs**(4S), **fstab**(5), **mtab**(5), **automount**(8), **mountd**(8C), **nfsd**(8)

BUGS

Mounting file systems full of garbage crashes the system.

If the directory on which a file system is to be mounted is a symbolic link, the file system is mounted on *the directory to which the symbolic link refers,* rather than being mounted on top of the symbolic link itself.

NAME

newaliases – rebuild the data base for the mail aliases file

SYNOPSIS

newaliases

DESCRIPTION

newaliases rebuilds the random access data base for the mail aliases file **/etc/aliases**. It is run automatically by **sendmail**(8) (in the default configuration) whenever a message is sent.

FILES

/etc/aliases

SEE ALSO

aliases(5), **sendmail**(8)

NAME

newfs – create a new file system

SYNOPSIS

/usr/etc/newfs [**–Nv**] [*mkfs-options*] *raw-special-device*

DESCRIPTION

newfs is a "friendly" front-end to the **mkfs**(8) program. On Sun systems, the disk type is determined by reading the disk label for the specified *raw-special-device*.

raw-special-device is the name of a raw special device residing in **/dev**, including the disk partition, where you want the new file system to be created. If you want to make a file system on **sd0[a-h]**, specify **sd0[a-h]**, **rsd0[a-h]** or **/dev/rsd0[a-h]**; if you only specify **sd0[a-h]**, **newfs** will find the proper device.

newfs then calculates the appropriate parameters to use in calling **mkfs**, and builds the file system by forking **mkfs**.

You must be super-user to use this command.

OPTIONS

–N Print out the file system parameters without actually creating the file system.

–v Verbose. **newfs** prints out its actions, including the parameters passed to **mkfs**.

mkfs-options

Options that override the default parameters passed to **mkfs**(8) are:

–a *apc* Number of alternates per cylinder (SCSI devices only).

–b *block-size*

The block size of the file system in bytes. The default is **8192**.

–C *maxcontig*

The maximum number of blocks, belonging to one file, that will be allocated contiguously before inserting a rotational delay. The default varies from drive to drive. Drives without internal buffers (or drives/controllers that don't advertise the existence of an internal buffer) default to **1**. Drives with buffers default to **7**.

This parameter is limited in the following way:

blocksize * *maxcontig* must be <= *maxphys*

maxphys is a read-only kernel variable that specifies the maximum block transfer size (in bytes) that the I/O subsystem is capable of satisfying. (This limit is enforced by **mount**(2), not by **newfs** or **mkfs**.)

Note: This parameter also controls clustering. Regardless of the value of *rotdelay*, clustering is enabled only when *maxcontig* is greater than 1. Clustering allows higher I/O rates for sequential I/O and is described in **tunefs**(8).

–c *#cylinders/group*

The number of cylinders per cylinder group in a file system. The default is **16**.

–d *rotdelay*

This specifies the expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive clusters/blocks in a file.

–f *frag-size*
The fragment size of the file system in bytes. The default is **1024**.

–i *bytes/inode*
This specifies the density of inodes in the file system. The default is to create an inode for each 2048 bytes of data space. If fewer inodes are desired, a larger number should be used; to create more inodes a smaller number should be given.

–m *free-space%*
The percentage of space reserved from normal users; the minimum free space threshold. The default is **10%**.

–o *optimization*
(**space** or **time**). The file system can either be instructed to try to minimize the time spent allocating blocks, or to try to minimize the space fragmentation on the disk. If the minimum free space threshold (as specified by the **–m** option) is less than 10%, the default is to optimize for **space**; if the minimum free space threshold is greater than or equal to 10%, the default is to optimize for **time**.

–r *revolutions/minute*
The speed of the disk in revolutions per minute (frequently 3600).

–s *size* The size of the file system in sectors.

–t *#tracks/cylinder*
The number of tracks per cylinders on the disk. The default is **16**.

–n *#rotational-positions*
The number of distinguished rotational positions. The default is **8**.

EXAMPLES

The following example verbosely displays the parameters for the raw special device, **sd0a**, but does not actually create a new file system:

```
example% /usr/etc/newfs –vN sd0a
mkfs –N /dev/rsd0a 16048 34 8 8192 1024 16 10 60 2048 t 0 -1 8 -1
/dev/rsd0a:     16048 sectors in 59 cylinders of 8 tracks, 34 sectors
        8.2Mb in 4 cyl groups (16 c/g, 2.23Mb/g, 896 i/g)
super-block backups (for fsck –b#) at:
 32, 4432, 8832, 13232,
example%
```

SEE ALSO

fs(5), **fsck**(8), **installboot**(8S), **mkfs**(8), **tunefs**(8)

System and Network Administration

DIAGNOSTICS

newfs: *special* **No such file or directory**
The device specified does not exist, or a disk partition was not specified.

special: **cannot open**
You must be super-user to use this command.

NOTES

To install the bootstrap programs for a root partition, run **installboot**(8S) after **newfs**.

NAME

newkey – create a new key in the publickey database

SYNOPSIS

newkey –h *hostname*
newkey –u *username*

DESCRIPTION

newkey is normally run by the network administrator on the Network Information Service (NIS) master machine in order to establish public keys for users and super-users on the network. These keys are needed for using secure RPC or secure NFS.

newkey will prompt for the login password of the given username and then create a new public/secret key pair in **/etc/publickey** encrypted with the login password of the given user.

Use of this program is not required: users may create their own keys using **chkey**(1).

OPTIONS

–h *hostname* Create a new public key for the super-user at the given hostname. Prompts for the root password of the given hostname.

–u *username* Create a new public key for the given username. Prompts for the NIS password of the given username.

SEE ALSO

chkey(1), **keylogin**(1), **publickey**(5), **keyserv**(8C)

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

nfsd, biod – NFS daemons

SYNOPSIS

/usr/etc/nfsd [*nservers*]

/usr/etc/biod [*nservers*]

DESCRIPTION

nfsd starts the daemons that handle client filesystem requests. *nservers* is the number of file system request daemons to start. This number should be based on the load expected on this server. Eight seems to be a good number.

biod starts *nservers* asynchronous block I/O daemons. This command is used on a NFS client to buffer cache handle read-ahead and write-behind. The magic number for *nservers* in here is also eight.

When a file that is opened by a client is unlinked (by the server), a file with a name of the form **.nfs***XXX* (where *XXX* is a number) is created by the client. When the open file is closed, the **.nfs***XXX* file is removed. If the client crashes before the file can be closed, the **.nfs***XXX* file is not removed.

FILES

.nfs*XXX*	client machine pointer to an open-but-unlinked file

SEE ALSO

exports(5), **mountd**(8C)

NAME

nfsstat – Network File System statistics

SYNOPSIS

nfsstat [**–cmnrsz**]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing the SunOS System Software* for information on how to install optional software.

DESCRIPTION

nfsstat displays statistical information about the NFS (Network File System) and RPC (Remote Procedure Call), interfaces to the kernel. It can also be used to reinitialize this information. If no options are given the default is

nfsstat –cnrs

That is, display everything, but reinitialize nothing.

OPTIONS

–c Display client information. Only the client side NFS and RPC information will be printed. Can be combined with the **–n** and **–r** options to print client NFS or client RPC information only.

–m Display statistics for each NFS mounted file system. This includes the server name and address, mount flags, current read and write sizes, the retransmission count, and the timers used for dynamic retransmission. The **srtt** value contains the smoothed round trip time, the **dev** value contains the estimated deviation, and the **cur** value is the current backed-off retransmission value.

–n Display NFS information. NFS information for both the client and server side will be printed. Can be combined with the **–c** and **–s** options to print client or server NFS information only.

–r Display RPC information.

–s Display server information.

–z Zero (reinitialize) statistics. This option is for use by the super-user only, and can be combined with any of the above options to zero particular sets of statistics after printing them.

DISPLAYS

The server RPC display includes the following fields:

calls The total number of RPC calls received.

badcalls The total number of calls rejected by the RPC layer (the sum of **badlen** and **xdrcall** as defined below).

nullrecv The number of times an RPC call was not available when it was thought to be received.

badlen The number of RPC calls with a length shorter than a minimum-sized RPC call.

xdrcall The number of RPC calls whose header could not be XDR decoded.

The server NFS display shows the number of NFS calls received (**calls**) and rejected (**badcalls**), and the counts and percentages for the various calls that were made.

The client RPC display includes the following fields:

calls The total number of RPC calls made.

badcalls The total number of calls rejected by the RPC layer.

retrans The number of times a call had to be retransmitted due to a timeout while waiting for a reply from the server.

badxid The number of times a reply from a server was received which did not correspond to any outstanding call.

timeout The number of times a call timed out while waiting for a reply from the server.

wait The number of times a call had to wait because no client handle was avaiable.

newcred The number of times authentication information had to be refreshed.

timers The number of times the calculated time-out value was greater than or equal to the minimum specified time-out value for a call.

The client NFS display shows the number of calls sent and rejected, as well as the number of times a **CLIENT** handle was received (**nclget**), the number of times a call had to sleep while awaiting a handle (**nclsleep**), as well as a count of the various calls and their respective percentages.

FILES

/vmunix	system namelist
/dev/kmem	kernel memory

NAME

listen, nlsadmin – network listener service administration for RFS

SYNOPSIS

nlsadmin [**–mx**] [**–edr** *service_code net_spec*] [**–ikqsv** *net_spec*]
[**–lt** *addr net_spec*] [**–a** *service_code* [**–p** *modules*] **–c** *command* **–y** *comment net_spec*]
[**–qz** *code net_spec*] [**–z** *code net_spec*] [*net_spec*]

/usr/etc/listen

AVAILABILITY

This program is available with the *RFS* software installation option. Refer to *Installing the SunOS System Software* for information on how to install optional software.

DESCRIPTION

nlsadmin configures, initiates and terminates network listener (**listen**) servers for the local host. Each network (transport provider) has an associated **listen** daemon to service it locally. The **listen** daemon for each is configured separately. A **listen** daemon accepts network service requests when they arrive, and spawns servers in response to those requests. It can be used on any network (transport provider) that conforms to the transport provider specification.

nlsadmin can also report on the listener processes on a machine, either individually (per network) or collectively.

Changing the list of services provided by the listener produces immediate changes, while changing an address on which the listener listens has no effect until the listener is restarted.

nlsadmin without any options gives a brief usage message.

The *net_spec* argument to **nlsadmin** refers to a particular **listen** daemon. Specifically, *net_spec* is the relative path name of the entry under **/dev** for a given network.

–x Report the status of all of the listener processes installed on this machine.

–e *service_code net_spec*
–d *service_code net_spec*
Enable or disable, respectively, the service indicated by *service_code* for the specified network. The service must have previously been added to the listener for that network (see the **–a** option). When a listener is disabled, processes serving prior requests continue until they complete.

–r *service_code net_spec*
Remove the entry for the *service_code* from that listener's list of services.

–i *net_spec* Initialize or change a listener process for the network specified by *net_spec*. That is, create and initialize the files required by the listener. Initializing a listener with this option does not start it running. The listener must be initialized before assigning addressing or services. Note: the listener should only be initialized once for a given network.

–q *net_spec* Query the status of the listener process for the specified network. If the listener process is active, **nlsadmin** exits with a status of 0. If no such process is active, the exit code is 1. The exit code will be greater than 1 if there is an error.

–s *net_spec*
–k *net_spec*
Start or kill, respectively, the listener process for the indicated network. When a listener is killed, processes that are still running as a result of prior service requests will continue unaffected. The listener runs under its own ID of **listen** with group ID (GID) **adm**. This GID appear in the system password file **/etc/passwd**; the **HOME** directory listed for the GID is concatenated with *net_spec* to determine the location of the listener configuration information for each network.

NAME

old-analyze, analyze – postmortem system crash analyzer

SYNOPSIS

/usr/old/analyze [**–dfmvD**] [**–s** *swapfile*] *corefile* [**system**]

DESCRIPTION

analyze is the post-mortem analyzer for the state of the paging system. In order to use **analyze** you must arrange to get a image of the memory (and possibly the paging area) of the system after it crashes (see **panic**(8S)).

The **analyze** program reads the relevant system data structures from the core image file and indexing information from **/vmunix** (or the specified file) to determine the state of the paging subsystem at the point of crash. It looks at each process in the system, and the resources each is using in an attempt to determine inconsistencies in the paging system state. Normally, the output consists of a sequence of lines showing each active process, its state (whether swapped in or not), its *p0br*, and the number and location of its page table pages. Any pages which are locked while raw I/O is in progress, or which are locked because they are *intransit* are also printed. (Intransit text pages often diagnose as duplicated; you will have to weed these out by hand.)

The program checks that any pages in core which are marked as not modified are, in fact, identical to the swap space copies. It also checks for non-overlap of the swap space, and that the core map entries correspond to the page tables. The state of the free list is also checked.

Options to **analyze**:

–d Print the (sorted) paging area usage.

–f Dump the free list.

–m Dump the entire coremap state.

–v (Long unused.) Use a hugely verbose output format.

–D Print the diskmap for each process.

In general, the output from this program can be confused by processes which were forking, swapping, or exiting or happened to be in unusual states when the crash occurred. You should examine the flags fields of relevant processes in the output of a **pstat**(8) to weed out such processes.

It is possible to look at the core dump with **adb**(1) if you do

adb –k /vmunix /vmcore

FILES

/vmunix default system namelist

SEE ALSO

adb(1), **ps**(1), **panic**(8S), **pstat**(8)

DIAGNOSTICS

Various diagnostics about overlaps in swap mappings, missing swap mappings, page table entries inconsistent with the core map, incore pages which are marked clean but differ from disk-image copies, pages which are locked or intransit, and inconsistencies in the free list.

It would be nice if this program analyzed the system in general, rather than just the paging system in particular.

NAME

openboot – start the system kernel or a standalone program

SYNOPSIS

SPARCstation 1 SYSTEMS

>b [*device* [(*c*,*u*,*p*)]] [*filename*] *boot-flags*

SPARCstation 2 SYSTEMS, SPARCsystem 600MP SERIES

>b [*device-specifier*] [*filename*] *boot-flags*

AVAILABILITY

Desktop SPARCsystems, SPARCsystem 600MP series only.

DESCRIPTION

The boot program is started by the PROM monitor and loads the kernel, or another executable program, into memory.

USAGE

Booting Standalone

When booting standalone, the boot program (**/boot**) is brought in by the PROM from the file system. The PROM contains drivers for all devices. The boot program simply accesses the PROM device drivers via the ROMVEC interface.

Booting a System Over the Network

When booting over the network, the system PROM obtains a version of the boot program from a server using the Trivial File Transfer Protocol (TFTP). The client broadcasts a RARP request containing its Ethernet address. A server responds with the client's Internet address. The client then sends a TFTP request for its boot program to that server (or if that fails, it broadcasts the request). The filename requested from a server must have a suffix that reflects the *kernel* architecture of the machine being booted. For these systems, the requested filename has the form:

ip-address.*arch*

where *ip-address* is the machine's Internet Protocol (IP) address in hex, and *arch* is a suffix representing its kernel architecture. These filenames are restricted to 14 characters for compatibility with UNIX System V and other operating systems. Therefore, the architecture suffix is limited to 5 characters; it must be in upper case. At present, the following suffixes are recognized: **SUN3** for Sun-3 systems, **SUN3X** for Sun-3x systems, **SUN4** for Sun-4 systems, **SUN4C** for Sun-4c systems, **SUN4M** for Sun-4m systems, **S386** for Sun386i systems, and **PCNFS** for **PC-NFS. arch**(1) may be used to determine the kernel architecture of a machine.

When the Sun server receives the request, it looks in the directory **/tftpboot** for *filename*. That file is typically a symbolic link to the client's boot program, normally **boot.***arch* in the same directory. The server invokes the TFTP server, **tftpd**(8C), to transfer the file to the client.

When the file is successfully read in by the client, the boot program jumps to the load-point and loads **vmunix** (or a standalone program). In order to do this, the boot program makes a broadcast RARP request to find the client's IP address, and then makes a second broadcast request to a **bootparamd**(8) bootparams daemon, for information necessary to boot the client. The bootparams daemon obtains this information either from a local **/etc/bootparams** database file, or from a Network Information Service (NIS) map. The boot program sends two requests to the bootparams daemon — the first, **whoami**, to obtain its hostname, and the second, **getfile**, to obtain the name of the client's server and the pathname of the client's root partition.

The boot program then performs a **mount**(8) operation to mount the client's root partition, after which it can read in and execute any program within that partition by pathname (including a symbolic link to another file within that same partition). Typically, it reads in the file **/vmunix**. If the program is not read in successfully, **boot** responds with a short diagnostic message.

System Startup

Once the system is loaded and running, the kernel performs some internal housekeeping, configures its device drivers, and allocates its internal tables and buffers. The kernel then starts process number 1 to run **init**(8), which performs file system housekeeping, starts system daemons, initializes the system console, and begins multiuser operation. Some of these activities are omitted when **init** is invoked with certain *boot-flags*. These are typically entered as arguments to the boot command and passed along by the kernel to **init**.

OPTIONS

device One of:

le	Lance Ethernet
sd	SCSI disk, CDROM
st	SCSI 1/4" or 1/2" tape
fd	Diskette (Desktop SPARCsystems only)
id	IPI disk (SPARCsystem 600MP series only)

c Controller number, **0** if there is only one controller for the indicated type of device.

u Unit number, **0** if there is only one driver.

p Partition number when booting off a disk, or tape file number when booting from a tape. Defaults to **0**.

device-specifier

The *device-specifier* is a device name or a device alias. A default boot device is used if the *device-specifier* is not given. If a *device-specifier* is given and it is not a device alias, it is checked to see whether it is a valid device name. If it is not a valid device name (indicated by a leading "/"), then the default boot device is used, and the word is considered to be the *filename.* Refer to the *Open Boot PROM 2.0 Toolkit User's Guide*. The Open Boot Prom(OBP) command **show-devs** can be used to retrieve all of the devices known to the system. A valid device name of a SCSI disk on a SPARCsystem 600MP, for example, is shown as follows:

/iommu@f,e0000000/sbus@f,e0001000/esp@f,80000/sd@3,0

The OBP command **devalias** displays all of the system built-in and user-defined device aliases. For example, the alias **disk** may represent the *device-path*

/iommu@f,e0000000/sbus@f,e0001000/esp@f,80000/sd@3,0

filename Name of a standalone program in the selected partition, such as **stand/diag** or **vmunix**. Note: *filename* is relative to the root of the selected device and partition. It never begins with a '/' (slash). If *filename* is not given, the boot program uses a default value (normally **vmunix**). This is stored in the **vmunix** variable in the **boot** executable file supplied by Sun, but can be patched to indicate another standalone program loaded using **adb**(1).

boot-flags The boot program passes all *boot-flags* to the kernel or standalone program. They are typically arguments to that program or, as with those listed below, arguments to programs that it invokes.

–a Prompt interactively for the device and name of the file to boot. For more information on how to boot from a specific device, refer to *Installing the SunOS System Software.*

–v Verbose. Print more detailed information to assist in diagnosing booting problems.

–b Pass the **–b** flag through the kernel to **init**(8) to skip execution of the **/etc/rc.local** script.

–h Halt after loading the system.

–s Pass the **–s** flag through the kernel to **init**(8) for single-user operation.

–d Pass the **–d** debug flag through the debugger (e.g., kadb) to a standalone program being debugged.

–w Pass the **–w** flag to the kernel to mount a root file system read-write.

–i *initname*
Pass the **–i** *initname* to the kernel to tell it to run *initname* as the first program rather than the default **/sbin/init**.

FILES

/boot	standalone boot program
/tftpboot/*address*	symbolic link to the boot program for the client whose Internet address, in uppercase hexadecimal, is *address*
/tftpboot/boot.sun4c	Sun-4c first stage boot program
/tftpboot/boot.sun4m	Sun-4m first stage boot program
/usr/etc/in.tftpd	TFTP server
/usr/kvm/mdec/installboot	program to install boot blocks from a remote host
/vmunix	kernel file that is booted by default
/usr/kvm/boot	
/etc/bootparams	file defining root and swap paths for clients

SEE ALSO

adb(1), **arch**(1), **tftp**(1C), **boot**(8S), **bootparamd**(8), **init**(8), **kadb**(8S), **monitor**(8S), **mount**(8), **ndbootd**(8C), **rc**(8), **reboot**(8), **tftpd**(8C)

Open Boot PROM 2.0 Toolkit User's Guide
Open Boot PROM Toolkit User's Guide
Installing the SunOS System Software
System and Network Administration

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

pac – printer/plotter accounting information

SYNOPSIS

/usr/etc/pac [**–cmrs**] [**–P***printer*] [**–p***price*] [*username*...]

DESCRIPTION

pac reads the printer/plotter accounting files, accumulating the number of pages (the usual case) or feet (for raster devices) of paper consumed by each user, and printing out how much each user consumed in pages or feet and dollars. The accounting file is taken from the **af** field of the **printcap** entry for the printer. If any *username*s are specified, then statistics are only printed for those users; usually, statistics are printed for every user who has used any paper.

OPTIONS

–c Sort the output by cost; usually the output is sorted alphabetically by name.

–m Disregard machine names. Normally, print jobs submitted by a user from different machines would be counted separately for each machine.

–r Reverse the sorting order.

–s Summarize the accounting information on the summary accounting file. The name of the summary file is the name of the accounting file with '**_sum**' appended to it.

–P*printer* Do accounting for the named *printer*. If this option is not used, the printer specified by the PRINTER environment variable will be used if it is present; otherwise accounting is done for the default printer.

–p*price* Use the value *price* for the cost in dollars per page/foot instead of the default value of 0.02.

FILES

/etc/printcap

SEE ALSO

printcap(5)

BUGS

The relationship between the computed price and reality is as yet unknown.

NAME

rarpd – TCP/IP Reverse Address Resolution Protocol server

SYNOPSIS

/usr/etc/rarpd *interface* [*hostname*]

/usr/etc/rarpd –a

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing the SunOS System Software* for information on how to install optional software.

DESCRIPTION

rarpd starts a daemon that responds to Reverse Address Resolution Protocol (RARP) requests. The daemon forks a copy of itself that runs in background. It must be run as root.

RARP is used by machines at boot time to discover their Internet Protocol (IP) address. The booting machine provides its Ethernet Address in an RARP request message. Using the "ethers" and "hosts" databases, **rarpd** maps this Ethernet Address into the corresponding IP address which it returns to the booting machine in an RARP reply message. The booting machine must be listed in both databases for **rarpd** to locate its IP address. **rarpd** issues no reply when it fails to locate an IP address. The "ethers" and "hosts" databases may be contained either in files under **/etc** or in Network Information Service (NIS) maps.

In the first synopsis, the *interface* parameter names the network interface upon which **rarpd** is to listen for requests. The *interface* parameter takes the "name unit" form used by **ifconfig**(8C). The second argument, *hostname*, is used to obtain the IP address of that interface. An IP address in "decimal dot" notation may be used for *hostname*. If *hostname* is omitted, the address of the interface will be obtained from the kernel. When the first form of the command is used, **rarpd** must be run separately for each interface on which RARP service is to be supported. A machine that is a router may invoke **rarpd** multiple times, for example:

/usr/etc/rarpd ie0 host
/usr/etc/rarpd ie1 host-backbone

In the second synopsis, **rarpd** locates all of the network interfaces present on the system and starts a daemon process for each one that supports RARP.

FILES

/etc/ethers
/etc/hosts

SEE ALSO

ethers(5), **hosts**(5), **policies**(5), **boot**(8S), **ifconfig**(8C), **ipallocd**(8C), **netconfig**(8C)

Finlayson, Ross, Timothy Mann, Jeffrey Mogul, and Marvin Theimer, *A Reverse Address Resolution Protocol*, RFC 903, Network Information Center, SRI International, Menlo Park, Calif., June 1984.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

rc, rc.boot, rc.local – command scripts for auto-reboot and daemons

SYNOPSIS

/etc/rc

/etc/rc.boot

/etc/rc.local

DESCRIPTION

rc and **rc.boot** are command scripts that are invoked by **init**(8) to perform file system housekeeping and to start system daemons. **rc.local** is a script for commands that are pertinent only to a specific site or client machine.

rc.boot sets the machine name and, if on SunOS 4.1.1 Rev B or later, invokes **ifconfig**, which uses **RARP** to obtain the machine's IP address from the NIS network. Then a "whoami" bootparams request is used to retrieve the system's hostname, NIS domain name and default router. The **ifconfig** and **hostconfig** programs set the system's hostname, IP address, NIS domain name, and default router in the kernel.

If coming up multi-user, **rc.boot** runs **fsck**(8) with the **–p** option. This "preens" the disks of minor inconsistencies resulting from the last system shutdown and checks for serious inconsistencies caused by hardware or software failure. If **fsck**(8) detects a serious disk problem, it returns an error and **init**(8) brings the system up in single-user mode. When coming up single-user, when **init**(8) is invoked by **fastboot**(8), or when it is passed the **–b** flag from **boot**(8S), functions performed in the **rc.local** file, including this disk check, are skipped.

Next, **rc** runs. If the system came up single-user, **rc** runs when the single-user shell terminates (see **init**(8)). It mounts 4.2 filesystems and spawns a shell for **/etc/rc.local**, which mounts NFS filesystems, runs **sysIDtool** (if on SunOS 4.1.1 Rev B or later) to set the system's configuration information into local configuration files, and starts local daemons. After **rc.local** returns, **rc** starts standard daemons, preserves editor files, clears **/tmp**, starts system accounting (if applicable), starts the network (where applicable), and if enabled, runs **savecore**(8) to preserve the core image after a crash.

Sun386i

These files operate as described above with the following variations:

fsck(8) is invoked with the **–y** option to prevent users being put in single-user mode by happenstance.

rc.boot invokes **netconfig**(8C) to configure the system for the network before booting. **netconfig** is invoked before the **/usr** filesystem is mounted, because **/usr** might be mounted from a server. **netconfig** writes **/etc/net.conf** unless the **–n** option is specified, controlling system booting.

rc.boot dynamically loads device drivers.

rc invokes any programs found in **/var/recover** to clean up any operations partially completed when the system crashed or was shut down.

rc.local starts the automounter.

The file **/etc/net.conf** stores these environment variables: The **VERBOSE** environment variable controls the verbosity of the messages from the **rc** script; its value is taken from **NVRAM**. The **NETWORKED** environment variable controls whether services useful only on a networked system are started in **/etc/rc.local**. The **PNP** environment variable, set up during initial system installation, controls whether local network configuration information is used or whether that information comes from the network. (Using automatic system installation causes all systems except boot servers to get this information from the network, facilitating network reconfiguration.) The **HOSTNAME** and **DOMAINNAME** environment variables, used together, help determine if this system is a boot server or, with **PNP** set to **no**, control the host name and domain name.

FILES

/etc/rc
/etc/rc.boot

/etc/rc.local
/etc/net.conf
/var/recover/*
/var/yp/*
/tmp

SEE ALSO

automount(8), **boot**(8S), **fastboot**(8), **hostconfig**(8), **ifconfig**(8C), **init**(8), **reboot**(8), **savecore**(8), **netconfig**(8C)

BUGS

The system message file **/var/adm/messages** is no longer created automatically.

NAME

rdate – set system date from a remote host

SYNOPSIS

/usr/ucb/rdate *hostname*

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing the SunOS System Software* for information on how to install optional software.

DESCRIPTION

rdate sets the local date and time from the *hostname* given as argument. You must be super-user on the local system. Typically **rdate** can be inserted as part of your **/etc/rc.local** startup script.

FILES

/etc/rc.local

BUGS

Could be modified to accept a list of hostnames and try each until a valid date returned. Better yet would be to write a real date server that accepted broadcast requests.

NAME

reboot – restart the operating system

SYNOPSIS

/usr/etc/reboot [**–dnq**] [*boot arguments*]

DESCRIPTION

reboot executes the **reboot**(2) system call to restart the kernel. The kernel is loaded into memory by the PROM monitor, which transfers control to it. See **boot**(8S) for details.

Although **reboot** can be run by the super-user at any time, **shutdown**(8) is normally used first to warn all users logged in of the impending loss of service. See **shutdown**(8) for details.

reboot performs a **sync**(1) operation on the disks, and then a multiuser **reboot** is initiated. See **init**(8) for details.

reboot normally logs the reboot to the system log daemon, **syslogd**(8), and places a shutdown record in the login accounting file **/var/adm/wtmp**. These actions are inhibited if the **–n** or **–q** options are present.

Power Fail and Crash Recovery

Normally, the system will reboot itself at power-up or after crashes.

OPTIONS

–d Dump system core before rebooting.

–n Avoid the **sync**(1). It can be used if a disk or the processor is on fire.

–q Quick. Reboots quickly and ungracefully, without first shutting down running processes.

Boot Arguments

If a boot argument string is given, it is passed to the boot command in the PROM monitor. The string must be quoted if it contains spaces or other characters that could be interpreted by the shell. If the first character of the boot argument string is a minus sign '–' the string must be preceded by an option terminator string '– –' For example:

reboot – – –s

to reboot and come up single user,

reboot vmunix.test

to reboot to a new kernel. See **boot**(8S) for details.

FILES

/var/adm/wtmp login accounting file

SEE ALSO

sync(1), **reboot**(2), **boot**(8S), **fastboot**(8), **fasthalt**(8), **fsck**(8), **halt**(8), **init**(8), **panic**(8S), **shutdown**(8), **syslogd**(8)

NAME

renice – alter nice value of running processes

SYNOPSIS

/usr/etc/renice *priority pid...*

/usr/etc/renice *priority* [**–p** *pid...*] [**–g** *pgrp...*] [**–u** *username...*]

DESCRIPTION

renice alters the scheduling nice value, and hence the priority, of one or more running processes. See **nice**(1) for a discussion of nice value and process scheduling priority.

OPTIONS

By default, the processes to be affected are specified by their process IDs. *priority* is the new priority value.

–p *pid...* Specify a list of process IDs.

–g *pgrp...* Specify a list of process group IDs. The processes in the specified process groups have their scheduling priority altered.

–u *user...* Specify a list of user IDs or usernames. All processes owned by each *user* have their scheduling altered.

Users other than the super-user may only alter the priority of processes they own, and can only monotonically increase their "nice value" within the range 0 to 20. (This prevents overriding administrative fiats.) The super-user may alter the priority of any process and set the priority to any value in the range –20 to 19. Useful nice values are 19 (the affected processes will run only when nothing else in the system wants to), 0 (the default nice value) and any negative value (to make things go faster).

If only the priority is specified, the current process (alternatively, process group or user) is used.

FILES

/etc/passwd to map user names to user ID's

SEE ALSO

pstat(8)

BUGS

If you make the nice value very negative, then the process cannot be interrupted.

To regain control you must make the priority greater than zero.

Users other than the super-user cannot increase scheduling priorities of their own processes, even if they were the ones that decreased the priorities in the first place.

NAME

savecore – save a core dump of the operating system

SYNOPSIS

/usr/etc/savecore [**–v**] *directory* [*system-name*]

DESCRIPTION

savecore saves a core dump of the kernel (assuming that one was made) and writes a reboot message in the shutdown log. It is meant to be called near the end of the **/etc/rc.local** file after the system boots. However, it is not normally run by default. You must edit that file to enable it.

savecore checks the core dump to be certain it corresponds with the version of the operating system currently running. If it does, **savecore** saves the core image in the file *directory*/**vmcore.***n* and the kernel's namelist in *directory*/**vmunix.***n*. The trailing **.***n* in the pathnames is replaced by a number which grows every time **savecore** is run in that directory.

Before **savecore** writes out a core image, it reads a number from the file *directory*/**minfree**. This is the minimum number of kilobytes that must remain free on the filesystem containing *directory*. If there is less free space on the filesystem containing *directory* than the number of kilobytes specified in **minfree**, the core dump is not saved. If the **minfree** file does not exist, **savecore** always writes out the core file (assuming that a core dump was taken).

savecore also logs a reboot message using facility **LOG_AUTH** (see **syslog**(3)). If the system crashed as a result of a panic, **savecore** logs the panic string too.

If the core dump was from a system other than **/vmunix**, the name of that system must be supplied as *system-name*.

OPTIONS

–v Verbose. Enable verbose error messages from **savecore**.

FILES

directory/**vmcore.***n*
directory/**vmunix.***n*
directory/**minfree**
/vmunix the kernel
/etc/rc.local

SEE ALSO

syslog(3), **panic**(8S), **sa**(8)

BUGS

savecore can be fooled into thinking a core dump is the wrong size.

You must run **savecore** very soon after booting — before the swap space containing the crash dump is overwritten by programs currently running.

Core images produced by SPARCstation1 systems, and from machines with discontiguous physical memory, are sparse and contain holes. For example, a core image of an 8 megabyte SPARCstation 1 might contain 3 to 4 megabytes of useful information, and thus only occupy 3 to 4 megabytes of disk space, yet contain enough holes to appear to be 36 megabytes in size. However, copying the core image will manifest the holes, so that this copy will require 36 megabytes of disk space. If it is necessary to move a core image, it is strongly recommended that the core image be compressed with **compress**(1) before the transfer. The compressed image may later be uncompressed on a system with sufficient disk space.

NAME

sendmail – send mail over the internet

SYNOPSIS

/usr/lib/sendmail [**–ba**] [**–bd**] [**–bi**] [**–bm**] [**–bp**] [**–bs**] [**–bt**] [**–bv**] [**–bz**]
[**–C***file*] [**–d***X*] [**–F***fullname*] [**–f***name*] [**–h***N*] [**–n**] [**–o***x value*] [**–q**[*time*]]
[**–r***name*] [**–R***string*] [**–t**] [**–v**] [*address* ...]

DESCRIPTION

sendmail sends a message to one or more people, routing the message over whatever networks are necessary. **sendmail** does internetwork forwarding as necessary to deliver the message to the correct place.

sendmail is not intended as a user interface routine; other programs provide user-friendly front ends; **sendmail** is used only to deliver pre-formatted messages.

With no flags, **sendmail** reads its standard input up to an EOF, or a line with a single dot and sends a copy of the letter found there to all of the addresses listed. It determines the network to use based on the syntax and contents of the addresses.

Local addresses are looked up in the local **aliases**(5) file, or by using the Network Information Service (NIS), and aliased appropriately. In addition, if there is a **.forward** file in a recipient's home directory, **sendmail** forwards a copy of each message to the list of recipients that file contains. Aliasing can be prevented by preceding the address with a backslash. Normally the sender is not included in alias expansions, for example, if 'john' sends to 'group', and 'group' includes 'john' in the expansion, then the letter will not be delivered to 'john'.

sendmail will also route mail directly to other known hosts in a local network. The list of hosts to which mail is directly sent is maintained in the file **/usr/lib/mailhosts**.

OPTIONS

–ba Go into ARPANET mode. All input lines must end with a LINEFEED, and all messages will be generated with a CR-LF at the end. Also, the ''From:'' and ''Sender:'' fields are examined for the name of the sender.

–bd Run as a daemon, waiting for incoming SMTP connections.

–bi Initialize the alias database.

–bm Deliver mail in the usual way (default).

–bp Print a summary of the mail queue.

–bs Use the SMTP protocol as described in RFC 821. This flag implies all the operations of the **–ba** flag that are compatible with SMTP.

–bt Run in address test mode. This mode reads addresses and shows the steps in parsing; it is used for debugging configuration tables.

–bv Verify names only — do not try to collect or deliver a message. Verify mode is normally used for validating users or mailing lists.

–bz Create the configuration freeze file.

–C*file* Use alternate configuration file.

–d*X* Set debugging value to *X*.

–F*fullname* Set the full name of the sender.

–f*name* Sets the name of the ''from'' person (that is, the sender of the mail). **–f** can only be used by ''trusted'' users (who are listed in the config file).

–h*N* Set the hop count to *N*. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop.

–M*id* Attempt to deliver the queued message with message-id **id**.

–n Do not do aliasing.

–o*x value* Set option *x* to the specified *value*. Options are described below.

–q[*time*] Processed saved messages in the queue at given intervals. If *time* is omitted, process the queue once. *time* is given as a tagged number, with **s** being seconds, **m** being minutes, **h** being hours, **d** being days, and **w** being weeks. For example, **–q1h30m** or **–q90m** would both set the timeout to one hour thirty minutes.

–r*name* An alternate and obsolete form of the **–f** flag.

–R*string* Go through the queue of pending mail and attempt to deliver any message with a recipient containing the specified string. This is useful for clearing out mail directed to a machine which has been down for awhile.

–t Read message for recipients. ''To:'', ''Cc:'', and ''Bcc:'' lines will be scanned for people to send to. The ''Bcc:'' line will be deleted before transmission. Any addresses in the argument list will be suppressed.

–v Go into verbose mode. Alias expansions will be announced, etc.

PROCESSING OPTIONS

There are also a number of processing options that may be set. Normally these will only be used by a system administrator. Options may be set either on the command line using the **–o** flag or in the configuration file. These are described in detail in the *Installation and Operation Guide*. The options are:

A*file* Use alternate alias file.

c On mailers that are considered ''expensive'' to connect to, do not initiate immediate connection. This requires queueing.

d*x* Set the delivery mode to **x**. Delivery modes are **i** for interactive (synchronous) delivery, **b** for background (asynchronous) delivery, and **q** for queue only — that is, actual delivery is done the next time the queue is run.

D Run **newaliases**(8) to automatically rebuild the alias database, if necessary.

e*x* Set error processing to mode **x**. Valid modes are **m** to mail back the error message, **w** to ''write'' back the error message (or mail it back if the sender is not logged in), **p** to print the errors on the terminal (default), 'q' to throw away error messages (only exit status is returned), and 'e' to do special processing for the BerkNet. If the text of the message is not mailed back by modes **m** or **w** and if the sender is local to this machine, a copy of the message is appended to the file **dead.letter** in the sender's home directory.

F*mode* The mode to use when creating temporary files.

f Save UNIX-system-style ''From'' lines at the front of messages.

g*N* The default group ID to use when calling mailers.

H*file* The **SMTP** help file.

i Do not take dots on a line by themselves as a message terminator.

L*n* The log level.

m Send to ''me'' (the sender) also if I am in an alias expansion.

o If set, this message may have old style headers. If not set, this message is guaranteed to have new style headers (that is, commas instead of spaces between addresses). If set, an adaptive algorithm is used that will correctly determine the header format in most cases.

Q*queuedir*
Select the directory in which to queue messages.

r*timeout*
The timeout on reads; if none is set, **sendmail** will wait forever for a mailer.

S*file* Save statistics in the named file.

s Always instantiate the queue file, even under circumstances where it is not strictly necessary.

T*time* Set the timeout on messages in the queue to the specified time. After sitting in the queue for this amount of time, they will be returned to the sender. The default is three days.

t*stz,dtz* Set the name of the time zone.

u*N* Set the default user id for mailers.

If the first character of the user name is a vertical bar, the rest of the user name is used as the name of a program to pipe the mail to. It may be necessary to quote the name of the user to keep **sendmail** from suppressing the blanks from between arguments.

sendmail returns an exit status describing what it did. The codes are defined in **sysexits.h**

EX_OK	Successful completion on all addresses.
EX_NOUSER	User name not recognized.
EX_UNAVAILABLE	Catchall meaning necessary resources were not available.
EX_SYNTAX	Syntax error in address.
EX_SOFTWARE	Internal software error, including bad arguments.
EX_OSERR	Temporary operating system error, such as "cannot fork".
EX_NOHOST	Host name not recognized.
EX_TEMPFAIL	Message could not be sent immediately, but was queued.

If invoked as **newaliases**, **sendmail** rebuilds the alias database. If invoked as **mailq**, **sendmail** prints the contents of the mail queue.

FILES

Except for **/etc/sendmail.cf**, these pathnames are all specified in **/etc/sendmail.cf**. Thus, these pathnames are only approximations.

/etc/aliases	raw data for alias names
/etc/aliases.dir	
/etc/aliases.pag	data base of alias names
/etc/sendmail.cf	configuration file
/etc/sendmail.fc	frozen configuration
/etc/sendmail.st	collected statistics
/usr/bin/mail	to deliver local mail
/usr/bin/uux	to deliver uucp mail
/usr/lib/mailhosts	list of hosts to which mail can be sent directly
/usr/lib/sendmail.hf	help file
/var/spool/mqueue/*	temp files and queued mail
~/.forward	list of recipients for forwarding messages

SEE ALSO

biff(1), **bin-mail**(1), **mail**(1), **aliases**(5) **newaliases**(8),

System and Network Administration

Su, Zaw-Sing, and Jon Postel, *The Domain Naming Convention for Internet User Applications*, RFC 819, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

Postel, Jon, *Simple Mail Transfer Protocol*, RFC 821, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

Crocker, Dave, *Standard for the Format of ARPA-Internet Text Messages*, RFC 822, Network Information Center, SRI International, Menlo Park, Calif., August 1982.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

set4, unset4, check4 – set, unset, and check the 4 megabyte process virtual address space limit flag in a Sun386i module

SYNOPSIS

set4 [**–d** *working_directory*] [– | *filename*] ...

unset4 *filename* ...

check4 *filename* ...

AVAILABILITY

Available only on Sun 386i systems running a SunOS 4.0.*x* release or earlier. Not a SunOS 4.1 release feature.

DESCRIPTION

set4 sets the 4 megabyte process memory flag in each *filename* program image, limiting the virtual address space for each program to 4 megabytes. If a '–' is used, **set4** reads the standard input for a list of files to set the 4 megabyte limit on. Lines in the standard input whose first character is '**#**' are ignored, so files may include comments.

unset4 clears the 4 megabyte process memory flag in the program image, so the process virtual address space is not limited to 4 megabytes.

check4 reports programs that do not have the 4 megabyte limit set, and does not report programs with the limit set.

OPTIONS

–d *working_directory*

This specifies a directory prefix for file names that **set4** processes.

EXAMPLES

Suppose that the file **small_progs** contains the following:

```
# These files should have their virtual address spaces limited to 4 MB:
/bin/date
/bin/true
```

Then the following command will run **set4** on **/build/bin/false**, **/build/bin/date**, **/build/bin/true**, and **/build/bin/cat**.

```
example% set4 –d /build /bin/false –
/bin/cat < small_progs
example%
```

In this example, **unset4** clears the 4 megabyte limit flag in **date**, and **clri**.

```
example% unset4 /bin/date /etc/clri
example%
```

In the last example, **check4** shows that **date** and **clri** are 4 megabyte processes, but **basename** is not.

```
example% check4 /bin/date /etc/clri /usr/bin/basename
basename is not a 4MB process
example%
```

SEE ALSO

execve(2V) **execl**(3V)

BUGS

There is a problem in the way that processes that have the 4 megabyte limit set **exec( )** processes that do not have the limit set. (See **execve**(2V) and **execl**(3V) for descriptions of **exec( )** processing.) For a short time during the **exec( )**, a child has the parent's data and stack limits. During this time, the program is checked

NAME

showmount – show all remote mounts

SYNOPSIS

/usr/etc/showmount [**–ade**] [*hostname*]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing the SunOS System Software* for information on how to install optional software.

DESCRIPTION

showmount lists all the clients that have remotely mounted a filesystem from *host*. This information is maintained by the **mountd**(8C) server on *host*, and is saved across crashes in the file **/etc/rmtab**. The default value for *host* is the value returned by **hostname**(1).

OPTIONS

–a Print all remote mounts in the format:

hostname:*directory*

where *hostname* is the name of the client, and **directory** is the root of the file system that has been mounted.

–d List directories that have been remotely mounted by clients.

–e Print the list of exported file systems.

FILES

/etc/rmtab

SEE ALSO

hostname(1), **exports**(5), **exports**(5), **exportfs**(8), **mountd**(8C)

BUGS

If a client crashes, its entry will not be removed from the list until it reboots and executes '**umount –a**'.

NAME

showrev – show machine and software revision information

SYNOPSIS

/usr/etc/showrev [**–a**] [**–p**] [[**–c**] *command*]

DESCRIPTION

showrev displays revision information for the current hardware and software. With no arguments, **showrev** shows the system revision information including hostname, hostid, kernel architecture, application architecture, kernel revision and release. On SunOS 4.1.1 Rev B or later, OS release revision, OpenWindows version, Sunview version and patch information may also be displayed.

If a command is supplied with the **–c** option, **showrev** shows the PATH and finds out all the directories that contain it. For each file found, its library information, SCCS Id, permission and checksum are printed as well.

OPTIONS

–a	Print all system revision information available. Window system and patch information are added.
–p	Print only the revision information about patches.
–c *command*	Print the revision information about *command*.

FILES

/usr/kvm/arch	system architecture information
/sbin/hostname	hostname information
/etc/hosts	general host information
/vmunix	kernel
/usr/sys/conf.common/RELEASE	OS release information - kernel name
/usr/bin/sunview	SunView executable
/usr/kvm/showrev.dat	OS release revision information
/etc/install/patch*	patch release information

SEE ALSO

arch(1), **cc**(1V), **ldd**(1), **sum**(1), **sccs**(1), **what**(1)

NAME

shutdown – close down the system at a given time

SYNOPSIS

/usr/etc/shutdown [**–fhknr**] *time* [*warning-message* ...]

DESCRIPTION

shutdown provides an automated procedure to notify users when the system is to be shut down. *time* specifies when **shutdown** will bring the system down; it may be the word **now** (indicating an immediate shutdown), or it may specify a future time in one of two formats: **+***number* and *hour***:***min.* The first form brings the system down in *number* minutes, and the second brings the system down at the time of day indicated in 24-hour notation.

At intervals that get closer as the apocalypse approaches, warning messages are displayed at terminals of all logged-in users, and of users who have remote mounts on that machine. Five minutes before shutdown, or immediately if shutdown is in less than 5 minutes, logins are disabled by creating **/etc/nologin** and writing a message there. If this file exists when a user attempts to log in, **login**(1) prints its contents and exits. The file is removed just before **shutdown** exits.

At shutdown time a message is written to the system log daemon, **syslogd**(8), containing the time of shutdown, the instigator of the shutdown, and the reason. Then a terminate signal is sent to **init**, which brings the system down to single-user mode.

The time of the shutdown and the warning message are placed in **/etc/nologin**, which should be used to inform the users as to when the system will be back up, and why it is going down (or anything else).

OPTIONS

As an alternative to the above procedure, these options can be specified:

–f Shut the system down in the manner of **fasthalt** (see **fastboot**(8)), so that when the system is rebooted, the file systems are not checked.

–h Execute **halt**(8).

–k Simulate shutdown of the system. Do not actually shut down the system.

–n Prevent the normal **sync**(2) before stopping.

–r Execute **reboot**(8).

FILES

/etc/nologin	tells login not to let anyone log in
/etc/xtab	list of remote hosts that have mounted this host

SEE ALSO

login(1), **sync**(2), **fastboot**(8), **halt**(8), **reboot**(8), **syslogd**(8)

BUGS

Only allows you to bring the system down between "now" and 23:59 if you use the absolute time for shutdown.

NAME

sticky – mark files for special treatment

DESCRIPTION

The *sticky bit* (file mode bit 01000, see **chmod**(2V)) is used to indicate special treatment of certain files and directories. A directory for which the sticky bit is set restricts deletion of files it contains. A file in a sticky directory may only be removed or renamed by a user who has write permission on the directory, and either owns the file, owns the directory, or is the super-user. This is useful for directories such as **/tmp**, which must be publicly writable, but should deny users permission to arbitrarily delete or rename the files of others.

If the sticky bit is set on a regular file and no execute bits are set, the system's page cache will not be used to hold the file's data. This bit is normally set on swap files of diskless clients so that accesses to these files do not flush more valuable data from the system's cache. Moreover, by default such files are treated as swap files, whose inode modification times may not necessarily be correctly recorded on permanent storage.

Any user may create a sticky directory. See **chmod** for details about modifying file modes.

BUGS

mkdir(2V) will not create a file with the sticky bit set.

FILES

/tmp

SEE ALSO

chmod(1V), **chmod**(2V), **chown**(2V), **mkdir**(2V)

NAME

sundiag – SunOS hardware diagnostic program

SYNOPSIS

/usr/diag/sundiag/sundiag [**–Cmpqtvw**] [**–a** | **–h** *hostname*] [**–o** *option_file*] [**–b** *batch_file*]
[**–k** *kernel_name*] [**–o** *saved_options_file*] [*sunview_arguments*]
/usr/diag/sundiag/*testname testname-specific_arguments* [**cprquvdt**] [**h** hostname]

AVAILABILITY

This program is in the diagnostics (*User_Diag*) software category of SunOS. Refer to *Installing SunOS* for information on how to install optional software.

DESCRIPTION

Sundiag is a diagnostic utility that tests hardware functionality. At start-up, Sundiag probes for the hardware installed on the system under test, and displays test options for the hardware it detects. If a hardware device connected to the system under test is *not* detected by **sundiag**, then it is not connected properly.

Only super-user can use Sundiag .

The Sundiag program consists of the **sundiag** window-based user interface, along with several binary modules and executable files containing the actual test code, all of which reside in **/usr/diag/sundiag**.

The Sundiag program can be run from the SunView window environment, from a **tty**, or individual tests can be run from the command line of a C-shell or Bourne shell. Sundiag cannot be used with the OpenWindows user interface yet.

OPTIONS

Running Sundiag in SunView or tty Mode

The following options are available when Sundiag is run from the SunView window environment, or a **tty** interface.

–C Redirect the console output from any existing console window to the **sundiag** console subwindow. If you are using the **tty** interface, the console message is displayed in the message line of the status screen.

–m Create any missing device files for the devices found during the kernel probe. **sundiag** uses the same major/minor device numbers and permissions declared in **/dev/MAKEDEV**.

–p Skip the Sundiag kernel probe for devices. If this argument is specified, Sundiag only runs the user-defined tests it finds in **.usertest**.

–q Automatically quit the Sundiag program when testing stops. This option is designed for use in **sh**(1) or **csh**(1) shell scripts, and can only be issued from a command line.

–t Run **sundiag** in **tty** mode.

–v Suppress Sundiag start-up messages, so they do not interfere with the display when SunView windows come up. This argument is used in your **.sunview** file.

–w Write the system hardware configuration to the **/usr/adm/sundiaglog/sundiag.conf** file.

–a *hostname*
Run **sundiag** in automated test mode. This option requires special Sun automated test equipment and is intended for use by Sun manufacturing.

–h *hostname*
Run Sundiag remotely. Specialized instructions are required; see the *Sundiag 2.3 User's Guide* for details.

–o *options_file*
Use the *options_file* to restore options. The default option file is **.sundiag**. **.sundiag** is used if the **–o** option is not used and if the default file exists.

–b *batch_file*
Run Sundiag in batch mode.

–k *kernel_name*
Specify the customized kernel name that was used to boot the system. The default kernel name is **/vmunix**. The performance monitor is disabled when this option is specified, since it depends upon **rstadt**(8C). **rstadt**(8C), in turn, relies on **/vmunix** as the kernel name.

sunview_arguments
Refer to **sunview**(1) for examples of generic tool arguments that may be used with **sundiag**.

Running Sundiag Tests from a Command Line

The following options are available when running individual Sundiag tests from a command line ("standard arguments").

c Create a core dump file if the system under test crashes.

p Skip any test loops.

r Continue testing after an error has occurred. The test continues with the next test sequence instead of exiting.

q Run a faster, abbreviated version of the test, if it exists.

u Display information on how to run the test. It shows three parts: command line usage, standard arguments and routine specific arguments.

v Display verbose messages regarding the test. These messages tell you more about the testing process that is going on. This mode is more valuable for some tests than others; graphics tests only return start and stop messages/failures.

d Display debug messages from the test. These messages provide more sophisticated information (mainly useful for test programmers).

t Display messages which allow you to trace down function calls and the sequences being used by the test code for some of the tests.

h *hostname*
Specify *hostname* to receive system messages regarding this test.

USAGE

Running the Sundiag Program from SunView

When **sundiag** is started from the SunView window environment, it brings up its own window with four subwindows:

- A test status panel on the upper left of the screen, which shows the test results.
- A performance monitor in the upper middle of the screen, which tracks system activity levels.
- A control panel on the upper right of the screen, which displays the hardware available for test. Select the hardware to be tested by clicking the left mouse button in the small boxes next to each of the hardware items. A "check-mark" will appear in the box next to the tests which have been selected for test. Most hardware items have option menus for changing test parameters. The option menus can be opened by clicking the left mouse button on the **Option** button to the right of each hardware item.
- A console window on the lower right of the screen, which displays system and error messages.

There are also some popup frames, including a text frame for viewing **sundiag** and system log files.

Running the Sundiag Program from a tty Interface

sundiag can be run from a terminal, by specifying the **–t** option (**tty** mode) when Sundiag is started. In **tty** mode, Sundiag emulates the window interface on a terminal screen. The tests and test options available in the window system are also available in **tty** mode. Commands and options are shown in brackets at the top of the **tty** screen, and are typed in at the command line on the bottom of the screen.

When executed from a terminal, **sundiag** uses **curses**(3V) to simulate subwindows on the screen.

Running Individual Sundiag Tests from a Command Line

Sundiag tests can be run individually from a shell command line using the syntax explained above in the SNOPSIS and OPTIONS sections.

FILES

/var/adm/sundiaglog/options/.sundiag	start-up option file
/usr/diag/sundiag/.usertest	user-defined test description file
/var/adm/sundiaglog/sundiag.info	Sundiag status log file
/var/adm/sundiaglog/sundiag.err	Sundiag status error file
/var/adm/messages.*	SunOS system log
/dev/MAKEDEV	

SEE ALSO

sunview(1), **curses**(3V), **rstatd**(8C)

Installing the SunOS System Software
Sundiag 2.3 User's Guide

NAME

suninstall – install and upgrade the SunOS operating system

SYNOPSIS

/usr/etc/install/suninstall

DESCRIPTION

suninstall is a forms-based subsystem for installing and upgrading the SunOS operating system. Unlike previous installation subsystems, **suninstall** does not require recapitulation of an interrupted procedure; you can pick up where you left off. A new invocation of **suninstall** displays the saved information and offers the user an opportunity to make any needed alterations before it proceeds.

Note: **suninstall** only exists in the mini-root and should only be invoked from there (see *Installing the SunOS System Software).*

suninstall allows installation of the operating system onto any system configuration, be it standalone, dataless, a homogeneous file server, or a heterogeneous server. It installs the various versions of the operating system needed by clients on a heterogeneous file server, from any Sun distribution media format. The number of different system versions that can be installed is only limited to the disk space available.

After the initial installation, the suninstall utility program **add_client**(8) adds clients while the server is running in multiuser mode. The suninstall **add_services**(8) program converts a standalone system or server into a heterogeneous file server, without rebooting, while the system is running in multiuser mode. To remove a diskless client, use the suninstall **rm_client**(8) program in multiuser mode.

To abort the installation procedure, use the interrupt character (typically CTRL-C).

USAGE

Refer to *Installing the SunOS System Software* for more information on the various menus and selections.

FILES

/usr/etc/install	directory containing installation programs and scripts
/usr/etc/install/xdrtoc	subsystem utility program
/etc/install	directory containing suninstall data files

SEE ALSO

add_client(8), **add_services**(8), **extract_unbundled**(8), **rm_client**(8)

Installing the SunOS System Software

NOTES

It is advisable to exit **suninstall** through the exit options from the **suninstall** menus.

NAME

sys-unconfig – undo a system's configuration

SYNOPSIS

/usr/etc/install/sys-unconfig

DESCRIPTION

sys-unconfig packs up a machine to make it ready to be configured again.

It restores a systems's configuration to an "as-manufactured" state. A system's configuration consists of hostname, Network Information Service (NIS) domain name, timezone and IP address.

sys-unconfig does the following:

- Restores the default **/etc/hosts** file.
- Removes the default hostname in **/etc/hostname.??[0-9]**.
- Removes the default domainname in **/etc/defaultdomain**.
- Removes the default **/usr/lib/zoneinfo/localtime** file.
- Disables the Network Information Service (NIS) if the NIS service was requested.

When **sys-unconfig** is finished, it will perform a system shutdown.

sys-unconfig is potentially a dangerous utility and can only be run by the super-user.

FILES

/etc/hosts
/usr/lib/zoneinfo/localtime
/usr/etc/install/sys_info

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

NAME

syslogd – log system messages

SYNOPSIS

/usr/etc/syslogd [**–d**] [**–f***configfile*] [**–m** *interval*]

DESCRIPTION

syslogd reads and forwards system messages to the appropriate log files and/or users, depending upon the priority of a message and the system facility from which it originates. The configuration file **/etc/syslog.conf** (see **syslog.conf**(5)) controls where messages are forwarded. **syslogd** logs a mark (timestamp) message every *interval* minutes (default 20) at priority **LOG_INFO** to the facility whose name is given as **mark** in the **syslog.conf** file.

A system message consists of a single line of text, which may be prefixed with a priority code number enclosed in angle-brackets (**<>**); priorities are defined in **sys/syslog.h**.

syslogd reads from the **AF_UNIX** address family socket **/dev/log**, from an Internet address family socket specified in **/etc/services**, and from the special device **/dev/klog** (for kernel messages).

syslogd reads the configuration file when it starts up, and again whenever it receives a HUP signal, at which time it also closes all files it has open, re-reads its configuration file, and then opens only the log files that are listed in that file. **syslogd** exits when it receives a **TERM** signal.

As it starts up, **syslogd** creates the file **/etc/syslog.pid**, if possible, containing its process ID (PID).

Sun386i DESCRIPTION

syslogd translates messages using the databases specified on an optional line in the **syslog.conf** as indicated with a **translate** entry.

The format of these databases is described in **translate**(5).

OPTIONS

–d	Turn on debugging.
–f*configfile*	Specify an alternate configuration file.
–m *interval*	Specify an interval, in minutes, between mark messages.

FILES

/etc/syslog.conf	configuration file
/etc/syslog.pid	process ID
/dev/log	AF_UNIX address family datagram log socket
/dev/klog	kernel log device
/etc/services	network services database

SEE ALSO

logger(1), **syslog**(3), **syslog.conf**(5), **translate**(5)

NAME

tfsd – TFS daemon

SYNOPSIS

/usr/etc/tfsd

DESCRIPTION

tfsd is the daemon for the Translucent File Service (TFS). This daemon is started by **inetd**(8C) whenever a TFS request is made.

tfsd looks up a file by looking in the frontmost directory (see **tfs**(4S)). If the file is not found in this directory, **tfsd** follows the *searchlink* from the frontmost directory to the directory immediately behind it. **tfsd** continues to search for the file until one of the following conditions is met:

- The file is found in a directory.
- There are no more searchlinks to follow.
- A *whiteout* entry for the file is found.

The searchlinks and whiteout entries are specified in **.tfs_info** files.

FILES

.tfs_info	holds searchlink and whiteout entries

SEE ALSO

unwhiteout(1), **lsw**(1), **tfs**(4S), **mount_tfs**(8)

NAME

tftpd, in.tftpd – TCP/IP Trivial File Transfer Protocol server

SYNOPSIS

/usr/etc/in.tftpd [**–s**] [*homedir*]

Sun386i SYNOPSIS

/usr/etc/in.tftpd [**–s**] [**–p**] [*homedir*]

AVAILABILITY

This program is available with the *Networking* software installation option. Refer to *Installing the SunOS System Software* for information on how to install optional software.

DESCRIPTION

tftpd is a server that supports the TCP/IP Trivial File Transfer Protocol (TFTP). This server is normally started by **inetd**(8C) and operates at the port indicated in the **tftp** Internet service description in the **/etc/inetd.conf** file; see **inetd.conf**(5) for details. The default **/etc/inetd.conf** file starts this server in secure mode, that is, with the **–s** option enabled. To run unsecure **tftpd**, modify this file and remove the **–s** option.

Before responding to a request, the server attempts to change its current directory to *homedir*; the default value is **/tftpboot.**

Sun386i DESCRIPTION

The **tftpd** daemon acts as described above, except that it will perform certain filename mapping operations unless instructed otherwise by the **–p** command line argument or when operating in a secure environment. This mapping affects only TFTP boot requests and will not affect requests for existing files.

The semantics of the changes are as follows. Only filenames of the format *ip-address* or *ip-address.arch,* where *ip-address* is the IP address in hex, and *arch* is the hosts's architecture (as returned by the **arch**(1) command), that do not correspond to files in **/tftpboot**, are mapped. If the address is known through a Network Information Service (NIS) lookup, any file of the form **/tftpboot/ip-address*** (with or without a suffix) is returned. If there are multiple such files, any one may be returned. If the *ip-address* is unknown (that is if the **ipalloc (8C)** service says the name service does not know the address), the filename is mapped as follows: Names without the *arch* suffix are mapped into the name **pnp.SUN3**, and names with the suffix are mapped into **pnp.** *arch*. That file is returned if it exists.

OPTIONS

–s Secure. When specified, the directory change must succeed; and the daemon also changes its root directory to *homedir*. This option is set in the default **/etc/inetd.conf** file.

The use of **tftp** does not require an account or password on the remote system. Due to the lack of authentication information, **tftpd** will allow only publicly readable files to be accessed. Files may be written only if they already exist and are publicly writable. Note: this extends the concept of "public" to include all users on all hosts that can be reached through the network; this may not be appropriate on all systems, and its implications should be considered before enabling this service.

tftpd runs with the user ID (UID) and group ID (GID) set to **–2**, under the assumption that no files exist with that owner or group. However, nothing checks this assumption or enforces this restriction.

Sun386i OPTIONS

–p Disable pnp entirely. Do not map filenames.

Sun386i FILES

/tftpboot/* filenames are IP addresses

SEE ALSO

tftp(1C) **inetd**(8C), **ipallocd**(8C), **netconfig**(8C),

Sollins, K.R., *The TFTP Protocol (Revision 2)*, RFC 783, Network Information Center, SRI International, Menlo Park, Calif., June 1981.

NOTES

The Network Information Service (NIS) was formerly known as Sun Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

Sun386i WARNINGS

A request for an *ip-address* from a Sun-4 can be satisfied by a file named *ip-address*.**386** for compatibility with some early Sun-4 PROM monitors.

NAME

tic – terminfo compiler

SYNOPSIS

tic [**–v**[*n*]] [**–c**] *filename*

AVAILABILITY

This command is available with the *System V* software installation option. Refer to *Installing the SunOS System Software* for information on how to install optional software.

DESCRIPTION

tic compiles a **terminfo**(5V) source file into the compiled format. The results are placed in the directory **/usr/share/lib/terminfo**. The compiled format is used by the **curses**(3V) library.

Each entry in the file describes the capabilities of a particular terminal. When a **use=***entry* field is given in a terminal entry, **tic** reads in the binary (compiled) description of the indicated *entry* from **/usr/share/lib/terminfo** to duplicate the contents of that entry within the one being compiled. However, if an *entry* by that name is specified in *filename*, the entry in that source file is used first. Also, if a capability is defined in both entries, the definition in the current entry's source file is used.

If the environment variable **TERMINFO** is set, that directory is searched and written to instead of **/usr/share/lib/terminfo**.

OPTIONS

–v[*n*]

Verbose. Display trace information on the standard error. The optional integer argument is a number from 1 to 10, inclusive, indicating the desired level of detail. If *n* is omitted, the default is 1.

–c Only check *filename* for errors. Errors in **use=** links are not detected.

FILES

/usr/share/lib/terminfo/?/* compiled terminal description data base

SEE ALSO

fork(2V), **curses**(3V), **curses**(3V), **malloc**(3V), **term**(5), **terminfo**(5V)

BUGS

Total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 1024 bytes.

When the **–c** option is used, duplicate terminal names will not be diagnosed; however, when **–c** is not used, they will be.

For backward compatibility, cancelled capabilities will not be marked as such within the terminfo binary unless the entry name has a '**+**' within it. Such terminal names are only used for inclusion with a **use=** field, and typically aren't used for actual terminal names.

DIAGNOSTICS

Most diagnostic messages produced by **tic** are preceded with the approximate line number and the name of the entry being processed.

mkdir *name* **returned bad status**

The named directory could not be created.

File does not start with terminal names in column one

The first thing seen in the file, after comments, must be the list of terminal names.

Token after a seek(2) not NAMES

Somehow the file being compiled changed during the compilation.

NAME

ttysoftcar – enable/disable carrier detect

SYNOPSIS

ttysoftcar [**–y** | **–n**] *tty* . . .

ttysoftcar –a

DESCRIPTION

For each *tty* specified **ttysoftcar** changes the carrier detect flag using the **TIOCSSOFTCAR ioctl( )** request (see **tty**(4)). If the **–a** option is specified, **ttysoftcar** sets all **tty**'s in the **/etc/ttytab** file to the carrier detection mode specified by their status field. If this field is set to **local**, software carrier detection is turned on. If this field is set to anything other than **local**, as is usually the case for modems, software carrier detection is turned off. **ttysoftcar** ignores devices in the **/etc/ttytab** file which do not exist.

If no options are specified, **ttysoftcar** returns the current status for *tty*. This status is reported as **y** or **n**.

OPTIONS

–a Reset **tty**s to appropriate values based on the status field of the **/etc/ttytab** file.

–y Turn on software carrier detect.

–n Turn off software carrier detect. Use hardware carrier detect.

SEE ALSO

termio(4), **zs**(4S), **ttytab**(5)

NAME

tunefs – tune up an existing (dismounted) file system

SYNOPSIS

/usr/etc/tunefs [**–a** *maxcontig*] [**–d** *rotdelay*] [**–e** *maxbpg*] [**–m** *minfree*] *special* | *filesystem*

DESCRIPTION

tunefs is designed to change the dynamic parameters of a file system which affect the layout policies. The parameters which are to be changed are indicated by the OPTIONS given below:

OPTIONS

–a *maxcontig*

This specifies the maximum number of blocks, belonging to the same file, that will be allocated contiguously before inserting a rotational delay (see **–d** below).

Formerly, this value was always 1 because neither the device drivers nor the Unix file system (UFS) could handle multiple block requests. UFS has been modified to cluster together several (up to *maxcontig*) blocks when doing sequential reads and writes, resulting in higher I/O rates. This is especially beneficial on drives or controllers with track buffers; in some cases it can double or triple the I/O rates.

Newfs sets the value to 7 for many drives, but the best value varies from drive to drive because of the track buffer size as well as the drive geometry. People interested in the getting the utmost performace from their drives can try something like this (note that only the read case is of interest - the write case goes into the cache):

```
for i in 1 2 3 4 5 6 7
do      umount /mnt
        tunefs -a $i /mnt
        mount /mnt
        dd if=/dev/zero of=/mnt/XXX bs=8k count=1000
        umount /mnt; mount /mnt  # to clear the cache
        time dd of=/dev/null if=/mnt/XXX bs=8k
done
```

This parameter is limited in the following way:

blocksize * *maxcontig* must be <= *maxphys*

maxphys is a read-only kernel variable that specifies the maximum block transfer size (in bytes) that the I/O subsystem is capable of satisfying. On most machines, *maxphys* is 56 Kbytes; on the **sun4c** architecture it is currently 124 Kbytes which allows *maxcontig* to vary from 1 to 15. This value is subject to change. (The limit is enforced by *mount*(2), not by *newfs* or *mkfs*.)

–d *rotdelay*

This specifies the expected time (in milliseconds) to service a transfer completion interrupt and initiate a new transfer on the same disk. It is used to decide how much rotational spacing to place between successive blocks in a file. For drives with track buffers a *rotdelay* of 0 is usually the best choice.

–e *maxbpg*

This indicates the maximum number of blocks any single file can allocate out of a cylinder group before it is forced to begin allocating blocks from another cylinder group. Typically this value is set to about one quarter of the total blocks in a cylinder group. The intent is to prevent any single file from using up all the blocks in a single cylinder group, thus degrading access times for all files subsequently allocated in that cylinder group. The effect of this limit is to cause big files to do

long seeks more frequently than if they were allowed to allocate all the blocks in a cylinder group before seeking elsewhere. For file systems with exclusively large files, this parameter should be set higher.

-m *minfree*

This value specifies the percentage of space held back from normal users; the minimum free space threshold. The default value used is 10%. This value can be set to zero, however up to a factor of three in throughput will be lost over the performance obtained at a 10% threshold. Note: if the value is raised above the current usage level, users will be unable to allocate files until enough files have been deleted to get under the higher threshold.

SEE ALSO

fs(5), **dumpfs**(8), **mkfs**(8), **newfs**(8)

System and Network Administration

BUGS

This program should work on mounted and active file systems. Because the super-block is not kept in the buffer cache, the program will only take effect if it is run on dismounted file systems; if run on the root file system, the system must be rebooted.

You can tune a file system, but you can't tune a fish.

NAME

unixname2bootname – convert SunOS device name to boot device name

SYNOPSIS

/usr/kvm/unixname2bootname *name*

AVAILABILITY

This program is available for desktop SPARCsystems and for the SPARCsystem 600MP series only.

DESCRIPTION

The **unixname2bootname** command converts a SunOS device name to a boot PROM device name. This is useful for OPENPROM systems that have more complicated (though absolutely general) device names.

EXAMPLES

On a SPARCstation 1 or a SPARCstation 1+ system, **unixname2bootname** would act like this:

```
example# /usr/kvm/unixname2bootname sd0b
sd(0,0,1)
example#
```

On a SPARCstation 2 system, **unixname2bootname** would act like this:

```
example# /usr/kvm/unixname2bootname sd0b
/sbus@1,f8000000/esp@0,800000/sd@3,0:b
example#
```

The output of **unixname2bootname** can be combined using various shell quote mechanisms to form arguments for **reboot**(8) and other utilities.

```
example# reboot "`/usr/kvm/unixname2bootname sd0b` –sw"
```

FILES

/dev/openprom

SEE ALSO

openprom(4S), **boot**(8)

BUGS

unixname2bootname does not deal with non-disk devices well.

Notes

Notes