PERQ Systems Corporation

ACCENT USER'S MANUAL

JUNE 1984

This manual is for use with Accent Release S5.

2600 Liberty Avenue
P. O. Box 2600
Pittsburgh, PA 15230
(412) 355-0900

Accent is a trademark of Carnegie-Mellon University.

Accent and many of its subsystems and support programs were originally developed by the CMU Computer Science Department as part of its Spice Project.

The information in this document is subject to change without notice and should not be construed as a commitment by PERQ Systems Corporation. The company assumes no responsibility for any errors that may appear in this document.

PERQ Systems Corporation will make every effort to keep customers apprised of all documentation changes as quickly as possible. The Reader's Comments card is distributed with this document to request users' critical evaluation to assist us in preparing future documentation.

ACCENT USER'S MANUAL

PREFACE

This manual contains documentation on using Accent, the operating system for the PERQ workstation. Accent was developed jointly by PERQ Systems Corporation and the Spice Project in the Computer Science Department at Carnegie-Mellon University. "Spice" is an acronym for Scientific Personal Integrated Computing Environment.

This manual contains introductory material on the PERQ workstation and the Accent operating system and instructions for using the window manager, the facilities, and the editor. If you are not going to program on the PERQ workstation, this is the only manual you will need. If you are going to do programming, you should read this manual and the Accent Programming Manual, which contains general information needed by all programmers. Then refer to one of the following manuals for information pertinent to your particular programming task:

Accent Microprogramming Manual
Accent Languages Manual
Accent Lisp Manual

Other manuals for Accent are the Accent Qnix Manual (forthcoming in a future release) and the Accent System Administration Manual.

Throughout the manuals the term "PERQ" refers to all models of the PERQ workstation unless stated otherwise. When a distinction is made between the PERQ workstation and the PERQ2 workstation, the term "PERQ2" refers to both Model LN-3000 and LN-3500.

The following symbols have been used throughout the Accent manuals:

< >	Material that is to be replaced by symbols or text as explained in the accompanying text. Do not type the angle brackets. Example: <filename> indicates that you should type the name of your file.
[ ]	Optional feature. Do not type the square brackets.
{ }	0 to n repetitions of an optional item. Do not type the braces.
CAPITALS	Literal, to be reproduced exactly as shown (although it may be reproduced in upper-case or lower-case). Example: <filename.CMD> indicates that the filename must contain the extension .cmd.
|	"Or"--choice between the items shown on either side of the symbol.
CTRL	Control key
ESC, INS	Escape key (labeled as ACC ESC or INS on various models)
DEL	Delete key (labeled as REJ DEL on some models)
HELP	Help key
LF	Linefeed Key
RETURN	Carriage return
<u>underlining</u>	input to be typed by the user

ACCENT USER'S MANUAL

TABLE OF CONTENTS

INTRODUCTION TO THE PERQ WORKSTATION

AND THE ACCENT OPERATING SYSTEM

June 8, 1984

2600 Liberty Avenue
P. O. Box 2600
Pittsburgh, PA 15230
(412) 355-0900

Accent is a trademark of Carnegie-Mellon University.

Accent and many of its subsystems and support programs were originally developed by the CMU Computer Science Department as part of its Spice Project.

The information in this document is subject to change without notice and should not be construed as a commitment by PERQ Systems Corporation. The company assumes no responsibility for any errors that may appear in this document.

PERQ Systems Corporation will make every effort to keep customers apprised of all documentation changes as quickly as possible. The Reader's Comments card is distributed with this document to request users' critical evaluation to assist us in preparing future documentation.

PERQ, PERQ2, LINQ, and Qnix are trademarks of PERQ Systems Corporation.

TABLE OF CONTENTS

INTRODUCTION TO THE PERQ WORKSTATION

AND THE ACCENT OPERATING SYSTEM

This document is an introduction to the PERQ and PERQ2 workstations and to the Accent operating system. Section 1 describes the hardware, and Section 2 describes the operating system.

1. PERQ Workstation Hardware

The PERQ and PERQ2 workstations, manufactured by PERQ Systems Corporation, provide an integrated computing system for a single user. Two or more PERQ workstations may be connected together via an optional 10MB Ethernet. The PERQ workstations provide a large, 32-bit, paged virtual address space for each process. Virtual addresses are mapped into a 20-bit physical address.

This section briefly describes the PERQ and PERQ2 hardware. For more details on the PERQ2 workstation, see the PERQ2 User's Guide.

The standard hardware system consists of a high resolution display, a keyboard, and a pointing device (tablet and mouse), all connected to the processor box. The processor box houses the processor, a fixed disk, and an optional floppy disk. Figure 1 shows a typical PERQ2 workstation.

1.1 Display

The PERQ and PERQ2 workstation displays are free-standing screens, connected to the processor box by cables about ten feet in length.

You can adjust the angle of the display to suit your needs. To tilt the display on the PERQ workstation, turn the thumbscrew at the rear of the display. On the PERQ2 workstation the display pivots on its base; simply tilt the display to the desired angle. Note that on some models the boot button and the diagnostic display (DDS), both discussed later, reside on the base of the display; to re-boot or read the DDS, you may have to tilt the display.

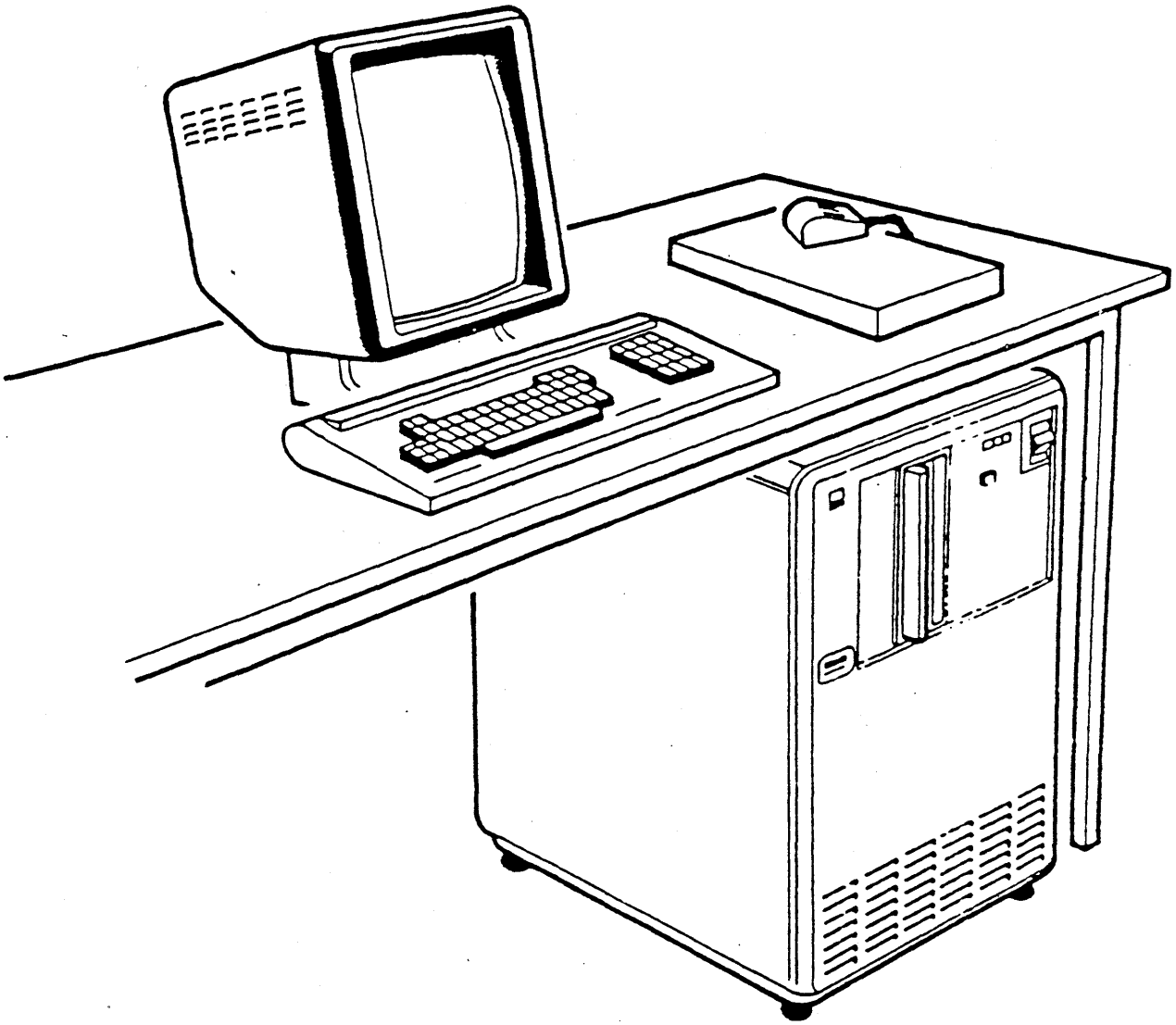


Figure 1

Typical PERQ2 Workstation System

The brilliance of the display can be adjusted to suit ambient lighting. On both the PERQ and PERQ2 workstations the control knob is on the back of the display.

The image on the display is known as a raster and consists of 1024 horizontal lines. Each line contains individual picture elements, referred to as pixels.

The standard display, referred to as a portrait display, has its long side vertical and resembles a sheet of 8-1/2" x 11" paper. Each of the 1024 horizontal lines contain 768 pixels.

An optional display, referred to as a landscape display, has its long side horizontal and resembles a sheet of 14" x 11" paper. Each of the 1024 horizontal lines contain 1280 pixels.

Each pixel can be either black or white. A user program can display various shades of gray by varying the proportions of black and white pixels in different areas.

One bit in the machine's main memory is used for each pixel. The area of memory containing these bits is known as the bit map. Since the bit map is simply a part of main memory, the whole of the bit map is directly accessible to user programs. For a portrait display, the bit map is 96K bytes. For a landscape display, the bit map is 130K bytes.

The display controller (an integral part of the memory board) transfers the bit map to the display 60 times per second, referred to as a 60Hz refresh rate. The refresh system is non-interlaced and thus provides a genuine 60 complete frames per second. This provides a flicker-free display.

The processor contains an instruction called RasterOp, implemented by special hardware, that enables programs to quickly and easily alter the bit map. In fact, RasterOp makes real animation possible.

The Q-code (PERQ workstation's machine code) RASTER-OP and the corresponding Pascal statement RASTEROP merge a section of the bit map, which corresponds to a rectangular section of the display (the destination), with an identically sized bit pattern (the source) which is often held in another part of main memory.

The first parameter to RASTEROP is the RASTEROP function. There are eight RASTEROP functions, of which RRpl (function 0) is probably the most commonly used. Function RRpl causes the source to entirely replace the destination. Most of the other functions perform binary operations between source and destination bits.

Refer to "PERQ/Accent Pascal Extensions" in the Accent Languages Manual for a complete description of RASTEROP and the RASTEROP functions.

For a portrait display, the first 48 words constitute the top line of pixels. For a landscape display, the first 80 words constitute the top line of pixels. The most significant bit (bit 15) of each word is the first pixel. The following diagram illustrates the pixels in the top left corner of a portrait display:

Line	Words(bits)
0	0(15,14, ... 1,0) 1(15,14 ... 1,0) 2 ... 47
1	48 ... 95
2	96 ...

The displays can also provide different fonts.

1.2 Keyboard

The PERQ and PERQ2 workstations use a detached, serial interface terminal keyboard. The PERQ workstation keyboard connects to the processor box by a cable about eight feet long. The PERQ2 workstation keyboard connects to the display by a coiled cable.

The keys are all solid state, including the LOCK key which contains a red light that is lit when the LOCK is on. On the PERQ workstation keyboard, the LOCK key functions like a shift lock--the key shifts alphabetic characters and numerals to their upper case equivalents. On the PERQ2 workstation keyboard, the LOCK key functions like a caps lock--the key shifts only the alphabetic characters to upper case.

All keys provide auto-repeat after the key has been held down for about half a second. On the PERQ workstation, the repeat continues at the rate of about 10 per second. On the PERQ2 workstation, the repeat continues at the rate of about 16 per second. N-key rollover ensures that a keystroke is not lost even if you have not released the previous key.

The keyboard generates ASCII (American Standards Code for Information Interchange) characters.

The keyboard employs standard typewriter layout (QWERTY) with the addition of special keys. The keyboard of the PERQ2 workstation has a numeric pad to the right of the standard keys; this pad is useful for entering large quantities of numbers. Figure 2 depicts

the PERQ workstation keyboard and Figure 3 depicts the PERQ2 workstation keyboard.

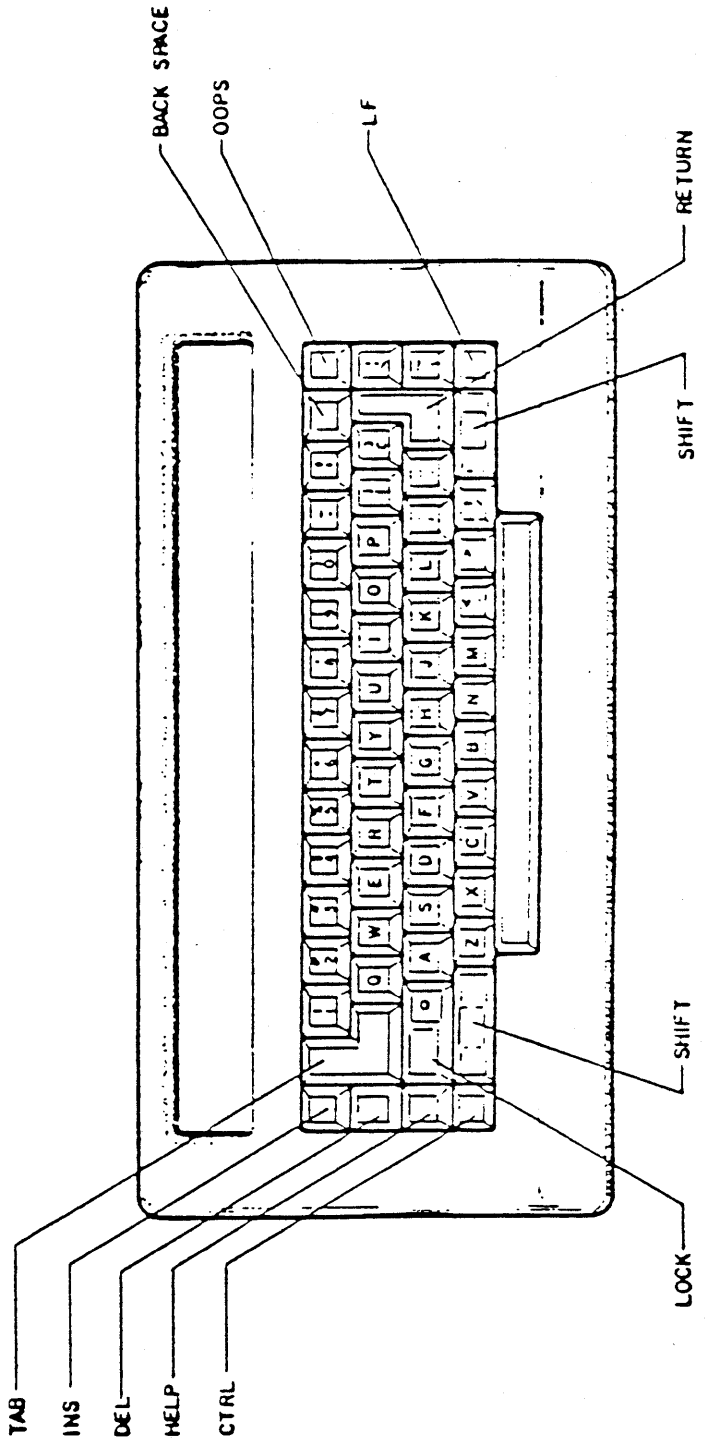


Figure 2

Keyboard of the PERQ Workstation

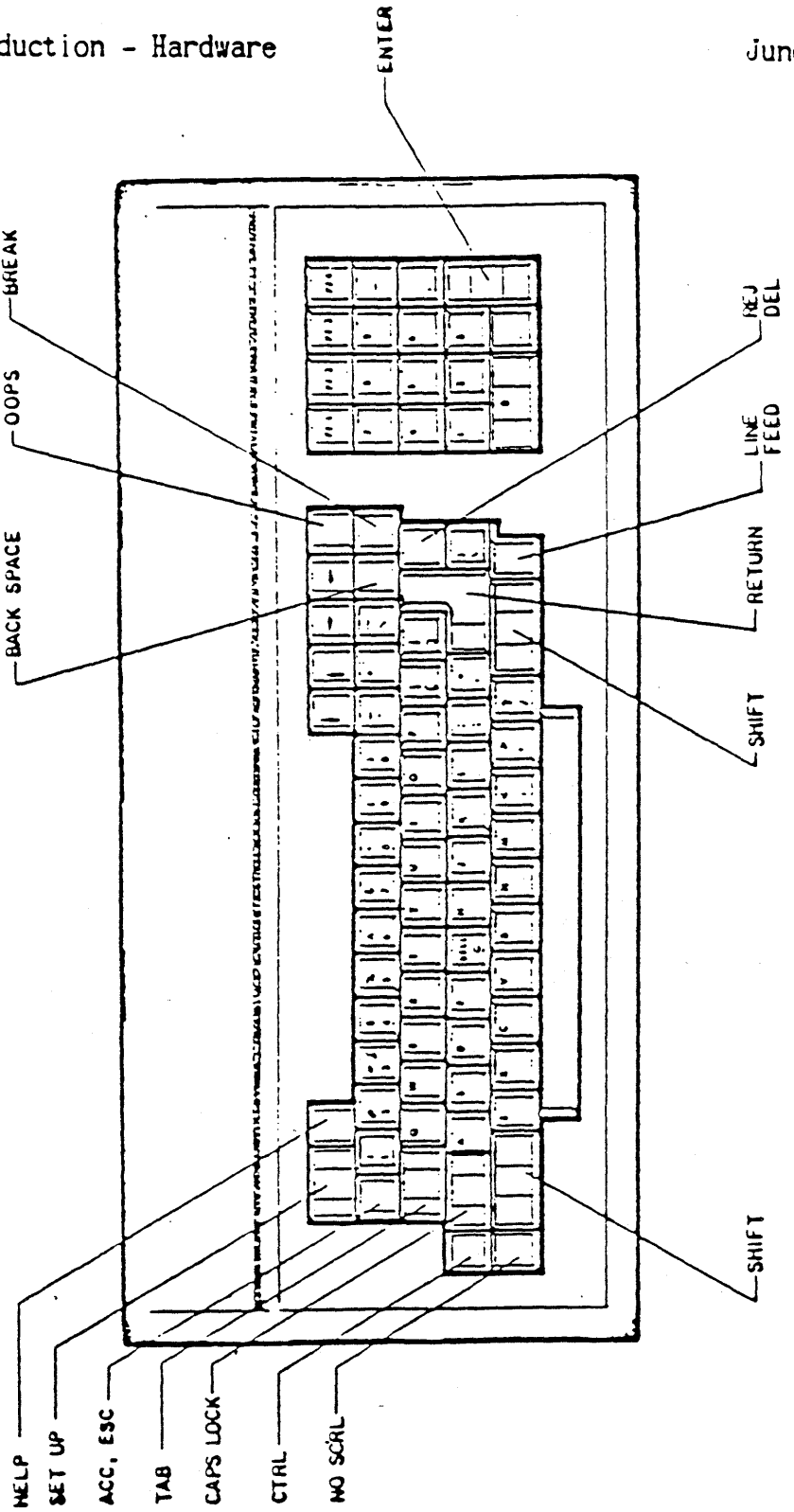


Figure 3

Keyboard of the PERQ2 Workstation

1.3 Tablet and Mouse

The PERQ and PERQ2 workstations use an electromagnetic tablet, referred to as the EM tablet, and three-button mouse as a pointing device. Figure 4 depicts the standard three-button mice. Optionally, you can choose a Summagraphics bitpad and four-button mouse. Figure 5 depicts the four-button mouse supplied with the optional bitpad. The bidpad has a reset button on the right edge of the tablet; the tablet has no reset button.

Throughout this document the term "tablet" refers to both the EM tablet and the Summagraphics bitpad.

The mouse position is read when the mouse is in contact with the tablet. (You have a choice of two modes--relative or absolute, as explained in the document "Basic Operations" in this manual.) The system displays a cursor on the screen to depict mouse position. The standard shape of this cursor is a small arrow pointing up and left, but you can redefine its shape (using the CursDesign program, available from the User Library). Note that this cursor is not placed in the bit map, but is superimposed on the display by the display controller at each refresh.

Pressing and releasing a button and moving the mouse with a button down are interpreted differently in different programs. See the documents on the window manager and the editor, both contained in this manual, for details on how to use the mouse with those programs.

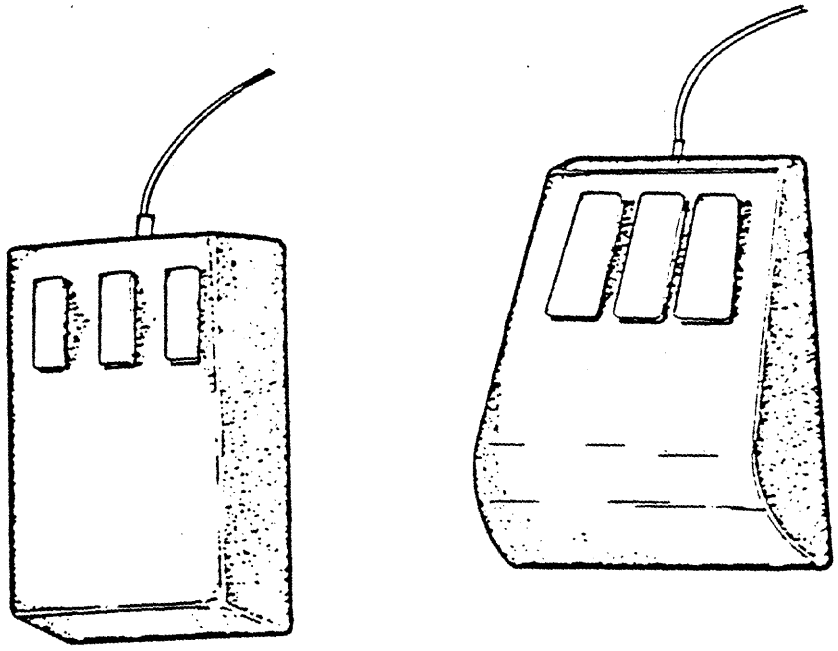


Figure 4

Three-Button Mice

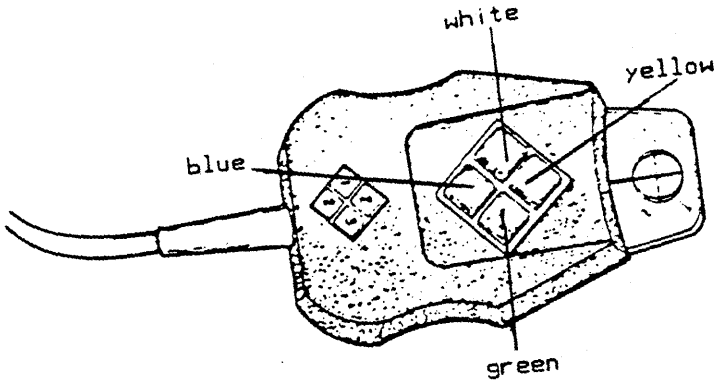


Figure 5

Four-Button Mouse

1.4 Processor Box

This section describes the components contained in the processor box. Figure 6 presents a conceptual diagram of the components.

1.4.1 Central processor unit

The processor is controlled by the Accent Operating System, which is written in Pascal. You can write programs to run under Accent in Pascal, FORTRAN 77, C, or Lisp.

Like most high-level languages, programs in all these languages must be translated into coded instructions (often called machine code) before they can be executed. The PERQ workstation's machine code is Q-Code, an adaptation of the University of California at San Diego (UCSD) Pascal p-System. The implementation of the Q-Code is in microcode.

The PERQ and PERQ2 workstation processors are actually microprogrammed to execute Q-code, whereas a conventional processor contains hardware that directly executes the machine code. On a microprogrammed processor, that hardware is replaced by a microcode engine controlled by a microprogram. The microprogram is written in microcode which the microcode engine executes directly. Figure 7 illustrates this concept. The net effect is that the whole appears to be a processor capable of executing the machine code (Q-code), even though the microcode is different from the machine code.

Accent supports the ability to change the instruction set portion of the microprogram for different languages on a per-process basis. For example, Lisp has its own instruction set which is dynamically loaded when a Lisp process starts.

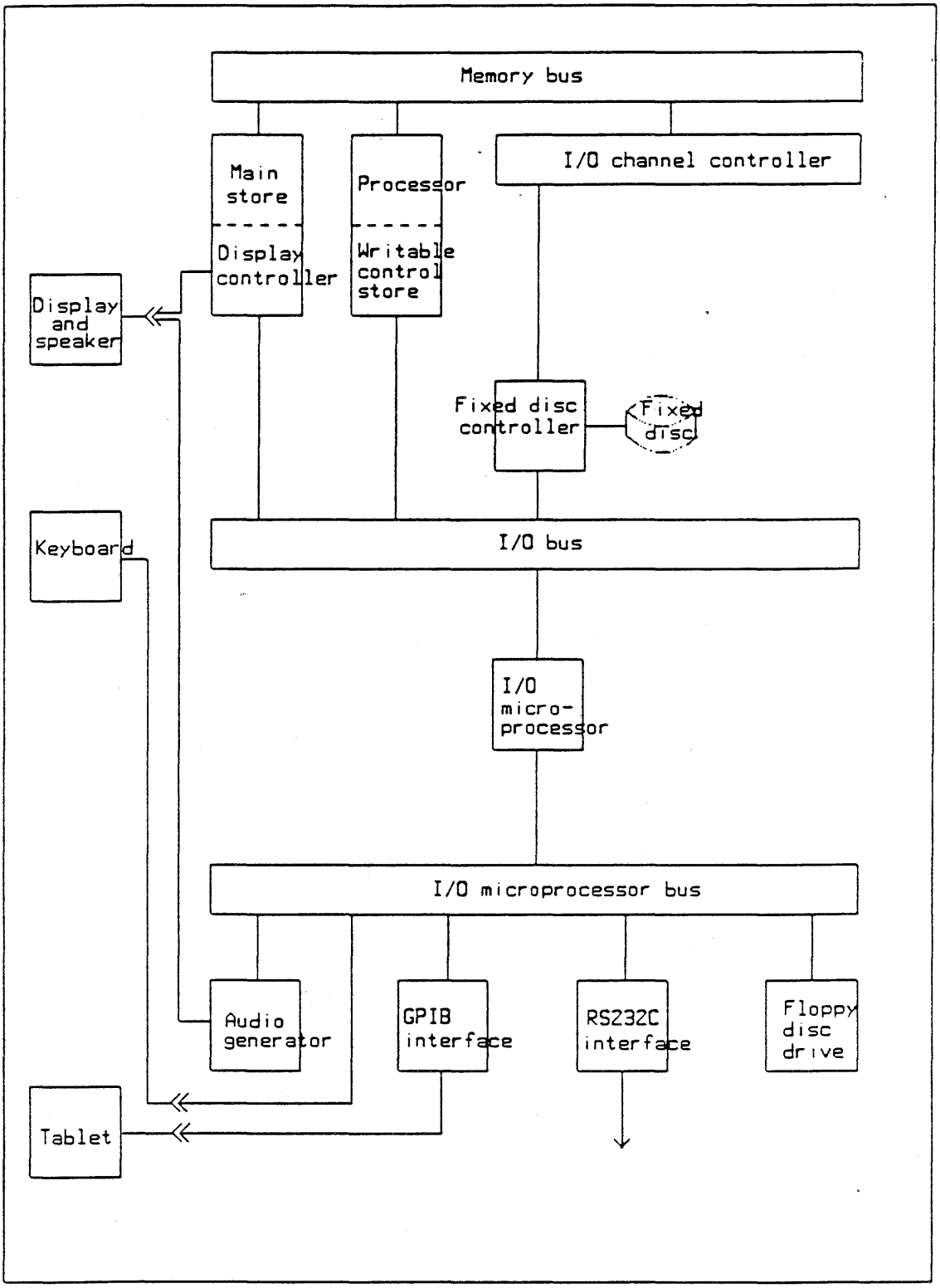


Figure 6

Components of Processor Box

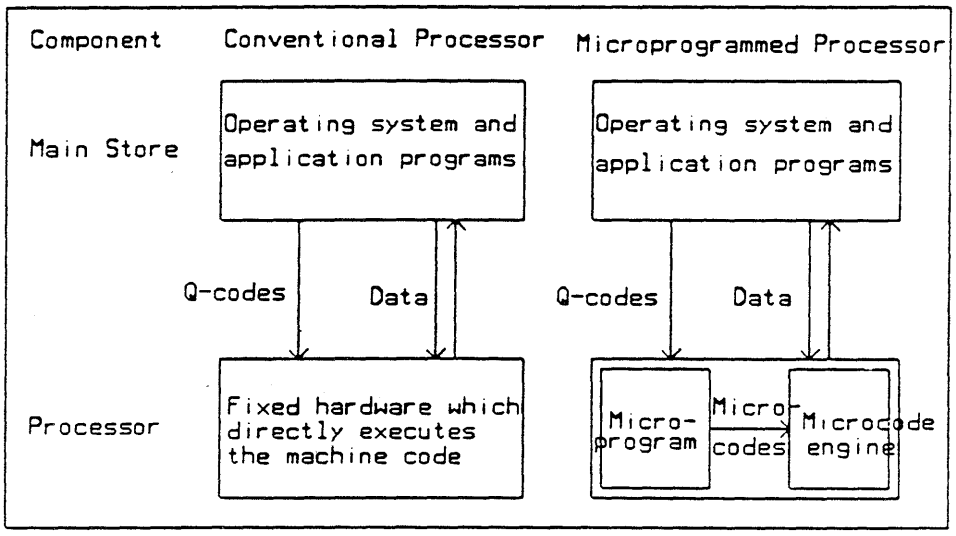


Figure 7

Microcode Executed by Microcode Engine

A microprogrammable processor provides several advantages. If Q-code is enhanced, a new microprogram enables an existing PERQ or PERQ2 workstation to execute the enhanced Q-code without hardware alterations. If a completely different machine code proves desirable for some application, a new microprogram could implement the new machine code. Also, a microprogrammed machine permits the transfer of some of the hardware controller's complexity into the microprogram and thus make the controller simpler, less expensive, and more reliable.

The microprogram is loaded during boot into Writable Control Store

(WCS) contained on the processor board. The PERQ workstation microstore is 16K words. Each WCS word is 48 bits long.

The WCS cycle time (that is, the time to execute one micro instruction) is 170 nanoseconds.

Data comes into and out of the processor on 16 bit wide paths. Internally, the processor uses paths that are 20 bits wide so that physical memory addresses can be calculated using a single precison arithmatic operation.

The PERQ and PERQ2 workstation processors have:

256 REGISTERS - these are 20 bit wide general purpose registers, which can hold main memory words or addresses.

OP FILE - this is an 8 byte buffer of Q-codes and parameters. It is loaded by copying the appropriate part of a code segment (Q-code program) from main memory. Thus, the OP FILE gives the processor quicker byte by byte access to the current part of the program than if it had to access each byte separately from main memory.

EXPRESSION STACK - this is a push-down stack containing 16 registers, each 20 bits, and is used for evaluating expressions by holding intermediate values.

1.4.2 Main memory

The standard PERQ and PERQ2 workstation main memory supported by Accent consists of 1 megabyte of Randomly Accessible Memory (RAM). Optionally, Accent will support 2.0 megabytes of RAM on either machine.

The PERQ workstation provides a 64-bit wide memory processor/memory bus. All stores and fetches are done in units of four 16-bit words known as quad words. The memory transfer time for a single quad word is 680 nanoseconds, providing an effective throughput of 11 megabytes per second.

1.4.3 Input and output control

The PERQ and PERQ2 workstations have two systems for handling data: the Input Output (Direct Memory Access or DMA) Channel Controller for fast data streams and the I/O microprocessor for slower data streams.

The DMA channel controller has direct access to the main memory. This controller is attached to the hard disk, the network, and two DMA channels for optional devices.

The I/O microprocessor, a Z80, controls the slower data streams, which are the keyboard, floppy disk drive, GPIB interface, RS232 interface (one on a PERQ workstation, two on a PERQ2 workstation), pointing device, and speech output.

1.4.3.1 Hard disk

Each PERQ and PERQ2 workstation is equipped with a high-speed, non-removable hard disk. On the PERQ workstation, the standard disk is a Shugart 24 megabyte disk. On the PERQ2 workstation, the standard disk is a Micropolis 8" 35 megabyte disk or a 5.25" Winchester-type disk with a formatted capacity of up to 34 megabytes. The disks are mounted in the processor box. On the PERQ workstation, the disks take approximately two minutes to spin-up and stabilize following power on; on the PERQ2 workstation this takes about 30 seconds.

Optionally, you may equip your PERQ2 workstation with up to two internal disks (two 40 megabyte Micropolis disks or two 140 megabyte Maxtor disks) or one internal disk and three external disks of either type.

1.4.3.2 Floppy disk

A floppy disk drive may be purchased as optional equipment for a PERQ and PERQ2 workstation. The drive uses an 8", single- or double-sided, single or double density diskette. The capacity of the diskette is 1 megabyte (double-sided, double-density).

Data on the diskettes is stored in RT-11 format, which is compatible with PDP-11 and VAX floppy disks. (RT-11, PDP-11, and VAX are registered trademarks of Digital Equipment Corporation.)

The primary purposes of the floppy disk systems are the receipt of

software, the exchange of programs and data with other systems, and the backup of files for archival and safety reasons.

On the PERQ workstation, the floppy drive is mounted horizontally at the top of the processor box. Insert the labeled side of the floppy at the front of the drive with the label facing up.

On the PERQ2 workstation, the floppy drive is mounted vertically at the top left of the processor box. Insert the labeled side of the floppy at the front of the drive with the label facing right.

Keep the door of the floppy disk drive closed, except when loading or removing a disk, to prevent dirt from entering the drive.

1.4.3.3 GPIB

Each machine is equipped with an IEEE-488 interface, known as the General Purpose Instrumentation Bus or GPIB. The GPIB is an 8-bit parallel, medium speed, industry standard bus. You can connect peripherals (for example, tape drives, printers, and plotters) to a PERQ or PERQ2 workstation via the GPIB. The GPIB can also acquire data from and control many types of laboratory instruments.

The GPIB interface can perform Talker, Listener, and Controller functions.

1.4.3.4 RS232

The PERQ and PERQ2 workstations are equipped with a fully programmable, serial line interface conforming to RS232 standards. Baud rate, character size, parity options, and modem control are all programmable.

The PERQ workstation uses a single RS232 port capable of transfer rates up to 9600 baud, while the PERQ2 workstation includes two RS232 ports (port A and port B) capable of transfer rates up to 19.2K baud.

1.4.3.5 Ethernet

You can optionally equip a PERQ or PERQ2 workstation with a Local Area Network interface conforming to Ethernet standards. This optional interface also conforms to ECMA and IEEE-802 standards.

The Ethernet interface provides high speed interconnection of computers at 10 megabits per second along a coaxial cable. Up to 2.5 kilometers of station separation is possible and up to 1024 stations can share the network.

On the PERQ workstation a separate board, which resides on the option IO board, is required for Ethernet support. On the PERQ2 workstation Ethernet support is an option on the system IO board. A transceiver cable connects the PERQ workstation interface to a transceiver (a small box usually located in the ceiling), which in turn connects via a tap to the coaxial cable.

The network uses a baseband transmission with Carrier Sense, Multiple Access/with Collision Detect (CSMA/CD).

2. The Accent Operating System

Accent is a sophisticated operating system intended for use on the PERQ workstation product line. It was developed and implemented jointly by the Spice Project in the Computer Science Department at Carnegie-Mellon University and PERQ Systems Corporation. The system is a multi-tasked, message-based system, providing for each process a paged virtual address space of 2^(32) 16-bit words. The system naturally supports the server-client paradigm of software development. Most system functions are invoked by sending messages to the relevant server process. This design approach extends gracefully to provide access to servers that are not on the local machine but that reside on other machines connected via the network.

This section provides an overview of the system. For more information see the document "Theory of Operations" in the Accent Programming Manual.

2.1 Kernel

The kernel provides low-level support to the various operating system functions. The main function provided is interprocess communication (IPC). All other operating system functions are provided by sending mssages, using the IPC facility, to server processes.

Virtual memory handling, which is normally provided by the kernel of an operating system, is provided by a separate process known as the pager. This process is reponsible for handling references to virtual addresses which are not currently resident in physical memory.

Process scheduling is provided by a combination of the kernel and the pager processes. Accent provides for priority scheduling of processes with pre-emption. An aging algorithm is used to lower the priority of a compute-bound process, thus guaranteeing quick response for highly interactive processes.

2.2 File System

The Accent file system supports a single level store view of the memory hierarchy. Access to files is through the virtual memory system. When a process wishes to read a file, that file is mapped into the process's address space as a set of contiguous virtual addresses. Note that this is only a mapping operation; no data is actually read from the disk. When a portion of the file is accessed that is not resident in memory, that portion of the file is then read from the disk using the same facilities as any other memory page fault.

The directory structure provided by the file system is a network-wide hierarchical tree. One of the features of the Accent file system is that access to data on remote disks is transparent to the user. The file system provides for protection using access control lists which are associated with each directory in the filing system.

The file system supports all of the disks listed in Section 1.4.3.1.

2.3 Inter-Machine Communication

In Accent all services are provided to clients by server processes. Requests are made of the servers by sending the request in a message to that server. Two processes, the network server and the message server, extend this method of interaction gracefully to servers that are on other machines. When a request is made for a service that is not on the local machine, that IPC message is sent to the message server on the remote machine via the local and remote network server. The remote message server then forwards the IPC message to the server that is going to perform the function.

In this interaction neither the client nor the server need know that the partner in the transaction is not on the local machine.

2.4 Process Manager

Accent provides a number of facilities that can be used to manipulate processes on the machine. Facilities are provided that will allow users to start and stop processes, debug processes, and terminate processes.

Processes can be stored into a hierarchy and then manipulated as if they were a single process.

2.5 Window Manager

Accent provides a high-level window manager and graphics support system. The window manager supports the covered window paradigm and provides the ability to manipulate windows in a large number of ways. Windows can be grown, shrunk, moved partially or completely off screen, and manipulated in other ways.

The window manager currently supports two basic types of windows. First, a process can specify the window to be a plain text window, in which case the window manager provides for automatic refresh and display of the window when that window is uncovered. It also provides for a history mechanism that can be used to view text that has been scrolled out of the window.

Second, the window manager supports graphic windows in which arbitrary graphics can be done inside of the window boundary. When a window of this type is uncovered, a notification is given to the process that owns the window, telling the process that the window state has changed and that it must redisplay any relevant information.

The window manager provides the ability to have numerous fonts which can be changed dynamically under program control.

In addition, it provides via separate processes (tracker and typescript) keyboard control and cursor tracking on a per window basis.

2.6 Shell

The Accent Shell provides a consistent user interface to the facilities of the Accent operating system. Its main function is to control the creation of processes that are going to execute programs that interact with the user. In addition, the Shell provides a small number of simple commands of its own. The Shell provides processes with an environment (search lists, default paths, logical names, etc.) that the processes can use during their execution.

The user can have more than one Shell active at any given time. Each of these Shells will be associated with a separate window.

2.7 Qnix, a UNIX Software Emulator

Accent provides the ability to run an optional server that emulates a UNIX System V environment. This emulation is at the system call level. Any program that uses the UNIX software system call interface and does not rely on the form of system data structures can be easily ported to execute under Accent using this server.

BASIC OPERATIONS

June 8, 1984

2600 Liberty Avenue
P. O. Box 2600
Pittsburgh, PA 15230
(412) 355-0900

Accent is a trademark of Carnegie-Mellon University.

Accent and many of its subsystems and support programs were originally developed by the CMU Computer Science Department as part of its Spice Project.

The information in this document is subject to change without notice and should not be construed as a commitment by PERQ Systems Corporation. The company assumes no responsibility for any errors that may appear in this document.

PERQ Systems Corporation will make every effort to keep customers apprised of all documentation changes as quickly as possible. The Reader's Comments card is distributed with this document to request users' critical evaluation to assist us in preparing future documentation.

TABLE OF CONTENTS

BASIC OPERATIONS

This document is intended to instruct a user of the PERQ workstation in the most commonly used functions. It does not include technical specifications or complete details on all of the functions covered. Other documents that contain more information are referenced throughout.

1. Booting the System

Upon delivery of your PERQ2 workstation you should read the documents Installing Your PERQ2 and PERQ2 User Guide. If you need help in getting your workstation into operating condition, see your system administrator. When new versions of the operating systems are released, your system administrator will load them into your workstation.

After your workstation has been installed, follow the instructions below each time you turn on your workstation.

1) Turn on the power.

   The PERQ workstation power switch is located on the front panel of the processor box in the groove below the floppy disk drive. The switch is right of center and is labeled OFF/ON. The PERQ2 workstation power switch is located on the front panel of the processor box adjacent to the floppy disk drive. The switch is labeled ON/OFF or 1/0 (1=on).

   When you power on either workstation, fans start. You will be able to hear the fans on a PERQ workstation but not on a PERQ2 workstation. If the workstation doesn't start, check to see that the workstation is plugged in and that there is power at the electrical outlet. If the workstation is plugged into a live outlet but won't start, notify your system administrator or PERQ Systems

representative.

2) Boot the system in one of the following ways:

   To get into the default operating system (Accent) when you turn on your workstation, let the workstation cycle through its boot sequence automatically. As the workstation cycles through the boot sequence, the disk spins up to speed. On the PERQ workstation this takes about two minutes; on the PERQ2 workstation, about 30 seconds.

   If the workstation is already on and you want to restart the boot cycle for the default operating system (Accent), press the boot (reset) button. On a PERQ workstation, the boot button is centered at the rear of the keyboard. On a PERQ2 workstation, depending on the model, the boot button is located either on the right side of the display base or on the front of the processor box. In either case the same sequence of events, described below, occurs.

   In addition to Accent, your workstation may contain another operating system that runs on the PERQ workstation (POS or PNX). To get into another system, press the boot button and then, just after the appearance of a black pattern on the screen, press the key several times that has been designated as the boot key for that system (see Bindboot in the document "User Facilities" in this manual).

This completes the booting sequence. If there were no problems, either the date and time will appear on the screen or you will be asked to supply it. This is the start of the login procedure explained in Section 2.1.

For your information in case of a problem, as the workstation goes through the boot sequence, numbers are shown on the Diagnostic Display (DDS). On the PERQ workstation the DDS is located on the underside of the keyboard. On the PERQ2 workstation, depending on the model, the DDS is located either on the base of the display (you may have to tilt the display to make the DDS visible) or on the front of the processor box. If any step fails, you can look at the Diagnostic Display and see where in the boot sequence the failure occurred. A list of the DDS values and their meanings is given in "PERQ/Accent Fault Dictionary: Key to the Diagnostic Display" (Appendix B to this manual). That document also describes in detail the sequence of events that occurs as the workstation boots.

In the event that the system crashes and you must reboot, there are several precautions that you should follow. See Section 10.

Also be aware that when you turn your workstation off, you should wait at least a minute until all disk activity has stopped before you turn it back on.

2. Logging In and Off

This section describes how to log in after you have booted the workstation as explained in the previous section or after someone else has used the workstation and left it turned on. This section also describes how to log off when you are done working.

2.1 Login

To login, proceed as follows:

1) Check the status of the workstation and do one of the following actions:

   If the screen is blank, the PERQ workstation is turned off. Follow the instructions in Section 1 to boot the system.

   If the screen is not blank, check to see if someone is still using the workstation and, if not, type bye to end the previous session.

2) The system sends a message across the network, asking for the time. If it gets a response, the system sets its clock. If the netserver is down or your site does not have a network, type the information in this format:

   DD-MMM-YY HH:MM:SS

   The time notation uses a 24-hour clock. The seconds are optional. If you were logging in on the 15th of January, 1984, at 2:30 p.m., you would type

   15-jan-84 14:30:00

   If your workstation is connected to the network but often it does not get the date and time, there may be a problem with your workstation or the network. Please let your system administrator or PERQ Systems representative know.

3) The system will ask for your username and then your password.

   See the ChangeUser command in the document "User Facilities" in this manual for details on how to set up your username and password and how to change your

password.

Note that your user name does not have to be the same as your SysName (the name by which your workstation is known on the network).

After you have logged in, the icon window (a portion of the screen that contains a small picture representing each window) appears at the bottom of the screen; then a window for the process manager appears at the top of the screen. The process manager window is used to display information about the system, such as which programs are currently running. Finally a large window appears in the middle of the screen, ready for your use. (Complete details on windows are given in the document "User's Guide to the Window Manager" in this manual.)

The process manager, window manager, network server, and message server were started when the system was booted. At login, the InitialShell.Cmd file is executed. Then the ShellCommands.Cmd file (referenced within the InitialShell.Cmd file) is executed and process priorities are set. Section 6 gives more information about these files, and it also explains how you can tailor the system to your preferences (for example, you may wish to have additional windows opened every time you login).

The process manager starts the message server and net server. If for some reason it cannot start these servers, the process manager will inform you. You will still be able to use your workstation, but you won't be able to communicate across the network. Report the problem to your system administrator or PERQ Systems.

2.2 Logoff

To logoff, type bye. After the system performs its maintenance functions associated with the Bye command, the system goes into the kernel debugger. (This may take a few seconds.) A message from the debugger will appear at the upper left corner of the screen in reverse video; it will stay on the screen until the workstation is turned off. If no one else is going to use the workstation, turn it off. (You should always issue the Bye command before turning the workstation off.)

3. Using the Equipment

The hardware is described briefly in the document "Introduction to the PERQ and the Accent Operating System" in this manual. The PERQ2 workstation is described in detail in the manual PERQ2 User Guide. This section tells you how to use the equipment.

3.1 Display

3.1.1 Adjustments

There are two ways to adjust the display to suit the lighting conditions and your preferences. First, you can adjust the brilliance of the display with the Brightness knob, located on the back of the display on both the PERQ workstation and PERQ2 workstation. Second, you can adjust the tilt of the display. On a PERQ workstation the tilt is adjusted by turning a thumbscrew on the back of the display. On a PERQ2 workstation, simply tilt the display to the desired angle.

You can also adjust the volume of the audio feedback you receive during certain operations. The Volume knob is located on the back of the display on both the PERQ and the PERQ2 workstations.

3.1.2 Scroll Control

If you wish you can have the system pause at the end of a windowful of output, rather than continuing and scrolling the output off the top of the window. To enable this pause facility, press CTRL-LINEFEED. From then on, until you logout, the last line in the window will be shown in reverse video (white letters over black background) if there is more output to follow. To resume output, press LINEFEED.

If you wish to disable the pause facility at any point while you are working, press CTRL-\.

3.1.3 Cursor

The cursor is an arrow on the screen which points up and left. As explained in Section 3.3, the cursor follows the movement of the mouse on the tablet or bitpad.

3.1.4 Lights

At the top of the screen above the process manager window is a row of lights in the form of small rectangles. The lights are used to indicate the state of the processor. There is no need to monitor these lights, but if you are an expert user and wish to monitor the lights, their meanings are listed below. A flashing light means that the condition is changing (for example, lights 23 and 24 flash when paging occurs). If all lights are off (white), the system is idle. If a particular light is white, it indicates the following condition (light numbers are counted from the left side the screen):

1. (unused)
2. Dirty
3. Idle
4. Memory cleanup (no free pages; looking for unmapped pages)
5. Create physical segment
6. Not enough room for physical segment (error condition; failure in 5)
7. Find a free VP record (same as 4 for VP pages)
8. Allocate a kernel AST block
9. (unused)
10. Msgs waiting non-nil (file I/O in progress)
11. (unused)
12. (unused)
13. Find least used page (when 4 doesn't find any free page)
14. (unused)
15. Write fault (probably disabled)
16. Read fault (probably disabled)
17. FindInTree (probably disabled)
18. Process page (probably disabled)
19. (unused)
20. (unused)
21. Create shadow segment
22. (unused)
23. Disk read
24. Disk write
25. (unused)
26. (unused)

3.2 Keyboard

The PERQ workstation keyboard is the standard QWERTY layout of typewriter keyboards, plus special function keys. The PERQ2 workstation keyboard also has a numeric pad, useful for entering large quantities of numbers. The special function keys are in different positions on different models and are not labeled identically, but they have the same effect on programs. Here is how the special function keys are used:

BACKSPACE and DEL - deletes the previous character

OOPS - deletes from the cursor back to the beginning of the line

CTRL - used in conjunction with other keys to issue commands. To give such a command, hold down the CTRL key and press the other key. The commands that can be given to the shell in this manner are listed in Section 4.2.6. The commands that can be given to a program will be listed in the documentation for that

INS (ESC) - completes a filename, as explained in section 4.2.7

HELP - used by itself to give an informative message on the Help utility. When pressed after typing a utility name, it gives information (if available) about that utility. For example, typing copy and then pressing HELP will give you a message telling how to copy files.

SETUP - (PERQ2 workstation only) used as a prefix to keyboard commands in the window manager program

BREAK - (PERQ2 workstation only) Retypes the previous command line (same as CTRL-p)

arrows - (PERQ2 workstation only) moves the cursor in the direction of the arrow; when used with the CTRL key, moves the cursor a larger unit, as follows:

Up-arrow - up one line
CTRL-up-arrow - up one page
Down-arrow - down one line
CTRL-down-arrow - down one page
Right-arrow - right (forward) one character
CTRL-right-arrow - forward one word
Left-arrow - left (back) one character
CTRL-left-arrow - back one word

The above functions apply to the shell and to the system editor. The keys may be used differently in other programs.

The LOCK key functions differently on the PERQ and PERQ2 workstations. On the PERQ workstation it shifts both alphabetic and numeric characters to the upper-case character of the key. On the PERQ2 workstation, LOCK shifts only alphabetic characters to upper case.

You communicate with the operating system (named Accent) by giving a command, either with the keyboard or the mouse. The command is sent to the command line interpreter, called the shell, which either executes the command or calls a program to execute the command.

The system stores keystrokes that you type while it is processing. This means that typing at high speed will not cause problems and also that in most cases you can type the next command without waiting for the previous command to finish executing.

If you hold a key down for more than half a second, it will automatically repeat.

3.3 Tablet and Mouse

Both the PERQ and the PERQ2 workstations use an electromagnetic tablet (referred to as an EM tablet) and a three-button mouse. Optionally you may use a Summagraphics bitpad and a four-button mouse (sometimes referred to as a "puck"). When the mouse is in contact with the tablet or bitpad, its position is read, and as you move the mouse the cursor on the screen follows the mouse. Pressing a mouse button causes an action, depending on the program you are using.

The mouse serves two purposes. First it cuts down on the amount of typing you must do, as many programs permit you to give commands using the mouse. In some programs, such as the window manager, you can use the mouse to obtain a list of commands (called a menu) and then you select the desired item by positioning the cursor on that item and pressing a certain button. Secondly, it allows you to use sophisticaed graphics programs to create or trace drawings. Documentation for the program you are using will tell you how to use the mouse in that program.

The mouse can be used in two modes: relative or absolute. In relative mode (the default in Accent), the cursor position is determined by the difference between previous and present tablet coordinates. The cursor does not necessarily match the position of the mouse. For example, if you lift the mouse from the upper left corner of the tablet and place it in the lower right corner, the cursor does not move; the cursor will move only when the mouse is moved on the tablet. In absolute mode, the cursor on the screen reflects the postion of the mouse on the tablet. In the above example, if you lift the mouse and move it from the upper left to the lower right corner of the tablet, the cursor will move to the lower right of your screen. Absolute mode can be put into effect for a process with the Launch facility explained in the document "User Facilities" in this manual.

3.4 Disk Drives

A fixed disk drive and optionally a floppy disk drive are contained in the processing box. A floppy disk drive enables you to use 8" floppy disks to exchange software and to make backup copies of important files. Section 8 tells how to use floppy disks. You should make backup copies of all files. You may wish to store confidential files only on floppy disks.

3.5 Peripheral Equipment (Printers, etc.)

Your system administrator will give you instructions for using the printers or other optional peripheral equipment that is installed at your site.

4. Using the Operating System

This section briefly explains the structure of the operating system and tells from a user's point of view how to interact with the system. The system is described in detail for programmers in the document "Theory of Operations" in the Accent Programming Manual.

4.1 Structure of the System

The Accent operating system consists of the kernel (the core of the system) plus several separate server processes. The server processes appear to the user as ordinary user programs. They are started automatically at boot time, and they perform their designated functions by executing calls to the kernel.

The main server processes are:

a) the environment manager

The environment manager maintains a set of variables and search lists that the shell and user programs may use as parameters. Changing these variables changes the default.

b) the window manager

This server enables you to divide the screen into different areas, called windows, and to monitor and control the processes running in the windows. The window manager is described briefly in Section 4.3 and in detail in the document "User's Guide to the Window Manager" in this manual.

c) the file system

This server enables you to divide the hard disk into a hierarchical structure of partitions, directories, and files, as described in Section 4.4.

d) the process manager

This server takes care of tasks related to the creation and termination of processes, as explained in Section 4.5, and also allocates display windows for the window manager.

e) the net server

This server provides access to the Ethernet, the Local Area Network interface between your workstation and others. (Ethernet is a trademark of the Xerox Corporation.)

f) the message server

This server sends messages to and from remote workstations on the network. The process is transparent to the user (that is, you will not be aware of the process).

4.2 Shell

The shell is a program that interprets commands from the user. When commands are issued (from the keyboard or a command file), the shell checks the command against the set of commands that it recognizes. If the command is an operating system command that the shell can execute directly, it does so. If the command is one that calls another program, the shell turns control over to that program. If the shell does not recognize the command, it gives you an error message.

4.2.1 Command Syntax

A command line consists of a command name, arguments, optional switches, and a line terminator (usually RETURN). Switches may take parameters. (All are explained in the sections below.)

The syntax of a command line is its grammar--that is, a definition of which strings of characters are valid. (In contrast, the term "semantics" denotes the meaning. For example, "copy" has a precise definition within the operating system, but in order for the system to know exactly what file to copy and where to place the new file, you must supply instructions to the system by using the proper syntax.)

The Accent operating system syntax conventions are as follows:

> If you are supplying as arguments only one input file and one output file, separating the names with a space is usually sufficient. The description for each command given in the document "User Facilities" in this manual will tell you if the command requires that the input and output names be separated with a tilde (~).
>
> If you are supplying more than one input file or more than one output file, separate the files with commas (,) and separate the list of input file(s) from the list of output file(s) with a tilde (~). You may put spaces before and after these punctuation marks, but you don't have to.
>
> If you want to insert a comment into a command line (for example, in a command file), begin any word with a number sign (#). The rest of the line after # will be ignored.
>
> To quote a character, precede it with a backslash (\). For example, if a filename contains the character * or ? (which can be used as wildcards, as explained in Section 4.2.9), when

you type the filename to the shell you must precede the character * or ? with \. The tilde character (~), used to separate input and output files in a command line, must also be quoted if used for itself.

To quote a string, enclose the string in single quotes ('...') or double quotes ("..."). For example, to find occurrences of the string Release 1 in all .pas files, you could type: findstring 'Release 1' *.pas.

To supply an environment variable name, enclose it in carets (^...^). For example, if you want to pass a standard list of switches to the compiler, create an environment variable with the switches and pass it on by enclosing the environment variable name in carets in the command line. You could also use this as an alternative to the Alias command described in Section 6.

To supply an environment variable name as above but as part of a quoted string, enclose the string in double quotes ("...^...^..."). .

With the exception of the environment variable name within a quoted string, explained in the preceding paragraph, none of the quoting or substitution constructions can be nested.

To specify a switch to a command, precede the switch with a space and a hyphen (-).

To specify an argument to a switch, precede the argument with an equals sign (=).

To specify standard input/output redirection and sequential execution, use the following characters:

> redirect output

< redirect input

| pipeline one program's output to another program's input

; execute processes sequentially

To give instructions to the shell on how to invoke a program in a redirect/sequential execution list, enclose one or more switches within square brackets [ ] and precede each switch with a hyphen (-). The switches presently supported are shown

below (they can be abbreviated).

-parallel

-debug

-loadbug

-SameEnvPort

Thus the entry

My_Program [-SameEnvPort -Debug]

means that the shell should invoke the process My_Program, giving it the same port to the environment manager as the shell and to invoke the system's debugger upon that process.

Note that as an alternative to the [-parallel] construct, you may simply enter an ampersand (&) as the last character of the command.

For programmers, the file system syntax and semantics are discussed in the document "The File System" and the inter-program argument format is discussed in the document "Theory of Operations," both in the Accent Programming Manual.

4.2.2 Command Names

The command name describes the action the system performs or the name of the utility that performs the action. All of the commands that can be executed by the shell or by a program that resides on the system when it is delivered to you are listed in Appendix A, "Command Summary," and explained in detail in the document "User Facilities" in this manual.

To get an on-line list of commands, type ?.

In deciphering commands the shell does not distinguish between upper and lower case letters; use whichever you prefer.

You cannot abbreviate a command unless you have established the abbreviation as an alias, as explained in Section 6.

4.2.3 Arguments

Input and output arguments further define the command action. The arguments are usually filenames. Some commands use a default file as the argument if you do not supply one; other commands require that you supply a filename (see details for each command in the document "User Facilities" in this manual).

If a command requires an input and an output argument, specify the arguments as:

```
Input~Output
```

Some commands that expect only one input and one output (e.g., copy and rename) permit you to separate them with a space and/or a comma:

```
Input Output

    or

Input,Output
```

Multiple spaces are acceptable.

If a command accepts multiple input or output arguments (such as the Link command, which connects any number of source files into a run file), you may separate like arguments with either spaces or commas and you must distinguish input from output with the tilde character (~). In such cases the correct form is either:

```
input1,input2,...inputN ~ output1,output2,...outputN

                    or

input1 input 2 ... inputN ~ output1 output2 outputN
```

If you neglect to supply a required argument, the utility prompts with a few words indicating the general nature of the missing argument. For example, the Rename command replies as follows (in this example user input is underlined):

```
rename
File to rename:  oldfile
Rename OLDFILE to:  newfile
```

The single line format for the above command is

```
rename oldfile ~ newfile
```

or

```
rename oldfile newfile
```

You can mix formats, and the utility will prompt for the missing information. For example:

```
rename oldfile
Rename OLDFILE to: newfile
```

Some commands accept wildcards in filenames, as explained in Section 4.2.9.

In some cases programs supply default answers as arguments or as replies to questions. The default answer is shown in square brackets ([ ]). You can choose this answer by pressing RETURN. To supply a different value, type the value and press RETURN. For example, if you used wildcards with the Delete command, the system would ask for verification before deleting each file by issuing a prompt like this:

```
Delete MyFile [No]:
```

You would press RETURN if you did not want to delete MyFile (or you could type no or just n). If you wanted to delete the file, you would type yes or y.

When there is no default for a prompt (as in the Rename example above), you must supply a response or request help. To request help type -help or press the HELP key. If you request help, you will receive a help message (if one is available for that utility) and then you will be returned to the shell; to execute the command, you must retype it. (Also see the command line editing capabilities described in Section 4.2.6.)

4.2.4 Switches

Switches modify the action of the command and therefore are typed after the command name. Switches are generally optional. They must be preceded with a space and a hyphen (-). If a switch accepts a parameter, specify the parameter after the switch but preceded with an equal sign (=), as follows:

```
command -switch=parameter
```

The effect of a switch is global, regardless of where it appears on the line--that is, it applies to every argument. Therefore if a command accepts multiple input or output arguments, no switch applies to one and not another argument. Some switches are mutually exclusive (for example, -ask and -noask). If you specify a switch that conflicts with a previously specified switch, the last occurrence has precedence. Likewise, if you change parameters by specifying a switch multiple times, only the last occurrence has an effect.

The -help switch overrides all other switches. The utility will display a help message (if one is available) and then exit to the shell.

4.2.5 Line Terminators

A command line given from the keyboard must be terminated by pressing one of the following keys:

a) RETURN to execute the command

b) HELP to obtain a help message (without executing the command)

c) INS(ESC) to complete the filename, as explained in Section 4.2.7. (You need to type only enough letters of the filename to distinguish it from other filenames.)

4.2.6 Issuing Commands

Commands may be issued from the keyboard, from a popup menu using the mouse, or from a command file.

From the Keyboard:

To issue a command from the keyboard, type the command line. If you wish to cancel a command, you can erase the line using the OOPS key or the BACKSPACE key if you have not yet pressed RETURN. If you have pressed RETURN, you can stop execution by typing CTRL-DEL-c (that is, press CTRL and DEL together and then lower case c). On the PERQ2 workstation you can use the SETUP key to stop execution.

Before you press RETURN you can edit the command using any of the following editing commands:

CTRL-a Move cursor to beginning of line

CTRL-b Move cursor back one character

CTRL-B Move cursor back one word

CTRL-d Delete character to right of cursor

CTRL-D Delete word to right of cursor

CTRL-e Move cursor to end of line

CTRL-f Move cursor forward one character

CTRL-F Move cursor forward one word

BACKSPACE or CTRL-h or DEL Delete character to left of cursor

CTRL-H Delete word to left of cursor

TAB or CTRL-i or CTRL-I Insert spaces up to the next tab stop (tabs are spaced every eight characters from left margin)

CTRL-k Delete characters to right of cursor, to end of line

CTRL-n Retrieves the next command in a ring buffer of previously issued commands (see Section 4.2.8)

CTRL-p Retrieves the previous command in a ring buffer of previously issued commands (see Section 4.2.8) The BREAK key on the PERQ2 workstation performs the same function.

CTRL-t Transpose two characters to left of cursor

CTRL-v Displays following windowful from transcript buffer (see Section 4.2.11)

CTRL-V Displays the previous windowful from the transcript buffer (see Section 4.2.11)

CTRL-w Turns "wrap" mode on (output lines that are too wide for the current window are continued on the next line). If the window width is changed, the lines are redisplayed for the new width.

CTRL-W Turns "wrap" mode off (output lines that are too wide for the current window are truncated). If the window width is changed, the lines are redisplayed for the new width.

OOPS or CTRL-u Delete characters to left of cursor, to beginning of line

CTRL-LINEFEED Enables scroll control (see Section 3.1.2)

CTRL-\ Disables scroll control (see Section 3.1.2)

LINEFEED Displays the next windowful of output, when scroll control is enabled (see Section 3.1.2)

INS or ESC Completes a partially specified filename (see Section 4.2.7)

CTRL-? List filenames that could match a partially specified filename (see Section 4.2.7)

From a Popup Menu:

The window manager and some programs provide popup menus. The procedure for using popup menus to issue commands to the window manager is covered in detail in the document "User's Guide to the Window Manager" in this manual. Briefly, the procedure is as follows. Use the mouse to position the cursor either in the icon for the window to which you wish to give a command or on either the right or left third of the title line for that window. Press the right button (green button on a 4-button mouse). A menu (a list of commands) with a black movable bar appears. Use the mouse to move the bar to the desired command and press any button. The command will be processed. (To remove the menu without giving a command, use the mouse to move the cursor outside the menu and press any button.)

From a Command File:

A command file can be used to store a series of commands. Create the file as you would any other file. Within the file list the command lines just as you would type them to the shell. After you have created the command file, you can invoke it by typing @<filename>. The commands contained within the command file will be executed.

A command file can also be used to invoke other command files; within the calling file insert the line @<filename>.

4.2.7 Automatic Completion of Filenames

The key labeled INS on the PERQ workstation and ESC on the PERQ2 workstation can be used to ask the shell to complete a filename which you have partially typed. If you have typed enough characters to uniquely identify a file, the shell inserts the rest of the filename and moves the cursor to the end of the filename.

You can also use INS(ESC) to expand a filename. Position the cursor at the desired place in the filename and press INS(ESC). If there is a file that has the characters you typed at the beginning and end of the name, the shell will supply the middle characters. For example, if you have a file named report5-25.txt you can type reptxt, position the cursor before the first "t", and press INS(ESC). The shell will fill in the middle of the name.

With either procedure explained above, if the shell cannot find a filename with the chararacters you typed or if more than one filename has these characters, the screen flashes. If there are several candidates, the names are then listed. If the shell can complete the name but not the extension, it will complete the name and stop. Either type one or more characters and try again, or type CTRL-?. CTRL-? lists all of the filenames that are candidates to complete the filename.

CTRL-? typed by itself at the beginning of a line will list all of the files in the current directory (or, if the current directory is empty, all of the files in the first non-empty directory in the default search list).

4.2.8 Retrieval of Previous Commands

Commands which you issue from the keyboard are kept in a ring buffer, and you can retrieve previously issued commands by moving around in this ring using CTRL-p and CTRL-n. Sixteen commands are stored. Typing CTRL-p retrieves the previously issued command. For example, if you've given 16 commands since login, CTRL-p retrieves the 16th, another CTRL-p retrieves the 15th, etc. Typing CTRL-n retrieves the next command in the ring, going forward. In this example, CTRL-n retrieves the first command you gave.

When the command is retrieved, the cursor will be positioned to the right of the line, just as though you had typed it. When you retrieve the desired command you can edit the command or press RETURN to execute it. (If you edit the command, you do not need to return the cursor to the end of the line before pressing RETURN.)

The SETUP key on the PERQ2 workstation performs the same function as CTRL-p.

4.2.9 Wildcards

A number of utilities use a wildcard convention when looking up files--that is, if two or more files have part of their name in common you can specify them by substituting special characters for all or part of the name. If a utility accepts wildcards, you can use any number of wildcards with it.

Accent uses the following wildcards:

*	matches 0 or more characters
?	matches exactly one character

Examples of wildcard usage are:

```
dir *.txt
```

and

```
dir ?boot*
```

The first command provides a directory of all files with a .TXT extension. The second command provides a directory of all files that have one character followed by the characters "boot" followed by zero or more characters.

When using wildcards in a pathname, only the last component (directory or filename) in the pathname can use a wildcard, except with the Directory command which will accept a wildcard in any position. Therefore, copy /sys/user/*/report.txt is invalid but dir /sys/user/*/report.txt is valid.

If a filename contains one of the characters used for wildcards, you can specify the character itself by preceding it with a backslash (\). For example, if a filename is Sch*Acct, you would specify it by typing Sch*Acct.

4.2.10 Command Files

The shell and some utilities permit the use of command files. A command file is a sequential file containing a list of utility specific commands. Instead of typing commonly used command sequences each time you use the commands, you can type the sequence once and store it in a file. Then when you use the utility, replace the command line with the command filename preceded by an at sign (@). (In some utilities you may not need to include the @ if the filename extension is .CMD.) The utility requesting input then accesses the specified file and starts to read and respond to the commands contained within it. For example, to initiate a file of Floppy commands named MyFile.Cmd, type the following in response to the Floppy prompt:

```
FLOPPY> @myfile.cmd
```

The Floppy utility would access the file and then execute the commands contained within the file. The default file extension for user command files is .CMD and therefore the above line could also be typed as follows:

```
FLOPPY> @myfile
```

You can nest user command files by including within a command file a line that calls another command file (@ followed by the other file's name).

To include comments in a user command file, start the comment line with a number sign (#). The # character is recognized only if it is the first character of a word; the rest of the characters on that line will be ignored. This is very useful for documenting the purpose of the file and perhaps the purpose of each line in the file.

For experienced users, several other options are available within command files that are executed from the shell:

> If the line has the character "&" at the end, the command will be executed in another process running in parallel with the shell and using another window. The process manager will prompt you to indicate the size and position of the new window. An exception to this, however, is that when the line begins with "@" and ends with "&", the "&" is ignored. If you want to run another command file in parallel, you must create another window and then invoke the command file in that window.
>
> The following redirection commands may be used: > to redirect

output, < to redirect input, and | to pipeline one program's output to another program's input. A semi-colon (;) may be used for sequential process execution.

4.2.11 Session Transcripts

The shell keeps a transcript of the current session in each window. The transcript contains all of the commands you have issued and any program output that has appeared in the window. The amount of text retained in the buffer is about three windowfuls, regardless of the window size. Thus, changing the size of a window changes the size of the buffer for that window. The size stays in effect until you change window size again or logoff.

To look at the transcript type CTRL-v to move forward in the buffer or CTRL-V to move backward.

The purpose of this buffer is only to let you look at the transcript. You cannot retrieve and execute commands in this manner. If you type anything while the transcript is displayed, the transcript is advanced to the end and what you have typed is inserted at the place where the cursor was positioned before you started looking at the transcript.

4.3 Window Manager

The Accent operating system includes a window manager that allows you to divide the screen into areas called windows. Each window usually has its own shell (command interpreter), so "window" and "shell" are often used interchangeably.

You can have several windows open at the same time, each running a different process. Each window has an icon (a smaller representation of the window) associated with it. When you login, several windows are opened. The default is three windows: a window at the top of the screen for the Process Manager, a window at the bottom of the screen for the icons, and a window in the center of the screen for your use. As described in Section 6, you may change the InitialShell.Cmd file if you wish so that it automatically opens other windows.

One way to open a new window is to type shell. After you type the command, a blinking cursor in the form of an upper left corner appears. Position the cursor where you want the upper left corner of the new window and press and release any button. Then the cursor will change to a lower right corner. Move the cursor to where you want the lower right corner and press and release any button. A window created in this manner is said to have been "spawned" from the parent window, and a spawned window inherits the environments of the parent window (such as the current directory and the current searchlist).

In order to communicate with the new window, you must designate it as the listener by moving the cursor into the window and pressing any button.

You may change the size of windows; put one on top of another; move them on and off-screen; and suspend, resume, and cancel processes that are running in the windows. All of these operations plus other ways to open windows are described in "User's Guide to the Window Manager" in this manual.

4.4 File System

This section discusses basic concepts of the file system from a user's point of view and tells how to accomplish some common tasks.

4.4.1 File System Structure

The PERQ workstation includes a software system to oversee the storage and handling of files. A file is an owner-named area on the hard disk.

The file system has a hierarchical structure. At the factory the hard disk (the device) is formatted for use with the file system. It is shipped ready to use and has the device name "Sys". You should then divide the device into a number of areas called "partitions" (for details, see the Partition utility in "User Facilities" in this manual). The recommended size for a partition is 10,080 or fewer 256-word blocks. (Files must fit entirely within a single partition; they cannot cross partition boundaries.) Examples of partition names are BOOT and USER. Partitions can then be divided into directories, and directories into files and/or other directories (these procedures are discussed in Sections 4.4.4 and 4.4.5 respectively).

The hierarchical structure of the file system can be seen by issuing the command dirtree to see the directory tree. The dirtree command is fully explained in "User Facilities."

4.4.2 Path Names

The route that the system travels through the hierarchical structure of the file system to reach a file is called a "path." A full pathname specifies device, partition, directories, and filename, in that order. The syntax of a pathname is:

/device/partition/{directory and optional filename}

The brackets { } surrounding the "directory" part indicate that there may be 0 to 9 nested directories (directories within directories) listed; do not type the brackets. The last name listed can be a filename instead of a directory name. You must type the slash (/) before each part of the pathname.

The "current" path is the path which is in effect; it is shown at the top of the window. The Path command shows you what path is in effect and gives you the option of changing it. Type path; the current path is displayed and you are prompted to either type a new path name or press RETURN to leave the current path in effect. Or you can just set a new path by typing path <NewPathname>.

A path name can have up to 255 characters, but each component of the path name can have only up to 80 characters (except a filename, which cannot exceed 25 characters).

There are three types of path names--absolute, logical, and relative. For each type, the root is always the device (whether it's stated or the default).

A) absolute - This is a complete path name:

/device/partition/{directory or filename}

An absolute pathname is used to specify a directory or filename independently of the current directory. The partition is followed by 0 or more directory names (the last one could be a filename). By using an absolute pathname, you can call a file on another workstation on the network, as explained in Section 9.

B) logical - This is a path name in which a synonym is substituted for part of the path name. At present you can use the following logical names:

Current: - the current directory, as set by the Path command

Default: - the search list for files, as set by Setsearch

Run: - the system's default search list for runnable (.run) and command (.cmd) files. When you boot, run is set to current:,boot: so the system will look first in the current directory before looking in the boot partition. You can change the list, as shown below.

Boot: - the partition from which you booted

Dev: - the device from which you booted. If you booted from your hard disk, the device is Sys. If you booted across the network from another workstation, it would be the device on that workstation.

To get a list of currently defined logical names, type show. To define new logical names, use the Define command. For example, to redefine the Run: searchlist, you could type:

```
define run: default:, /sys/user/utilities
```

Thereafter when Run: is specified in a path name, the system would look for .RUN and .CMD files first in Default: and then in the Utilities directory.

C) relative - This is a path name relative to the current directory. It is interpreted differently by different commands and programs--in some cases the path name is considered to be prefixed by the logical name Current:, in some cases it is considered to be prefixed by the logical name Default:. Default: is more encompassing, as it includes Current:.

The Delete command assumes only Current:, so that you won't accidentally delete a file in another directory. The editor assumes Default: because no harm is done if you access the wrong file. For information on how other commands treat path names, see the document "User Facilities" in this manual.

If Default: is assumed, the directories in the Default: search list are searched in order. Once one or more files are found in the directory the search terminates, so not all the directories on the search list may be searched.

If you are accessing a directory or file on the hard disk of your own workstation, you will use a logical or relative path name. If you use a relative path name, the amount of information you must supply depends on where the directory or file is in relation to the current path. The methods are:

a. If the directory or file is in the current directory or in a directory on the search list, explained below, you can just type:

    directory or filename

b. If the directory or file is in the device from which you booted and in the current partition, but in a different directory, you may omit the device and partition, as follows:

    {directory}/optionally a filename

    The search then starts at the first directory you specify.

There are two time-saving devices you may use with the above methods:

A period (.) can be used to refer to the current directory. This is convenient in those few instances when the current directory is not the default or it's not on the default searchlist. For example, suppose you edit a file that's in a different directory and want to write the revised version to the current directory. When you instruct the editor on where to put the revised file, you can type a period for the directory in the path name.

Two periods (..) can be used to refer to the parent node in the directory tree. For example, to switch to a different directory named New which is nested under the same directory as the current one, you could type ../new.

4.4.3 Search List

In parallel with the concept of a current directory, the file system provides a search list. A search list is a set of directories to be searched, in a specific order, whenever you specify a filename without specifying a pathname.

To use this command, type setsearch. The command displays the paths currently on the search list and prompts for input. To add a path, type the pathname and press RETURN. To remove the top name on the list, use the -Pop switch. To return to the command level without changing the search list, press RETURN.

Do not remove /Sys/Accent from the search list (a great deal of useful system software resides there).

SETSEARCH stays in effect until you change it again or until you logoff. The next time you login, SETSEARCH will be back to its original default.

For more details on the SetSearch command, see the document "User Facilities" in this manual.

4.4.4 Directories

A directory is a collection of files or other directories. A directory that is created within another directory is called a nested directory.

To create a directory, path to the partition or directory under which you want to create the new directory. Then type makedir followed by the name you wish to assign to the new directory. For example, to make a new directory named "Plans" under the current directory, type makedir plans.

If you then wish to create a file in the new directory, you must either use the Path command to set the current path to the new directory or include the new directory in the pathname (for example, edit /sys/smith/plans/newfile).

To see what files are contained in the current directory, type the command:

```
directory
```

The command can be shortened to dir. If the command is typed as shown above with no arguments, an asterisk (*) denoting all files is assumed. Wildcards, listed in Section 4.2.9, can be used in any combination. For example, to display all of the filenames ending with the characters .cmd, type:

```
dir *.cmd
```

or for all files beginning with the letter p, type:

```
dir p*
```

To store a useful directory listing in a file rather than displaying the listing on the screen, follow the command with a tilde and a filename. For example, to store a list of all the files ending in .cmd (command files) in a file named "CMDFILES" type:

```
dir *.cmd ~ cmdfiles
```

4.4.5 Files

A file is an owner-named area on the hard disk or a floppy disk, containing a unit of information (for example, a program or a module of a program or the text of a paper or report).

Filenames can contain letters, digits, underscores (_), periods (.), hyphens (-), plus signs (+) and dollar signs ($). The maximum length permitted is 25 characters. The part of the filename that appears at the end of the name following the last period is called the extension. (If there is no period, the filename has a null extension, that is, no extension).

When you specify a filename, you may use upper or lower case letters. (However, if you plan to use Qnix, be aware that case does matter.)

Following are brief descriptions of how to perform some common functions with files. All of these commands and the switches that can be used with them are covered in more detail in the document "User Facilities" in this manual.

Creating and Editing

Files are usually created and edited using one of the two editors that are supported in the Accent operating system:

a) the standard system editor, which is a "point and act" editor. First you position the cursor where you want to insert text or make changes (the cursor is positioned by moving the mouse or issuing commands). Then you type the text to be inserted or give a command to make a change. Details are given in the document "The Editor" in this manual. The editor makes a backup copy of each file you edit and attaches a dollar sign ($) to the extension of the backup copy.

b) Hemlock, an editor written in Common Lisp. Details are given in the document "Hemlock Command Implementor's Reference" in the manual Accent Lisp.

Copying

Files can also be created by copying existing files on your own or another workstation. Issue the Copy command as follows (the angle brackets indicate that you should insert a filename; do not type the brackets):

copy <SourceFilename> <DestinationFilename>

The SourceFilename can have wildcards in it as long as the DestinationFilename has the same wildcards in the same order. For example, if you have written two papers (filenames beginning with "theory") to be presented to one type of audience and you wish to copy the papers and revise them for another type of audience, you could type copy theory*.* revtheory*.*. All files that match the SourceFilename will be copied by taking the characters that match each wildcard and putting these characters in the corresponding place in the DestinationFilename.

Renaming

Files can be renamed by typing

rename <OldFilename> <NewFilename>

Appending

One or more files can be appended to the end of a file by typing

append <File1,File2,File3,...>

The append command changes only the first file listed. File2 is attached to the end of File1, then File3 is attached at the end of the combined File1-File2, etc. File2, File3, etc. are retained as separate entities.

Typing

If you want to look at a file without making any changes to it, you can have it typed on the screen by typing

type <Filename>

Printing

The command to print a hardcopy of a file will depend on the devices installed at your location.

Deleting

To delete a file, the command is

delete <Filename>

The Delete command can be abbreviated to del.

If you use wildcards the Delete command will ask you to confirm file by file that you want to delete each file matching the wildcards.

When you delete a file you cannot retrieve it.

If you think that you might want to use a file again someday but you don't want to keep it on-line, copy it to a floppy as explained in Section 8.

4.4.6 Filename Extensions

An extension is a sequence of characters that appears at the end of a filename, following the last period in the name. The operating system treats extensions simply as part of filenames. When documentation states that a filename should be supplied, the extension should be included in most cases. However, certain utilities interpret extensions in special ways, and in some cases the utility will attach a default extension (see the document "User Facilities" in this manual).

Backup files conventionally have a dollar sign ($) as the last character of the extension.

The following is a list of standard extensions and how they are used:

.PAS	- Pascal source file
.FOR	- FORTRAN source file
.DAT	- FORTRAN data file
.SEG	- segment file produced by the Pascal or FORTRAN compiler when they compile a .PAS or .FOR file. The .SEG files are used as input to the Linker and contain the code that will be executed when the program is run.
.RUN	- executable file produced by the Linker. The .RUN file refers the system to the executable .SEG files.
.MICRO	- PERQ microcode files. They can be used as input to the micro assembler, PRQMIC.
.BIN	- file containing the runnable version of PERQ microcode. .BIN files are produced by the micro placer PRQPlace.
.REL, .RSYM, and .INT	- temporary files produced by the micro assembler and read by the placer. These files can be deleted after a .BIN file has been created.

.DFS - file used to communicate definitions between programs that may be in different languages (for example, between Pascal and microcode)

.CMD - file that contains a number of commands, all of which are executed when the file is called by typing @<Filename>. Some programs automatically check for certain command files when invoked.

.KST - file containing a character set definition

.KEYTRAN and .KTEXT - files which the window manager uses to determine the mapping of keys on the keyboard and buttons on the mouse to commands for the servers, facilities, and user programs. (The KeyTran compiler uses .KTEXT files as source files and produces object files with the extension .KEYTRAN.)

.MBOOT and .BOOT - files containing microcode and the Pascal operating system, respectively. These files are read when the PERQ workstation is booted.

.ANIMATE and .CURSOR - files containing pictures put into the the cursor that follows the mouse. (Various utilities use the pictures to depict progress or action.)

.SCURSOR - file containing a header block and a set of cursors

.INDEX - file containing an index to help files

.DOC - file containing formatted documentation (the output of the Prose formatting program)

.HELP - file containing a help message

4.5 Process Manager

The process manager controls the programs that are running in each of the windows. It tells you the status of the processes by displaying the following symbols in the windows' icons:

keyboard - program is waiting for user input

bug - error encountered

exclamation point - window needs attention

progress bar 1 - progress in program (proportion completed or random progress)

progress bar 2 - progress in command file (proportion completed or random progress)

In addition to the above symbols, each icon contains a 6-letter text string that identifies the window and three dots if the window is off-screen.

You can give commands to control the processes that are running in any of the windows by issuing the window manager commands listed below, either from the keyboard or a menu.

To issue a command from the keyboard, the window in which the process is running must be the listener (use the mouse to position the cursor within the window and press any button). The command prefix on a PERQ workstation is CTRL-DEL. On a PERQ2 workstation you may use either CTRL-DEL or SETUP.

To issue a command from the menu, you can request the menu from either the window itself (use the mouse to position the cursor in the left or right section of the title line and press the right button) or from the icon for that window (position the cursor in the icon and press the right button).

Debugging - To send a process into the debugger, select "debug process" on the command menu or type <prefix>d. You will be asked to open a window for the debugger.

Suspending - To suspend a process select "suspend process" on the command menu or type <prefix>z.

Resuming - To resume a process that you had suspended, select "resume process" on the command menu or type <prefix>q.

Cancelling - To cancel (abort) a process, select "cancel process"

on the command menu or type <prefix>c.

Killing - To abort a process as well as any command files and to return the shell to the top level, select "kill process" on the command menu or type <prefix>k.

5. Getting On-Line Help

There are four ways to get on-line help:

A) the Help command

Type help followed by a word or phrase that describes the facility. To get a list of the valid words and phrases, type help without an argument.

B) the Help switch

Some commands and programs recognize the help switch. If you cannot get a Help message using Method A, try typing <name> -help.

C) the HELP key

Pressing the HELP key will give you a short message on the Help facility itself and the subjects on which help is available. Pressing the HELP key after typing a utility name will give you information (if available) about that utility.

D) the ? command

Typing ? will give you a list of all the built-in shell commands, followed by the user-defined commands.

6. Tailoring the System to Your Preferences

The following files are used to identify your workstation and to tailor the system to your preferences. Do not create files with these names for any purpose other than that described below.

boot:SysName

This file consists of a single line of text containing the name by which your workstation is known throughout the network. (This name has nothing to do with the Login procedure, and your SysName and user name do not have to be the same.)

If you change SysName, the new name becomes effective the next time you boot your workstation.

We suggest that you use your last name, unless another user has the same name or your system administrator prefers a different naming convention.

boot:InitialShell.Cmd

After booting, the shell looks for this file and then executes the commands contained in that file. This file can define new commands and environment variables, set server priorities, and define logical names. This file can also be used to call other command files by including a line @<Filename>.

The following priorities are recommended for the window manager and should be placed in this file:

```
priority 10mhznetserver 13
priority 10mhzmsgserver 13
priority sapphire 12
priority typescript 11
priority proc 10
```

You may include Launch commands to startup programs, as explained in the document "User Facilities" in this manual.

run:ShellProfile and run:ShellCommands.Cmd

When you open a new window using a window manager command, that window inherits global environment

variables and, if these two files exist, they will be executed in the order shown. (However, if you open a window using the Shell command, the new window inherits the environment of the parent window and these files are not executed.)

ShellCommands.Cmd (but not ShellProfile) should be referenced in the InitialShell.Cmd file (@shellcommands.cmd).

You can use the Define and Alias commands to define environment variables and new commands. Both commands are explained briefly below and in more detail in the document "User Facility" in this manual.

Define

To define or redefine a variable in the environment, type

define VariableName VariableValue

The VariableValue can be followed by the switch -local or -global. The -local switch (the default) makes the variable applicable only to the current window or any windows created from it with the shell command. The -global switch makes the variable applicable to all windows.

An example of a Define command is the following line which identifies the printserver to be used when a print command is given without specifying a server (in this example the printerserver is named "Duke"):

define printservername duke -local

The above command changes the default printserver for only the current shell. A local environment takes precedence over a global environment variable, so the printservername in effect for other shells remains unchanged by the above command.

The variable stays in effect until you log off. If you wish to have a variable value put into effect each time you log on, insert the define command into the InitialShell.Cmd file.

To see what variable values are in effect, type show.

To remove a variable from the environment, define the variable with an empty value--that is, just type

```
define VariableName
```

Alias

To define new commands, type

```
alias <CommandName> <Definition> { -Switch}
[Documentation]
```

The Definition is a previously defined command or program name, including arguments and switches. Thereafter, typing just the word you have designated as CommandName is equivalent to typing the Definition.

Documentation is any information that you wish to have appear with the command when you type ?. When you supply the documentation you must enclose it in quotes if it includes any special characters, including blank spaces.

The alias substitution does not include RETURN, so when you type the CommandName you can add more arguments and switches and then press RETURN.

An example of an Alias command is the following line, which equates the command "ls" used in some systems for a list of files to the Accent command "direct" for a directory of files:

```
alias ls direct 'Alternative directory lister'
```

After typing this command you could then use the ls command.

7. Running Programs

The following sections tell you how to run programs that came with the system and, if you wish, how to prepare your own programs to run on the system.

7.1 Programs That Reside on the System

Executable programs are stored in files with the extension .RUN. For example, the Scavenger program is contained in a program named Scavenger.Run.

Once a program exists in such a form, it can be executed by supplying its name (without the .RUN extension). For example, to run the Scavenger program, just type scavenger.

Files in /Sys/Accent are automatically available to all users of the workstation. If a file that is needed in order to use a program does not reside in the current directory or in /Sys/Accent, it will be necessary to use the Setsearch command explained in Section 4.4.3 to add the path to the directory that contains the file or define a new logical name for the Run: search list, as explained in Section 4.4.2.

7.2 Your Own Programs

The Accent operating system supports the use of Pascal, FORTRAN, C, Lisp, and microcode.

There are four steps in writing your own program and preparing it to run:

1. creating a source file;
2. compiling the source file to produce a segment file (a file of Q-Codes);
3. linking the segment file to create a .Run file; and
4. running the program.

Each of the steps involves manipulation of at least one file.

To create the source file, type in the text using the system editor

(see "The Editor" document in this manual) or the Lisp editor.

The Compile, Link, and Run commands are covered in "User Facilities" in this manual.

At the present time preparing C programs involves the cpp, pcc, asm, and lnk commands, all explained in the document "User Facilities."

Complete instructions for programming on a PERQ workstation are given in the following manuals:

Accent Programming Manual
Accent Microprogramming Manual
Accent Languages Manual
Accent Lisp Manual
Accent Qnix Manual

8. Backing Up Files

On-line files can be lost due to a disk failure or human error. Therefore it is very important that you back up your files regularly (that is, make an off-line copy of the files). You are protected to an extent by the editor, which always saves a transcript of the latest editing session and a backup copy of any file you edit; however, this will not help if you can't access on-line files.

There are two ways to save files off-line:

a) on floppy disks (details are given in the following sections); or

b) on streamer tapes, if your company owns a streamer (see your system administrator).

You do not need to back up files that came with your operating system. If you lose those files, your system administrator can help you re-load the system.

If you have confidential files you may wish to store them only on floppy disk, rather than on the hard disk.

You should take the following precautions in using floppy disks:

Do not place a floppy disk on your tablet or bitpad. Because both are magnetized, the floppy may be damaged.

Do not touch the recording surface.

Do not bend or fold the floppy disk.

Do not smoke while handling floppy disks.

Insert the floppy disk carefully into the floppy disk drive until the back stop is reached.

Do not use paper clips to attach anything to the floppy disk.

After attaching the label to the floppy, do not write on the label with a sharp pen or pencil; use a felt-tip pen.

Always return the floppy disk to its protective dust jacket after use.

Avoid heat and cold. Store at 10 degrees C to 52 degrees C (50 degrees F to 125 degrees F).

8.1 Formatting Floppy Disks

A new floppy disk contains no recorded information. Before it can be used on the PERQ workstation, a floppy disk must be formatted (a technical term that means writing onto the floppy disk the basic information that the PERQ workstation needs) and zeroed (a directory created on the floppy disk).

In order to format and zero a floppy disk, it is first necessary to disable "write protection"--the mechanism that protects a floppy from being accidentally overwritten. To do this, cover the largest of the three notches on the floppy's perimeter with a special sticker made for that purpose. Then place the floppy in the floppy disk drive (label side up on the PERQ workstation, label side facing right on a PERQ2 workstation) and close the door of the floppy disk drive.

To format and zero a disk the commands to use, in order, are floppy format and zero. After each of these commands, a short series of self-explanatory prompts is issued. Until you become familiar with the Floppy utility, answer yes to each question by typing y or pressing RETURN if "yes" is shown as the default ([Yes]). (For more information, see "Floppy" in the document "User Facilities" in this manual.)

The floppy disk is now ready for use on the PERQ workstation.

A floppy disk needs to be formatted only the first time it is used. Thereafter if you wish to destroy all the files on the floppy disk and reuse it, you need only to zero it.

8.2 Transferring Files to and from Floppy Disks

Files placed on a floppy disk can be used on other PERQ workstations or on the same one at a later date. To guard against accidental erasure when there are important files on the floppy disk, the write-protect mechanism should be reactivated by removing the sticker.

To put files onto a floppy or to retrieve files from floppy to hard disk, invoke the Floppy utility by typing:

floppy

Commands that you type will then apply to the files on the floppy disk, not the files on the hard disk. Therefore direct or delete could be issued to display the names of the files on the floppy disk or to delete some of them.

To copy a file from hard disk to floppy disk, the command is

put <filename>

To go in the other direction, i.e., from floppy to hard disk, the command is

get <filename>

After either of the above commands you will have two copies of the file, one on the hard disk and one on floppy disk. If the copy on the floppy disk is not needed, delete it now. If the copy on the hard disk is not needed, delete it when you leave the Floppy utility.

In the Floppy utility, filenames on the floppy disk can be only six characters long and extensions three characters long. If the name and extension you specify is longer, each will be truncated.

To leave the Floppy utility, type q for quit.

For more information on the Floppy utility, see "User Facilities" in this manual.

9. Using the Network

With the Accent operating system, two or more workstations can be linked into a network via an Ethernet. You can copy files or run programs on other workstations on the network, and other users can access your files and programs (assuming in both cases that access privileges have been granted).

If your site does not have a network, this section does not apply to you.

9.1 Your Workstation Name

You should assign your workstation a name so that others on the network can communicate with you. Create a file named SysName in your Boot: or Accent partition and in the first line of the file type in the name by which you want your workstation to be known. We suggest your last name (unless another user on the network has the same name or your system administrator has a naming scheme that you should follow).

When you boot your workstation, the name that you have stored is entered into the network system. Therefore, after you enter or change your name, people cannot reach you by that name until you reboot.

9.2 Communications with Other Workstations

You can access files and execute programs on another workstation if the following conditions exist:

1) the other workstation is on and running Accent;

2) the other user allows network access to his or her files; and

3) you know the other user's workstation name (SysName) and the partition, directory, and filename for the desired file.

If the above conditions are met,you can use any command that you would use to access files on your own workstation. For example, to copy from SysName Smith the file Status.Txt and to name it Status-Smith.Txt in the current directory on your workstation, you would type

```
copy /smith/reports/status.txt status-smith.txt
```

9.3 Access Privileges

A file is owned by the user on whose directory it resides. The owner determines read privileges (permission to look at the file) and write privileges (permission to change the file). Read privileges include typing, copying, and appending (but not appending to the file). Write privileges include editing and deleting. These privileges are set separately for other users and yourself (for example, you might want to prevent yourself from accidentally deleting a very important file).

The command to set access privileges is

setprotect <directory or filename>

followed by one or more non-conflicting switches:

-OwnerRead	(default)
-OwnerWrite	(default)
-WorldRead	(default)
-WorldNoWrite	(default)
-OwnerNoRead	
-OwnerNoWrite	
-WorldNoRead	
-WorldWrite	

You may use wildcards in the directory or filename.

The access privileges you set for a directory apply to all the sub-directories and files in that directory. For example, suppose in the User partition you have several nested directories under the directory Reports and the path to one of the files is as follows:

/sys/user/reports/prod/inv

Access privileges are checked at every step in the path, so the access privileges set for the Reports directory would apply to the Prod directory and the Inv file, as well as all the other files or directories nested under the Reports directory.

To find out what access privileges are in effect for any directory, give the directory command with the -All or -Protect switch (direct -all or direct -protect).

In order for access privileges to work on your network, your system administrator must set up an authorization server. If there is no authorization server at your site or if it's down, you will have complete access to your own files but not other users' files.

10. Handling Problems

Following is a list of problems that you may encounter. If you need help in resolving these problems, see your system administrator or call your PERQ Systems representative.

System won't boot

Report the number displayed on the Diagnostic Display (see Section 1).

Kernel debugger is invoked

Occasionally when you are using a program an exception will be encountered, and the process manager will invoke the kernel debugger. A message from the debugger will appear in reverse video in the upper left corner of the screen. Report the problem to your system administrator and then reboot your workstation. (The kernel debugger is for use by system programmers in controlling, observing, and modifying a process interactively, and it can be invoked intentionally by pressing all three mouse buttons, CTRL, and INS(ESC) simultaneously.)

The kernel debugger is automatically invoked by the Bye command and in this instance does not indicate a problem.

Note that the kernel debugger is not the same as the Kraut debugger discussed below.

Kraut debugger is invoked

If a program encounters a fault, it will invoke the Kraut debugger. A message will appear from Kraut will appear in reverse video in the upper left corner of the screen (just as it does for the kernel debugger), and you will be prompted to open a new window for Kraut. If you are not a programmer, cancel the request for the window and exit the program by pressing the DEL key. Report the problem to your system administrator. If you are a programmer and wish to use the debugger, see the document "Kraut: PERQ Pascal Remote Symbolic Debugger" in the Accent Programming Manual.

System crashes

When this happens the kernel debugger types out a message in white characters on a black background. The message consists of a number and perhaps a short explanation of the problem. Make a note of what the message says in case you must get

help.

The first thing to try is typing g. If that doesn't work, try rebooting the system (but see the precautions below). If neither works, report the problem.

If the message says "Running out of paging space" this means that the paging partition used by Accent to support the virtual memory is exhausted. This condition may be due to having allocated too little space to the partition or to having used the system for many hours of work during which many process and file pages have been "paged out." Usually you will be able to solve the problem by rebooting.

You should not reboot while a disk input/output operation is in progress, and therefore you should observe the following precautions when rebooting:

> If the shell is still capable of prompting for and executing commands, type bye. This invokes the Accent kernel debugger. When the debugger prompts with the ">" character, it is safe to reboot.
>
> If the shell will not accept commands, wait about a minute to insure that all disk activity has stopped (when activity has stopped, no lights on the right-hand side at the top of the screen will be flickering). Then reboot.

Program won't run

Sometimes when you are trying to run a program, you may get an error message of the following form (in this example, the user was trying to execute the program CALC):

```
ALoad:  Incorrect length for run file
     /sys/Accent/LibPascalInit.RUN
Shell:  Error in spawning /sys/user/smith/CALC.Run
     Code is 124
```

This error occurs when the program you are trying to run is not linked with the current LibPascalInit.Run file (a system program normally linked with most Pascal programs). The most probable cause is that you got a new version of the operating system and did not relink programs that you had written or copied from other users.

If the program is one you wrote and therefore you have the .SEG files (object code), issue the same Link command that you used the first time you linked the program.

If you do not have the .SEG files, link the .RUN file. Continuing with the above example, the command would be:

link -relink calc

(See details on the Link command in the document "User Facilities" in this manual.)

System suggests running Scavenger

If the file system runs out of free space to store files in the partition in which you are working, the system will suggest that you run the Scavenger program on the partition. You should do this immediately; see details on Scavenger in the document "User Facilities" in this manual.

USER'S GUIDE

TO THE WINDOW MANAGER

June 8, 1984

2600 Liberty Avenue
P. O. Box 2600
Pittsburgh, PA 15230
(412) 355-0900

Accent is a trademark of Carnegie-Mellon University.

Accent and many of its subsystems and support programs were originally developed by the CMU Computer Science Department as part of its Spice Project.

The major part of the design and most of the implementation of the window manager was done by Brad Myers of PERQ Systems Corporation. The design grew out of his discussions with many people, including Gene Ball, the window designers at International Computers Limited, and various interested parties at CMU and PERQ Systems Corporation. Amy Butler and Dave Golub of PERQ Systems were instrumental in completing the window manager's implementation.

The information in this document is subject to change without notice and should not be construed as a commitment by PERQ Systems Corporation. The company assumes no responsibility for any errors that may appear in this document.

PERQ Systems Corporation will make every effort to keep customers apprised of all documentation changes as quickly as possible. The Reader's Comments card is distributed with this document to request users' critical evaluation to assist us in preparing future documentation.

TABLE OF CONTENTS

1. Introduction

The window manager is the program that allows you to divide the screen into different areas, called windows, and to monitor and control the processes running in the windows. This document describes from a user's point of view how the window manager operates with the Shell. Section 10 of this document is a glossary of terms relevant to the window manager, and the last page of this document is a summary of the commands.

The details of window management as used within a particular application program will be covered in the document for any such program. A separate document ("The Window Manager" in the Accent Programming Manual) describes how programmers can incorporate the window manager into application programs.

1.1 Key Concepts

For readers who are not familiar with window managers, this section gives background information about window managers in general and the Accent window manager in particular.

1.1.1 Windows

If one were working on a number of different pieces of paper at a desk, one would typically separate the pages physically by having them at different places on the desk. Similarly, a window manager separates a number of different computer tasks by assiging each a separate area of the computer screen. Each area of the screen is called a window. The PERQ, like most personal work stations that support multiple processes, allows the user to assign the different processes to different windows in order to keep track of them separately.

In general, the window manager supports what is known as the "Covered Window Paradigm," a method for presenting multiple windows on the screen at the same time and allowing the various windows to overlap (see Figure 1). The windows can be thought of as pieces of paper. Windows can be moved around on the screen, just as papers can be moved around on a desk. One window can be brought on top of another, thereby partially or wholly covering that window. A window that cannot be entirely seen because other windows are on top of it is said to be covered. A window which is covered by all other windows occupying the same area of the screen is on the bottom. A window can extend off-screen in any direction (somewhat like papers hanging off the edge of a desk). The graphics and text in one window do not affect the

graphics and text in other windows. A program can be running in a window even if it is partially or wholly covered by other windows or it is off-screen.

PERQs are equipped with a tablet and a three-button or four-button mouse. When the mouse is moved around on the tablet, the computer echoes the mouse's position by displaying a cursor on the screen whic follows the mouse movement. Commands can be given to the window manager in three ways: with the mouse, from a command menu, or from the keyboard.

1.1.2 Listener

While there is only one keyboard and mouse on a machine, the screen c contain many windows. Therefore, one window has to be identified as the window to which keyboard operations are currently directed. This window is called the Listener because it is the window that is currently listening to the keyboard.

1.1.3 Icons

To allow you to monitor and control the programs that are running in windows (uncovered, covered, or off-screen), the window manager provides icons (small pictures which give information about each window, regardless of its position). The icons tell you whether a program is still running, is waiting for input from you, or is finished, and whether the window is off-screen. All of the window manager's icons are collected together in one window, the Icon Window (usually at the bottom of a portrait screen or at the side of a landscape screen), so that all of the icons can be manipulated as a group (for example, if you want to cover them with another window or move them off-screen).

The Process Manager Window (which the system places at the top of the screen when you enter Accent) and the Icon Window can be treated like any other window.

Figure 1 shows five user's windows plus the Icon Window. In the title line of each window is arbitrary text set by the program being run in each. The first window is in the Shell and the previous program run there exited with an error (the icon Shell2 with the bug). DirTree is executing in the second window and is showing random progress in the title line (below the text) and in the icon. The third window, like the first window, is in the Shell, and the last program run there als exited with an error. This shell is waiting for user input (the

keyboard). The Editor window has a gray border to show that it is currently the Listener. Its icon is similarly marked. It is trying to get the user's attention by using the exclamation point picture in the icon. The next window is off-screen (shown by the three dots at the bottom of the icon). The compiler is running in this window. The top progress bar shows random progress and the bottom progress bar shows that it is part of a command file that is about 25% done. The final window is also compiling. It is working on a program which starts with "WinM" and random progress is shown in both the icon and the title line for the window.

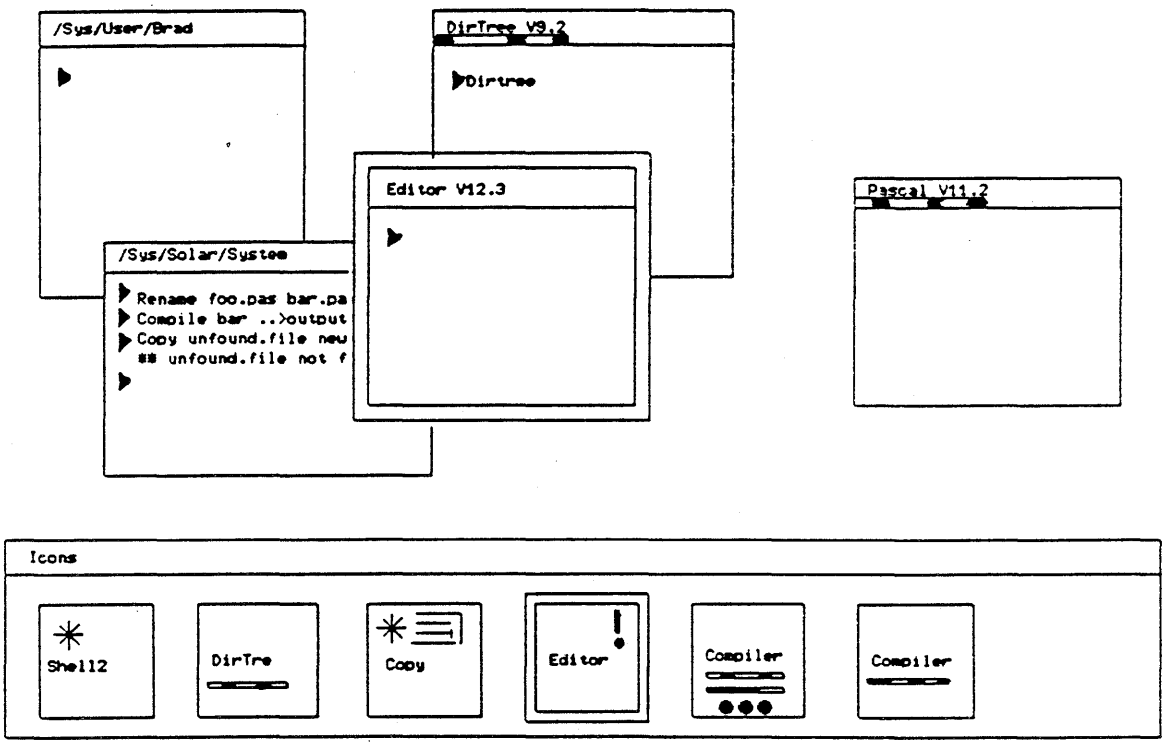


FIGURE 1. SAMPLE WINDOWS

1.1.4 Cursors

The window manager uses a number of cursors, each of which is associated with a function. The cursor pictures and their functions a shown in Figure 2. When you give the window manager a command, the cursor changes to the appropriate picture or to the window manager default cursor. If you are issuing a command using the mouse, the cursor picture appears when you press the button and stays until you release the button. When you release the button, the command is executed. The cursor pictures indicate their functions so that you c tell at a glance whether you have given the correct command. If you see that you have given the wrong command, just move the mouse until you see the "Cancel" cursor before you release the button.

When the user presses down on a mouse button while the cursor is in t title line, border, or icon of a window, the cursor will change to on of the window manager's cursors to indicate what will happen next. T command is executed when the button is released. If the operation requires multiple presses (for example, growing a window), the cursor picture will change at each step to show what is expected next. The cursors are shown on the following page.

1. The Accent system default cursor.
2. The window manager default cursor(waiting for input)
3. Make the window in which the cursor is located the Listener.
4. Bring the window to the top.
5. Send the window to the bottom.
6. From the icon, make the window on-screen, on-top, and Listener.
7. Get a popup menu from a window.
8. Make the window smaller or take it off-screen (the < command).
9. Bring the window on-screen (icon only) or make it full-screen (the > command).
10. Pick a corner or side from which to move or resize the window.
11. Resize the window.
12. Move the window.
13. Cancel command.
14. Specify the top left corner of the window.
15. Specify the top side of the window.
16. Specify the top right corner of the window.
17. Specify the right side of the window.
18. Specify the bottom right corner of the window.
19. Specify the bottom side of the window.
20. Specify the bottom left corner of the window.
21. Specify the left side of the window.
22. Identify the window for this icon.
23. Get a popup menu from an icon.

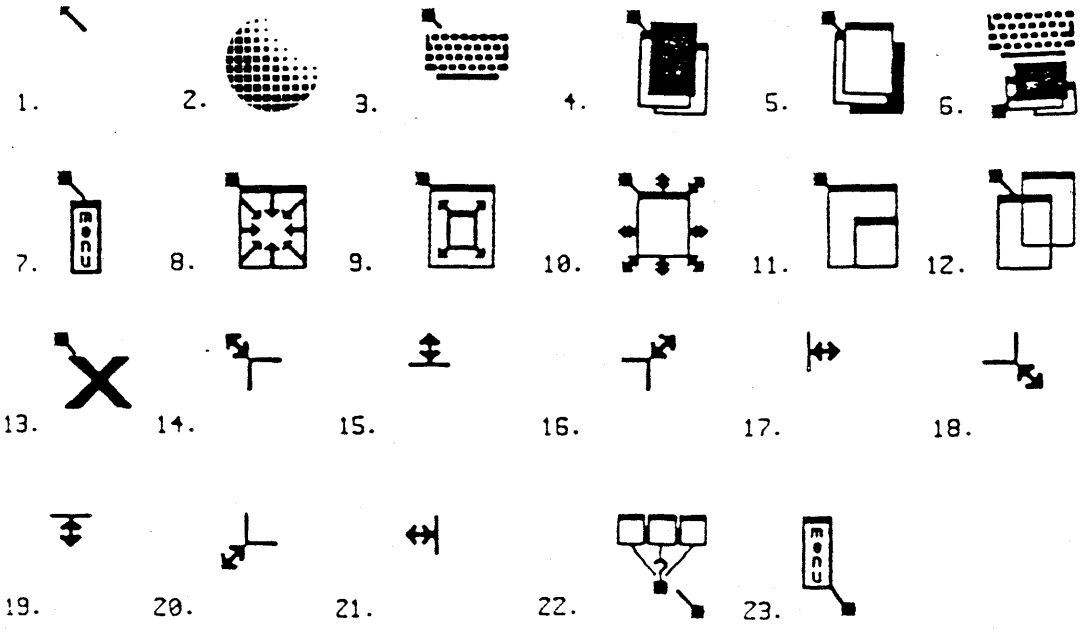


FIGURE 2: THE WINDOW MANAGER CURSORS

1.2 Window Features

Each window is rectangular and may extend off the screen in any direction. A window can be moved; changed in size; brought to the to or sent to the bottom of the stack of windows occupying the same area made full screen; sent off-screen; or restored to its original position. For each window its title line, borders, and icon are special areas. The window manager commands are given with the mouse positioning the arrow cursor (cursor #1 in Figure 2) in one of those areas, as explained in detail in Sections 2 through 10.

Figure 1 shows a screen containing several windows. The following sections give more information about title lines, borders, progress bars, and icons.

An application program may give windows a number of special propertie For example, the windows may not have borders, title lines, or icons, they can be restricted from moving or growing, they can be transparen so that the picture underneath shows through, and they can remember t picture underneath so that the area reappears later (as happens with popup menus in the Shell).

1.2.1 Title line and borders

The title line, located across the top of each window, is black with white letters. It serves three functions. First, it displays a line arbitrary text specified by the Shell or by an application. Second, i contains a progress bar, explained in the next section. Third, it can be used in conjunction with the mouse to perform operations on the window (by locating the cursor within a certain portion of the title line and pressing a mouse button, as explained in Section 1.3.1).

The border goes entirely around the window (outside of the title line The Listener window is identified with a special border around the outside of the window (a gray area between two hairlines), both in th window itself and in its icon.

1.2.2 Progress bars

A progress bar in the title line of each window shows how much of an operation has been completed. The progress bar appears on a line belo the text string and fills in (in reverse video) from left to right as the process runs. In some cases the application cannot determine how complete an operation is, but nevertheless it is helpful to the user show "random progress" (an indication that the operation is still bei

processed). In such cases random progress is shown by displaying blinking lines in the progress bar.

The progress bar is repeated in the icon so that the user can determine the progress of operations in windows that are off-screen or covered. In the icon only, there is a second progress bar that is used for command files so that you can tell how much of a command file has been processed.

1.2.3 Icons

An icon is a small picture that represents a window (see Figure 1). Icons contain information about the status of the corresponding window and about the process running in the window. They permit the user to easily monitor and control multiple processes and windows.

All the icons are grouped together in one window, the Icon Window. The title line of the Icon Window can be used to issue commands to control the icons as a group. (Button presses inside the title line of the Icon Window mean the same thing as for any other window.) Button presses inside an icon, however, are interpreted as an operation on the window associated with that icon.

The Icon Window can be covered or moved off-screen like any other window. There is no icon for the Icon Window itself.

The order of icons within the Icon Window is random. A new icon goes in the first available space. When a window is deleted, the associated icon goes away but the rest of the icons do not move. The user can compact the icons to remove these spaces if desired (see section 9.3).

The user is free to move or grow the Icon Window. The icons will be automatically organized for the new size or shape. The default position for the icons is one row across the bottom of the screen for Portrait screens (which hold 11 icons) and down the right side of the screen for Landscape screens (which hold 15 icons). If the Icon Window is full and another icon is needed, the Icon Window automatically expands along its shortest dimension and moves so that all the icons are on the screen.

Each icon is divided into a number of sections. The top of the icon can contain three pictures, used to show the state of the process running in the window. On the left will be a "bug" when an error has been reported in the window. When there is no error, the bug will be absent. In the center is a "keyboard" when the window is waiting for input from the user. On the right is an exclamation point which is for application's use. For example, a Mail program might bring up the

exclamation point (called the "attention" picture) when new mail come in.

Below the pictures is a six-letter text string used to identify the window. This will usually be the first six letters of the text name the program being run in the window. The name that appears in the icc is always guaranteed to be unique with respect to all other icons. If an application specifies a name that conflicts with another window's name, a number is appended to the end. If the name is already six characters long, the last character of the name is changed to a numbe Thus, if "foo" exists, another foo would become "foo2". If "Pascal" existed, another Pascal would become "Pasca2". The Shell will set th name to the program being run, but some programs may explicitly set i to some other name.

Below the name are two progress bars. The first bar operates like th progress bar in the title line of the window itself. The second bar shows progress in command files. Below the progress bars is an area that contains three big dots (an ellipsis) if the window has been sen off-screen.

The border of each icon changes the same way as the associated window border to show whether the window is the Listener or not. Thus it wi be either a hairline if not the listener or two hairlines surrounding gray area if it is.

1.3 Commands - General Information

There are three ways to issue commands: with the mouse, with the command menu, and with the keyboard. For most operations you have a choice between at least two of these methods.

The following sections give some general instructions for issuing commands in each of these three ways. The specific commands are given in sections 2 through 10, organized by function, and they are summarized on the last page of this document.

If you receive an error message from the window manager, please report it to your system administrator or PERQ Systems. The error messages are listed in the document "The Window Manager" in the Accent Programming Manual.

1.3.1 Mouse commands

Using the mouse to issue commands to the window manager involves thre steps:

1) positioning the cursor in the appropriate area,

2) pressing the appropriate button (after which a cursor appears, depicting the action you've chosen), and

3) releasing the button (after which the action is completed).

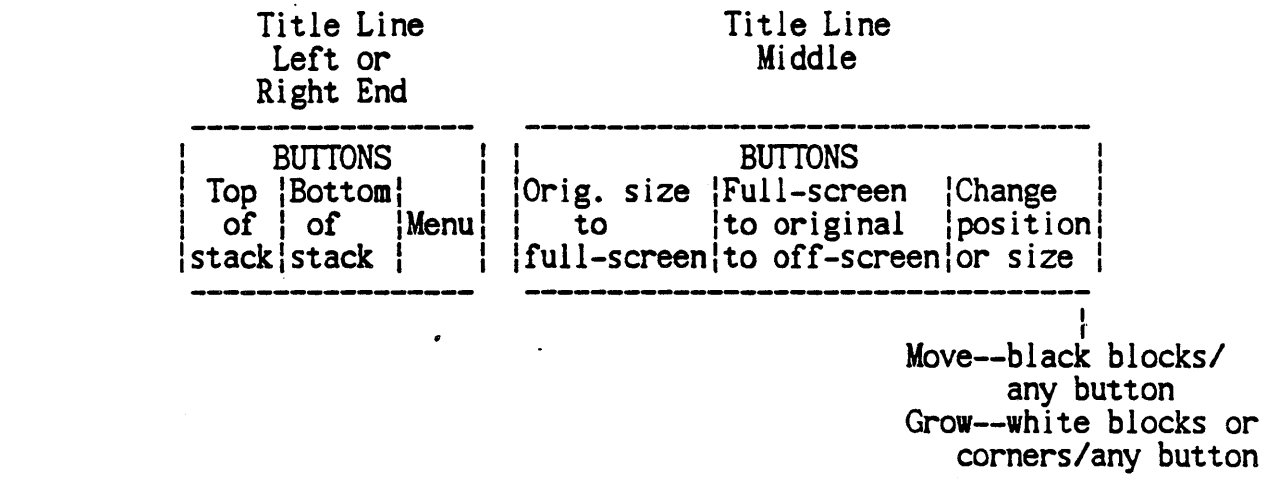
The window manager is written for a three-button mouse. If you have four-button mouse, the white button corresponds to the left button, t yellow to the middle button, and the green to the right button; the blue button serves no function.

A cursor in the form of an arrow echoes on the screen the position of the mouse on the bitpad. To invoke the window manager commands using the mouse, you must first position the arrow cursor on either the tit line or the icon of the window you wish to affect (it does not have t be the Listener). Be careful not to press a button within the Listener, as all button presses within the Listener are interpreted b the application program running in that window.

The title line of each window is divided into three sections: left, middle, and right. Since often only one side of a window is visible, the left and right sections of the title line serve the same function Using the three buttons on the mouse and the two areas of the title line, there are six functions that can be performed from the title li (discussed in detail in the section shown below):

send to top of stack of windows (Section 6.1)
send to bottom of stack of windows (Section 6.1)
show the command menu (Section 5)
switch from original size to full-screen (Section 6.3)
switch from full-screen to original size to
off-screen (Section 6.4)
change position or size (Section 6.5)

The cursor positions and the buttons for these commands are as follows:



After you position the arrow cursor in the desired area and press the appropriate button for the command you wish to give, the cursor will be changed to a picture that represents the action that will be completed when the button is released. Figure 2 shows the cursors used in the window manager. Release the button to complete the action.

When you are issuing a command to the window manager, you cannot press a second button before you release the first button. This is because the window manager treats the pressing and releasing of a button as one mouse event--pressing brings up the appropriate cursor and releasing executes the command. If the first button pressed is not the same as the last button released, the command is ignored. (This may differ in applications, because presses and releases are treated as separate events in applications.)

If after you press a button and obtain a cursor you realize that you have picked the wrong area or wrong button, you can correct the action before you release the button. Move the cursor across the title line to switch to another function (the cursor picture will change) or move the cursor outside the title line to abort the action.

Pressing a button within an icon affects the window represented by th icon. There are three functions that can be performed from an icon:

make the window on-screen, on-top, and Listener
in one operation
identify the window represented by the icon
show the command menu

The buttons used to issue these commands are as follows:

BUTTONS		
On-screen, on-top, & Listener	Identify window	Menu

1.3.2 Menu Commands

The command menu contains nearly all of the commands (there are a few commands that can be issued only from the keyboard). Section 5 explains how to obtain and use a menu.

You can request a menu from any window or its icon, whether or not it the Listener. If the command applies to one regular window only (rather than the Icon Window or all windows), the command will be executed in the window from which you request the menu.

1.3.3 Keyboard Commands

All operations that can be performed with the mouse or from the menu, plus a few others, can be performed with keyboard commands. Keyboard commands affect only the Listener (except for a few commands that affect only the Icon Window or that affect all windows).

All keyboard commands must be preceded with a prefix of CTRL-DEL on the PERQ and with CTRL-DEL or SETUP on the PERQ2. (To use CTRL-DEL press and hold the CTRL key while you press the DEL key.) Release the prefix key before you type the command. When the window manager prefix key is pressed, the window manager suspends any process that is running in the Listener and changes the cursor shape to a window manager cursor. The next character that is typed will be interpreted as a window manager command. The commands can be typed in lower or upper case. Typing another character returns control to the process in the Listener; therefore if you want to give more than one command you must repeat the prefix (except for the + and - commands explained in section 3).

If some process has the mouse trapped or turned off or otherwise disabled, you can resume giving mouse commands by first giving a keyboard command (which will release the mouse).

2. Creating Windows

When you enter Accent, a window is created automatically for the Process Manager. Another window is created which runs your initial shell command file.

Additional windows can be opened in one of the following ways:

A) type <prefix>s; or

B) select "shell" on the command menu.

After you do either of the above, you must indicate the upper left an lower right corner as follows. After the appearance of a blinking cursor in the form of an upper left corner with an arrow (cursor #14 Figure 2), position the cursor where you want the upper left corner o the window and press and release any button. The cursor will change a lower right corner with an arrow (cursor #18). Move the cursor to where you want the lower right corner (a hairline box showing the outline of the window will follow the cursor) and press and release a button.

The window manager will not permit you to create a window that is smaller than a minimum size (which is very small) specified by the window manager.

If after you open a window you find that you need to make it bigger, obtain the menu (from the title line or the icon) and select the reshape command explained in Section 6.6.

After you create a window, you cannot use the window until you make i the Listener.

3. Designating the Listener

As explained in the Introduction, the Listener is the window that is receiving commands from the keyboard. The window that is currently the Listener has a narrow gray band around the border.

There are several ways to designate a window as the current Listener:

A) Move the cursor into the desired window and press any button. A picture of a keyboard (cursor #3 in Figure 2) appears. When the button is released, the border changes to show that the newly selected window is now the Listener. (Moving the cursor without pressing does not change the Listener, nor does moving the cursor outside a window while pressing.)

B) Either select "listener menu" on the command menu or type <prefix>l. After either action, a menu appears which lists all of the windows; select the window you wish to make the Listener.

C) To bring a window back if it is off-screen, bring it to the top if it is covered, and make it the Listener all in one operation, press the left button while in the icon (cursor #6 in Figure 2 appears).

D) Type <prefix>+ or <prefix>- to move around a ring of windows. (The window manager remembers which windows have been the Listener in what order and keeps their names in a circular buffer.) To go back to the window that was the most recent Listener before the current Listener, move back by typing <prefix>-. To go to the window that was the least recent Listener (or the most recently created window that has never been the Listener), move forward by typing <prefix>+.

   These two characters, - and +, are easy to remember in conjunction with these functions, but either the upper- or lower-case characters of each key will work (_ or - to move back and + or = to go forward).

   Whether moving forward or backward the window manager default cursor (#2 in Figure 2) appears. The ring is re-arranged each time you change Listener. You can give multiple +'s and -'s without repeating the prefix key. After you have used the + or - command, you can leave the window manager and return to an application program by typing <prefix>x (for "exit window manager"). If you want to go back to the window that was Listener before you gave the + or - command and resume the application program, type any illegal command such as the BACKSPACE key.

4. Obtaining Help

On-line help for the window manager can be obtained in any of the following ways:

A) select "help" on the command menu;

B) press the <prefix>HELP keys; or

C) type <prefix>h.

A small window in reverse video appears in the center of the screen. contains a summary of the mouse and keyboard commands. To remove the help window and restore the portion of the screen that it covers, typ any character.

5. Obtaining the Command Menu

The command menu can be obtained in any of the following ways:

A) press the right button while on the left or right section of the title line (cursor #7 in Figure 2 appears);

B) press the right button while in the icon (cursor #23 in Figure 2 appears); or

C) type <prefix>? (Listener only).

When you release the button, the menu appears. Commands on the menu can then be selected by moving the cursor to the desired command and pressing and releasing any button. The command is executed on the window from which the menu was requested. To abort the menu without making a selection, press outside the menu area.

Nearly all of the commands that are available in the window manager are listed on the menu (see the summary on the last page of this document).

6. Moving and Changing the Size of Windows

6.1 Top or Bottom of Stack

If a window is on top of the stack, it is not covered by any other windows. If a window is on bottom, it is covered partially or wholly by any other windows that occupy the same area of the screen.

A window can be brought to the top or sent to the bottom in any of th following ways:

A) position the cursor in the left or right section of the title line. Press and release the left button to bring the window to the top or the middle button to send it to the bottom (cursor #4 or #5 appears);

B) on the command menu select either "top" or "bottom," or

C) type either <prefix>t for "top" or <prefix>b for "bottom" (Listener only).

If a screen is on the bottom and is not accessible, you can move it t the top by requesting the command menu from its icon.

Sending a window to the bottom is useful for finding the other window that are covered by that window.

6.2 On-screen, On-top, and Listener

There is a special command which can be used to bring a window on-screen, put it on top of the stack, and make it the Listener in on operation. Press and release the left button while in the icon of th desired window (cursor #6 appears).

6.3 From Off-screen to Original to Full-screen

There is a special command which takes a window up through a hierarchy, from off-screen to full-screen. If a window is off-screen when you issue the command, the window will be returned to its original position on the screen. If the window is on-screen, it is made full-screen. (Thus, to bring a window from off-screen to full-screen requires issuing the command twice.) This operation can be performed in any of the following ways:

A) if the window is on-screen and you want to make it full-screen, press and release the left button while on the middle section of the title line (cursor #9 appears); or

B) type <prefix> > (Listener only);

C) if the window is off-screen, from its icon select ">Bigger" on the command menu.

When a window is made full-screen, its old position is remembered so that it can be returned to its original position in a single "go back" operation (explained in section 6.4 below). In the Shell, "full-screen" uses the entire window outside the Icon Window, but maintains the Icon Window. This is useful for making the window for the current process (for example, the editor) large while monitoring other processes through their icons. (Application programs may define "full-screen" differently.)

The size of the window is saved each time the full-screen operation is executed. Therefore if you specify full-screen and then change the size of the window, the "go back" command will return the window to the original size (before the full-screen command). However, if you specify full-screen, change size, then specify full-screen again, the intermediate size will be used for the "go back" command.

6.4 From Full-screen to Original to Off-screen

A special "go back" command takes a window down through a hierarchy from full-screen to off-screen. If the window is full-screen when yc issue the command, the window is returned to its original size (see section 6.3 for a discussion of what size is considered to be the "original" size). If the window is not full-screen, it is sent off-screen. This operation can be performed in any of the following ways:

A) press and release the middle button while on the middle section of the title line (cursor #8 appears);

B) select "<Smaller" on the command menu; or

C) type <prefix> < (Listener only).

Three dots in the icon indicate that the window is off-screen.

Sending a window off-screen is useful for two reasons. First, it allo the screen to be less cluttered since off-screen windows will not sho up when other windows are moved around or sent to the bottom. Second sending a window off-screen makes the refresh calculations for all other windows more efficient.

A window that has been taken off-screen can be brought back to its original on-screen position by using its icon to request the command menu and choosing ">Bigger."

You should use one of the methods shown above to take a window off-screen rather than just moving it off-screen with the move comman discussed in the next section. Otherwise, the window is included in the refresh operation even if it's off-screen and its on-screen position is not remembered.

6.5 Position or Size

Windows can be moved or "grown" (increased or decreased in size) in any direction. To perform either operation, perform the sequence described below.

1) Indicate in one of the following ways that you want to move or grow the screen:

   a) press and release the right button while on the middle section of the title line (cursor #10, size/move, appears);

   b) select "move window" or "grow window" on the command menu; or

   c) type <prefix>m for "move" or <prefix>g for "grow" (Listener only).

2) Indicate the direction in which you wish to move or grow the window by selecting a control point as follows:

   a) To move a window, position the cursor precisely on the black rectangle on either side and press any button. Cursor #12 appears and stays until you release the button; then a different cursor (#14 through #21) appears.

   b) To grow a window, position the cursor precisely on the white rectangle on the side or corner in the direction you wish to increase or decrease and press any button. Cursor #11 appears and stays until you release the button; then a different cursor (#14 through #21) appears.

3) Move the cursor to the desired location. As the cursor is moved, a hairline box is drawn to show the bounds of the window.

4) When the window reaches the desired location or size, press and release any button.

If you have trouble at first remembering which blocks to use, remember to look at the cursor which appears when you press a button. The command is not executed until you release the button, so if you picked the wrong block move the cursor before you release the button.

To abort the procedure at any time, press any character on the keyboard.

Windows can be moved partially off-screen in any direction if you wis

As an alternative to Move or Grow you may wish to use Reshape, explained in section 6.6 below.

6.6 Reshape

Reshape is a useful alternative to Move or Grow, as it allows you to specify entirely new corners for a window. To use Reshape, do the following:

1) Either select "reshape window" on the command menu or (Listener only) type <prefix>r. Cursor #14 appears.

2) Position the cursor at the new location for the upper left corner and press and release any button. Cursor #18 appears.

3) Repeat step 2 for the lower right corner. (As you move the cursor, a hairline box will show the new window size.)

To abort the process, press DEL; the original window will return.

7. Debugging or Stopping Processes

7.1 Debugging

To send a process into the debugger, select "debug process" on the command menu or (Listener only) type <prefix>d. You will be asked to open a window for the debugger.

7.2 Suspending a Process

To suspend a process, select "suspend process" on the command menu or (Listener only) type <prefix>z.

7.3 Resuming a Process

To resume a process that you had suspended, select "resume process" on the command menu or (Listener only) type <prefix>q (for "quit window manager").

7.4 Cancelling a Process

To cancel (abort) a process, select "cancel process" on the command menu or (Listener only) type <prefix>c.

7.5 Killing a Process

Killing a process aborts the process as well as any command files, and the Shell returns to the top level, waiting for user-input. To kill a process, select "kill process" on the command menu or (Listener only) type <prefix>k.

If you wish to also delete the window, see section 8.

7.6 Quitting the Window Manager and Returning to the Application

To leave the window manager and go back to the application program, type <prefix>x (for "exit window manager"). This command should be given after moving in the Listener ring with the + or - commands.

8. Deleting Windows

To kill a process and delete the window in which it was running, select "eradicate window" on the command menu or (Listener only) type <prefix>e. This kills all processes running in that window, including the Shell.

9. Working with Icons

9.1 Bringing the Icon Window On-screen, On-top

To bring the Icon Window back if it is off-screen and to bring it to the top of the stack in one operation, select "icon window" on the command menu or type <prefix>i. The default window manager cursor (# appears. You can execute this command from any window.

9.2 Identifying the Window Represented by an Icon

To identify which window goes with an icon or vice versa, do one of t following:

A) from the icon, press and release the middle button (cursor #22 appears);

B) from the window, select "find icon" on the menu; or

C) from the window (Listener only), type <prefix>f.

The icon and its matching window are video-inverted, and lines are drawn from the four corners of the icon to the four corners of the window. The lines are blinked on and off a number of times. If the window is covered, the screen area where it would be if it were on to is shown. If the window has been sent off-screen using the off-scree command (see section 6.4), then the icon simply blinks and no lines a drawn. If the window has been moved partially off-screen, however, lines are drawn toward the ends of the window.

9.3 Reorganizing the Icons

When a window is deleted, its icon is erased but none of the other icons move. To reorganize (compact) the Icon Window to remove the blank spots and to redraw the window, select "organize icons" on the command menu or type <prefix>o. The default window manager cursor (# appears. This command can be executed from any window.

9.4 Monitoring the Icons

The following symbols are used in the icon to tell you what is happening in the corresponding window:

keyboard	program is waiting for user input
bug	error encountered
exclamation point	window needs attention (symbol may be used differently in application programs)
6-letter text string	window name (a unique identification)
progress bar 1	progress in program (proportion completed or random progress)
progress bar 2	progress in command file (proportion completed or random progress)
3 dots	window is off-screen

10. Glossary

The following definitions are given for your information. Several of the items (noted below) are not apparent to the user.

Accent: a multi-tasked, message-based operating system for the PERQ. The window manager described in this document runs only under Accent.

Application: any program that uses the window manager.

Application Interface: the set of messages or procedures and exceptio that are provided to the application by the window manager. The windo manager allows applications to display text and graphics in windows a to create, manipulate and delete windows. The interface is not appare to the user.

Covered Window Paradigm: a method for presenting multiple windows on the screen at the same time and allowing the various windows to overlap. When two windows overlap, one is said to be on top when it can be seen in its entirety. The other window is said to be covered since it is at least partially obscured by the first window. With mo than two overlapping windows, there is an ordering of windows from to to bottom where the top one is fully uncovered, and the bottom one is "underneath" all other windows.

Cursor: a picture that follows a pointing device. In the window manager the cursors are pictures that indicate the action the user ha chosen.

Icon: a small picture that represents a window. Each icon will be associated with a particular window and will contain information abou the status of that window and the process running in it.

Listener: the process that is currently receiving keyboard input. In multi-process multi-window system, the keyboard needs to be multiplex over the processes and windows. Only one process at a time may recei the keyboard characters. This is the Listener process. "Listener" i also used to refer to the window containing the Listener process.

Mouse: a device which the user moves on a tablet in order to move a cursor on the screen. It may have one, three or four buttons (used t issue commands), with three being the most common number. The user interface to the window manager uses only three buttons. These map t the four colored buttons on a Summagraphic mouse as follows: Left = White, Middle = Yellow, and Green = Right. The user interface to the window manager does not use the blue button; however, application

programs using the window manager may use it. (A mouse, especially if it has four buttons, is often referred to as a puck.)

Output: as used in this document, all graphical and textual output to the screen. Included are all RasterOps to the screen (including display of text), line drawings, pictures, etc. This also includes the characters echoed when typing.

RasterOp: a low-level PERQ function that operates on rectangles. The user does not need to be concerned with this.

Shell: a program or process that is in charge of running other programs. The shell takes requests from a user or a command file and executes the programs represented by those requests.

Tablet: a device attached to a computer and used with a mouse to move a cursor on the screen (therefore a tablet is referred to as a pointing device).

Typescript: normal text input to the system from the keyboard. The typescript is linear and can be thought of as a "glass teletype" or the typing that would appear on a printing terminal. Thus, it excludes typing to screen editors and other application programs that interpret individual characters as commands. Most user typing will be handled by the typescript package.

User: a human who is sitting in front of the PERQ typing on the keyboard and using the mouse.

User Interface: the part of the window manager that the user sees and uses directly. This includes the pictures displayed on the screen, the operations the user performs (using the keyboard and mouse) to affect the windows, and the responses of the window manager to those actions.

Viewport: the basis for the implementation of windows in the window manager. There is one or more viewports for each window. There may be viewports that do not correspond to windows if they are created by application programs. Such a viewport will not be controlled by the window manager. Since windows are implemented using viewports, sometimes the word "windows" is used in documentation where more accurately it should read "windows and viewports". However, the user does not need to be concerned with viewports.

Viewport Layer: the part of the window manager that implements the Viewport abstraction. The viewport layer also provides some simple graphical operations that allow the window manager (and application programs) to display text and pictures in viewports. The user does n need to be concerned with the different layers.

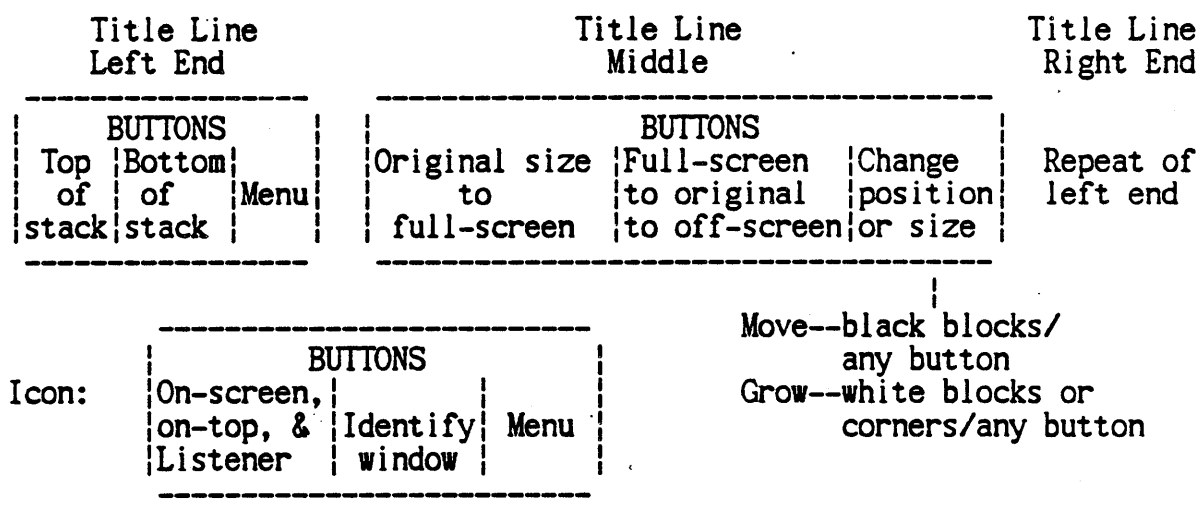
Window: any rectangular area of the screen that is controlled by the window manager. It may extend off the screen in any direction, or may even be off the screen entirely. Windows are clipped to the screen's rectangle (that is, the visible portion is cut off at the boundary).

Window Layer: the part of the window manager that supports windows. Windows are a separate layer on top of Viewports that implement the user-interface to the window manager. The window layer includes the procedures that implement the icons, the titles and borders for windows, and the entire mouse and keyboard interface to the window manager. Again, the user does not need to be concerned with the different layers.

Window Manager: a component of the Accent operating system that allc users and programs to create, modify and output text and graphics to windows. It may also be in charge of collecting and distributing inp from the user to the appropriate process. The Window Manager (at the user's request) changes which windows are covered and uncovered by moving the windows to the top and bottom.

SUMMARY OF THE WINDOW MANAGER COMMANDS

Mouse Commands



Menu Commands	Keyboard Commands*
Shell (create new window)	<prefix> s
Eradicate window (kill process and delete window)	<prefix> e
Help	HELP or <prefix> h
Listener menu (to designate new Listener)	<prefix> l
Bottom (send to bottom of stack)	<prefix> b
Top (send to top of stack)	<prefix> t
>Bigger (from off-screen to original to full-screen)	<prefix> >
<Smaller (from full-screen to original to off-screen)	<prefix> <
Grow window (using corners or white blocks)	<prefix> g
Move window (using black blocks)	<prefix> m
Reshape window (redefine upper left & lower right corners)	<prefix> r
Icon window (bring on-screen, on-top)	<prefix> i
Organize icons (compact the icons)	<prefix> o
Find icon (identify corresponding window)	<prefix> f
Debug process (send window to debugger)	<prefix> d
Cancel process	<prefix> c
Kill process (including command files)	<prefix> k
Suspend process	<prefix> z
Resume process	<prefix> q

Keyboard Commands Not Available from Menu	
Obtain menu	<prefix> ?
Forward in ring	<prefix> +
Backward in ring	<prefix> -
Exit window manager and return to application	<prefix> x

*Prefix is CTRL-DEL on the PERQ and CTRL-DEL or SETUP on the PERQ2.

USER FACILITIES

June 8, 1984

2600 Liberty Avenue
P. O. Box 2600
Pittsburgh, PA 15230
(412) 355-0900

Accent is a trademark of Carnegie-Mellon University.

Accent and many of its subsystems and support programs were originally developed by the CMU Computer Science Department as part of its Spice Project.

The information in this document is subject to change without notice and should not be construed as a commitment by PERQ Systems Corporation. The company assumes no responsibility for any errors that may appear in this document.

PERQ Systems Corporation will make every effort to keep customers apprised of all documentation changes as quickly as possible. The Reader's Comments card is distributed with this document to request users' critical evaluation to assist us in preparing future documentation.

PERQ, PERQ2, LINQ, and Qnix are trademarks of PERQ Systems Corporation.

TABLE OF CONTENTS

USER FACILITIES

1. Introduction

This document describes all of the facilities that are available to a user under Accent, including the commands that are implemented directly by the shell and the utilities (commands that are implemented as run files).

To issue a command, type a command line in response to the default Accent prompt (a right-pointing triangle). The document "Basic Operations" in this manual discusses the use of commands, including arguments, switches, syntax, popup menus, and user command files. It also discusses conventions of the file system, such as path names.

All numeric values required in a command are decimal values.

Section 2 lists the facilities by functional groups. In Section 3 the facilities are listed in alphabetical order and each is discussed in detail. Section 4 is a summary, which is repeated in Appendix A of this manual for your convenience as a quick reference.

2. Functional Groups

The user facilities are listed below according to function.

INITIALIZATION COMMANDS

Alias
ChangeUser
Define
Launch
Login
Path
SetSearch
Shell
Unalias
UserControl

PROGRAM DEVELOPMENT COMMANDS

Asm (assemble for C)
Compile (Pascal)
cpp (C)
Link (Pascal)
Lisp
Lnk (C)
Make
Matchmaker
Pascal
PasMac
pcc (C)
PRQMic
PRQPlace
QDis

INFORMATIONAL COMMANDS

Details
DirTree
Help
IconWallclock
Show
Statistics
Version
Wallclock

PROCESS CONTROL COMMANDS

Bye
Debug
Kill
Mace
Pause
Priority
Quit
Resume
Run
Suspend
Verbose

DEVICE MANAGEMENT COMMANDS

Dismount
Mount
Partition
Scavenger

COMMUNICATIVE COMMANDS

Chatter
FTP
Listen
On
Remote
Speak

FILE MANAGEMENT COMMANDS

Append
Compare
Copy
Delete
Direct
Edit
FindString
MakeDir
Patch
Print
Rename
Stut
TypeFile

SYSTEM MAINTENANCE COMMANDS

Bindboot
ExpandTabs
Floppy
KeyTranCom
SetBaud
SetProtect
SetTime

3. Facilities

Each facility is described in detail in the following pages. The facilities are listed in alphabetical order.

ALIAS

Alias assigns another name to an existing command. (This is the only safe way to do so.) You may also specify that a default filename be remembered for the command. An Alias Table is initialized each time a shell is created. When you type the new CommandName or an unambiguous abbreviation of it, the shell looks for the string in the Alias Table and substitutes the predefined value (the definition), including any default filenames. Aliases are also useful to define commands that override defaults by using switches.

Format:

ALIAS <CommandName> <Definition> {-switch} [Documentation]

If Alias is invoked without an argument, it types out a syntax reminder. The CommandName is any name you wish to assign. The Definition is a previously defined command or program name, including any arguments and switches that are valid for that command. If the Definition contains a nonalphanumeric character, including a blank, it must be quoted by it with a backslash (\). (Or you can enclose the Definition in quotes - '...'.) Do not abbreviate the command name you are using in the Definition.

Do not use a previous Alias as the Definition.

If you wish to use one of the Alias switches described below, it must come after the switches that are part of the Definition and before the Documentation.

Documentation is any information that you wish to give about the command. The information will appear with the command when you give the ? command. If the Documentation contains any nonalphanumeric characters, enclose the Documentation in quotes ('...').

After you have set an Alias, typing the CommandName is equivalent to typing the Definition. This substitution does not include RETURN, so when you type the CommandName you can add more arguments and switches and then press RETURN.

Switches are:

Switch	Description
-SETDEFAULT	The first argument you supply each time you invoke the new command will become the default filename. (This is useful because certain commands use default filenames when you don't supply a filename.) If you have not specifed an argument, no default filename is set.
-USEDEFAULT	If you type the new command without an argument, the default filename remembered by the shell will be used.
-HELP	Displays information about the Alias facility.

Examples:

```
alias ls direct\ \-sort\=size
```

This command defines a command "ls" which is equivalent to the command "direct" and specifies that the entries are to be sorted in order of size. Note the use of \ to quote non-alphanumeric characters, including spaces.

```
alias ls "direct -sort=size"
```

This is equivalent to the example above. Note that the quotes may be used to enclose an entire string of characters containing nonalphanumeric characters.

```
alias edit run\ editor -usedefault -setdefault
alias compile run\ pascal -usedefault -setdefault
```

This command allows

```
edit <file>
compile
```

or

```
compile <file>
edit
```

to work by implicitly using the same file in the second command.

alias help /sys/user/smith/myhelp.run

This command defines the help file as the MyHelp.Run file instead of the system help file.

You may redefine existing aliases by issuing the Alias command again with new information (but do not use a previous Alias as the Definition).

To remove an Alias entry, use the Unalias facility.

APPEND

Append copies one or more files to the end of an existing file. The command accepts a list of input files, separated by either commas or spaces, and appends each file in the order listed to the end of the first file.

Format:

```
APPEND [file][,file2][,...filen] [-Help]
```

Append prompts for any missing arguments. The -Help switch displays a description of the Append command.

The Append command alters only the first file you specify. For example, the command:

```
APPEND file1,file2
```

appends file2 to the end of file1, but file2 is retained as a separate entity.

The append operation is successive. For example, the command line:

```
APPEND file1,file2,file3
```

first appends file2 to the end of file1 and then appends file3 to the end of combined file1file2.

ASM

Asm (Assembly) enables programs written in either the Assembly language or the C language to be run on the PERQ workstation. The C assembler is an optional part of the operating system and is documented in the Accent Languages Manual.

Format:

```
ASM -O -o <filename.o> <filename.s>
```

where

-O	is the optimize switch.
-o	is the output switch.
<filename.o>	is the name of the file that is to contain the object code.
<filename.s>	is the name of the input file that contains the assembly language code.

BINDBOOT

Bindboot attaches a boot file to a boot character. A boot character is a lower-case letter on the keyboard which brings up a particular operating system; the boot key is pressed after the workstation is turned on and a black pattern appears briefly on the screen.

There are two types of boot files. The system file (standard extension .Boot) contains the operating system code. The Interpreter file (standard extension .MBoot) contains the microcode for the workstation. The system file is constant across various models of the PERQ workstation, but there is a specific Interpreter for each type of workstation.

Format:

BINDBOOT {-switch}

Switches are:

Switch	Description
-SYSTEM=<FILENAME>	Uses <filename> as the system boot file
-INTERPRETER=<FILENAME>	Uses <filename> as the interpreter boot file.
-NOSYSTEM	Does not write the system boot file.
-NOINTERPRETER	Does not write the interpreter boot file.
-BOOTCHARACTER=<Char>	Binds the boot to the specified character.

-HELP

Displays information about the Bindboot facility

If you do not supply switches, Bindboot will prompt for the required information.

To bind only the system boot file, supply the following command line:

```
bindboot -BootChar=<c> -System=<filename> -NoInterpreter
```

To bind only the interpreter boot file, supply the following command line:

```
bindboot -BootChar=<c> -Interpreter=<filename> -NoSystem
```

BYE

The Bye command logs you off the system.

Format:

BYE

The Bye command traps to the Accent kernel debugger (this may take a few seconds). After the appearance of a message from the debugger in the upper left corner of your screen, you may turn off your workstation. The debugger provides routine maintenance functions, and therefore you should always issue the bye command before turning off a workstation.

CHANGEUSER

ChangeUser changes your user name (the name you login under), password, shell, or profile. ChangeUser prompts for all inputs.

Format:

CHANGEUSER

To run the ChangeUser program, simply type:

Changeuser

The program then asks a series of questions about what you want to change. If you do not want to change a particular definition, just hit a carriage return after the prompt.

The only valid switch is -HELP.

The questions this program asks are as follows:

1. Current password:

   You enter your password.

2. Should I change your password ? :

   You enter y or n. If you enter a y the program will ask:

   ```
   New Password :
   Again :
   ```

   If you enter an n, the program will proceed to the next question.

3. Profile [Boot:Default.Profile] :

   You enter a new name only if you want to change your profile name.

   NOTE: Currently the Profile is not used by the system. All functions normally served by a profile are handled by ShellCommands.Cmd.

4. ShellName [Boot:Shell.s5.run]:

Once you have entered a new shell name or hit the return key, the program will type:

```
Login information changed.
```

CHATTER

The Chatter utility allows a PERQ workstation to act as a terminal on a host machine such as a VAX, using an RS232 line for communication. If your PERQ workstation is not connected to an RS232 line, you cannot use this utility. On a PERQ2 workstation the command supports either RS232 port (RSA or RSB).

Format to enter Chatter:

CHATTER

To issue a Chatter command after you have entered Chatter, type CTRL-r. Chatter then displays a menu at the top of its window. Enter the desired command, in uppercase, and press RETURN. You must type CTRL-r before every Chatter command. The following describes the valid Chatter commands:

Command	Description
D	(Device) This command selects the RS232 port (RSA or RSB) to use with a PERQ2 workstation.
S	(Save) This command writes into a file everything that comes from the remote computer. The command prompts you for a file name. In order to be able to access this file later, you must issue a Chatter Close command before exiting the utility.
T	(Transmit) This command transmits a file across the RS232 line, as if it were being typed at the keyboard.
C	(Close) This command closes the file you opened with the Save command.
B	(Baud) This command changes the RS232 baud rate. Valid baud rates are: 110, 150, 300, 600, 1200, 2400, 4800, 9600, and 19200. The default is 4800 baud.
Q	(Quit) This command exits Chatter and returns you to the Accent shell. It does not log off or disconnect the remote host or the PERQ workstation.

If while in Chatter the characters you type are not echoed properly, either the baud rate is incorrect or the RS232 cable is not properly connected.

Chatter cannot be used as a half-duplex terminal since it does not echo characters locally.

COMPARE

Compare looks for differences between two files and prints the results into an output file, or if no output file is specified, prints the results on the screen.

Format:

Compare <FilenameA> <FilenameB> ~ [OutputFile] {-Switch}

If input files are not specified, Compare will prompt for them.

Switches are:

Switch	Description
-DIFFERENCES	Lists the lines that do not match. FileB.
-UNEQUALCOLUMNS	Marks unequal columns.
-MATCH=<n>	The file is compared in chunks and this command specifies the number of lines to be compared in each chunk. You may specify a number from 1 to 100. Use of -match=1 would compare the files one line at a time. The default is 6.
-LINELENGTH=<n>	Compares up to this length of line. You may specify from 1 to 255. The default is 100 characters.
-HELP	Displays information about the Compare facility.

COMPILE (PASCAL)

The PERQ Pascal compiler translates Pascal source code into a .Seg file that you can link and run. There are several ways to invoke the compiler and several options that you can use with it. The Pascal compiler is an optional part of the operating system and is documented in the Accent Languages Manual.

Format:

COMPILE [inputfile] [~] [outputfile] {-switch}

You can also invoke the compiler by typing pascal instead of compile in the above line. The commands are the same with the exception that the Pascal command does not set default filenames.

The inputfile is the name of the source file to compile. If you do not specify an inputfile, the compiler uses the default file name remembered by the shell. If you want to compile a program immediately after editing it, you need not specify its name since the shell remembers the last file edited, compiled, linked, or run. If you specify an inputfile and the compiler does not find the file, it appends the extension .Pas and searches again. If inputfile is still not found, the compiler prompts for an entire command line.

The outputfile is the name of the file to contain the output of the compiler. The compiler appends the extension .Seg if you do not specify an extension. If you omit outputfile, the compiler uses the file name from inputfile. Then, if the inputfile has a .Pas extension or no extension, the extension .Seg is used for the outputfile. The compiler rewrites the outputfile if it already exists.

You can specify any number of switches. Note that if you specify a switch multiple times, the last occurrence is used. If you specify the -HELP switch, the compiler ignores other information on the command line and displays a Help message.

Switches are:

Switch	Description

-AUTO

The compiler automatically generates a RESET(INPUT) and REWRITE(OUTPUT). This switch enables automatic initialization. This switch is the default.

-NOAUTO

This switch disables automatic initialization of input and output, that is, you must specifically reset (input) on rewrite (output) to read or write information to the display.

-COMMENT=<string>

The -COMMENT switch permits the inclusion of arbitrary text in the first block of the .Seg file. This string has a maximum length of 80 characters. Because a space or a hyphen (-) terminates the version string, you must enclose the string in double quotes if it contains more than one word or a hyphen. The switch is particularly useful for including copyright notices in .Seg files.

-ERRORFILE[=filename]

This switch allows compilations to be left unattended. Normally when the compiler detects an error in a program, it displays error information (file, error number, and the last two lines where the error occurred) on the screen and then requests whether or not to continue. The -ERRORFILE switch overrides this action. When you specify the switch and the compiler detects an error, the error information is displayed and written to a file and there is no query of the user. Lastly, the compiler does display the total number of errors encountered on the screen. The compiler appends the extension .Err if it is not already present. If you do not specify a filename, the compiler uses the source file name. If the .Pas extension is present, it is replaced with the .Err extension. If the .Pas extension is not present, the compiler appends the .Err extension.

The error file exists after a compilation if and only if you specify the -ERRORFILE switch and an error is

encountered. If the file filename.Err already exists from a previous compilation, it is rewritten, or deleted in the case of no compilation errors.

-GLOBALINOUT

This switch causes the code that accesses INPUT and OUTPUT to treat them as globals, even if compiling a program (as opposed to a module).

-NOGLOBALINOUT

This switch accesses INPUT and OUTPUT locally if compiling a program, producing slightly faster code. It is the default.

-HELP

The -HELP switch provides general information and overrides all other switches.

-LIST[=filename]

The -LIST switch controls whether or not the compiler generates a program listing of the source text. The default is to not generate a list file. If the -LIST switch is given, the compiler prints with each source line the line number, segment number, and procedure number. The compiler appends the extension .Lst to filename if it is not already present. If you omit filename, the compiler uses the source file name. If the .Pas extension is present, it is replaced with the .Lst extension. If the .Pas extension is not present, the .Lst extension is appended.

-QUIET

This switch disables the Compiler from displaying the name of each procedure and function as it is compiled.

-VERBOSE

This switch enables the Compiler to display the name of each procedure and function as it is compiled. This switch is the default.

-RANGE

This switch enables the generation of additional code to perform checking on array subscripts and assignments to subrange types. This switch is the default.

-NORANGE

This switch disables the generation of additional code to perform checking on array subscripts and assignments to subrange types. Note that programs compiled with range checking disabled run slightly faster, but invalid indices go undetected. Therefore, until a program is fully debugged, it is advisable to keep range checking enabled.

-SCROUNGE=[filename]

This switch enables the generation of .Qmap and .Sym files for use by the debugger. It is the default.

-NOSCROUNGE

This switch disables generation of .Qmap and .Sym files.

-VERSION=string

The -VERSION switch permits the inclusion of a version string in the first block of the .Seg file. This string has a maximum length of 80 characters. Because a space or a hyphen (-) terminates the version string, you must enclose the string in double quotes if it contains more than one word or a hyphen. Currently this string is not used by any other PERQ software, but it may be accessed by user programs to identify .Seg files.

The version string is terminated by either the end of the command line, the occurrence of a '-' character, or a space. Therefore you must enclose the string in double quotes if it contains more than one word or a hyphen.

-NOMIXEDMODEPERMITTED

Prohibits the useage of mixed-mode expressions.

MIXEDMODEPERMITTED

Permits the useage of mixed-mode expressions (default).

PEEPOPT

Enables Peep-Hole code improvement of output code.

NOPEEPOPT

Disables Peep-Hole code improvement of output code (default).

DISASSEMBLE

Enables printing of disassembly listing on the listing file.

NODISASSEMBLE

Disables printing of the disassembly listing on the listing file (default)

MAP

Enables printing of variable allocation map on the listing file.

NOMAP

Disables printing of variable allocation map on the listing file (default).

You can include certain switches in the source program text. Refer to the "PERQ Pascal Extensions" in the Accent Languages Manual. for more detailed information on switches and other compiler features.

Examples of legitimate compiler calls include:

```
COMPILE Program.pas

COM ProgramX
   (Note that the .pas extension is implicit;
   if ProgramX does not exist, the compiler
   looks for ProgramX.PAS.)

COMP Program2~Program1
  (creates the output file Program1.SEG)
```

```
COM
  (compiles the default file)

COM -NOSCROUNGE
  (compiles the default file with symbolic
  debugging information production disabled)
```

COPY

The Copy command creates a new file identical to the specified source file.

Format:

COPY SourceFile[~]DestinationFile {-switch}

Copy prompts for missing arguments and accepts wildcards in file specifications. If the names of SourceFile and DestinationFile are identical, a new file is not created; if you want two files, use different names.

You can copy across devices and partitions and you can specify the non-file-structured devices CONSOLE: and RS:. The use of CONSOLE: as a source file allows you to create a file on the screen and immediately write it to a file. After you type COPY CONSOLE: Copy will prompt for a filename. Type a filename and press RETURN, type the contents of the file, and then close the file by typing CTRL-z and pressing RETURN. The file will be written. CTRL-DEL-c or CTRL-DEL-k aborts the copy operation. Note that you cannot include control characters in the file when you copy from the console. If you copy a file from the RS232 interface, by default the interface is driven at 9600 baud; use the SetBaud command to change the baud rate (see the SetBaud facility description).

Switches are:

Switch	Description
-ASK	When you specify a wildcard, this switch requests verification for each file copied. -ASK is the default when you use a wildcard.
-NOASK	This switch overrides verification requests for individual files. -NOASK is the default when you do not specify wildcards.

-CONFIRM

This switch requests verification before overwriting an existing file. -CONFIRM is the default.

-NOCONFIRM

This switch overrides requests for confirmation before overwriting an existing file. -NOCONFIRM also sets -NOASK.

-NOSUPERSEDE

When the destination file is the same name as an existing file, this switch copies the source file only if the source file is newer than the existing file. If you want to be sure you have the latest version of a file, copy it using this switch; if you already have the latest version, unnecessary copying will not occur.

-UPDATE

When the destination file has the same name as an existing file, this switch compares the dates of the two files. If the file to be copied is newer, it is copied. If it is older, Copy asks you whether you wish to copy it. If the dates are the same, the Copy command is ignored. The date of the file that is copied will be preserved on the copy.

-HELP

This switch displays a description of the Copy command and the associated switches, but does not copy files.

If the source contains wildcards, the destination must contain the same wildcards in the same order. (The wildcard * matches 0 or more characters; ? matches exactly one character.) When you specify a wildcard, Copy matches all files in the directory with the source pattern. For each match, the part of the source file name that matches each wildcard replaces the corresponding wildcard in the destination. For example,

COPY foo*.abc? anotherdir/*baz.rmn?z

copies the input file "FOOZAP.ABCD" into a new file named "anotherdir/ZAPbaz.rmndz".

Copy asks for verification of each file copied when you specify a

wildcard. The -NOASK switch disables the verification request. Copy also requests confirmation before overwriting an existing file. The -NOCONFIRM switch overrides the confirmation request. (Note that the -NOCONFIRM switch implies the -NOASK switch.)

Wildcards are not valid in the directory part of the source file. However, Copy uses the search list to try to find the source. Note that this is different from the Rename and Delete commands, both of which look in only one directory.

When the source file name contains no wildcards, the destination file name can contain, at most, one occurrence of the asterisk wildcard (*). In this case, the whole non-directory part of the source replaces the asterisk in the destination. For example,

```
COPY /sys/Boot/newOS/myprog.Pas dir3/new.*
```

copies the file "/sys/Boot/newOS/myprog.Pas" into a new file named "dir3/new.myProg.Pas". This is most useful when you want to copy a file from one directory to another, keeping the same filename. For example,

```
COPY dir1/prog.Pas *
```

copies prog.pas from the directory "dir1" into the current directory.

If there are no wildcards in the source, an attempt to include wildcard characters other than a single asterisk (*) in the destination name leads to problems later. The file is created using these extra wildcards as simple literal characters; however, a file name with wildcards in it is hard to specify. For this reason Copy requires confirmation before creating files with wildcards in the name.

If an error occurs with the use of wildcards, Copy asks whether or not to continue processing the rest of the files that match the input, regardless of switches you specify.

CPP

Cpp is the C programming language pre-processor. Using Cpp is the first step in preparing a C program to run on the PERQ workstation. The pre-processor is an optional part of the operating system and is documented in the Accent Languages Manual.

Format:

CPP <filename.c> <filename.i>

The first argument is the name of the C file to be compiled and the second argument is the intermediate file created by Cpp (which is then used as input to the Pcc facility).

The pre-processor allows compiler directives such as #include and #define files to be handled for compilation.

In a future release Cpp and Pcc will be replace by a CC (C Compiler) facility similar to the UNIX software cc facility.

DEBUG

Debug invokes the Pascal debugger which enables you to control, observe, and modify a process interactively.

Format:

Debug [Process]

To obtain a list of process names and numbers, type details -systat. If the process name is ambiguous (i.e., more than one instance is currently running), you must use the process number.

For online documentation after you enter the debugger, type ?.

In addition to the Debug command, you can invoke the debugger in other ways:

a) with the window manager command <prefix>d or the popup menu (the prefix is CTRL-DEL on a PERQ and CTRL-DEL or SETUP on a PERQ2 workstation; see the document "User's Guide to the Window Manager" in this manual for more details);

b) by ending the command line that starts execution of a process with the special shell switch [-debug], in brackets as shown.

Note that this user-level debugger is totally distinct from the Accent kernel debugger, over which the user has no control. The kernel debugger is invoked when the Bye command is given.

DEFINE

Define manipulates environment manager variables. Commands are of the forms:

DEFINE <name> [name_switch]... [[element [, element]...]]
To set or modify the value of a variable

DEFINE <name> [scope_switch] -show
To show a variable value

DEFINE [scope_switch] -show
To show all variable values

where

name is a name followed by a colon (foo:) for search lists or a simple name for string-list variables;

name_switch is either scope_switch or command_switch;

scope_switch is either -GLOBAL, -LOCAL, or -NORMAL;

command_switch is either -AFTER[=<INDEX>] or -REPLACE[=<COUNT>];

element is element_string [element_switch]...;

element_string is a space or comma delimited string value. If name specified is a search-list, the string syntax is checked for validity as a search-list item;

element_switch is either -RESOLVE or -FULL.

Local environment manager variables take precedence over global ones for the current process.

Command switches, which affect the ways in which a variable is set, are:

-AFTER=<index>	Specifies the position in the old list where insertion or deletion (replacement) is to be performed. The default value for index is 0.
-REPLACE=<count>	Specifies the number of elements to be removed from the original list before any new items are inserted. The default value is 1.

-HELP is also a switch, used to obtain information about the Define facility.

If neither -AFTER, -REPLACE, nor -SHOW are specified, then the new value for the variable will simply be the list of elements specified.

Scope switches may be used to either set or show the local or global value for a variable. If -NORMAL is used for a -SHOW, then the most local value will be shown.

Both element switches cause a search list value to be recursively evaluated before insertion into the list. If -FULL is not specified, then only the first element in the resultant list is inserted.

Examples:

```
Define current: /sys/user/blair/
          -- same as path /sys/user/blair/

Define default: -repl -after=1 foo
          -- same as sets -pop foo

Define sys:,current:, dev:User/System/,
         <boot>System/ -- New slist

Define default: -show
         -- Examine default searchlist

Define terminal PERQ
         -- A string-valued variable
```

The environment manager variables supplied with the system are listed below. Actual values of these variables are installation dependent. You may also define your own variables.

```
Boot:              = /Sys/Accent/ -Global
BootCharacter      = 8 -Global
Current:           = /Sys/Accent/ -Local
Default:           = current:, /Sys/Accent/LibPascal/,
                       boot: -Local
Dev:               = /Sys/ -Global
MachineName        = Smith -Global
OopsKeyDown        = FALSE -Global
PrinterServerName  = Printer1 -Global
ProfileName        = Boot:Default.Profile -Global
Run                = current:, boot: -Global
ShellCommands      = ShellCommands -Global
ShellName          = Boot:Shell.s5.run -Global
ShellProfile       = ShellProfile -Global
SidServerName      = Printer1 -Global
SystemVersion      = S5 -Global
UserID             = 2 -Global
UserName           = Jane -Global
```

DELETE

The Delete command deletes the name of one or more files from the directory and places the blocks the file occupied on the free list, thus making those blocks available for use by other files. When you delete a file, you irrevocably destroy it.

Format:

DELETE <Filename>[,file2][...,filen] {-switch}

You can specify wildcards in the file specification, but not in a directory specification (wildcards are discussed under the Direct command).

The Delete command will search only one directory (the current directory or the specified directory).

The switches are:

Switch	Description
-CONFIRM	This switch requests verification before deleting a file. -CONFIRM is the default when you use a wildcard.
-NOCONFIRM	This switch overrides requests for confirmation. -NOCONFIRM is the default when you specify a filename without a wildcard.
-HELP	This switch displays a description of the Delete command and the associated switches. Note that -HELP does not delete any files.

DETAILS

Details gives information about the current state of the system.

Format:

DETAILS {-switch[=switch value]}[~outputfile]

More than one switch can be specified if you wish. If no switches are given, a default selection of information is given.

Switches are:

Switch	Description
-ALL	Gives all of the information available with the other switches.
-BOOTCHAR	Gives the boot character for the current system.
-BOOTS	Gives all the valid boot characters.
-DISKTYPE	Gives the type of disk.
-ETHERNET	Gives the Ethernet address and machine name.
-FREEPAGE	Gives the size of the free paging space.
-IOBOARD	Gives the type of I/O board.

-MACHINENAME

Gives the name of the workstation (the name in the file SysName).

-MEMORYSIZE

Gives the size of the memory.

-MONITOR

Gives the type of monitor.

-NAMES

Gives all names (machine, profile, etc.)

-OOPSKEYDOWN

Shows whether ignore-run-file is set.

-PARTITION

Gives information about the partition.

-PATH

Gives the boot partition and current path.

-PROFILE

Gives the name of the current profile file.

-RS232STATUS

Gives the number of RS232 connections.

-SEARCH

Lists the current search list.

-SERIALNUMBER

Gives the serial number of the workstation.

-SHELLNAME

Gives the name of the current shell run file.

-SYSTAT

Gives the status of processes. A sample -Systat is as follows:

	State	Priority		RunTime	Window	What
4.	S	S	7	0.300002		Time Server
10.	S	S	7	3.350027		Sesamoid
15.	S	S	7	0.75006		Environment Manager
42.	S	S	7	0.883340		Startup Process
113.	S	S	13	0.400003		10MHzMsgServer.S5.RUN
125.	P	S	13	0.116668		10MHzNetServer.S5.RUN
148.	S	S	7	5.300042		Initial Shell
183.	S	S	7	0.150001	Detail	Details.Run
51.	S	S	12	8.066731		Sapphire
62.	P	S	15	0.733339		Tracker
79.	S	S	11	4.800038		Typescript
1.	U	0	10	0.383336		Process Manager
194.	S	S	0	1.683347	Clock	Clock
214.	S	S	7	0.066667	remote	remote
229.	P	S	15	0.000000	Floppy	FloppyServer
249.	S	S	0	0.100001	listen	listen
164.	S	S	7	0.050000	speak	speak

A status line has five fields:

1. The first field is a unique process number (the Process Manager's port number for the Kernel Port of the process being described.) This field may always be used for Process on any process control command.

2. The second field contains two subfields:

   1) the process's privilege state:

   * S - All supervisory privileges
   * P - Physical memory access capability
   * U - No privileges -- user

2) the process's execution queue (priority):

* '# #' - two digit (0-15) run queue index
* 'P' - pending queue
* 'S' - sleeping queue
* 'O' - other

3. The third field is the elapsed run time, in seconds.
4. The fourth field is the Process Manager's port number for the Window (port) of the process.
5. And lastly, the fifth field is the complete name of the process. This field may be used for Process on any process control command as long as it is unique.

-TIME

Gives the current time.

-USERNAME

Gives the current user's name.

-VERSION

Gives the version numbers of the net server, process manager, and Sapphire (internal name of the window manager).

-WCSSIZE

Gives the writable control store size.

-HELP

Displays information about the Details facility.

DIRECT

The Direct command provides a list of files in a directory. By default, the command provides an alphabetical list of the file names, but you can specify other sort algorithms. You can display the directory listing at your terminal or you can direct the listing to a file.

Format:

DIRECT [dirSpec/][fileSpec] {-switch}[~][outputfile]

If invoked without a switch the command lists, in alphabetical order, all files in the current directory.

To write the output of a Direct listing to a file, specify an output filename.

If you specify a wildcard, Direct matches the dirSpec part of the command line against all directories and the fileSpec part against all files in the directories that matched dirSpec. Wildcards are not valid in partition or device names. The following wildcards are valid:

*	matches 0 or more characters
?	matches exactly 1 character

Note: A wildcard character can be specified as a literal character in a filename by preceding it with a backslash (\)--for example, foo*.pas for filename foo*.pas.

Switches are:

Switch	Description

-ALL

This switch provides the following information for each file:

Number of blocks
Number of bits
Kind of file
Creation date
Last access date
Access privileges

-DELIMITER

This switch lists the file names as

name ¦ name

The switch is most useful when used in conjunction with an output file specification (for example, when you create a command file from the Direct listing).

-DIRECTORIES

This switch lists directories only.

-FAST

This switch lists only the file names, a short directory. -FAST is the default.

-HELP

This switch displays a description of the Direct command and the associated switches. Note that -HELP does not provide a directory listing.

-LISTDIRECTORIES

When performing a multi-directory operation, Direct normally lists only the directories that have valid matches for the fileSpec. This switch instructs Direct to list all directories that match the dirSpec,

regardless of whether or not they contain matches for fileSpec.

-MULTICOLUMN

This switch instructs Direct to list files in four columns. -MULTICOLUMN is the default when Direct displays a short (-FAST) listing on the screen. Note that -ALL and -SIZE override this switch.

-ONECOLUMN

This switch lists all files in a single column. -ONECOLUMN is the default when you specify an output file.

-PARTITIONS

This switch provides partition information (for each partition) after the Direct listing.

-PROT

This switch displays the owner of the file and the owner's ID number (except for files that are part of the operating system) and the protection code for each file specified. (See the SetProtect facility.)

-SIZE

This switch displays the number of blocks and bytes for each file in the Direct listing.

-SORT=option

This switch permits you to specify the algorithm Direct uses to sort and list the file names. The options are:

NOSORT - lists the files in essentially random order.

NAME - sorts by the name of the file and produces an alphabetical listing. NAME is the default.

SIZE - sorts by file size. This option lists files in decreasing order, with the largest file first.

CREATEDATE - sorts by creation date. This option lists the most recent file first.

ACCESSDATE - sorts by last access. This option defines access as the last read operation performed on the file.

Examples:

DIRECT
lists every file in the current directory

DIRECT */*
lists all files in all directories starting with the current directory and including all subdirectories.

DIRECT /BOOT/x*/*.run~run.list
looks in the Boot partition for all the run files in directories whose names start with "x" and writes all of these names into the file "run.list".

DIRECT Program*
tells you what files beginning with "Program" are in the current directory, for example, Program.pas, Program.seg, Program.run.

DIRECT *zing*
lists all files in the current directory with "zing" in their names.

DIRECT Program* -SIZE
lists files beginning with "Program" and tells how much disk space each occupies.

DIRTREE

The DirTree command provides a graphic representation of the file system tree structure. The command erases the screen and displays all the directories starting from the root directory on the left. If the window is not large enough to display the tree, DirTree enlarges the window.

Format:

DIRTREE [RootDirectory] {-switch}

You can specify any directory as the root of the tree (for example, you could type dirtree /sys/user/). The default is the default device; DirTree displays all the partitions, all the directories in each partition, all the subdirectories, and so on. Lines connect each directory to its parent. If you specify a directory, DirTree simply starts the search from that directory.

If the directory structure is too deep to fit on the screen, DirTree puts an asterisk (*) on the right of the parent. To see more, reinvoke DirTree with this parent directory as the root.

The valid DirTree switches are:

Switch	Description
-BLOCKS	Updates the tree, displaying the number of blocks in use in each directory. Note that the count does not include blocks used for file headers (at least one such block exists for each file) nor does the count include blocks used to hold the directory itself. For each directory that has subdirectories, DirTree displays the total number of blocks in the parent directory followed by a tilde (~), followed by the total of all blocks in the parent directory and in all its subdirectories. For example, imagine a directory, User, with two subdirectories, Oldsource and Newsource. In the User directory, 300 blocks are used for files, in Oldsource 150 blocks are used, and in Newsource 45 are used. The count for User would be:

User/
300~495

Displaying the tree with blocks takes about 4 times as long as without blocks.

-HELP

Displays a description of the DirTree command and the associated switches. Note that -HELP does not display the tree structure.

After the tree is drawn you can give any of the following commands:

Command	Description
b	(blocks) Same as the Blocks switch described above.
h	(help) Same as the Help switch described above.
q	(quit) Quits the program and returns you to the shell.
r	(return) Returns to the current DirTree, after you have issued the h command.
u	(update) Updates DirTree. You can keep DirTree in a spare window on or off screen (DirTree consumes no processing time when it is idle) and at any point while you are working return to that window and issue this command. The tree will be redrawn, reflecting the current status. (Note that if you want the tree updated with the number of blocks, you should use the b command which updates and furnishes the number of blocks. The u command does not display number of blocks.)

DISMOUNT

The Dismount command detaches a partition from the file system.

Format:

```
DISMOUNT
```

You will be prompted for the partition name.

Note: The Mount facility and the Dismount facility are not related.

EDIT

Edit is used to create or alter text files on the PERQ workstation using the editor for the Accent operating system.

Three basic uses of the edit command are discussed briefly below: creating a new file, changing an existing file, and reading a file at leisure. Exiting the editor is also explained briefly. This utility is explained in detail in the document "The Editor" in this manual.

Format:

EDIT [filename]

or

EDIT <filename> -replay

If you omit the filename, Edit assumes that you want to edit the default file remembered by the shell. The -REPLAY switch runs a transcript of the last editing session. This recovers editing lost due to the system crashing or your exiting a file without saving it. For details, refer to the editor document.

The editor performs automatic completion on the filename you specify. Type as much of the filename as needed to distinguish it from other filenames and press INS (ACC/ESC on PERQ2 workstations). The editor will also look for certain filename extensions. If you do not specify an extension, the editor tries to find the filename specified with these extensions (listed in the order tried): .pas, .pasmac, .mss, .cmd, and .micro.

Creating a New File

To create a new file, invoke the Editor giving the name of the new file. The Editor clears the screen to give you a blank page. The cursor is positioned in the filename; press RETURN to move it into the text. The normal mode is "insert" so there is no need to issue an insert command--you can begin to type the text.

Changing an Existing File

To make changes to an existing file, invoke the Editor with the name of the file. The cursor will be positioned at the first character of the file. Commands for moving the cursor and making changes are given in the editor document.

Reading a File

To read a file without making any changes to it, you can access it as though you wished to edit it and then scroll through the file at your own pace. To safeguard against your having made accidental changes to the file, type CTRL-x CTRL-f to exit and then press OOPS and RETURN. (You can also read a file by using the Typefile command.)

Exiting the Editor

To exit the Editor, type CTRL-x CTRL-f. You are then prompted to do one of the following:

press RETURN to write the changes to the default file;

type a different filename to write the changes to a different file;

press DEL to return to the window; or

press OOPS and RETURN to exit without saving.

EXPANDTABS

ExpandTabs simulates tabs in every 8th column by replacing tabs in the input file with the correct number of spaces. ExpandTabs is used when the input file was written for another system and put onto a PERQ workstation, which does not support tabs. Its command line takes the form:

EXPANDTABS <SourceFile> <DestinationFile> [-Help]

The SourceFile and DestinationFile must be different.

The only switch is -Help, which displays information about Expandtabs.

FINDSTRING

The FindString command searches through a number of files for a particular string. The command operates in two modes: context and nocontext. In context mode (the default), FindString prints the line number and the leading and trailing characters for each occurrence of the specified string. In nocontext mode, FindString prints only the word *Match* when it reaches the first occurrence of the specified string. You specify the mode with the -CONTEXT or -NOCONTEXT switches.

Format:

FINDSTRING string,filelist {-switch}[~outputfile]

The first argument to FindString is the string to search for. To include a non-alphanumeric character, either precede it with a backslash (\) or surround it with quotes. The next argument is the file to search. A pathname must be included if the file is not in the current directory. You cannot list more than one file but you can specify a wildcard. Switches are described below. If you wish, you can direct FindString to write the occurrence(s) to a file by specifying an output file.

Switch	Description

-CASESENSITIVE

This switch specifies that case is significant (for example, if you specify the switch and the string to search for is XYZ, FindString does not view XyZ as a match.

-NOCASESENSITIVE

This switch specifies that case is not significant; FindString ignores upper and lower case. -NOCASESENSITIVE is the default.

-CONTEXT

This switch directs FindString to list leading and trailing characters for each occurrence of the string. -CONTEXT is the default.

-NOCONTEXT

This switch directs FindString only to notify you when it finds the first match, if any.

-HELP

This switch displays a description of the FindString command and the associated switches. Note that -HELP does not search for string occurences.

Example:

```
FindString screen, /sys/boot/os/*.pas~screen.users -nocontext
```

This command directs FindString to search all files with a .PAS extension in the OS directory of the BOOT partition for an occurrence of the string screen. FindString writes the output to the file "screen.users". By default, case is not significant; in the example above, Screen would match screen. You can force FindString to match case exactly by specifying the -CASESENSITIVE switch.

FLOPPY

The FLOPPY utility formats, tests, reads, and writes RT-11 format floppy disks. You can use FLOPPY to transfer files between the hard disk and the floppy disk. Note that the RT-11 file format restricts filenames to six characters with a three character extension. Thus, if you wish to copy a file from the hard disk to the floppy disk, Floppy limits the file name on the floppy to six characters and the extension to three characters (you can enter the full file name, but only the first six characters will name the file on the floppy). Valid characters are the upper and lower case alphabetics, the digits 0 through 9, the dollar sign, and the period; Floppy uses the RAD50 character set.

The environment variable FloppyServerName can be set to pick a particular floppy server. If this environment variable does not exist, the server used will be [<MachineName>]FloppyServer.

Format:

FLOPPY [command] {-switch}

To execute a single FLOPPY command, enter a command on the command line.

To execute multiple FLOPPY functions, type FLOPPY and press return. In this case, the utility prompts with FLOPPY>. You can then enter commands. All of the Floppy commands prompt for missing arguments.

You can also direct FLOPPY to access a command file containing a list of FLOPPY commands. Simply enter an at sign (@) followed by the file name that contains the FLOPPY commands (@file).

Some FLOPPY commands require confirmation. This confirmation comes from the keyboard even if a user command file is in use. If you use the FAST command (see below), no confirmation is required. The Zero and Format commands, however, require an explicit -switch to override the confirmation request.

While FLOPPY is processing a command, the progress bar in the window title line, as well as the progress bar in the icon, moves from left to right. When it reaches the right side, the operation is complete.

FLOPPY only accepts wildcards for the DELETE and DIRECT commands (see the respective command descriptions). In addition, the only wildcard available is the asterisk (*) and it can only be used by itself in either the filename or extension part of a file specification (or both).

Valid FLOPPY commands and their switches are:

Command	Description
ASK	Require confirmation on each file name.
NOASK	Do not require confirmation on each file name.

COMPARE

This command takes a disk file and a floppy file (in that order) and ensures that all bytes are identical.

Format:

COMPARE [diskfile][~][floppyfile]

You can specify multiple disk files and multiple floppy files for the comparison. If you specify multiple arguments, you must separate like arguments by a comma (,) and delimit input from output arguments with the tilde character (~).

The COMPARE command prints a message for every block that contains a difference.

COMPRESS

This command moves files so that all the unused blocks are at the end of the floppy; the command coalesces free space on the floppy.

Format:

COMPRESS {-switch}

The command does not accept input or output arguments, but has a switch that turns verification on or off. The default is verify--COMPRESS checks every transfer to

assure there are no errors. The -NOVERIFY switch overrides the default.

Do not interrupt this command once it starts; interrupting the Compress command renders the floppy data unreadable.

CONFIRM

Require confirmation before deleting or overwriting a file.

NOCONFIRM

Do not require confirmation before deleting or overwriting a file.

DELETE

This command deletes a file or multiple files from the floppy.
Format:

DELETE [file][...,filen] {-switch}

To delete multiple files, you must separate the filenames with a comma (,). By default, the DELETE command requests confirmation before deleting a file. You can override this by specifying the -NOCONFIRM switch.

You can specify an asterisk (*) only in place of the file name and/or in place of the file extension. For example, *.txt, *.*, or myprogram.* are valid uses of the wildcard (you cannot specify my*.*).

DENSITY

This command displays whether the floppy is single or double density.

Format:

DENSITY

The command does not accept arguments nor switches.

DIRECTORY

This command lists the files contained on a floppy and

optionally writes the directory listing to a file.

Format:

DIRECTORY [file][~][outputfile] {-switch}

If you specify the DIRECTORY command with no arguments, it lists all the files contained on the floppy. If you specify an input argument, DIRECTORY lists all files that match the specified filename.

By default, the DIRECTORY command prints a full listing. To override this and print only the file names, specify the -SHORT switch.

If you specify an output argument, DIRECTORY only writes the file names (-SHORT is the default when you specify an output file) to a disk file with that name.

Note that when you direct the output to a file, Directory does not display the filenames on the screen.

FAST

This command turns off requests for confirmation for all subsequent commands except Format and Zero; the command sets the -NOASK and -NOCONFIRM switches for all subsequent commands except Format and Zero.

Format:

FAST

The command does not accept arguments or switches.

Use the Fast command only in conjunction with command files.

FORMAT

This command formats a floppy disk and destroys its current contents. Format command switches permit you to specify the floppy density, number of sides, and whether or not to test after formatting. The -Interleave switch permits you to specify the number of floppy sectors between sequentially numbered sectors. For example, if the -Interleave value is one, the sectors are numbered sequentially. If the -Interleave value is two, every other sector is numbered sequentially.

Format:

FORMAT {-switch}

The switches are:

-DblDensity
-SingleDensity (default)
-Noconfirm
-Singlesided
-DblSided (default)
-Test
-NoTest (default)
-Interleave=value (default is 2)

GET

This command copies one or more floppy files to the hard disk.

Format:

GET [floppyfile][~][hardfile] {-switch}

In its simplest form, you specify only an input filename to copy from the floppy to the hard disk. GET then copies the floppy file to the hard disk using the same filename. If the filename already exists on the hard disk, GET requests confirmation before overwriting it. If you omit the input file name, GET prompts for the file to copy. Other forms of the command permit you to name the hard disk file. For example

GET floppyfile~harddiskfile

To specify multiple files, use the following construct

GET f1,f2,...fn~h1,h2,...hn

Like the COMPRESS command, GET takes the -VERIFY and -NOVERIFY switches. The default is -VERIFY.

The GET command requires confirmation before overwriting

a file on the hard disk. You can specify the -NOCONFIRM switch to override this action.

HELP

When issued as a command, HELP provides general information for the FLOPPY utility. You can get specific help by using the -HELP switch. Note that HELP and -HELP override any other commands or switches; FLOPPY displays the requested help and then reprompts.

PATH

This command changes the current hard disk directory.

Format:

Path [pathname]

Specify the directory for the new path. If you omit the directory, Path prompts for it.

PAUSE

This command suspends FLOPPY execution.

Format:

PAUSE

Press carriage return to continue.

PUT

This command parallels the GET command, but transfers a file or files from the hard disk to the floppy disk.

Format:

PUT [hardfile][~][floppyfile] {-switch}

PUT also requires confirmation before overwriting an existing file on the floppy (override this with -NOCONFIRM) and, like GET, prompts for a missing file name and accepts the -VERIFY and -NOVERIFY switches.

QUIT

This command exits the FLOPPY utility.

Format:

QUIT

RENAME

This command changes the name of a floppy file.

Format:

RENAME [oldname][~][newname][-NOCONFIRM]

If the new filename already exists, RENAME asks for confirmation before overwriting it. You can override this with the -NOCONFIRM switch.

SAFE

ASK only on wildcards (default).

TYPE

This command displays a floppy file on the screen. A progress bar moves from left to right in the window title line, as well as in the icon, to show how much of the file has been displayed.
Format:

TYPE [filename]

VERIFY

Verify correctness of data transfers.

NOVERIFY

Do not verify correctness of transfers.

ZERO

This command creates a new directory on the floppy.

Format:

ZERO {-switch}

By default, the directory matches the number of sides on the floppy. You can override this by specifying the

-SINGLESIDED switch. The Zero command always requests confirmation before creating the directory. You can override this only by specifying the -NOCONFIRM switch. In the process of creating the new directory, the ZERO command destroys the contents of the current floppy. Use the ZERO command after formatting a floppy (see the FLOPPY FORMAT command description) or whenever you wish to destroy the current content of a floppy.

Note that when you issue the ZERO command, you irrevocably destroy the current contents.

FTP

The FTP utility copies files between PERQ workstations or any other computer that supports the FTP protocol. (If both PERQ workstations are running Accent there is no need to use FTP, as the Copy command can be used between workstations. However, if one or both workstations are running the POS operating system, FTP must be used to copy files from one to another.)

Do not use FTP to transfer directories.

Each workstation must have a file called SysName in the Accent partition which contains the name by which that workstation is known on the network.

Both the workstations must run FTP in order to transfer a file. One workstation must go into Poll mode and the other workstation must connect to it (see the POLL switch below). A machine such as the Print Server is in POLL mode continuously.

Format:

FTP [command] {-switch}

If not given with a command, FTP prompts with FTP>. You can enter commands directly to FTP or type help to receive an on-line summary of commands and switches.

Commands are:

Command	Description

GET <SourceFile>[~][DestinationFile]

This command copies a file from a remote node to the local node. You can optionally include a filename for the file on the local node. By default, FTP requests verification before overwriting an existing local file (to override this, use the NoConfirm command). If you do not specify the name of the file to transfer (SourceFile), the command prompts for it.

PUT <SourceFile>[~][DestinationFile]

This command copies a file from the local node to a remote node. You can optionally include a filename for the file on the remote node. The file you copy from the local node overwrites an existing file on the remote node. If you do not specify the name of the file to transfer (SourceFile), you will be prompted for the filename.

POLL

This command permits a node to send or receive transfers at the request of a remote node. The remote machine issues a Connect command specifying the polling machine as the node and then issues one or more Get or Put commands to transfer files.

QUIT

This command exits FTP.

The following may be either commands or switches. If they appear as switches on a line without a command, they are used as the new default values. If used as switches on a line that has a command on it, it will set that value only for the duration of that command.

To use a command as a switch, precede the command name with a hyphen.

CONNECT <nodename>

Establishes the Ethernet connection between the local node and the specified remote node (nodename).

LOCALPREFIX <string>

Specifies a character string to precede all local file pathnames. The string you specify can be a simple character string or the string can be a path specification. For example, if you specify Foo as the LocalPrefix string, all files transferred to your workstation will begin with the prefix Foo. However, if you specify Foo/ as the LocalPrefix string, the string specifies a directory; all files transferred to your workstation are placed in directory Foo/. When specifying a directory, you must explicitly state the complete path unless the directory is a subdirectory of the current path.

REMOTEPREFIX <string>

Specifies a character string to precede all remote file pathnames. The string you specify can be a simple character string or a path specification (see the LocalPrefix command).

ASK

Instructs FTP to ask permission before doing each transfer.

NOASK

Turns off the permission request.

CONFIRM

Requests confirmation before overwriting a file on the local node. Note that the confirmation request does not

apply to the remote node; when you transfer a file to another node, the file overwrites an existing file with the same name. Confirm is the default.

NOCONFIRM

Overrides confirmation requests for files on the local node.

DISPLAY

Displays information about the status of FTP.

PKTTOGGLE

Toggles whether or not packets are displayed. Initially, FTP assumes that you do not want to see the packet traffic.

HELP

If used as a command, displays information about the FTP facility. If used as a switch, displays information for the command on that line only.

Switches enable you to issue one-line FTP commands if you want to transfer only one file; for example,

FTP GET <FILENAME> -CONNECT=<MACHINE>

Control is returned to the command interpreter shell after the above command is executed. You do not have to give the Quit command to FTP.

To transfer more than one file, type ftp and then specify the desired commands. Each command will apply to all succeeding lines (until you reset or override it).

You can transfer files with either workstation taking the active role; the following describes the alternate scenarios:

1. PERQ Workstation #1 takes the active role and PERQ Workstation #2 is passive. They'll follow this script:

   Workstation #1 (Jones) tells Workstation #2 (Smith) that she wishes to give him the file /jones/user/report.txt and that she will name it Report7-15.txt on his machine.
   Workstation #2: Smith types ftp and then poll.
   Workstation #1: Jones paths to /jones/user and types ftp and connect smith. Then she issues the command:

   put report.txt report7-15.txt

   A progress bar appears on both workstations and indicates progress through the transfer.

   Jones could have stayed in the current directory and included the pathname for the file.

2. Workstation #2 has the active role:

   Workstation #2 (Smith) tells Workstation #1 (Jones) that he wishes to copy the file Logo.Kst from her.
   Workstation #1: Jones types ftp and then poll.
   Workstation #2: Smith types ftp and then connect jones. Then he issues the command:

   GET /jones/user/logo.kst

   A progress bar appears on both workstations and indicates progress through the transfer.

The passive workstation polls while the active workstation processes the transfer.

If an FTP transfer fails, the current transfer aborts and FTP waits for another command (or exits if invoked with a command line). However, if FTP was invoked from a command file and the transfer fails, it restarts the transfer and tries repeatedly until it succeeds. If FTP detects a syntax error in the command file, it aborts that command and asks whether or not you wish to continue.

HELP

The Help command displays general information about other commands and utilities.

Format:

HELP [command]

When a command name is specified as an argument, information is given about that command, if available. If no argument is given, a general Help message is received and you can then type ? to get a list of commands.

ICONWALLCLOCK

IconWallclock displays the time on an analog clock in the icon window.

Format:

WALLCLOCK

As an alternative to Wallclock, you can use the command details -time.

KEYTRANCOM

KeyTranCom creates a "key translation file" (.KEYTRAN) from a .KTEXT source file. A key translation file is a mapping of keyboard keys or key sequences to commands (called "Events") accepted by an application program. Examples of programs using this facility include the shell (technically, the TypeScript process) and the editor.

Format:

```
KEYTRANCOM [filename.KTEXT]
```

The document "The Window Manager" in the Accent Programming Manual shows the format for creation of a key translation table text file.

KILL

The Kill command is used to terminate a specified process.

Format:

KILL [Process] [-switch]

For Process you may use the process number or process name (both obtained by typing details -systat) or the name in the icon window. If you use the name in the Icon window, the Kill command affects all the processes controlled by that window.

The only switch is -Help, which displays information about the Kill command.

You can also use the window manager command <prefix>k or "kill process" on the popup menu. The prefix is CTRL-DEL on the PERQ and CTRL-DEL or SETUP on the PERQ2 workstation.

LAUNCH

Launch allows you to startup programs in specific windows. Launch reads the filename given for commands to create windows and spawn programs (.RUN files). Launch takes each line in the file (long lines can be split by using a backslash character as the last character of a split line), creates a window according to the window descriptor, and spawns the process according to the shell command line.

Launch is not the shell program and so cannot perform directory listings or file deletions or any other function built into the shell.

Format:

LAUNCH <filename> {-switch}

The syntax of Launch file lines is given by the following informal BNF with the additional provision that the items in a list of name-value pairs are separated by commas:

```
<Launch-Command-Line> :== '[' <Window-Descriptor> ']'
                           <Shell-Command-Line>

<Window-Descriptor> :== {<Global-Switches>} <Window-Switches>

<Global-Switches> :== ABSOLUTE ¦ LISTENER ¦ NOBORDER ¦
                      NOCLIP ¦ NOICON ¦ NOTITLES
                      REMOVE ¦ RANK ¦ = <integer>
                      SCREENLEFTX = <integer> ¦
                      SCREENTOPY = <integer> ¦
                      SCREENHEIGHT = <integer> ¦
                      SCREENWIDTH = <integer>

<Window-Switches> :== <Ask-User> ¦ { <Use-Map-File> } ¦
                      { <Use-Rectangle> } ¦ <Use-Quadrant>

<Ask-User> :== <nil>

<Use-Map-File> :== MAPFILE=<filename> ¦ INDEX=<character>

<Use-Rectangle> :== LEFTX = <integer> ¦ TOPY = <integer> ¦
                    WIDTH = <integer> ¦ HEIGHT = <integer>
```

<Use-Quadrant> := FULLSCREEN ¦ UPPERHALF¦ LOWERHALF ¦
RIGHTHALF ¦ LEFTHALF ¦
LEFTUPPERQUADRANT ¦ LEFTLOWERQUADRANT ¦
RIGHTUPPERQUADRANT ¦ RIGHTLOWERQUADRANT

<Global-switches> which can be placed in the file are listed below. They should appear first in the file as some of them change the value of other constants.

ABSOLUTE

Specifies that the tracking of the mouse in the new window is to be absolute; the default is relative tracking. (For an explanation of absolute and relative mode, see the document "Basic Operations" in this manual.)

LISTENER

Makes the new window the listener.

NOBORDER

Creates the window with no borders; the default is to create the borders.

NOCLIP

Specifies that the window will not be clipped against the physical screen; the default is to clip.

NOICON

Creates the window with no icon in the icon window; the default is to give the window an icon.

NOTITLE

Creates the window with no title line; the default is to create a title line for the window.

REMOVE

Removes the window from the screen area after creating it; this is the same as moving it off-screen with a window manager command. The default is not to remove the window.

RANK

Specifies the rank (or Z-axis) for the new window. Rank 1 is the top of the screen, rank 2 just below that, etc. Zero or negative rank puts the window on the bottom of the stack of windows.

SCREENLEFTX

Specifies a value for the default LEFTX; the default value is 0. See note below.

SCREENTOPY

Specifies a value for the default TOPY; the default value is 0. See note below.

SCREENWIDTH

Specifies a value for the default WIDTH; the default value is the width of the screen. See note below.

SCREENHEIGHT

Specifies a value for the default HEIGHT; if not given, the default value is the height of the screen minus 100 pixels so as to not cover the icon window. See note below.

NOTE: The four screen switches are mutually exclusive. If you have no window switches, Launch will ask you to specify the window. (This is equivalent to the mechanism "reshape window" in the window manager program.) Otherwise, if you wish to use a map-file, include one or both of the map-file switches. The map file is a file of characters that specifies a proportional mapping of windows onto the screen. The default window map file is DEFAULT.WMP. For instance, suppose the map file contains the following:

```
1...
....
2..1
.2..
```

Using index 1 will give you a window three-quarters of the height of the screen and full width. Using index 2 will give you the lower left quarter of the screen for the window. Note that these two windows will overlap.

The index specifies what character to look for in the map file to find the corners of the window. Space ( ) or period (.) can be used as a filler in the file. Only one instance of the character need be present. For example, the following map file effectively divides the screen into three horizontal bands of equal height:

```
1
2
3
```

If you wish to specify the shape of the rectangle precisely, use at least one of the rectangle switches. The default values of rectangle switches are:

LEFTX defaults to the value of the SCREENLEFTX global switch. If that is missing, it defaults to 0.

TOPY defaults to the value of the SCREENTOPY global switch. If that is missing, it defaults to 0.

WIDTH defaults to the value of the SCREENWIDTH global switch. If that is missing, then it defaults to the width of the screen.

HEIGHT defaults to the value of the SCREENHEIGHT global switch. If that is missing, then it defaults to the height of the screen (minus 100 pixels to leave room for the icon window).

If you wish to use the quadrant descriptors, include one of the quadrant keywords. These parameters perform the obvious mapping to values for LEFTX, TOPY, WIDTH, and HEIGHT by using the values of SCREENLEFTX, SCREENTOPY, SCREENWIDTH, and SCREENHEIGHT respectively for the default full screen size.

Switches for the Launch command are:

-QUIET

Disables all but critical error messages from appearing on the screen.

-HELP

Displays information about the Launch facility.

Example:

A simple Launch command file that will start the editor and a new shell which will be the listener is as follows:

```
[height=975]editor
[upperhalf,listener]shell.s5
```

Following is a standard Launch.Cmd file:

```
[noicon, remove,leftx=0,topy=0,width=400,height=300,priority=9]
Iconwallclock
[remove, leftx=0, topy=0, width=768, height=500]listen
[remove, leftx=0, topy=501, width=768, height=500]speak
[remove, leftx=0, topy=0,width =200,height=200]lazydisk
[remove, leftx=0, topy=0, width=200, height=200]remote
![remove, noicon, topy=0,width=200, height=200]rs232aserver
![remove, noicon, topy=0,width=200, height=200]rs232bserver
![remove, noicon, topy=0,width=200, height=200]speechserver
![remove, noicon, topy=0,width=200, height=200]gpibserver
```

LINK

Link invokes the Pascal linker for Accent. It produces a runfile (.RUN) from .SEG, data, or .RUN (library) files. The runfile is created by linking together all of the separately compiled modules that make up a program.

Format:

LINK {-switch} Source[,Source2,...SourceN] [~RunFile]

Common uses are:

```
link test            (links user program test.SEG -> test.RUN)
link test,t2 ~xtext  (links test.SEG and t2.SEG -> xtest.RUN)
```

Any number of source files can be listed separated by commas. The main program should be listed first. All of the files that the program imports will be added to the runfile by the Linker. However, if you wish to import other files, you may specify them as arguments to the command.

If you do not specify a name for the output runfile, the Linker uses the name of the source file and the extension .RUN.

Switches are:

Command	Description
-INCLUDE	Copies the bodies of all .SEG files (except those which came from a library .RUN file) into the output .RUN file. Individual files, including .RUN files, can be included using a local -INCLUDE switch.
-NOINCLUDE	Does not copy the bodies of .SEG files into the output .RUN file.

-MAKELIBRARY

Builds libraries out of all the .SEG files on the command line including all that are referenced.

-MAP

Creates a .MAP file containing a map of the .RUN file. This includes the segment numbers of the routines and information regarding imports and exports.

-RELINK

Takes old .RUN files and relinks them with newer versions of libraries. The libraries are listed first in the command line with the program to be relinked at the end.

-SYSTEM

Builds .RUN files to be used by MakeVMBoot to build Accent boot files.

-SYMREFERENCE

References the .SYM and .QMAP files, if they exist, in the .RUN file, which causes them to be present in the address space of the process when it runs. This is slow, but keeps the .RUN file small.

-SYMINCLUDE

Copies the .SYM and .QMAP files, if they exist, into the .RUN file, which makes the debug information present in the address space of the process when it runs. This is faster than using -Symreference, but causes the run file to be much larger.

-MAIN

Specifies that one of the inputs other than the first contains the main program. This is used primarily when re-linking an existing .RUN file with some replacement modules.

-FORCELOAD

Loads all the files on the command line, whether they are referenced or not. Without this switch, only .SEG

files that are referenced are loaded. This switch applies to .RUN files only if a local -INCLUDE switch is specified for the particular .RUN file.

-DATA

Includes all data files (inputs that are marked with a local -DATA switch) in the .RUN file. This switch MUST be specified immediately after the actual data file.

-VERBOSE

Displays the name of each procedure and function as it is linked. It may be disabled by specifying -QUIET.

-QUIET

Prevents the display of procedure and function names during linkage. This is the default; it may be disabled by specifying -VERBOSE.

-LIBRARY=name

Specifying a library as a global switch is equivalent to appending <Name>.RUN to the list of input files. At link time the file <Name>.RUN will be searched

-NODEFAULTLIBRARY

If this switch is not specified, the default library LibPascalInit.Run is linked with your program.

-NOINITIALIZATION

If this switch is not specified, the module PascalInit is made the initial entry point of the output file. It performs process initialization, then invokes the user main program. PascalInit is included in the default library.

HELP

Displays information about the Link facility.

Examples:

link test

Builds Test.RUN from Test.SEG and .SEG files that it imports.

```
link -incl test,x,y,util.run
```

Builds Test.RUN which includes x.SEG and y.SEG and references Util.RUN for utilities.

```
link -lib=LibPas test,dl -data ~main
```

Builds main.RUN which links to test.SEG, references library LibPas.RUN, and includes data file "dl".

```
link -forceload -include test,oldmain.RUN -main ~newmain
```

Builds newmain.RUN as a copy of oldmain.RUN, substituting test.SEG for the module test in oldmain. The ordering of the inputs is important; the opposite order would use the module test in oldmain, not the replacement in test.SEG.

LISP

Accent Lisp is an implementation of the Common Lisp language for the PERQ Model LN-3500 workstation. Lisp is an optional part of the operating system and is documented in the Accent Lisp Manual.

LISTEN

Listen allows you to receive short messages from another user who is running the Speak facility. Listen prints the messages on its window or to a specified transcript file.

Format:

```
LISTEN [Name] {-switch[=value]}
```

Name is the name by which you are known to anyone who is trying to send you a message using Speak. If you invoke Listen without a name, you can be reached with your workstation name (contained in your SysName file) or UserName (the name you used at login).

Switches are:

Switch	Description
-TRANSCRIPT=<filename>	Takes a transcript of everything received and writes the transcript to a file.
-HELP	Displays information about the Listen facility.

If you wish to reply to a message, you must invoke the Speak facility.

LNK

Lnk links programs written in either the C programming language or Assembly language with other required files. The C linker is an optional part of the operating system and is documented in the Accent Languages Manual.

Format:

LNK -o <filename.EXE> libc.lib {x.o, y.o z.o} <filename.o>

where

-o is the output switch;

<filename.EXE> is the name of the target file that is to contain the executable code;

libc.lib is the name of the file containing all of the required libraries; it should always be linked with the main module (although it does not have to come first in the command line);

{x.o, y.o, z.o} are the names of any additional files you may wish to link with the main module;

<filename.o> is the name of the input file that contains the object code for linking.

LOGIN

The Login command initiates a session at a PERQ workstation. Login is called when you boot the workstation.

Format:

LOGIN [UserName]

After you type the above command the system prompts for your password. Login then searches the System.Users file on the authentication workstation and validates the UserName and the password.

MACE

Mace is a low level debugger used primarily in system implementation. For most situations, the Debug facility is more appropriate.

Format:

```
MACE
```

For online documentation after you enter the Mace debugger, type '?.

MAKE

Make helps to create or maintain computer programs that are comprised of many separate modules.

Format:

MAKE {-switch} [TargetFileName]

Switches are:

Switch	Description
-f=<MakeFileName>	Uses the named file as the input MakeFile.
-c	Creates a command file and executes it (unless in No Execute mode).
-d	Runs in Debug mode.
-h	Displays information about the Make facility.
-n	Traces and prints but does not execute the commands needed to update the targets.
-s	Suppresses the printing of each shell command line as it is executed.

Make executes commands in the MakeFile to update one target name. The target name typically refers to a program. If no -f option is present, the file MakeFile is used. More than one -f option may

appear. If TargetFileName is not specified, the first target name that appears in the MakeFile is used.

Make updates a target if it depends on prerequisite files that have been modified since the target was last modified or if the target does not exist.

The MakeFile contains a sequence of entries that specify dependencies. The first line of an entry is a single target, then a colon, then a list of prerequisite files. All the following lines that begin with spaces are shell commands to be executed to update the target. Each shell command must appear on a separate line. These commands are printed as they are executed unless the option -s is used. If a name appears on the left of more than one "colon" line, it depends on all of the names on the right of the colon on those lines, but only one command sequence may be specified for it.

To include comments, start the comment with the character #. Everything from the # to the end of the file will be ignored.

The following MakeFile says that Pgm.Run depends on two files--A.Seg and B.Seg--and that they in turn depend on .PAS files and a common file Incl.Pas:

```
pgm.run=a.seg b.seg
        link pgm
a.seg=incl.pas a.pas
      comp a
b.seg=incl.pas b.pas
      comp b
```

Only one target may be listed on a line at a time, although it is possible to have multiple targets if a dummy name is given with no succeeding commands, e.g.,

```
allfiles=file1 file2 file3
file1= ...
     commands for file1
file2= ...
     etc.
```

Make accepts lines of up to 255 characters. After that it prints a warning and truncates the line.

MAKEDIR

The MakeDir command creates a new, empty directory.

Format:

MAKEDIR [FileSpecification] [-Help]

where FileSpecification is the name for the new directory. If you do not specify the name, MakeDir prompts for one. Do not specify an extension. The -Help switch displays a description of the MakeDir command.

MakeDir does not accept the name of any existing directory; it prints an error message if you supply such a name.

MATCHMAKER

Matchmaker is a language for defining remote procedure calls. Remore procedure calls are message-based interfaces with declarations. These interfaces are generally between multiple processes in a multiple-language programming environment. Once an terface between two processes has been declared, then Matchmaker can be used to generate procedures for sending and receiving messages. Both processes can understand the messages even though the processes may be written in different languages. Matchmaker does all the work of appropriately packing the procedure arguments into messages, and extracting message fields for incoming procedure arguments.

Matchmaker allows a programmer to write a server process and declare the types of all data to be exchanged between the server and its user processes. Procedural interfaces can be declared based on those types for sending data between processes in messages. When these procedures are implemented they can begin to send and receive messages, rather than code within the same process. Remote procedure call interfaces simplify and increase the reliability of writing message based code.

Matchmaker is described in detail in the document "Matchmaker: The Accent Remote Procedure Call Language" in the Accent Programming Manual.

Format:

```
Matchmaker FileName -Lang1=Opt1 ... -LangN=OptN
```

where FileName is the name of the specification file tto be compiled without the .mm extension. The Lang names are currently members of the set Pascal, C, and Accent Lisp. The Opt values must be members of the set:

All	Generate all files for Lang
Defs	Generate Defs file for Lang
User	Generate User file for Lang
Server	Generate Server file for Lang
NoUser	Generate all but User file for Lang
NoServer	Generate all but Server file for Lang

The order of switches and the filename does not matter. Unique abbreviations for both switches and options are accepted.

Switches:

Switches are:

Switch	Description
-HELP	Displays information about Matchmaker.
-VERBOSE	Displays progress information about Matchmaker.
-QUIET	Does not display progress information about Matchmaker.

MOUNT

The Mount command attaches devices to the file system.

Format:

MOUNT

After you invoke Mount, you will be prompted for the interface on which the device is attached (cio for the PERQ or eio for the PERQ2 workstation). Then you will be prompted for the unit number. If your workstation has only one disk, type 0. If it has more than one disk, type the unit number of the disk to be mounted.

ON

On executes a command on another workstation that is running the Remote facility.

Format:

ON <MachineName> <CommandString>

If no parameters are specified, On prompts for them. In CommandString you cannot abbreviate the command name or use an alias.

There are no switches for the On facility itself but the CommandString can contain switches.

PARTITION

Partition creates new partitions or modifies the names and sizes of existing partitions on a disk. Most uses of Partition include initializing pages, which destroys all old data in the partition's physically contigious disk area.

The file system restricts partitions on a disk to fewer than 64K logical blocks [pages]. However, all partitions start and end on cylinder boundaries, and the Partition program enforces this constraint. The maximum size allowed is 65,040 blocks. The maximum number of partitions allowed is 15. Note however, that the maximum usable size for POS is 32520 blocks and the maximum number of partitions allowed by POS is 10 (in the first 31MB of disk addressed by POS).

Format:

PARTITION [-HELP] [-DEBUG]

The -Help switch gives information about the Partition program. The -Debug switch is a learning tool that leads you through the series of questions that the Partition program asks, without making any changes to your disk.

Partition may be used on any one of the multiple disks (units 0-3) that may be mounted on your PERQ. Partition can create new partitions on unused portions of a disk. For existing partitions, Partition can split, merge, rename or initialize. In addition, Partition can rename the whole device.

There are several points to keep in mind when using Partition:

1. You can set free list address schemes to be device-relative or partition-relative. Device-relative free lists are satisfactory for most partitons. However, if a partition includes a logical block number >64K, the free list must be partition-relative because a link in the free list chain cannot hold an address >64K.

2. If you change a free list from device-relative to partition-relative (or vice versa), you must initialize the pages in that partition. Initializing the pages rewrites the links in the free list chain in the current relative manner.

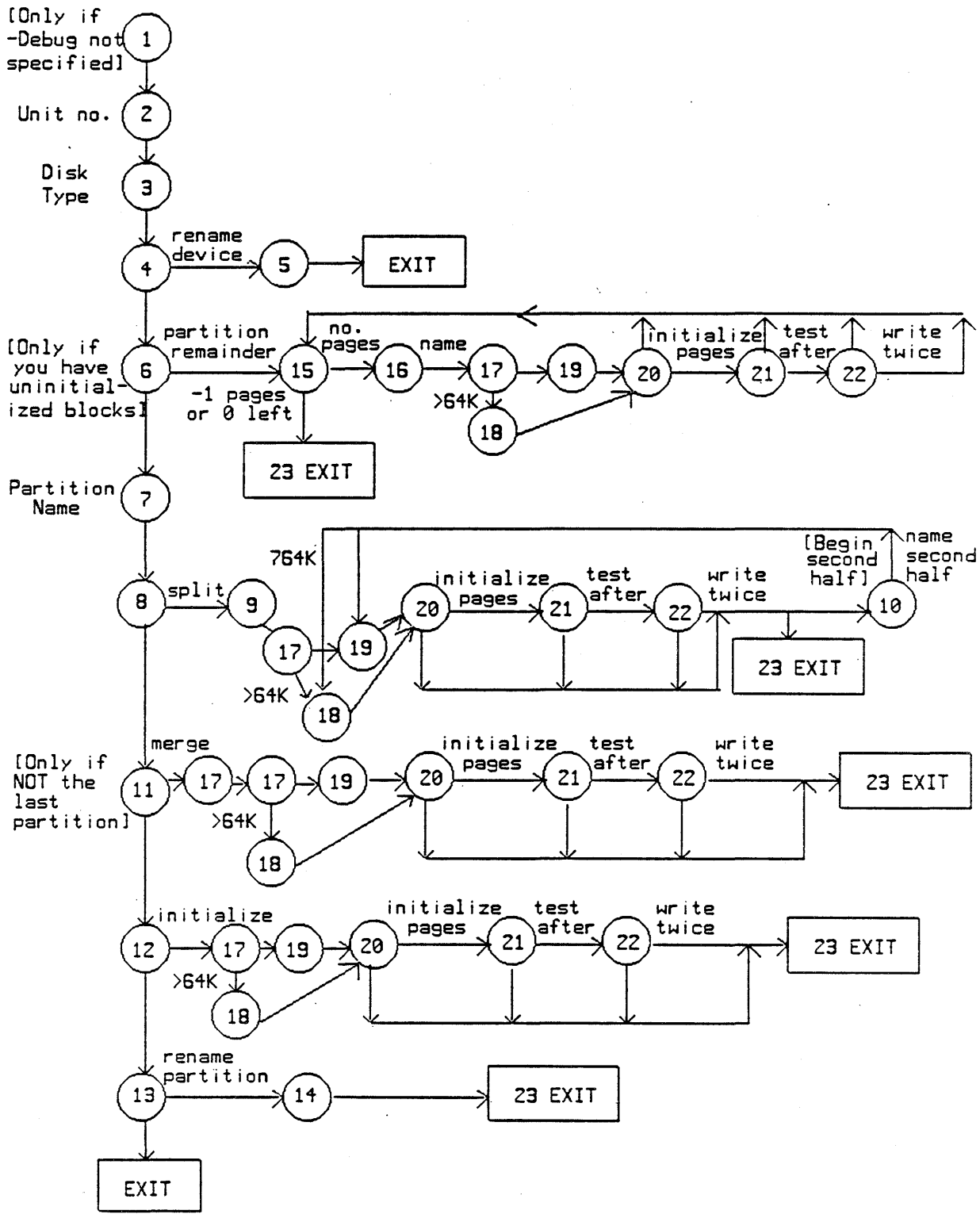
3. Initializing pages also rewrites each data block within a partition, and destroys any data that was in the block.

4. Initializing a partition rewrites header information in the Partition Information Block. If you initialize the partition, you also should initialize the pages within it.

5. You should not run Partition on the partition you are currently pathed to, or on partitions accessed by the operating system (such as your boot and paging partitions).

6. You should not abort Partition by using CNTL-Del C or CNTL-Del K once Partition has begun writing to the disk.

All of the questions that Partition asks are listed on the following pages, with explanatory text for each one. Note that your responses control the logical flow of the questions. Depending on your responses, not all the questions appear. The flow of questions is shown in Figure 1.



Flow of Partition Questions

1. Do you want to debug? (does not do any writes) [No]

   The response to this question specifies whether or not the Partition program actually initializes or modifies a disk.

   Replying Yes is the same as specifying the -Debug switch. (If you specified the switch, you will not receive this question.) If you respond Yes, the program simply continues with the question sequence, but the disk is not modified. Respond Yes to familiarize yourself with the logical flow of Partition questions and to prepare your responses.

   If you respond No, the program modifies the disk based on your responses to succeeding queries.

   The default response is No (the program modifies the disk).

   In case you do not remember whether you gave the -Debug switch or how you replied to this question, the window title line shows "Debug ON" if you are only running through the questions or "Debug OFF" if you are actually making changes to the disk.

2. Which unit do you wish to Partition (0-3)? [0]

   Respond with the unit number of the disk you wish to partition. The default is Unit 0. Use the default if there is only one disk in your workstation.

   Depending on the type of disk in your workstation, the next question will be 3a, 3b, or 3c.

3. You will be asked one of the following questions:

   a. This seems to be a MICROPOLIS 8-inch disk.
      Is this right? [Yes]

   b. This seems to be a SHUGART 14-inch disk.
      Is this right? [Yes]

   c. This seems to be a 5.25" disk.
      Is this right? [Yes]

   The above questions confirm the type of disk. The next question is 4.

4. Do you want to rename the device? [No]

   This question begins the disk modification sequence. This first question in the sequence allows you to change the name of

the disk.

Before renaming a disk, note that the POS linker incorporates the name of the disk (device) where a program was linked into the Runfile. Therefore, if you rename the disk, you cannot run existing POS programs on that disk. This includes the Shell, Login, and the Partition program itself.

If you respond Yes, Partition requests confirmation and then prompts for the new disk name. See question 5.

If you respond No, the program asks whether or not you want to partition the remainder of the disk. See question 6.

The default response to this question is No (to not rename the disk).

5. New device name [current name]

If you responded Yes to question 4 and to the subsequent request for confirmation, this question allows you to enter a new disk name. Enter a disk name from 1 to 8 characters. The Partition program writes the name you enter in the Device Information Block (DIB words 114 through 117).

Note that the default response to this question is the current name of the disk.

6. Do you want to partition the remainder of the device? [No]

This question may be asked if you responded No to question 4. It is not asked if you have no uninitialized cylinders on your disk. Respond YES if your disk was not fully initialized and has unused cylinders (at the end).

If you respond Yes to this question, Partition asks for information about each new partition in turn. See questions 15-22. This allows you to create more partitions in the unitialized area at the end of your disk. You may stop before the end of the disk by answering -1 to question 15.

7. Which partition do you want to modify?

This question is asked if you responded No to question 6. Enter the name of the partition you wish to modify. The program displays a summary of that partition's Partition Information Block and then prompts for the modification. See questions 8 through 13.

There is no default response to this question.

8. Do you want to split this partition? [No]

Since files cannot cross partition boundaries, it is never a good idea to split a partition that contains files. The system destroys any file that has blocks in both partitions. However, it is safe to split an empty partition or to split a partition that you plan to erase.

If you respond Yes to this question, Partition requests confirmation and then prompts for the number of pages in each partition and the name of the new partition. See questions 9 and 10.

If you respond No to this question, Partition continues with the modification queries. See question 11.

9. How many pages would you like in the first half?

The response to this question specifies how much of the partition to leave with the old partition (the partition being split). The remainder of its pages are put in the new partition. The Partition program then asks if you want to initialize the partition pages and thus create a new free list. See questions 17 through 22. After initializing the pages (or immediately, if you respond no to initializing pages), the program requests a name for the new partition. See question 10.

The number of blocks per cylinder governs the size of a partition on a disk. All partitions must be multiples of the number of blocks per cylinder for a given disk. The minimum size for partitons is the number of blocks per cylinder, as follows:

35-megabyte Micropolis disk -- 120

24-megabyte Shugart disk -- 240

5.25" disks -- depends on the particular type and capacity of the disk

10. Name of second half (new partition)? Partition name (up to 8 chars):

Enter the name of the new partition. The Partition program then asks if you want to initialize the pages for the new partition. See questions 18 through 22.

11. Do you want to merge this partition with the next? [No]

If you responded No to question 8 and the partition you named in question 7 was not the last partition, this question allows you to combine two partitions. If either contains files and you wish to keep them, then both partitions must have disk relative free list hints and you must NOT initialize pages (question 20). Otherwise you will destroy the files if you merge. If in doubt, do a Debug run--this will tell you what kind of free list hints each partition has.

If you respond Yes to this question, you are asked to confirm your answer, then the program joins the partition you specified in question 7 with the partition which is next on the disk. This is the only time the order of the partitions on the disk matters. Partition then asks if you want to initialize the new partition. See questions 17 through 22. If you want to erase the new, bigger partition, respond Yes to question 20 to initialize the pages. To save all the files from both partitions, respond no to question 20 (but see the first paragraph above). Then, after Partition exits, run the Scavenger program on the new partition. When running the Scavenger in this case, be sure to tell it to rebuild the directories so that the directories of the two partitions can be joined together. You must not answer No to question 20 if either partition has partition-relative free list hints.

If you respond No to this question, Partition continues with the modification queries. See question 12.

12. Do you want to initialize this partition? [No]

If you responded No to question 11, this question allows you to initialize the partition specified in question 7.

Question 20 will ask whether or not you wish to initialize the partition pages.

If you respond Yes to this question, the next questions are 17 through 22. There are few reasons to initialize the partition without initializing its pages. However, if you respond No to question 20, you must use the Scavenger program to recreate the directory immediately after running Partition (but see Question 19).

If you respond No to this question, Partition continues with the modification queries. See question 13.

13. Do you want to change the partition name? [No]

    If you responded No to question 12, this question allows you to change the name of the partition specified in question 7.

    Before renaming a partition, note that the POS linker incorporates the name of the disk (device) and the partition where a program was linked into the Runfile. Therefore, if you rename the partition, no POS program in that partition can be run. Do not change the name of the partition that is used in the current path since this invalidates the default path name. Also, all entries in the search list that refer to the renamed partition will no longer work. After a rename, the POS system may not be able to find the shell or any other programs.

    If you respond Yes, Partition requests confirmation and then prompts for the new partition name. See question 14.

    If you respond No, the program exits (and has done nothing to the disk).

    The default response to this question is No (to exit the program, rather than rename the partition).

14. New partition name [current name]

    If you responded Yes to question 13 and to the subsequent request for confirmation, this question allows you to enter a new partition name. Enter a partition name from 1 to 8 characters. The Partition program writes the name you enter in the Partition Information Block (PIB words 114 through 117).

    Note that there is no default response to this question.

15. How many pages would you like in it? (0 => all, -1 => finish)

    If you replied Yes to question 6, this question is asked for each partition in turn until the disk is full or you have reached the maximum number of partitions. The file system restricts disk partitions to fewer than 65,040 logical blocks (pages), however to be useable by POS there should be no more than 32520 blocks and there should be no more than 10 partitions in total in the first 31MB of the disk.

    The number of blocks per cylinder governs the size of a partition on a disk. All partitions must be multiples of the number of blocks per cylinder for a given disk. The minimum size for partitions is the number of blocks per cylinder, as follows:

35-megabyte Micropolis disk -- 120

24-megabyte Shugart disk -- 240

5.25" disks -- depends on the particular type and capacity of the disk

If you reply -1 the sequence which started at question 6 is ended and there will still be some uninitialized blocks at the end of your disk. You may initialize these on a subsequent run of Partition.

After you enter the number of blocks for a partition, the program requests the partition name. See question 16.

16. Partition name (up to 8 chars):

You must name each of the partitions (from one to eight characters). Examples of partition names used for the hard disk include "boot", "user", and "exp". A partition can have the same name as a disk, but all the partitions must have unique partition names.

See question 17.

17. Do you want to dismount the Partition? [Yes]

This question begins the partition initialization sequence. (You will not be asked this question if you are in -Debug mode.)

The usual reply is the default [Yes]. If you have any problems dismounting, e.g., if Accent could not mount the partition, you may reply No and then continue as normal.

If the last page of the partition is greater than 64K pages, the next question is 18; otherwise the next question is 19.

18. * Partition extends beyond 64K and so must use partition relative free list hints -- partition unusable by POS. OK to continue? [Yes]

The next question is 20.

19. Do you want partition relative free list hints? [No]

IMPORTANT - If you are not going to initialize the partition pages (answer NO to question 20) for an existing partition,

then you must be sure that you do not change free list hints from partition relative to disk relative or vice versa. Look at the partition information block summary.

Also you must always initialize pages when merging or splitting partitions if either or both have partition relative free list hints.

If you reply Yes, you will be asked to confirm your decision.

20. a. Do you want to initialize partition pages? [Yes]

Your response specifies whether or not the program initializes the partition's logical blocks (pages). If you are partitioning the remainder of the disk, that is, you responded Yes to Question 6, Partition repeats this question for each new partition on the disk. If you are modifying a partition (you responded No to question 6) and you want to initialize it (you responded Yes to questions 8, 11, or 12), Partition asks this question once for merge or initialize or twice for split.

A Yes response to this question places all the logical blocks in the partition on the free list. If any logical blocks are found to be bad, they are removed from the free list and thus are never accessed again.If you are partitioning the remainder of the disk, that is, you replied Yes to Question 6), you should respond Yes to this question. If you are modifying a partition and initializing it (you responded YES to questions 8, 11, or 12), it is recommended that you respond yes to this question also. There are few reasons to initialize the partition itself without initializing the pages, and in general it is a dangerous thing to do (see question 19).

If you respond Yes to this question, Partition asks whether you want to change the default interlace (Question 20b) and then whether or not to test after initializing the pages (Question 21).

If you respond No to this question, you are asked to confirm your answer, and you are warned that you must use the Scavenger program to recreate the directory immediately after running Partition (but see Question 19).

The default response is Yes (to initialize the partition pages).

b. Do you want to change the default interlace? [16] [No]

The usual reply is No. You may use this to optimize the speed of initializing your particular disk type.

21. Do you want to test after initializing? [Yes]

You are asked this question if you replied Yes to question 20.

Although testing after initializing slows the process somewhat, it is good practice to do this testing. If any logical blocks are found to be bad during testing, they are removed from the free list and thus are never accessed again.

If you respond Yes, Partition asks if you want to write every page twice. See question 22.

If you respond No, Partition simply initializes the logical blocks.

The default response is Yes (to test after initializing).

22. Do you want to write each block twice? [Yes]

You are asked this question if you replied Yes to question 21.

Writing every logical block twice provides some additional protection from bad pages; random data is written into the header and body of each block and then read back and compared. In practice, some bad blocks on the disk pass the first test and fail this one.

23. Do you want the partition remounted? [Yes]

In the current release, you will be asked this question only if you have not written to the disk (e.g., if you aborted before any writes were done). You will not be asked this question if the disk was written to (see Exit below) or if you're in Debug mode.

Only mounted partitions can be accessed, so if you plan to use the disk respond Yes.

EXIT

In the current release if you have written to the disk you will be asked to reboot.

PASCAL (Compile)

The Pascal compiler is explained under the Compile facility. You may invoke it by typing either pascal or compile. The Pascal compiler is an optional part of the operating system and is documented in the Accent Languages Manual.

Format:

```
PASCAL [inputfile] [~] [outputfile] {-switch}
```

NOTE: Invoking the PERQ Pascal compilation facility through the use of the Pascal command by-passes any use of the default file remembered by the shell.

PASMAC

PasMac allows programmers to declare and use macros in their Pascal source code. These macros are expanded by the PasMac preprocessor before compilation.

Format:

PASMAC <InputFile> <OutputFile>

When invoked with no arguments, PasMac prompts for the information it needs.

PATCH

Patch allows you to examine and modify the contents of any file on your hard disk. It shows every byte that is stored in the file. This command is basically for use by programmers in making low-level changes to binary files.

Format:

PATCH [filename]

If you do not specify a filename or if the specified file does not exist, PATCH prompts for the filename. When it has a valid file, it prompts with:

Read Block [0]?

Press RETURN to look at the block number in brackets or type a new block number. You can also type HELP for online documentation that describes the commands you can use.

When you ask to read a block, Patch displays it byte by byte or word by word on your screen in 32-rows. You can reference each byte with the indices 0 to 511. Patch permits you to make temporary or permanent changes to your file.

To access all hard disk blocks, you can patch the /SYS/ file.

PATH

The Path command designates a different directory to be the current directory. The "current directory" designation is important for the following reasons. When a file is specified either as a command or as an argument to a command but a pathname is not given, the system first looks in the current directory and then in the directories specified in your search list (see the SetSearch command description). New files are created in the current directory if a different one is not specified. Therefore, if you plan to create or work on several files in the same directory, making that directory current will save you from typing the full pathname for that directory each time.

The "pathname" is the full name of a directory; it includes the names of the partition, all the directories above the current directory in the hierarchy of the file system, and the current directory. For example, if you wish to make the current directory Reports, located in the Smith directory of the User partition, you would type path /sys/user/smith/reports.

Format:

PATH [pathname]

If you do not specify a pathname, Path displays the current path. You can then type a new path or press RETURN to exit without changing the path. Path prints an error message if you try to Path to a nonexistent directory.

Pathnames are explained more fully in the document "Basic Operations" in this manual.

PAUSE

Pause can be used to suspend the execution of a command file and wait for user confirmation before proceeding. It takes a message as a parameter, prints it on the screen, and then waits for a carriage return on the keyboard before continuing. This command can be given by the user, but it is most useful in command files when some user action is required before proceeding (for example, changing floppies).

Format:

PAUSE [message]

PCC

PCC is the PERQ C compiler. After you have used the Cpp (preprocessor) facility, running Pcc is the next step in preparing a C program to run on the PERQ workstation. The C compiler is an optional part of the operating system and is documented in the Accent Languages Manual.

Format:

```
PCC <filename.i> <filename.s>
```

The first argument is the intermediate file which was created by the Cpp facility, and the second argument is the file created by Pcc (which will contain the assembly code).

The output file from pcc is then assembled with the Asm facility.

In a future release Cpp and Pcc will be replaced by a CC (C compiler) facility similar to the UNIX software cc facility.

PRINT

The Print command sends files to a printer. If the file does not begin with a form-feed character, Print supplies one. Also, Print supplies a form-feed at the end of every printed file.

For printers connected to the RS232 port, Print requires Z80 PROMs version 8.5 (or higher) for proper operation. You can find these two PROMs on the "IO" board (red color-coded) labeled with the version number. For printers connected to the GPIB (IEEE-488) port, the PROM versions are not relevant. Contact PERQ Systems field service to exchange earlier PROMs.

Format:

PRINT <Filename>[,filename2,...filenameN] {-switch}

Switches are:

Switch	Description
-ADDR=<n>	This switch sets the GPIB device address to <n>. The default is 4 for -GPIB, 1 for all others.
-BAUD=<n>	This switch sets the baud rate for the print device. The default is -BAUD=9600.
-BREAK	This switch forces a blank page between each page of the printed listing.
-NOBREAK	This switch omits the blank page between each page of the printed listing. The default is -NOBREAK.

-CONTINUOUS

This switch prints the pages continuously, without pausing between pages. It is the default switch.

-COPIES=<n>

This switch allows you to specify the number (n) of listings you wish to print. The default is -COPIES=1.

-DIABLO

This switch directs Print to initialize for the Diablo 630 daisy wheel printer.

-ELECTROSTATIC

This switch directs Print to initialize for the Versatec V80 GPIB printer. Note: The electrostatic printer has only one print format, so the switches -tall, -short, -wide, and -narrow do not apply.

-GPIBCORRESPONDENCE

This switch directs Print to initialize for the RICOH correspondence printer attached to the GPIB.

-HELP

This switch displays a description of the Print command and the associated switches. Note that -HELP does not print a file.

-HP

This switch directs Print to initialize for the Hewlett-Packard 7310A printer.

-LINEPRINTER

This switch directs Print to initialize for the TI 810 (or similar) printer.

-MATRIX

This switch directs Print to initialize for the OKIDATA matrix printer.

-ML82A

This switch directs Print to initialize for the Microline 82a printer.

-NARROW

This switch specifies narrow characters (12 to 16.5 characters per inch, depending on the printer). The default is -NARROW.

-NOPAGE

This switch directs Print not to format the file.

-PLAIN

This switch directs Print to perform no initialization for a specific printer. Use -PLAIN with generic printers. The default is -PLAIN.

-RS232CORRESPONDENCE

This switch directs Print to initialize the RICOH correspondence printer attached to its RS232 port.

-SHIFT=<n>

This switch instructs Print to shift the listing a number (n) spaces to the right. You cannot specify a negative number (shift the listing to the left). The default is -SHIFT=0.

-SINGLE

This switch instructs Print to print one page, then pause. Thus, this switch permits user interaction (for example, to change paper). Press RETURN to continue printing.

-SHORT

This switch specifies short characters (eight lines per inch). The default is -SHORT.

-SERVER=<name>

This switch sets the name of the server. The default is RS232AServer for RS232 printers and GPIBServer for GPIB printers. You may also define the environment variable LinePrintServer to set the server.

-START=<n>

This switch permits you to specify the page number (n) of the document to begin printing. The default is -START=1.

-STOP=<n>

This switch permits you to specify the last page number (n) of the document to print. The default for -STOP prints the last page of the document.

-TABS=<n>

This switch provides tab stops every n characters. The default is -TABS=8.

-TALL

This switch specifies tall characters (six lines per inch).

-TITLE

This switch instructs Print to print a title line plus one blank line at the top of each page. This is the default for all files except files with a .DOC extension (the output files from the Prose formatting program).

-NOTITLE

This switch instructs Print to omit the title line at the top of each page. This is the default for all files with a .DOC extension (output files of the Prose text formatter).

-WIDE

This switch specifies wide characters (10 characters per inch).

PRIORITY

Priority sets the execution priority of the specified process to the specified level. Priority levels range from 0 (lowest priority) to 15 (highest).

Format:

PRIORITY [<Process> [<LevelNo.>]] [-Help]

For Process you may use the process number or process name (both obtained by typing details -systat) or the name in the icon window. If you use the name in the Icon window, the Priority command affects all the processes controlled by that window. If specifying a name, any prefix that is unique will suffice. If multiple instances of a program are running (and therefore you can't supply a unique name for a particular process), you must use the process number.

If Priority is invoked without an argument, you will be prompted for the process and level number.

The following priorities are recommended for the window manager and should be placed in your InitialShell.Cmd file (note: Sapphire is the internal name of the window manager):

```
priority 10mhznetserver 13
priority 10mhzmsgserver 13
priority sapphire 12
priority typescript 11
priority Proc 10
```

The -Help switch can be used to obtain summary information about the Priority facility.

PRQMIC

PrqMic is the PERQ microcode assembler. It creates files to be used by the placer (PrqPlace) to make the binary files. The input to the microcode assembler is a file of microcode source. For information on microcode syntax and other information, refer to the document "Microprogrammer's Reference" in the Accent Microprogramming Manual.

Format:

PRQMIC <InputFile> [-Help]

If the input filename is not supplied, PRQMic prompts for it.

PRQPLACE

The PrqPlace facility uses the output of the PERQ microcode assembler (PrqMic) to create binary files. This facility is run after the source has been assembled by PrqMic. For information on microcode syntax and other information, see the document "Microprogrammer's Reference" in the Accent Microprogramming Manual.

Format:

PRQPLACE <RootFile> <ListingFile> {-switch}

RootFile is the name of the microcode source file which has been assembled (without the .MICRO suffix). ListingFile is the file to which the microcode listing is to be written.

Switches are:

Switch	Description
-DELETE	Deletes all intermediate files (this is the default).
-NODELETE	Saves the intermediate files.
-HELP	Displays information about PrqPlace.

QDIS

QDis is a disassembler for Q-Code. It decodes a .Seg file into Q-code and lists various pieces of information, depending on the switches used.

Format:

QDIS <SegFile> ~<ListFile> {-switch}

You must specify the name of a segment (.Seg) file that contains the Q-Codes to disassemble. The .Seg file you specify can contain wildcards. If QDis does not find the specified .Seg file, it appends the extension .Seg and tries again.

The listfile specifies a file to contain the QDis output. If you omit a listfile, QDis outputs to the console.

When you initiate QDis, it identifies itself as the QCode Disassembler and displays switch settings. QDis then displays the following information (depending on switch settings):

Name of program or module
Name of source file from which generated
QCode version number
Size of global data block
Length of identifiers
Routine descriptor block number, if one exists
(this information pertains only to FORTRAN generated .Seg files)
Version string
Comment string
Language
Number of imported segments
Import list block number
Diagnostic block number, if one exists
(this information pertains only to FORTRAN generated .Seg files)
Unresolved references block number, if exists
(this information pertains only to FORTRAN generated .Seg files)
List of imports

If you did not specify a routine name or number to disassemble,

QDis displays a list of the program's routines and the following information about each:

Routine name
Routine number
Lexical level
Parameter size
Result + Parameter size
Local + Temporary size
Entry address
Exit address

When QDis asks which routine you'd like disassembled, type a routine name or number. The program then types a listing of that routine's Q-code translation and prompts for another routine. You can also type the -DICTIONARY switch at the routine name prompt to view the list of the program's routines again. Press RETURN to exit.

Switches are:

Switch	Description

-DICTIONARY, -NODICTIONARY

This switch enables or disables the printing of the names of all routines contained within SegFile. DICTIONARY is the default.

-DISASSEMBLE,-NODISASSEMBLE

This switch enable or disables the printing of a disassembly listing. DISASSEMBLE is the default.

-IMPORTS, -NOIMPORTS

This switch enables or disables the printing of filenames imported by SegFile. IMPORTS is the default.

-MISCELLANEOUS,-NOMISCELLANEOUS

This switch enables or disables the printing of various information relative to SegFile. MISCELLANEOUS is the default.

-ROUTINE=<name>

This switch specifies which routine name or number (or, if name is ALL, every routine) within SegFile for QDis to process. A question mark (?) is permitted in the switch. For ?, after each routine QDis prompts for the next routine to process. -ROUTINE=? is the default.

-HELP

This switch displays information on QDis and its associated switches.

QUIT

The Quit command causes the current shell to be terminated and its window deleted.

Format:

QUIT [yes|no]

If no argument is supplied, Quit will prompt for confirmation of your desire to leave the shell.

REMOTE

Remote allows a command issued by a user on another workstation using the On facility to be executed on your workstation.

Format:

```
REMOTE [-Help]
```

The only switch is -Help, which displays information about Remote.

RENAME

The Rename command changes the name of an existing file.

You can rename a file from one directory to another (move the file) as long as both directories are in the same partition. Any program that uses the .SEG file and was not linked with the -Include switch for the Link command must be relinked.

Format:

RENAME <SourceFile>[~]DestinationFile {-switch}

Rename prompts for any missing arguments.

Switches are:

Switch	Description
-ASK	This switch requests verification before renaming a file. It is the default when wildcards are used. (The use of wildcards with the Rename command is discussed below.)
-NOASK	This switch overrides verification requests, even if wildcards are used. It is the default when wildcards are not used.
-CONFIRM	This switch requests confirmation before changing the name of a file to an existing file name (therefore overwriting the existing file). This switch is the default.
-NOCONFIRM	This switch overrides confirmation requests.

-HELP

This switch displays a description of the Rename command and the associated switches. Note that -HELP does not rename a file.

The source file for Rename can contain wildcards. (The wildcard * matches 0 or more characters; the wildcard ? matches exactly one character.) If the source contains wildcards, the destination must contain the same wildcards in the same order. If you include wildcards in the source file, Rename finds all files in the directory which match the source pattern. For these files, the part of the source file name that matched each wildcard is used to replace the corresponding wildcard in the destination. As an example, for the command:

RENAME foo*.abc? anotherdir/*baz.rmn?z

The input file "FOOZAP.ABCD" would be renamed to the new file "anotherdir/ZAPbaz.rmnDz".

If wildcards are used, Rename asks for verification of each file renamed. This can be disabled with the switch "-NOASK" or enabled with the switch "-ASK." The default is -ASK.

Wildcards are not allowed in the directory part of the source file. The Rename command will search only one directory (the current one or a specified directory).

When the source file does not contain wildcards, the destination file can contain, at most, one occurrence of the asterisk (*) wildcard. In this case, the non-directory part of the source replaces the "*" in the destination. For example,

RENAME /sys/Boot/newOS/myprog.Pas dir3/new.*

renames the file "/sys/Boot/newOS/myprog.Pas" to a new file named "dir3/new.myProg.Pas". This is most useful when you want to rename a file from one directory to another directory, keeping the same filename. For example,

RENAME /sys/dir1/prog.Pas *

moves prog.pas from the directory "dir1" into the current directory. The directories must be in the same partition in order for this command to work.

If there are no wildcards in the source, an attempt to include wildcard characters other than the single "*" in the destination leads to problems later. Rename treats these extra wildcards as simple literal characters, creating a filename containing characters normally used as wildcards. Such filenames are hard to specify with other commands, and therefore Rename requires confirmation before creating such a file.

If the the destination file already exists, Rename requests confirmation before deleting. You can disable the confirmation request with the -NOCONFIRM switch. The default is -CONFIRM. When you specify the -NOCONFIRM switch, you implicitly specify the -NOASK switch.

If an error in the use of wildcards is discovered, Rename asks whether or not to continue processing the rest of the files that match the input. This confirmation is required regardless of switches you specify.

RESUME

The Resume command causes a process which had been stopped with the Suspend command to be resumed.

Format:

RESUME [Process] [-Help]

For Process you may use the process number or process name (both obtained by typing details -systat) or the name in the icon window. If you use the name in the Icon window, the Resume command affects all the processes controlled by that window. If specifying a name, any prefix that is unique will suffice. If multiple instances of a program are running (and therefore you can't supply a unique name for a particular process), you must use the process number.

If you do not specify a process in the command, Resume will prompt for one.

The -Help switch may be used to obtain summary information about the Resume facility.

RUN

Run executes a program that resides on the system.

Format:

RUN <ProgramName> [argument1 [argument 2...]]

There are two ways to run a program--by typing just the program name or by using the Run command. The Run command when typed without a filename uses the default file remembered by the shell. Therefore if you have been editing, compiling, and linking a file, simply typing run executes that default file.

The Run command can also be used to change which file is the default, by giving the new default file with the command. Thus, the command

Run <ProgramName> [arg1] [arg2...]

is the same as typing

<ProgramName> [arg1] [arg2...]

except that the first command sets the default file to the file specified.

The arguments which will be accepted are the arguments that are valid for the specified program.

SCAVENGER

The Scavenger provides disk maintenance. It checks all files for consistency, rebuilds the free list, and creates a new directory structure for the partition. The Scavenger also removes bad boots.

The Scavenger should be run only on devices initialized by the Partition program.

The maintenance provided by Scavenger may be required in the following situations:

> If you turned off the workstation without using the Bye command, the maintenance functions associated with the command were not performed. In a subsequent session the system may instruct you to run Scavenger.
>
> If a file was not closed properly because a program aborted, the directory in which it resides may be affected. If the file was a binary file, you will experience problems running the file. If the file was text, you will receive a message when you try to edit or type the file.
>
> When a partition is nearly full, the system may not have enough space to perform operations. If the system cannot perform for this reason it will ask you to delete some files and then run Scavenger.

You can run the Scavenger any time you suspect a file system problem.

Format:

> SCAVENGER [partition]

If you omit the partition name, Scavenger asks a number of questions before it begins processing the partition. The questions and responses are discussed below. Note that some of the questions have a default response. If you include a partition name, Scavenger uses all defaults and runs until completion unless there are any serious errors. If errors are discovered, Scavenger requests

confirmation before exiting. Note that the partition you specify must be on the hard disk.

Do not type CTRL-DEL-c or CTRL-DEL-k after Scavenger begins writing on the device. (Either can be typed during read pass.) If you type either command after the Scavenger begins rebuilding the directory, you may not be able to access anything in the directory. If this happens, rerun Scavenger from another partition.

The Scavenger cannot recreate the directory if there are no free blocks in the partition. Therefore, if your partition is full, you must delete some files before running the Scavenger. If you cannot delete any files due to a bad directory and there are no free blocks, then you cannot rebuild the directory. In this case, you must initialize the partition, thus losing all files there. Fortunately, this is a rare occurrence.

The Scavenger program fixes one partition at a time. You can scavenge the current partition or scavenge partitions other than the one you are currently running in.

If you are going to scavenge a 5.25" disk that you did not buy from PERQ Ssystems, you should add the following information to the Disk.Params file in your boot partition: disk name, number of heads, number of cylinders, write precompensation cylinder, boot size, and scetors per track. Otherwise, you will be required to provide this information each time you run Scavenger.

Phases

The Scavenger program has three separate phases. In phase one, it checks and updates some of the system information and deletes bad boots. If you define a boot with BindBoot, but either the microcode or system code files have been deleted, the boot is known to be bad and the Scavenger deletes it.

In the second phase, the Scavenger checks the partition you specify for consistency. The program reads all blocks and generates a new free list in ascending disk order (the Scavenger discards the old free list). Also during this phase, the Scavenger marks as "bad" those blocks that are not readable. If the blocks marked as bad cannot be rewritten, they are marked as "incorrigible" and removed from the file system. All blocks that were allocated to files containing bad or incorrigible blocks are put in a bad file. Additionally, malformed chains are added to the bad file. You specify a name for this bad file in phase three.

During phase three, the Scavenger rebuilds the directories for the partition. You can direct the Scavenger to delete the old directories, if desired. Otherwise, the old directories are marked as such and the program adds a dollar sign to the end of the directory name. The Scavenger aborts if there is not enough room in the partition to create copies of all the directories, leaving the directories only partially rebuilt. In this case, delete some of the files in the directory and then rerun the Scavenger.

Prior to entering a name in a directory or creating a new directory, the Scavenger validates the filename. If a bad name is found (the name includes a control character, "<", "/", ":", or is null) or two files have the same name, Scavenger requests a new filename. In addition, the name cannot end with a ">" or contain ">..>" or ">.>". After Scavenger is finished, you can examine the files with bad names to see whether they contain useful information. If so, rename or edit the files to recover the data. Otherwise, simply delete the files.

Also during the third phase the Scavenger checks the length of all files and allows you to specify a new length. Note that the length refers to the stored length (in the FIB) rather than the number of blocks in a file. Certain files, like directories and swap files, do not set the length field. File lengths usually become garbled when the file is opened and written, but not properly closed (for example, when a transfer is aborted).

Questions

Following is a list of the questions Scavenger will ask, with an explanation of each.

1. Which unit to scavenge?

   In response to this question, specifiy the device that contains the partition you wish to modify.

   Scavenger checks to see what type of disk drive you are using and asks whether your disk is a MICROPOLIS 8-inch drive, a Shugart 14-inch drive, or a 5.25-inch drive. See question 2, 3, or 4.

2. This seems to be a MICROPOLIS 8-inch disk.
   Is this right? [Yes]

   This question confirms the disk choice.

   The next question is number 8.

3. This seems to be a SHUGART 14-inch disk.
   Is this right? [Yes]

   This question confirms the disk choice. Scavenger then checks to see whether the disk is a 12- or 24-megabyte disk and asks for confirmation of the size chosen. See question 7.

4. This seems to be a 5.25" disk.
   Is this right? [Yes]

   This question confirms that a 5.25-inch disk is being used. The next question is 5.

5. Enter name of disk, <HELP> for help, [UNKNOWN].

   For the 5.25-inch disk, you must supply a recognized disk name or disk parameters. If you supply the name of a disk, the next question is 9. If you specify Unknown (the default), the next question is 6. To obtain a listing of known names, press the HELP key.

   The Disk.params file contains the name and parameters of various disks. If you wish, you may edit the file to add other disks (enter the information in exactly the same format as the existing entries). Thereafter when you run Scavenger, you can give the name in response to this question.

6. Would you like to enter the parameters yourself? [YES]

   If you specify YES, questions 6a through 6e are presented. A NO response returns you to Question 5.

   6a. No. of heads:

   There is no default response to this question.

   6b. No. of cylinders:

   There is no default response to this question.

   6c. Write precompensation cylinder:

The inner cylinders are more densely packed in terms of bits per linear inch, and whenever two bits are written in close proximity to each other, bit shift may occur. It may also occur, but to a lesser degree, because of phenomena such as random noise, speed variations, etc. The technique called write precompensation reduces bit shift by detecting which bits will occur early and which will occur late and writing these bits in the opposite direction of the expected shift. Supply a cylinder number.

6d. Boot size:

The size of the boot area is typically 32.

6e. Sectors per Track:

PERQ Systems typically supplies disks with 16 sectors per track, but the number may be different on the disk you are using.

The next question is 8.

7. Is this a nn-MByte disk? [Yes]

This question confirms the size of the Shugart disk. The default is the size of disk. In the actual question, the size (12 or 24) replaces nn above.

The next question is number 8.

8. Can I make changes to your disk [Yes]?

The response to this question specifies whether or not the Scavenger program can make changes to the device in the first two phases of the program (the Scavenger fixes directories later). This is similar to the debug option in the Partition program (refer to question 1 under Partition).

If you respond No, the Scavenger checks the partition for errors and reports them, but does not fix anything. If you are running the Scavenger only to fix the directory, it is about twice as fast to respond No to this question; otherwise, Yes is a good idea.

The default is Yes.

The logical header of every block contains information about the

state of that block. The information includes the count of the block in the file (the file relative LBN) and a two-word identifier for the file the block belongs to (the serial number). The next two questions allow you to check the correctness of these values.

9. Do you want logical block consistency checking [Yes]?

   This question asks whether or not the Scavenger should check the Logical Block Number field in the block's logical header. Remember that this field specifies the file relative logical block number (the first, second, third, and so on block of the file).

   If you respond Yes, the Scavenger compares the LBN field in the logical header with the random index for each file.

   If you respond No, the Scavenger omits this test.

   The default is Yes (to check the file relative logical block number).

10. Do you want serial number consistency checking [Yes]?

    This question asks whether or not the Scavenger should check the Serial Number field in the block's logical header. The serial number of all blocks in a given file is the Logical Disk Address of the file's File Information Block.

    If you respond Yes, the Scavenger compares the Serial Number field in the logical header with the LDA of the FIB.

    If you respond No, the Scavenger omits this test.

    The default is Yes on workstations with more than 256k bytes of main memory and No on 256k byte workstations.

If the Scavenger continually aborts due to FullMemory, respond No to either question 10 or question 11. The default response of No for serial number checking on workstations with 256k bytes of main memory avoids the FullMemory condition; simply press return to get the default answer.

11. Is there enough room to do it in one pass [Yes]

    This question asks if there is enough memory to do the scavenge in one pass. If your partition has 10080 or fewer blocks in it and if the screen has been shrunk (the shell shrinks the screen when it knows you are running the Scavenger), then respond Yes. If your partition is larger

than 10080 blocks, respond No.

If you respond No, the Scavenger uses three passes, which correspondingly slows the program.

12. How many tries for a suspect read? [1]

This question asks for the number of retries for a suspect read. The default is 1.

If you responded Yes to question 8 (whether or not the Scavenger could change your disk), the Scavenger asks questions 13 through 17.

If you responded No to question 8, the Scavenger asks question 18.

13. Do you want temporary segments deleted [Yes]?

This question asks whether or not the Scavenger should delete temporary files. Temporary files exist for swapping; all user files are permanent.

The recommended response is Yes, which is the default.

14. Do you want old bad segments deleted [Yes]?

This question asks whether or not the Scavenger should delete old bad segments. Bad segments contain files with incorrigible blocks from a previous Scavenger run.

If you respond Yes to this question, then Scavenger adds the bad file created by the previous Scavenge of this partition to the free list.

If you respond No to this question, Scavenger retains the old bad segment.

15. Can I rewrite bad blocks [No]?

This question asks whether or not Scavenger can rewrite bad blocks. If a block cannot be successfully read in the specified number of retries (see question 12), writing new data onto the block could fix the problem. However, because writing new data has only a small likelihood of fixing the problem, the default response is No.

If you respond Yes, the Scavenger writes new data in the bad blocks. If the write or a subsequent read fails, then the block is "incorrigible," otherwise it is "bad."

If you respond No, the block is marked "incorrigible" as soon as the read fails. If there are only transient read errors, the block is left alone.

16. Type pairs to ignore (cyl head cyl head ...): []

    This question permits you to enter cylinder/head pairs that you assume are bad.

    If you suspect a bad cylinder or track, enter the cylinder number followed by a space followed by the head number. Continue this sequence for the suspect cylinder/head pairs. Note that a carriage return signifies end of input. Therefore, if the bad pairs require more than a single line for input, simply wrap the pairs to the next line; do not press carriage return until you enter all the bad pairs.

    The default ignores no cylinder/head pairs.

17. Type other blocks to ignore: []

    This question permits you to enter block numbers that you assume are bad.

    If you suspect a bad block, enter the Logical Block Number of the bad block. You can enter up to 15 bad blocks, separated by spaces. Again, carriage return signifies end of input. If you suspect that more than half the logical blocks are bad on a track, writeoff the entire track. Disks have the following number of blocks per track: Shugart (either a 12M or 24M), 30; Micropolis, 24; 5.25" disk, 16; and floppy, 6.

    The default ignores no logical blocks.

    Scavenger then asks whether or not you want complete error listing; see question 18.

18. Do you want complete error listing [Yes]?

    This question permits you to specify whether or not you want notification of individual errors.

    Respond Yes for complete error listing or No for a summary of errors. Yes is the default.

After you answer the above questions, the Scavenger begins fixing the partition.

The Scavenger updates the title line of the window to show what it is working on and uses various cursors to show progress of the different passes.

First, the Scavenger fixes discrepancies if you allowed changes (responded Yes to question 8). If the Scavenger finds a problem with the partition or device information blocks it cannot fix, it asks for help. The problem could be that you specified the wrong device type in question 1 or that the device is not a file system device (for example, an RT-11 format floppy or a hard disk that has not been initialized by the Partition program). These are easily remedied. However, the problem could be that the device discrepancies are beyond repair. Unfortunately, the only fix in this case is to re-partition the device from scratch.

After the device and partition information checks out, Scavenger displays a summary of the Device Information Block (including partition names) and asks which partition it should work on.

19. Which partition do you want to scavenge?

    Enter the name of the partition to scavenge. There is no default response to this question.

When you enter the partition name, the Scavenger displays a summary of the specified partition's Partition Information Block. Note that the values are those stored in the PIB prior to the scavenge.

The Scavenger now makes a read pass through the partition, building tables of each block's next and previous link (this is the data usually visible in the lower portion of the screen). The pass is done in eight parts for efficiency; the cursor goes down the screen eight times before the next step.

The Scavenger then checks the tables built during the read pass for consistency. The cursor changes to show that consistency checking is in progress. If any loops are found, the Scavenger breaks the loops and blinks the screen to show that a loop has been fixed.

Following the consistency check, the Scavenger rebuilds the free list. Note that the Scavenger rebuilds the free list only if you responded Yes to 8 (you permitted changes to the disk). Rebuilding the free list requires a write pass for all blocks on the free list. The Scavenger displays a write cursor. If any bad blocks were found, some more reads and writes are necessary to make the bad blocks into a well-formed chain.

At this point, the Scavenger writes the new Partition Information Block and then begins the directory building pass. The Scavenger

asks whether it should rebuild the directory.

20. Do you want to rebuild the directories?

    This question permits you to rebuild the partition's directory structure.

    Sometimes the Scavenger recommends that you do this, in which case the default is Yes. Otherwise there is no default.

    If the Scavenger recommends that you rebuild the directories or if you suspect there is a problem with the directories and there are enough free blocks, respond Yes.

    If you respond Yes, the Scavenger asks if you want to delete the old directories (see question 21).

21. Delete old directories [Yes]?

    If you responded Yes to question 20, this question permits you to delete the old directories.

    If you respond Yes, the Scavenger deletes the old directories.

    If you respond No, the Scavenger saves the old directories for later inspection; the Scavenger appends a dollar sign to the end of the old directory names and changes their file type to ExDirFile (directories all have the type DirFile). The old directories are just files that you can delete after the scavenge.

The Scavenger then creates new directories as needed. Note that empty directories are not recreated.

Since the directory reappears after a scavenge, the directory rebuilding process makes it easy to recover from overwriting or deleting a directory.

The Scavenger checks and allows fixing file lengths if desired. For each file, it checks the stored length with the actual number of blocks in the file. If they do not match, it allows you to specify a new stored length. This can be any value, but making it bigger than the number of blocks in the file is not recommended. The default for the stored length is the number of blocks in the file. The Scavenger does not check the lengths for directory files or files with their type field set to "SWAPFILE."

Each file's random index allows the file system to find a random logical block without searching down the chain from the file start.

The Scavenger, as part of the directory building phase, can rebuild the random indices for all files. There is a separate question for this with a default answer of No. There is usually no reason to rebuild the indices unless Scavenger asks you to. Building the random indices for large files takes a long time.

If a bad file was created, the Scavenger asks for a name for the file at the end of the directory building phase. If you allow Scavenger to enter and fix the indices for the bad file, you can then type and edit it as a normal file. In this way, you can reclaim some useful information.

SETBAUD

SetBaud is used to set the baud rate of the PERQ workstation.

Format:

SETBAUD [baudrate] {switch}

Valid baud rates are 19200 (valid only for PERQ2 models), 9600, 4800, 2400, 1200, 600, 300, 150, and 110. SetBaud assumes the server names are RS232AServer and RS232BServer.

Switches are:

Switch	Description
-PORT	Specifies the port on which to set the baud rate. "RSA" or "RS" indicate the "A" port (the default). "RSB" indicates the "B" port.
-HELP	Displays information about SetBaud.

SETPROTECT

The SetProtect command sets access privileges on files. A file is owned by the user who was logged in when the file was created. The owner determines read privileges (permission to look at the file) and write privileges (permission to change the file). Read privileges include typing, copying, and appending (but not appending to the file). Write privileges include editing and deleting. These privileges are set separately for other users and yourself (for example, you might want to prevent yourself from accidentally deleting a very important file).

Format:

SETPROTECT <directory or filename> <-switch>

Switches are:

-OwnerRead	(default)
-OwnerWrite	(default)
-WorldRead	(default)
-WorldNoWrite	(default)
-OwnerNoRead	
-OwnerNoWrite	
-WorldNoRead	
-WorldWrite	

You may use wildcards in the directory or filename.

The access privileges you set for a directory apply to all the sub-directories and files in that directory. For example, suppose in the User partition you have several nested directories under the directory Reports and the path to one of the files is as follows:

/sys/user/reports/prod/inv

Access privileges are checked at every step in the path, so the access privileges set for the Reports directory would apply to the Prod directory and the Inv file, as well as all the other files or directories nested under the Reports directory.

To find out what access privileges are in effect for any directory, give the Direct command with the -all or -prot switch (dir -all or dir -prot).

In order for access privileges to work on your network, your system administrator must set up an authorization server. If there is no authorization server at your site or if it's down, you will have complete access to files on your machine but not to files on other machines.

SETSEARCH

The SetSearch command allows you to add and remove paths on the default search list and to change their order on the list. The default search list is the list of directories searched anytime you specify a file (thus you do not have to give a full pathname when specifying files in any directory that is on the default search list).

Format:

SETSEARCH [pathname][,pathname] {-switch}

If you do not specify a pathname, SetSearch displays the current search list and then gives you three choices:

a) Name to push

If you specify a pathname, SetSearch pushes that name onto the search list and retypes the list.

b) "-pop" to remove an item

The -pop switch removes the first item (excluding the current directory) off the list. (Pushing and popping operations work on the second position of the search list so that the current directory always remains at the top.)

c) CR to exit

Pressing RETURN exits SetSearch with no change. (Thus to find out at any time what is on the search list, type setsearch and after the list is displayed press RETURN.)

When you change the search list, the changes stay in effect only during the current session. When you log off, the list reverts to its default state.

Switches are:

Switch	Description
-POP[=<n>]	This switch removes the <n> items below the first item from the search list. The default for <n> is 1.
-RESET	This switch pops all but the first (which is usually the boot) directory from the search list.
-HELP	This switch displays a description of the SetSearch command and its switches.

SETTIME

SetTime changes the settings on your workstation for the time and/or date. SetTime can get a new time setting from one of three sources: the user; the network time server; and on PERQ2 workstations, the EIO clock.

Format:

SETTIME [DD MMM YY] [HH:MM][:SS] {-switch}

If you specify the time, note that the time notation uses a 24 hour clock. The seconds are optional. You cannot specify a year prior to 1980 (80). Correct only the time by omitting the date. The following example sets the date to June 3, 1983 and the time to 3:30 PM:

SetTime 3 Jun 83 15:30

If you direct SetTime to acquire the time, omit the date and time and simply specify the switch to identify the source.

Switches are:

Switch	Description
-USER	This switch permits the user to enter a date and time, using the notation described above. If you do not enter a new date or time, SetTime prompts for a value. -USER is the default.
-SERVER	This switch sets the date and time based on the current time on the network time server.
-EIOBOARD (PERQ2 workstations only)	This switch, valid only on PERQ2 workstations, sets the time from the EIO clock. Note that SetTime adds an offset to the EIO clock time to get the current date and

time. You can change this offset using the -SetGMTOffset switch.

-SETGMTOFFSET

This switch sets the local to Greenwich Mean Time (GMT) offset. The GMT offset is added to the running time from the EIO board to obtain the current date and time.

SHELL

The shell is the command interpreter for the Accent operating system. Typing the Shell command tells the system that you want to open a new window for a new shell.

Format:

```
SHELL [-quit] [command] [-Help]
```

After you give the Shell command, you will be prompted to specify the window corners. If a command is specified, that command will be executed as soon as the new window is opened. If the -quit switch is specified, the new window will be eradicated as soon as the specified command has been executed. The -quit switch must precede the command.

SHOW

Show displays the values of environment manager variables.

Format:

```
SHOW [<Name>] {-switch}
```

If no variable is specified, all variable values will be shown. If a variable is specified, its value will be shown. Variable names are names followed by a colon for search lists and simple names for string-list variables. The environment manager variables supplied with the system are listed below (you may also define your own).

Here is an example of the information which you will obtain with the Show command. If you have defined additional variables, they will be shown also.

```
Boot:               = /Sys/Accent/ -Global
BootCharacter       = 8 -Global
Current:            = /Sys/Accent/ -Local
Default:            = current:, /Sys/Accent/LibPascal/,
                        boot: -Local
Dev:                = /Sys/ -Global
MachineName         = Smith -Global
OopsKeyDown         = FALSE -Global
PrinterServerName   = Printer1 -Global
ProfileName         = Boot:Default.Profile -Global
Run                 = current:, boot: -Global
ShellCommands       = ShellCommands -Global
ShellName           = Boot:Shell.s5.run -Global
ShellProfile        = ShellProfile -Global
SidServerName       = Printer1 -Global
SystemVersion       = S5 -Global
UserID              = 2 -Global
UserName            = Jane -Global
```

Switch	Description
-LOCAL	Displays the local value of the specified variable.

-GLOBAL

Displays the global value of the specified variable.

-NORMAL

Displays the most local value of the specified variable.

-RESOLVE

Recursively expands any reference to a search list. (If this switch is omitted, a search list reference is shown as the name of the search list.)

-HELP

Displays information about the Show facility.

Examples:

show default:	Shows the default searchlist. It will be current:, ...
show default: -resolve	Shows the default search list, expanding it. It will be /sys/user/...

SPEAK

Speak allows you to send short messages to any other user who is running the Listen facility.

Format:

SPEAK [UserName] [-switch]

After you have invoked Speak, send a message by prefixing it with the name of the listener enclosed in slashes, as shown in this example:

/ets/ good morning

From then on until you change the destination by prefixing a message with another designated user, any messages you type will go to that person. If for any reason Speak cannot find the designated user, an error message will be printed and your destination will be changed to your own listener name. Speak indicates the current destination in its prompt and on its title line.

Typing /who/ will give a list of all users on the network who are currently running Listener.

If you are running both Speak and Listen, you should use the same name when you invoke each facility.

The only switch is -Help, which displays information about the Speak facility.

To leave the Speak facility, type CTRL-DEL-c or CTRL-DEL-k.

STATISTICS

The operating system constantly collects statistics about the performance of the swapper. The Statistics command permits you to enable or disable the display of information after each process executes.

Format:

STATISTICS [Yes|No] [-switch]

The option is "Yes" to display the statistics after each process executes, or "No" to end the displaying of statistics.

The only switch is -Help to obtain information about the Statistics facility.

After each process executes, the shell displays in the process manager window statistics for that program in the following format:

```
Load     1.6 secs.
Exec     3.6 secs.
IO       0.8 secs.
Swap     0.1 secs.
Move     0.0 secs.
Duty    97.2 percent.
```

"Load" is the time spent loading the program and its modules into memory.

"Exec" is the total time spent executing including IO, Swap and Move times.

"IO" is the time spent doing Unit-level IO not including IO time spent swapping.

"Swap' is the amount of time spent swapping.

"Move" is the amount of time spent moving segments from one place in memory to another to try to find room for new allocation.

"Duty" is the proportion of time spent in actual work. The system computes the duty factor as follows:

100 * (Exec - Swap - Move) / Exec

STUT

STUT is the streamer utility program used to save and retrieve files on tape using the streamer. The use of this facility is described in the Tape Streamer User's Guide for the Accent Operating System.

Format:

```
STUT [-Help]
```

The -Help switch displays information about STUT.

SUSPEND

The Suspend command temporarily stops a process that is running.

Format:

SUSPEND [Process] [-Help]

For Process you may use the process number or process name (both obtained by typing details -systat) or the name in the icon window. If you use the name in the Icon window, the Suspend command affects all the processes controlled by that window. If specifing a name any prefix that is unique will suffice. If multiple instances of a program are running (and therefore you can't supply a unique name for a particular process), you must use the process number.

If you do not specify a process in the command, Suspend will prompt you for one.

The -Help switch gives summary information about the Suspend facility.

TYPEFILE

The TypeFile command displays on the screen any file or files you specify.

Format:

TYPEFILE [FileSpecification][,file2][...,fileN] {-switch}

TypeFile does extension completion on the file name you specify. If the file to print is FOO.PAS, you only need type FOO. The extensions that TypeFile knows about, in order tried, are: .Pas, .For, .Micro, .Cmd, .Dfs, and .Doc.

If you omit the file name, TypeFile displays the default file. Unlike other commands, TypeFile does not change the default file name.

Switches are:

Switch	Description
-WAIT	This switch instructs Typefile to stop when it finds a formfeed character (CTRL-L) in the file. Thus the switch permits you to read the page before the screen is erased and the next page displayed. To continue, press RETURN. -WAIT is the default.
-NOWAIT	This switch instructs TypeFile to display the file without interruption.
-HELP	This switch displays a description of the TypeFile command and the associated switches. Note that -HELP does not display a file on the console.

UNALIAS

Unalias removes a command that was set with the Alias facility. The command to be removed must be typed exactly as it was initially given to Alias (as it appears when you type ?).

Format:

UNALIAS [command]

If Unalias is invoked without a command, it prompts for the command name.

USERCONTROL

UserControl is a facility available only to system administrators. Its purpose is to maintain the System.Users file on the authentication workstation which contains information about valid users and their passwords. This information is used by the Login program.

Format:

USERCONTROL [command] {-switch}

UserControl is described in detail in the Accent System Administration Manual.

VERBOSE

Verbose sets the flag that controls whether the shell command line is printed on the screen before it is executed. The flag applies only to shell lines that are read in command files, not those typed at the console.

Format:

VERBOSE [Boolean] [-switch]

If Verbose is set to True, each command line will be printed before it is executed. If Verbose is set to False, command lines are executed without being printed. You may type true, t, or yes for the boolean True and false, f, or no for boolean False. If Verbose is invoked with no argument, it will print out the value of the flag.

The only switch is -Help which displays information about the Verbose facility.

VERSION

Version displays the shell version number followed by a brief comment explaining the changes in this version of the shell.

Format:

```
VERSION
```

4. USER FACILITIES SUMMARY

```
100      Path [pathname]
101      Pause [message]
102      Pcc <filename.i> <filename.s>
103      Print <Filename>[,filename2,...filenameN] {-switch}
107      Priority [<Process> [<LevelNo.>]] [-Help]
108      PRQMic <InputFile> [-Help]
109      PRQPlace <RootFile> <ListingFile> {-switch}
110      QDis <SegFile> ~<ListFile> {-switch}
113      Quit [yes¦no]
114      Remote [-Help]
115      Rename <SourceFile>[~]<DestinationFile> {-switch}
118      Resume [Process] [-Help]
119      Run <ProgramName> [argument1 [argument 2]]
120      Scavenger [partition]
131      SetBaud [baudrate] {switch}
132      SetProtect <directory or filename> <-switch>
134      SETSEARCH [pathname][,pathname] {-switch}
136      SetTime [DD MMM YY] [HH:MM[:SS] {-switch}
138      Shell [-quit] [command] [-Help]
139      Show [<Name>] {-switch}
141      Speak [UserName] [-switch]
142      Statistics [Yes¦No] [-switch]
143      Stut [-Help]
144      Suspend [Process] [-Help]
145      TypeFile [FileSpecification][,file2][...,fileN] {-switch
146      Unalias [command]
147      UserControl [command] {-switch}
148      Verbose [Boolean] [-switch]
149      Version
```

Making Full Use of Speak and Listen

Ed. Zayas
19 June 1984

==========

Introduction

There are a lot of features of the Speak and Listen programs that have managed to escape the public eye, such as typescripting, broadcasting and multicasting. This document is meant to bring these little-known goodies up to the light of day.

Syntax/Switches

The Speak command line is patterned as follows:

```
Speak [MyName] [-PostPend=string] [-help]
```

I will use the convention that anything surrounded by brackets is considered to be optional. If MyName appears in the command line, Speak will call itself by that name. If it does not appear, then Speak will try to get its name from the environment. It will try TalkName, UserName and MachineName, in that order, taking the first one that is defined.

PostPend is a switch that should only be set by implementors. It has to do with choosing an appropriate postfix to the string name actually appearing in messages passing between Speak and Listen.

The Listen command line is patterned as follows:

```
Listen [MyName] [-Transcript=filename] [-help]
```

The optional MyName works exactly as in Speak. The Transcript switch allows all messages received by Listen to be stored in the named file (the default is no transcripting).

Namelists in Speak

It is well known that to address a message to a particular person, one has simply to start the message with that person's name (as known by Listen) surrounded by slashes. It is NOT commonly known that you may use a LIST of names instead of a single name. So, for example, if I want to address my messages to both Zayas and das, I would type the following to Speak:

```
/Zayas das/ Hey guys, this is going to both of you.
```

The above message would appear in both their Listen windows, and the next prompt you'd receive in Speak is /Zayas das/. Note that no commas are necessary (or allowed) between names. Until another destination is specified, all further messages will be directed to both Zayas and das.

Special Names

Some names have been reserved by Speak, and you are not allowed to use them for your own. If one of these names appears as the first name in a namelist, then special operations are invoked.

Who — Typing in /Who/ will result in the receipt of a message in your Listen window consisting of the names of the people currently running Listen, and hence every one who can currently be Spoken to. The responsibility for providing Who service is shared by everyone running Listen, and is totally invisible to the user. There is a single Listener at any time that has designated him/herself the WhoServer by checking in a special name with the network NameServer. This Listener will handle all Who requests. When the current Listener can no longer be the WhoServer (when it

taken on by the next Listener to request Who service.

- Finger Typing in /Finger [Name]/ results in a message to your Listen window containing "interesting" information about that person (if they are currently running Listen). The source for this information is that person's "Plan File", a normal text file containing whatever you want other people to know about you. If you define the PLANFILE environment variable, Listen will take that filename and use if to find your plan. If it's undefined, it will look for a file named PLAN in your boot partition. Note that if Name is not provided (just typing in /Finger/), then this is equivalent to a /Who/ request.

* Typing in /*/ [Message] causes your Message to be broadcast to everyone currently Listening.

Built-in Test for Listen

If you type in a line of this nature to Speak

<Name [NumMessages]

then you will send a series of messages to Name. If NumMessages has been specified, then that is the number that will be sent. A reasonable default will be selected otherwise. Note that there cannot be any blanks between the "<" and the start of the specified name.

Some Examples

```
/who/ ..........................  See who's out there.
/finger jbb/ ...................  What's in John Brodie's plan file?
/ssj das/ Accent question ......  Multicast your question.
/*/ Anyone want some pizza? ....  Broadcast around lunchtime.
<Zayas 20 ......................  Send Zayas 20 (numbered) messages.
```

PERQ Systems Corporation

TAPE STREAMER
USER'S GUIDE
FOR THE
ACCENT OPERATING SYSTEM

JUNE 1984

2600 Liberty Avenue
P. O. Box 2600
Pittsburgh, PA 15230.

Accent is a trademark of Carnegie-Mellon University.

Accent and many of its subsystems and support programs were originally developed by the CMU Computer Science Department as part of its Spice Project.

The information in this document is subject to change without notice and should not be construed as a commitment by PERQ Systems Corporation. The company assumes no responsibility for any errors that may appear in this document.

PERQ Systems Corporation will make every effort to keep customers apprised of all documentation changes as quickly as possible. The Reader's Comments card is distributed with this document to request users' critical evaluation to assist us in preparing future documentation.

PREFACE

The tape streamer is a recording device that is used to load a new operating system onto a PERQ workstation, save large groups of files on tape, or restore files from tape to a PERQ workstation. This guide describes how to connect and disconnect the tape streamer and how to user STUT, the streamer utility, with the Accent operating system. This guide assumes that you are familiar with basic operations on the PERQ workstation.

The following symbols have been used throughout this manual:

< >	Material that is to be replaced by symbols or text as explained in the accompanying text. Do not type the angle brackets. Example: <filename> indicates that you should type the name of a file.
[ ]	Optional feature. Do not type the square brackets.
{ }	0 to n repetitions. Do not type the braces.
CAPITALS	Literal, to be reproduced exactly as shown (although it may be reproduced in upper-case or lower-case). Example: <filename.CMD> indicates that the filename must contain the extension .cmd.
|	"Or"--choice between the items shown on either side of the symbol.
CTRL	Control key
RETURN	Carriage return
underlining	input to be typed by the user

Throughout the manual the term "PERQ" refers to both the PERQ workstation and PERQ2 workstation unless stated otherwise.

This manual is for using the tape streamer with the Accent

operating system only. If you are using POS, consult the STREAMER TAPE USER'S GUIDE FOR THE POS OPERATING SYSTEM.

TABLE OF CONTENTS

1. TAPE STREAMER INSTALLATION

A tape streamer is a recording device used to:

- load a new operating system
- save large groups of files (entire partitions with all of their directories and subdirectories)
- restore files from tape to a PERQ workstation.

One tape holds several large partitions, each in its own partition on the tape. A tape is a convenient way to store a large number of files for archival and safety purposes or to transfer the files from one PERQ workstation to another.

Individual files cannot be written to or read from tape, as is done with floppies. An entire partition must be transferred. If you are transferring more than one partition to tape, you must specify all the partitions to be transferred before the copying begins--you cannot write at the end of a tape.

1.1 Connecting the Tape Streamer

The tape streamer is box-shaped and comes with a power cord, a ribbon cable, and an interface PC board. The ribbon cable and the board provide the interface between the streamer and the PERQ workstation.

Obtain the streamer and a new or used tape cartridge from your system administrator. (If the arrow in the upper left corner of the cartridge points to "Safe," this indicates that the tape contains information that should not be overwritten. Do not use the tape.)

Proceed as follows:

1) Turn off the workstation.

2) Remove the front cover of the workstation.

3) If the machine is a PERQ workstation rather than a PERQ2 workstation, remove the optional Ethernet (IOB) board.

4) Unplug the ribbon cables in the workstation.

5) Insert the PERQ workstation/Streamer Interface PC Board in the I/O option slot.

6) Plug the 26-pin edge connector of the ribbon cable onto the JC connector (bottom connector of three) of the interface PC board, making sure that the red edge of the workstation/Streamer cable is up.

7) Plug the other end of the ribbon cable onto the back of the streamer drive unit, making sure that red marks are aligned.

8) Plug an A.C. power cord into the back of the streamer drive unit and turn on the power switch (put the switch in the "1" position).

   When power is applied to the streamer unit, it should make a clicking noise indicating head registration alignment.

9) Insert a tape cartridge into the front of the streamer, with the write-protection device on the cartridge placed in the upper left corner (away from you).

10) Turn on the PERQ workstation.

The tape streamer is installed and ready for use. Note that as operations are performed, STUT (the streamer utility, described in Section 3) may send messages to inform you of progress or to report problems. These messages are explained in Section 5.

1.2 Disconnecting the Tape Streamer

1) Turn off the workstation.

2) Turn off the streamer.

3) Disconnect and remove the PERQ workstation/streamer ribbon cables and interface PC board.

4) Reconnect the ribbon in the workstation. For a PERQ workstation but not a PERQ2 workstation, replace the Ethernet board in the IOB slot. Install the front cover on the machine.

5) Remove the tape cartridge, set the write protection to "Safe," and label the cartridge. Store it in a cool, dry place.

2. STUT, THE STREAMER UTILITY

STUT is the streamer utility program used to save and retrieve files on streamer tapes.

2.1. Installing STUT

The code of STUT occupies the following files, which are included with the Accent operating system. Both must be present in order to run the utility.

Stut.Run
TSIO.Bin

2.2 Running STUT

After loading STUT, as explained in the previous section, use this procedure to run STUT:

1) Connect the streamer and insert a tape cartridge as explained in Section 1.1.

2) Use the Path or Setsearch command to move to the partition where STUT is located.

3) Type stut [STUTcommand].

   If you have previously created a command file containing a sequence of STUT commands, invoke the file by typing stut @<filename>.

4) STUT checks to see if there is a tape in the streamer drive. If not, it prompts you to insert a tape. Insert a tape and press RETURN.

5) If a STUT command or a command filename was given in Step 3, the command(s) are executed. If not, STUT prompts for a command by displaying "STUT>".

6) At the "STUT>" prompt enter any of the STUT commands.

3. STUT COMMANDS AND OPERATION

The STUT commands are described below. They may be typed in any mixture of upper and lower case. A command can be abbreviated to just the number of characters needed to distinguish it from other commands. For example, s, p, and g are sufficient for the Select, Put, and Get commands respectively.

3.1 STUT Commands

HELP

Displays a summary of the commands.

QUIT

Terminates execution of the STUT program. (If the quit command is given in a command file invoked with "@", the command file is terminated and STUT resumes taking commands from the source of "@" (the keyboard or another command file).

SELECT <source> [~destination]

This command specifies which partitions will be used with the Put or Get commands, described below. All partitions to be copied or read must be specified before the Put or Get command; there is no way to write at the end of the tape or to start reading in the middle of a tape. Type the partition names one per line, each line beginning with select. The program will accept up to 10 Select lines; all of the lines will be executed during Put or Get.

The terms "source" and "destination" are relative to whether you are going to put or get files. To put files onto tape, the tape is the destination. The format of this command is:

SELECT <source> [~destination]

where "source" is the name of the partition to be copied and "destination" is the name of the partition where the copied files will be placed.

The name of a partition on the tape cannot exceed 25 characters.

If the destination name is omitted, the partition will be given the same name as the source. For example,

SELECT User

selects the User partition on disk and writes it to a partition on tape that also will be named "User." If you are writing to hard disk from tape and the disk does not contain a partition with the same name as the selected partition, you must specify a destination partition.

If the current path is to the floppy, the name of files from hard disk must include the full pathname. For example, to put the contents of hard disk partition User on tape under partition name StreamerUtility, type:

SELECT User ~StreamerUtility

Files retrieved from tape to hard disk must specify the hard disk partition where the file is to be placed. Continuing with the example above, to retrieve StreamerUtility from tape and place it in hard disk partition User type:

SELECT StreamerUtility ~User

If you make a mistake while entering Select statements, there is no way to delete a single selection. You can remove all Select statements by typing clear. However, there is no need to correct the error if you have specified a name that doesn't exist; that particular Select line will be ignored.

To get a list confirming the selections entered, type list.

PUT [annotation]

Transfers the partitions selected with the SELECT command from disk to tape. The annotation is a header line of information that identifies the contents of the tape (for example, user name, date, partitions copied). It is recorded on the tape and can be displayed by giving the HEADER command. If you do not give an annotation with the Put command or if you use the Put command in the form given below, STUT will prompt for an annotation.

PUT <source> [~destination]

This form of the command appends the new selection to the already selected partitions and then transfers all selected partitions to tape.

GET

Transfers from tape to disk the partitions selected with the SELECT command.

GET <source> [~destination]

This form of the command appends the new selection to the already selected partitions and then transfers all selected partitions to disk.

@<filename>

Specifies a command file from which commands will be taken. If the file has the extension .CMD, you do not need to type the extension. When "end of file" or "quit" is encountered in that file, commands are then taken from the keyboard (or from the command file in which the file is embedded, if any). The file may be "Console:" to take commands from the keyboard.

HEADER

Displays the tape header that was specified with the PUT command.

DIRECT

Displays the tape header plus the names of all files on the tape. "Direct" is short for "directory."

WIND

Moves the tape from its initial rewound position to the other end and then rewinds it. This procedure puts the proper tension on the tape and cleans off possibly damaging oxide particles. It is helpful to use the WIND command before writing on a new tape or before writing or reading a tape that hasn't been used for awhile.

REWIND

Moves the tape to its initial rewound position. After you write or read a tape, use this command.

ERASE

Erases the contents of the tape by moving the tape through the drive and then rewinding. This is useful if the contents of the tape are no longer needed and you do not wish to have anyone else see the contents. (Move the write-protection off "Safe" before erasing a tape.)

BUFFERS <nn>

Sets the number of buffers to allocate from non-screen memory. The default value is 90. This command is useful primarily with the GET command, as described in Section 3.3.

CLEAR

Removes all selections entered previously with the Select command.

3.2 Saving a Partition

STUT saves the entire contents of a partition with its directories and subdirectories:

1) Connect the tape streamer and insert a tape as explained in Section 1.1.

2) Use the Path or Setsearch command to move to the directory containing the STUT files.

3) Type stut.

4) If you have booted from floppy, set the number of buffers by typing buffers <20>.

5) If the tape is new or has not been used recently, type wind.

6) Use the Select command to specify each partition that is to be copied. Each partition must be specified on a separate line, beginning with select. Up to 10 "Selects" can be specified.

   When using Accent STUT, total pathnames for files are limited to 100 characters. You should try to keep your partition, directory, subdirectory, and file names reasonably short if you are planning to use STUT. Even though the partition name is all you type when using Select, STUT writes each file individually and uses the entire pathname. If the pathname is longer than 100 characters, the name is truncated and a warning message (listed in section 5.1) is issued. If the truncated name is the same as another filename on the tape (perhaps another truncated name), the previous file will be overwritten.

   If you make a mistake while entering Select statements, there is no way to delete a single selection. You can clear all selections by typing clear. However, there is no need to correct the error if you have specified a name that doesn't exist; that particular Select line will be ignored.

   If you want to get a list confirming what selections you have entered, type list.

7) Put a header line on the tape to indicate the contents by typing put <annotation>. If you type put, STUT will prompt for the annotation.

8) STUT then copies all of the selected partitions onto the tape. After each one, STUT will give the total number of blocks and files. Make a note of these figures so that when you retrieve files you will be sure you have them all. (It is a good idea to write these figures on the label of the tape cartridge.) When STUT is all through copying, it will display the total number of blocks and files, as well as final statistics.

9) Type rewind. (If the tape has rewound itself, a message appears.)

10) Leave STUT by typing quit.

3.3 Retrieving a Partition from a Streamer Tape

To retrieve one or more partitions from tape to the hard disk, follow the steps below.

1) Connect the tape streamer and insert a tape as explained in Section 1.1.

2) Use the Path or Setsearch command to move to the directory containing the STUT files or put the pathname where the STUT program resides in your Setsearch list.

3) Type stut.

4) If you wish to see the annotation line on the tape, type header. If you wish to see a list of the files on the tape, type direct.

5) If one of the following instances applies to you, set the number of buffers by typing buffers <nn>.

   a) If you have booted from floppy, set the buffer size to 20.

   b) If the partition(s) that you are GET-ting from the tape only contains small files (less than 60 blocks each), set the buffer size to be large (for example, 120).

   c) If the partition(s) that you are GET-ting from the tape only contains large files (like boot files), set the buffer size to be small (for example, 10).

   d) If you have a 2 megabyte memory board, set the buffer size to be huge (for example, 800 works nicely for streaming Accent).

6) If the tape has not been used recently, type wind.

7) Use the Select command to specify each partition that you wish to copy. Each partition must be specified on a separate line, beginning with select. Up to 10 "Selects" can be specified.

   If you do not have a partition on your workstation with the same name as the partition you are copying, you must specify a destination partition with the Select command. If the destination partition does not have enough room to

hold the partition being copied, you will receive an error message when the destination partition becomes full. You will have to delete some files, scavenge, and rerun the STUT commands. You do not need to disconnect the tape streamer.

If files on the tape have the same name as files already on the hard disk, the files on hard disk will be overwritten with those from the tape when the Get command is given. Other pre-existing disk files are not disturbed.

If you make a mistake while entering Select statements, there is no way to delete a single selection. You can clear all selections by typing clear. However, there is no need to correct the error if you have specified a name that doesn't exist; that particular Select line will be ignored.

To get a list confirming what selections you have entered, type list.

8) Type get. All of the files specified will be copied onto the hard disk. After each one, STUT will give the total number of blocks and files. (If you previously made a note of these figures when the files were put onto the tape, you can verify that you have received each file in its entirety.) When STUT is all through copying, it will display the total number of blocks and files, as well as final statistics. The statistics are discussed in Section 5.2.4.

   If you do not have a partition on your workstation with the same name as the partition you selected and you did not specify a destination partition, you will receive an error message. Then STUT will ask if you wish to continue getting the next selection, if any. You will have to rerun STUT to get the partition that was not copied.

9) Type rewind. (If the tape has rewound itself, you will receive a message.)

10) Leave STUT by typing quit.

The next section explains a shortcut method to use when you regularly backup one or more partitions.

3.4 Using a Command File for Routine Backup

It may be useful to periodically save the entire contents of a partition. A command file can be set up to simplify this task:

1) Create a file named Backup.Cmd in the partition to be copied. Put the following commands into the file:

```
SetSearch /Sys/<Dir>
Stut @Backup.StutCmd
```

where <Dir> is the location of the STUT utility files.

2) Create another file named Backup.StutCmd in the same partition. Put the following commands into the file:

```
Wind
Select <partition>
Put
Quit
```

where <partition> is the name of the partition to be copied.

3) Thereafter to backup the <partition>, just connect the streamer, insert a tape, and type:

```
path /sys/<dir>/
@Backup
```

After reading the Put command in the command file, STUT will prompt for a header to put on the tape.

4. STUT MESSAGES

STUT issues messages to keep you informed of progress and to report problems and statistics, using the following convention:

*<warning message> - These messages will not stop the execution.

**<error message> - These messages will stop the execution when they occur.

4.1 Initial Messages and Warnings

"Waiting for rewind"

This message may be followed by a sequence of dots while the streamer rewinds the tape. These dots merely indicate that the PERQ workstation is still working. When the rewind is complete an asterisk will appear and the program execution will continue. If more than one asterisk appears, an error has occurred.

"Initial status before writing"
"Initial status before reading"
"Status when starting tape movement"

These messages will usually end with the signal "- OK." If there has been a problem, STUT will display one or more of the messages listed in Section 4.2.

"*<message>"

Messages in this form are warnings and are accompanied by a beep. Among the warnings are these:

"Unable to make directories for file <filename>"

The filename includes an illegal partition name or directory name or that a file already exists with the

same name as one of the directory names.

"Could not create file"

The last file listed could not be entered in the directory for some reason. Possibly its name is illegal. (However, this message will not appear solely because the file previously exists so it is not a safeguard against overwriting files.)

"Replacing existing file"

The last file listed has the same name as one that previously existed on the disk. The older file is replaced.

"Name truncated to 100 characters"

The pathname for the last file listed is longer than one hundred characters and therefore only the first one hundred will be used when the file is put onto the tape.

4.2 Status Messages

"Streamer status - <message>"

This message appears at the end of tape operations, whether terminated normally or by some abnormal condition. If the operation was halted, the next line(s) that appear will indicate the conditions that caused the halt.

There are four categories of streamer status messages--hardware errors, operating problems, program errors, and information messages. These messages are described in the following sections.

4.2.1 Streamer hardware errors

"Streamer hardware error: Exception group 0 bit missing"
"Streamer hardware error: Superfluous exception group 0 bit"
"Streamer hardware error: Exception group 1 bit missing"
"(Undefined bit 4 was set)"
"(Undefined bit 2 was set)"

These messages all indicate that the streamer hardware is not performing as expected. Report the problem to your system administrator or PERQ Systems Corporation.

"Streamer not responding
Waiting at point <nn>
Last command was <mm>"

This is the most common problem with the streamer. There are three probable causes:

1. The streamer is not hooked up. This may be because the boards or cables are not tightly plugged together or because you have an older workstation that has not been modified for streamer use. (An older model is modified by removing and slightly rewiring chip U112.) Check that the machine has wires connecting the J166 pins on the backplane and, if so, verify that your IO board can use the streamer.

2. There is an unrecoverable read or write error on the tape. This will cause the tape to move back and forth sixteen times, which takes longer than the time-out period. You can try using the WIND command to put correct tension on the tape,

which will clean off oxide particles. If this fails when you are writing, try another tape. If you are reading, either keep trying or use another (perhaps earlier) input tape.

Note that the streamer makes sixteen retries for read errors. The "not responding" message occurs on those occasions when the time-out precedes the last retry. However, the first eight have all failed already.

3. The tape is wound too tight or loose, causing bits to get out of sync. In this case the streamer will simply stop moving. The WIND command will fix this situation.

4.2.2 Operating problems

"No cartridge detected in drive"

This message is self-explanatory. There must be a tape cartridge in the drive, and it should not be removed during execution of STUT.

"Write protected tape"

The arrow on the tape cartridge is pointing to "Safe." If you really want to write on this tape (thereby destroying existing data on the tape), turn the arrow around.

"No data found on streamer tape"

Possibly this tape has never been written on. If you are trying to retrieve information, you may have inserted the wrong tape.

"End of tape detected before end of data"

The output tape is full. There is no way to recover from this error. However, the tape still can be used later to transfer the files back to disk. All files will be loaded successfully up to the last file completely written to tape.

"Could not recover from tape data error"

Unfortunately, the tape is unreadable. Using the WIND command might make it readable again.

"Couldn't read a block, couldn't even find it"

This message may indicate that nothing has been written on tape. There is no way to recover from this error.

4.2.3 Program errors

"Program error: drive not on-line"
"Program error: Undefined command from PERQ"

If either of these errors occurs, please report it to your system administrator or PERQ Systems Corporation.

4.2.4 Information messages

"PERQ reset the streamer"
"Streamer at beginning of tape"
"File mark detected on tape"

These messages do not indicate an error; STUT is merely reporting on what is happening.

"<eee> Soft errors counted"
"<ooo> Speed mismatch count"

At the end of a read or write, these two lines reflect the performance of the streamer. The number <eee> indicates the number of times the streamer had to rewrite or reread a block because the first try failed. The count <ooo> is the number of times the streamer stopped and the PERQ workstation processed the buffers.

CARNEGIE-MELLON UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

SPICE PROJECT

Mercury

Mike Horowitz, Dave Nichols, Ed Smith

14 February 1984

Keywords and index categories: <not specified>
Machine-readable file: [x]/usr/spicedoc/manual/spiceuser/hg

Supported by the Defense Advanced Research Projects Agency, Department of Defense, ARPA Order 3597, monitored by the Air Force Avionics Laboratory under contract F33615-78-C-1551. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

Table of Contents

1. Introduction

Mercury is the electronic mail system for Spice. Mercury provides commands for reading mail, composing new mail and organizing old mail. Mercury under Spice is almost identical to Mercury under Unix[1] and is very similar to the RdMail system on Tops-10 systems. The Spice system provides a transport mechanism known as Mailman for getting electronic mail to and from a Perq. A separate document describes the Mailman program. Mailman is run as a background server and, except for initialization, will not bother the user.

1.1. Conventions

Mercury is a typescript oriented program, so commands and their arguments must be typed on a single line terminated by a *<return>*. Commands and many keywords can be abbreviated as long as the abbreviation is unique to Mercury. Mercury will complain, of course, if it does not understand your intentions. We will use **bold face** to indicate information typed by Mercury and *italics* to indicate something typed by a user. Note that though Mercury comes from a Unix environment, it is *not* case-sensitive.

Mercury and its accomplice Mailman assume they are the only processes running that can access your mail. Therefore, it is *foolish and dangerous* to run multiple instances of Mercury or Mailman on a single Spice Machine or to run multiple instances of Mailman on different Spice Machines for the same user.

Mercury uses the file *mail.hg* to store mail. This file is kept in the identical format used by Mercury under Unix. In particular, you can transfer your *mail.hg* file from your Vax area to use on your Spice Machine. (Be sure to ship the file in *BINARY* mode!)

New mail arrives from Mailman and is put in the file *<boot>perqinbox*. Outgoing mail is written into files that can be matched in a directory scan as *<boot>perqoutbox**. If you see any of these files in your *<boot>* area it is best to leave them alone unless you know what you are doing.

2. How to Get Started

To start Mercury, type:

mercury

Mercury will identify itself, tell you the number of messages found in your *mail.hg* file and give the number of new messages delivered since you last checked. Mercury will then give you the so-called "command prompt":

Hg>

Mercury uses several different prompts. The command prompt indicates that you can type ordinary Mercury commands. Other prompts, detailed below, accept a different set of commands and will *not* accept ordinary Mercury commands.

---

[1] TM, Bell Labs

3. How to Get Out

(subtitle: Sorcerer's Apprentice Section) To get out of Mercury, give the following command to Mercury at the command prompt:

Hg> *quit*

This command will get you back to the Spice Shell. This command, along with other ways of getting out of Mercury, are detailed below.

4. How to Get Help

Once inside Mercury, various help facilities are available. Typing:

Hg> *?*

will get you a list of commands and keywords used by Mercury. (This list was optimized to fit on precisely the screen of a *Concept 100*, and is thus fairly dense.) You can give the *help* command with an argument specifying what you want information about, and Mercury will oblige, often giving you more than you wanted. For instance, typing:

Hg> *help quit*

will tell you all about the *quit* command. Note that if there are several different topics that match the argument to the help command, you will be prompted for each topic. This prompt will ask for one of:

[Yes, No, Abort]

indicating whether you want to see this topic (answer *yes*), skip this topic and go to the next (answer *no*) or whether you have had enough and want to return to the **Command prompt** (answer *abort*).

Another help command is *syntax*. This command gives you the syntax that Mercury expects for a particular command. The following would give you the syntax for the *quit* command.

Hg> *syntax quit*

5. Basic Commands

Mercury maintains a list of all your messages and numbers them sequentially. Each message can be referenced with this *message number*. A *message sequence* is any combination denoting a series of message numbers or a message number. A *message sequence keyword* is an attribute of the messages, such as those that were answered or deleted The basic commands for reading and sending mail take either a message number sequence or a message keyword as its argument. A message sequence can be one of the following:

- empty, meaning the *current message*(your current message is the number of the last message in your mail file upon first entering mint)
- a single message number, meaning just that message (e.g. 57)
- a sequence of message numbers separated by commas, indicating those messages (e.g. 63, 25, 75)
- a message number, followed by an exclamation mark, followed by an integer *count*, indicating that message and *count* number of messages following (e.g. 37 !4 which is equivalent to 37, 38, 39, 40)
- a message number, followed by a colon, followed by another message number, indicating the messages from the first number to the second, inclusive (e.g. 37:41)

- the keyword *all*, indicating all messages (e.g. all)
- any message sequence keyword (explained later in the header command paragraph).

A *header* is one line of information containing various caracteristics about a message or group of messages. A header has the format of the following example:

41?-+30 Oct 83 John.Doe@Spice...foo(267)

This header indicates the following (from the left):

- **29** - the message number
- **?-+** - various flags indicating message attributes (described later with the *headers* command)
- **30 Oct 83 John.Doe@Spice** - the date of creation of the message and the sender's name
- **foo** - the Subject of the message
- **(187)** - the length, in characters, of the message.

The commands to be explained in the reading mail section are checkmail, current, type, previous, next, delete, expunge, undelete, exit, kill, headers. The commands in the sending mail section are mail, retry, answer, forward, and remail. The full details will be described in the more advanced commands section.

5.1. Reading Mail

When you start Mercury, it announces whether you have any new mail. For each piece of new mail, Mercury gives you a *header*. For example, Mercury might type the following when it first comes up:

[One new message found in mail box: 29]
29?-+30 Oct 83 John.Doe@Spice... foo (187)
Hg>

At any time in Mercury, the user can check on the existence of new mail with the *checkmail* command:

Hg> *checkmail*
[One new message found: 34]
Hg> *checkmail*
[No new mail found in mail box.]

If you type the *current* command, Mercury will show the message number of the current message, the total number of messages, and the mail file that is open.

hg> *current*
current message is 23 of 23 messages: mail.hg

To read an entire message, the user types the *type* command:

Hg> *type 29*
---- Message 29 is ----
Date: 30 Oct 83, 11:24:22
From: John.Doe@Spice
To: You
Subject: foo

Hi. Bye.
Hg>

Two other ways of typing messages are the *next* command and the *previous* command which type the next or previous message, respectively. An example (assuming the current message number is 3):

Hg> *previous*
---- Message 2 is ----
Date: 30 Oct 83, 11:24:22
From: John.Doe@Spice
To: You
Subject: foo

How's the Mercury document coming?
Hg> *next*
---- Message 4 is ----
Date: 30 Oct 83, 11:24:22
From: John.Doe@Spice
To: You
Subject: foo

Just checking.
Hg>

To delete a message, the user types the *delete* command:

Hg> *delete 29*

The *delete* command only marks messages to be deleted when the user executes the *expunge* command. Doing an *expunge* command will show you the headers of all messages to be deleted and prompt you for what to do.

Hg> *expunge*
29?-+30 Oct 83 John.Doe@Spice foo (187)>
Expunge deleted messages? [Yes, No, Abort]:

If you say *yes*, all messages marked for deletion will be deleted from the mail file and destroyed. If you say *no* or *abort*, the messages will not be deleted. Note that when you destroy messages with *expunge*, the messages that are left are renumbered.

The user can "undelete" a deleted message with the *undelete* command:

Hg> *undelete 29*

In the section above on how to get out of Mercury, we mentioned the *quit* command. Another way of getting out is to use the *exit* command.

Hg> *exit*
4?-+30 Oct 83 John.Doe@Spice foo (187)>
Expunge deleted messages? [Yes, No, Abort]:

Notice that the *exit* command performs an *expunge* command before leaving Mercury. Answering *yes* will destroy all deleted messages and leave Mercury. Answering *no* will simply leave Mercury. Answering *abort* will not destroy any messages and not leave Mercury. You will be returned to the command prompt.

A useful shorthand is the *kill* command, which does a *delete* followed by a *next*. So, if the current message is 3, then executing the *kill* command will delete message 3 and type message 4, like so:

Hg> *kill*
---- Message 4 is ----
Date: 30 Oct 83, 11:24:22
From: John.Doe@Spice
To: You
Subject: foo

Just checking.
Hg>

This is particularly useful when going through the morning's mail, deleting the message you just read and showing you the next one. Getting into the habit of typing this to read all your newest mail can unfortunately delete mail you wish to keep: be careful!

The *headers* command can be used to type the headers for the message or messages indicated.

Hg> *headers 3*
3 * John.Doe@Spice... foo (145)

In particular, using the keyword *all* with the headers command:

Hg> *headers all*
1 John.Doe@Spice... more foo (153)
2 John.Doe@Spice... foo (512)
3 * John.Doe@Spice... foo (145)
...

prints the headers for all messages in the user's mail file. Each header has a set of flags printed after the message number. These flags have the following meaning:

- ? the message has not been answered (see the *answer* command below on sending mail)
- + the message is new, (that is, this message was read in from the mailbox during this run of Mercury)
- - the message has not been examined (that is, typed)
- * the message has been marked for deletion
- B this is a *blind* copy of a message (described below on sending mail)
- D this is a draft of message (described below on sending mail).

Each of these flags can be used as message sequence keyword. For instance, to see the headers for all deleted messages, type:

Hg> *headers deleted*
3 * John.Doe@Spice... foo (145)

Or, to type all answered mail, type:

Hg> *type answered*

5.2. Sending Mail

There are various ways to create and send mail with Mercury. The basic command is just:

Hg> *mail*

The *mail* command takes as an optional argument the name of the recipient of the message. This command runs the editor, which under Spice is Oil. Upon invoking the editor, the screen will display a *mail template* which initially looks something like the following:

Date: 30 Oct 83, 11:31:22
From:
To:
cc:
Subject:
Message-ID: <1983.10.30.11.31.22.mumble>

<< Put body of message here >>

When you fill in the template, it is essential to fill in the **From** and To fields in order for Mercury to deliver your mail. You can leave the cc field (a remarkable holdover from the days when typewriters struck paper and could make "carbon copies") blank. You can also include the value of the To field with the original *mail* command. So, typing the command *mail Joe* creates a template whose To field is already filled in with "Joe". In order to let Mercury know where the message heading ends and your message begins, be sure to *leave a blank line between the Message-ID and the first line of your message!* Using the Spice editor, remove the line about the "body of message" and include your message there.

After exitting the editor normally, you will then be back running Mercury. Mercury will then give you the "action prompt", which looks like this:

Action (ABort, Blind, Draft, Edit, Mail, Type):

The prompt expects one of the following:

- *Abort* - this will return you to the command prompt, destroying the message you created
- *Blind* - this will create a blind copy of the message, place it in your mail file, and return to the action prompt
- *Draft* - this will create a draft copy of the message, place it in your mail file, and return to the command prompt
- *Edit* - this will return you to the editor to work some more on your message
- *Mail* - this will actually mail your message, returning you to the command prompt
- *Type* - this will type your message, and then return you to the action prompt.

Note that several of the above options return you to the action prompt. Thus, you can make a blind copy of the message, type it out, and then mail it before returning to the command prompt. Draft messages are mail that Mercury assumes you will eventually get back to editing again.

The *retry* command on a draft message puts you in the action prompt for that message, allowing you to edit it some more or mail it. So, if message 6 is a draft of a message, then you can do a *retry* command on it, and you will see the following:

Hg> *retry 6*
Action (ABort, Blind, Draft, Edit, Mail, Type):

A message can be retried (i.e. - re-edited) as many times as desired as long as you type *Draft* at the action prompt. As with all the other basic commands, *Retry* can be given a message sequence of more than one number. Mercury will succesively prompt for each message. Draft messages are deleted after they are mailed. Note you cannot use the *mail* command with a message number to send draft mail since the *mail* command will interpret the message number as the value of the recipient of a new message template!

Another way to create mail is to use the *answer* command. This command, followed by a message sequence allows you to "answer" messages. That is, it creates a message template whose To field is a list containing the names of the senders of all the messages in the sequence, the cc field is a list of all people who received copies of the original mail, and the Subject field is a line begining with "Re:" followed by the subject of the first message in the sequence. Except for these changes in the message template, this commands acts like the *mail* command. Of course, as with all commands that require a *message sequence*, you can leave it blank to act on the current message. The following command will answer the current message:

Hg> *answer*

You can forward a message with the *forward* command. This command takes as its arguments a message sequence, optionally followed by a slash character (/) and a list of recipients. This acts much like the *mail* command. The difference is that the message template created for you to edit will contain a copy of the message you are forwarding. So, executing the forward command:

Hg> *forward*

might create a template like the following:

Date: 30 Oct 83, 11:31:22
From:
To:
cc:
Subject:
Message-ID: <1983.10.30.11.31.22.mumble>

<< Put body of message here >>
--- Begin forwarded message ----
Date: 30 Oct 83, 11:31:22
From: John.Doe@Spice
To: You
cc:
Subject: foo
Message-ID: <1983.10.30.11.31.22.mumble>

Hi.
---- End forwarded message ----

A faster way of forwarding mail, especially when you don't need to add any text to the message, is the *remail* command. This command takes as its arguments a message sequence, a slash character (/), and a list of recipients. If the list of recipients is omitted, the command prompts for them.

Hg> *remail 3/John.Doe@Spice*

6. Advanced Commands

This section details more advanced features of Mercury. Included in this section are more details about messages and message headers as well as the various ways Mercury provides to organize and search through messages.

6.1. Conventions

Mercury uses a couple of environment variables to find some information about the user. The environment variable *MailAddress* is looked up when Mercury initializes to find the mail address of the user. The value of this variable is used to fill in the **From** field in message templates. You should make this something like <VaxAccount>@CMU-CS-SPICE or wherever you have mail sent. If this value is null, then the **From** field will always be blank. The user must fill in this field or Mercury will refuse to deliver the mail. This field should contain something that a recipient of your mail could use to send a reply to.

Another environment variable looked up by Mercury is *MailFile*. The value of this variable is taken as the name for the user's default mail file. If this value is null, then the name "mail.hg" is used. Note that no specific directory is given, so the search path will be used to find this file. The user can change mail files at any time with one of the commands *read* or *qread*, described below in the section on mail files.

6.2. Message Headers

Each message contains a large amount of information in its header. The header is that part of the message at the top that contains various fields such as **Date**, **From**, **To**, and so on. You can see the entire header of a message with the *wholeheaders* command:

Hg> *wholeheader 3*
3 + 7 Nov 83 John.Doe@Spice foo (154)
Date: 7 Nov 83 1039 EST (Monday)
From: John.Doe@Spice
To: You
Subject: foo
Message-Id: <07Nov83.103922.mumble>

Another command to see information about a message is the *whereis* command. This takes a message sequence and returns all sorts of atributes and parameters kept by Mercury about your messages:

Hg> *whereis 3*
3 + 7 Nov 83 John.Doe@Spice foo (154)
System attributes(s): Answered Examined New
Class name(s): foo

Class names are discussed in a subsequent section on using mail files as a data base.

6.3. Mail Files

Mercury provides various mechanisms for organizing mail. One way is with multiple mail files. When Mercury first comes up it looks for your default mail file. This file is found in the environment variable *MailFile*. (If *MailFile* is null, then the name "mail.hg" is used.) You can change mail files within Mercury

with the *read* command. This command takes the name of a mail file as its only argument. The current mail file is exitted:

Hg> *read old>mail.hg*
[There are 20 messages in >sys>spice>old>mail.hg]

Note that the *read* command does an *exit* on the current mail file before reading in the new one. If you wish to *quit* out of the current mail file instead of *exit* (that is, you do *not* want to do an *expunge* of deleted messages), then do a *qread* command.

You can create a new, empty mail file with the *create* command. This command takes the name of the mail file you want to create and creates it:

Hg> *create newmail.hg*
[Creating newmail.hg]

Of course, if you try to create a mail file that already exists, Mercury will complain.

To put messages into a mail file from the current mail file, use the *put* command. This command takes a message sequence and a mail file name (separated by a slash, "/") and creates copies of the messages in the mail file. Mercury tells you the current number of messages in the mail file before it starts, then the number of messages when it's done:

Hg> *put 3/newmail.hg*
[There are zero messages in newmail.hg]
[There are now one messages in newmail.hg]

Another similar command is the *move* command. This takes the same arguments as *put*, but marks the messages moved for deletion in the current mail file. These messages will of course be deleted during the next expunge.

6.4. Classification of Messages

Another way of organizing messages is to use message classes within an individual mail file. First, to create a message class, you use the *allocate* command:

Hg> *allocate myclass*

To see a listing of all classes, use the *display* command:

Hg> *display*
myclass 0 Unclassified 4

Note that there is an implicit class of "unclassified" messages. The number to the right of the class name is the number of messages in each class.

To add a message to a class, you use the *classify* command. This command takes a message sequence, followed by a slash, followed by a class name, like so:

Hg> *classify 1,2,3/myclass*

Doing a *display* command after this reveals what you would expect:

Hg> *display*
myclass 3 Unclassified 1

Note that the number of messages in all classes will *not* necessarily add up to the total number of messages in the mail file. This is because a message may belong to more than one class:

Hg> *alloc anotherclass*
Hg> *classify 1/anotherclass*
Hg> *display*
anotherclass 1 myclass 3 Unclassified 1
Hg> *headers all*
1 7 Nov 83 John.Doe foo1 (153)
2 7 Nov 83 John.Doe foo2 (134)
3 7 Nov 83 John.Doe foo3 (255)
4 7 Nov 83 John.Doe foo4 (231)

A quick way of seeing all messages that have not yet been classified (to me, that means messages I haven't dealt with yet), do a *headers not classified* command.

Messages can be removed from classes with the declassified command:

Hg> *declassify 1/anotherclass*

Classes are destroyed with the *deallocate* command:

Hg> *deallocate anotherclass*

If you attempt to deallocate a class that is not empty, Mercury will complain. A quick way to declassify all messages in a class is:

Hg> *declassify anyclass / anyclass*

This creates a message sequence of all messages in the class *anyclass* and deletes all of those messages from the class *anyclass.*

6.5. Using Mail Files as Data Bases

Mail files are actually maintained as data bases of messages and related information. You can, for instance, substitute for any message sequence an expression that evaluates to a message sequence. This expression can test message attributes, class names, fields of messages headers (such as **From**, **Subject**, **To**, CC, etc.), and so on. This expression can includes unions, intersections, subtractions and negations. The expression can use the following attribute names to match messages: **Answered**, **Blind**, **Deleted**, **Draft**, **Examined**, **Kept**, **New**, **Newest** or **Private**. The expression can also use the following message sequence keywords to match messages: **All**, **BackReference**, **Before**, **BodySearch** **Classified**, **Context**, **During**, **FieldSearch**, **Handled**, **HeaderSearch**, **Last**, **MessageSearch**, **NextLast**, **Processed**, **Reference**, or **Since**.

The best reference to such expressions is in the help facilities provided within Mercury. Help is provided about "message sequence", "system attributes" and "set expressions" that should explain more than you will want to know about using this facility.

An interesting command to use with this mechanism is *context*. This will establish a sub-set of messages in the current mail file that are visible. Mercury will change its prompt (by adding an extra greater-than sign) to indicate that you are nested within a context. This command can be executed again, within a context, to restrict even further the messages visible to the user. An example:

Hg> *context since 7-nov-83*
[3 messages]
Hg>>
context myclass
[1 message]
Hg>>>

To get out of a context, execute a *pop* command. This will leave the current context and return to the next higher context. Mercury's prompt will change indicating the change in context.

7. Summary of Commands

The following sections gives a brief summary of the commands in the order described above.

7.1. Getting Started

- *quit* - exits Mercury immediately; no messages marked for deletion are actually deleted;
- *?* - gives a brief summary of all commands and keywords known to Mercury;
- *help <command-or-keyword>* - takes a command name or keyword as its argument and types out much help information about the topics in Mercury that match the argument;
- *syntax <command>* - takes a command name for its argument and returns a brief description of the syntax and arguments expected for the command;

7.2. Basic Commands

- *current* - shows you some information about the "current" message, including its number and the mail file currently open (default is *mail.hg*);
- *checkmail* - checks on new messages in the mail box; if there are any, Mercury reads them and assigns them message numbers;
- *type <message-sequence>* - types the messages specified; if no message is specified, this types the current message;
- *next* - types the message after the current message;
- *previous* - types the message preceding the current message;
- *delete <message-sequence>* - marks for deletion the specified messages; if no message is specified, then this deletes the current message;
- *expunge* - destroys all messages that have been marked for deletion;
- *undelete <message-sequence>* - undeletes the specified messages (that is, removes the mark for deletion for each message in the *<message sequence>*); this command will *not* restore a message destroyed by the *expunge* command;

- *kill* - performs a *delete* command on the current message, followed by a *next* command;
- *headers <message-sequence>* - types a header for the messages specified;
- *mail <recipient>* - lets the user edit a message template, then upon return from the editor, delivers the message to the recipient;
- *retry <message-sequence>* - allows the user to retry a draft message;
- *answer <message-sequence>* - answers the specified messages; this command works just like the *mail* command, obtaining the recipient of this message from the source of the message(s) being answered;
- *forward <message-sequence>/<recipient>* - allows the user to forward mail to another recipient; the mail being forwarded is included as a part of the message template;
- *remail <message-sequence>/<recipient>* - is a short version of the *forward* command; this command simplies mails the messages to the recipients specified without letting the user edit the messages before they are delivered;

7.3. Advanced Commands

- *wholeheaders <message-sequence>* - types the entire message header for each of the messages given;
- *whereis <message-sequence>* - types system attributes for all messages given;
- *read <mail-file>* - exits from the current mail file (doing an expunge of all deleted messages) and reads in the given mail file;
- *qread <mail-file>* - reads in the given mail file, doing a *quit* of the current mail file;
- *create <mail-file>* - creates a new, empty mail file of the given name;
- *put <message-sequence> / <mail-file>* - puts the specified messages in the specified mail file;
- *move <message-sequence> / <mail-file>* - puts the specified messages in the specified mail file, marking the messages for deletion in the current mail file;
- *allocate <class-name>* - allocates a new class of the given name;
- *display <class-name>* - displays all classes and the number of messages in each class;
- *classify <message-sequence> / <class-name>* - adds messages to a specified class;
- *declassify <message-sequence> / <class-name>* - removes messages from a specified class;
- *deallocate <class-name>* - destroys a class;
- *context <message-sequence>* - establishes a context out of the specified messages;
- *pop* - returns from the current context to the next higher context;

7.4. Action-Prompt Commands

The following are possible answers to the action prompt:

- *Abort* - cancel the command that is awaiting action;
- *Blind* - create a blind copy of the message involved and return to the action prompt;

- *Draft* - create a draft copy of the message involved and return to the command prompt;
- *Edit* - enter the editor on the message involved; Mercury will display the action prompt again when the user returns from the editor;
- *Mail* - mail the message and return to the command prompt;
- *Type* - type the message and return to the action prompt;

THE EDITOR

June 8, 1984

2600 Liberty Avenue
P. O. Box 2600
Pittsburgh, PA 15230
(412) 355-0900

Accent is a trademark of Carnegie-Mellon University.

Accent and many of its subsystems and support programs were originally developed by the CMU Computer Science Department as part of its Spice Project.

The system described in this manual is based upon the Editor program by Keith Wright; Ray Scheboth of PERQ Systems assisted in implementation.

The information in this document is subject to change without notice and should not be construed as a commitment by PERQ Systems Corporation. The company assumes no responsibility for any errors that may appear in this document.

PERQ Systems Corporation will make every effort to keep customers apprised of all documentation changes as quickly as possible. The Reader's Comments card is distributed with this document to request users' critical evaluation to assist us in preparing future documentation.

TABLE OF CONTENTS

THE EDITOR

1. Introduction

The Editor is a text-editing program provided with the Accent operating system. As with any editor, you may use the Accent Editor as you create a new file or to make changes to an existing file.

Two especially convenient features of the Editor are:

1) The Editor allows you to work in up to nine files at a time, which is very useful for making similar changes to more than one file, copying parts of files, or looking up information in other files as you work. Each file is opened in a new window, and you can easily move from window to window to read or make changes.

2) In giving commands, you may press either the CTRL key on the keyboard or a button on the mouse (the middle button on a 3-button mouse or the yellow button on a 4-button mouse).

Sections 2 through 8 of this manual contain basic information on using the Editor: how to enter the Editor; how to close windows and leave the Editor; what the display looks like; how to use the keyboard and mouse; how to save your work; how to get on-line help; and how to move quickly through a file by thumbing or scrolling. Section 9 covers the commands for making changes, broken down by tasks, and Section 10 explains how to use multiple windows. Section 11 covers changing the keys associated with commands. A summary of the commands appears at the end of this manual.

It is recommended that you do not edit in Accent a file that was created in POS or vice versa, as the Accent Editor takes out all carriage returns.

Unless specified otherwise, throughout this manual "PERQ" refers to both the PERQ and PERQ2.

2. Invoking the Editor

To create or edit a file, type ed /pathname/<filename> -{switch}. Valid switches and their purposes are:

-help	gives a message telling you to hit the help key once the Editor is running.
-log	creates a transicrpt file of the editing session which can be played back to recover lost changes in the event of a problem. This is the default.
-nolog	does not create a transcript of the session. Use this switch with caution, since any changes lost with it in effect cannot be recovered.
-replay	replays transcript of the last editing session to allow recovery of lost changes. See Section 9.5 for complete instructions on the use of this switch.

Alternatively, you may simply type ed, and the Editor will access the last file edited. However, if you specified switches the first time you edited the file, they are disregarded and the default -log switch is used.

If you make a mistake in typing the command or filename, you may use the commands explained in this manual for editing within a file (except those commands which move the pencil cursor from line to line). If you include an extension with the filename, the Editor will attempt to find that file. If you do not specify an extension, the Editor will try to find such a filename with the extension .pas, .pasmac, .mss, .cmd, or .micro (in that order). If it finds no such file, a new file is created.

The Editor also has an escape completion feature whereby you can type just as much of the filename as is needed to distinguish the file from other files and then press INS on a PERQ or ACC(ESC) on a PERQ2. (NOTE: Throughout the rest of this manual, we will refer to this key as the INS key. If you have a PERQ2, you should recognize that this is the same as your ACC(ESC) key.) If you have supplied enough of the filename to identify it, the Editor will supply the rest of the name. If you have not typed enough of the filename, the Editor will list existing files beginning with the letters typed so far, then echo your line and wait for more information.

After you have entered the Editor, the filename is shown in the prompt window. The pencil cursor is positioned at the beginning of the file if you have accessed an existing file. If a new file is

being created, the pencil cursor is positioned after the filename so that you can specify a different name if you wish; to move the pencil cursor from the filename into the file, press RETURN.

After you have accessed one file you can open windows for other files, as explained in Section 10.

The format of the display is described in Section 4.

3. Closing Windows/Leaving the Editor

To close all of the windows you have opened and exit the Editor, give the command xf. (While you type each letter of the command, hold down either the CTRL key or the middle or yellow button on the mouse. If you don't know how to use CTRL characters, see Section 5.) If you have opened more than one window, each of the windows is closed in turn.

If you want to stay in the Editor but close a particular window, move the pencil cursor into that window and issue the command xd.

As each window containing an updated file is closed, the Editor turns the window to reverse video and prompts for an action from you. A default name is shown (the original file or the file into which you last wrote the changes during the current editing session). Do one of the following:

> To write the changes to the default file, press RETURN.
>
> To write the changes to a different file, edit the filename. (If you wish to write the file to a different directory than that shown in the prompt region, but keep the same filename, retype the pathname. If no pathname is specified, the file goes into the current directory.)
>
> To return to the window, press DEL (to delete the exit command).
>
> To exit without saving the changes you have made, press the OOPS and RETURN keys in succession.

The Editor has several safeguards to help you avoid costly mistakes:

a) As mentioned above, as you close each window the Editor asks for the name of a file into which to write the changes.

b) When you write out changes, the system keeps a copy of the original file as it was before you edited it. This copy has the same name but with a dollar sign appended to the end of the name. This provides a safeguard in case you decide you want to keep the file as it was before editing or in case the system or program malfunctions while you're editing. (If the program malfunctions, do not turn off the machine or reboot; see the instructions in Section 6.)

If for some reason you decide to keep both the original file and the edited file, you should rename the original. Otherwise, when you edit and save again, the original file will be lost, as only one backup copy is kept.

c) If you have accessed a file in order to read it and wish to safeguard aginst keeping any changes you may have accidentally made, pressing OOPS and RETURN exits without saving.

d) The Editor retains a transcript of the text and all of the commands you issue. This transcript can be replayed, as explained in Section 9.5, and therefore it provides a way to recover the changes you've made even if you exited the editor without saving your work or in case of machine crash.

4. The Display

When you invoke the Editor, two windows are provided initially--a small prompt window and a large text window.

The prompt window presents information on the current state of the editor, editor version number, error messages, prompts, and the contents of the yank buffer (explained in Section 9.4).

The text window is divided into three areas:

1. The status bar, a horizontal bar containing the name of the file being edited and a thumb bar for searching through text to a particular place.

2. The text of the file you specified (or for a new file, blank space).

3. The scroll bar--a vertical bar to the left of the text which allows you to move sequentially through the file.

You may open more windows at any time, as explained in Section 10.

The screen contains an arrow (which is manipulated with the mouse) and a pencil cursor (a short vertical blinking bar between two characters). The arrow is used to position the pencil cursor, as explained in Secton 5. The arrow can be moved freely within a window or from one text window to another. The pencil cursor marks the "current position." Any characters you type will be inserted at that point and any commands you issue will affect the text before or after the pencil cursor (depending on the command).

5. Using the Keyboard and Mouse

As you are editing, you may use these keys to perform their normal functions:

BACKSPACE - delete the character before the pencil cursor

OOPS - delete the line before the pencil cursor

RETURN - insert a new line and move the pencil cursor to the beginning of that line

Editor commands consist of one or two characters or keys, typed either while the CTRL key is pressed or while the mouse button (middle or yellow) is pressed. If the command involves two letters, such as CTRL-x CTRL-f, hold the CTRL key or button down while you press each of the letters.

In the Editor the commands are case sensitive, that is, f is a different command than F.

The mouse has three general uses:

1) To change the position of the arrow and pencil cursor

   The mouse controls the location of the arrow on your screen. The arrow's position does not affect editing unless you press the mouse. However, if the arrow bothers you, you may move it out of the area in which you're working by picking up and moving the mouse.

   In the Editor, the Sapphire window manager causes the mouse to operate in relative mode--that is, the pencil cursor position is determined by the difference between the previous and present tablet coordinates. The pencil cursor does not necessarily match the position of the mouse. For example, if you lift the mouse from the upper left corner of the tablet and place it in the lower right corner, the pencil cursor does not move. The next pencil cursor position is determined from mouse movement while in contact with the tablet. You can move the pencil cursor large distances by making several swipes with the mouse; be sure to lift the mouse far enough from the tablet.

   To change the current position of the pencil cursor, slide the mouse until the arrow points to the exact spot where you want the pencil cursor. Then press and release the middle or yellow button. The pencil cursor will move to

the position indicated, and anything you type will be inserted at this new position and any commands you issue will affect the text before or after the pencil cursor (depending on the command).

2) To invoke an editing command without using the CTRL key

   Editor commands can be given using the mouse instead of the CTRL key. Simply depress the middle or yellow mouse button, issue the desired command, and release the button.

3) To thumb and scroll through your text

   This procedure is explained in Section 8.

Because the middle or yellow button on the mouse is used both to position the pencil cursor and to give commands, it is convenient to position the arrow, press the button (thereby moving the pencil cursor to that location), keep the button down, and issue a command.

6. Saving Your Work/Recovering from Disasters

If you wish, at any time you may save the changes you've made by issuing the command xs (save). The changes will be written to the file shown in the status bar. However, you do not have to remember to save your work, as the Editor will ask if you want to save before it closes any window. The Editor also has other safeguards built in, as explained in Section 3.

If a malfunction should occur while you're in the editor and you can't save or exit the file, do not turn off the machine or reboot. You may lose your file. Instead type <prefix>C. On a PERQ workstation, the prefix is CTRL-DEL; on a PERQ2, it is SET-UP. This will take you out of the editor and you can then regain the changes you had made by playing the transcript (see Section 9.5).

7. Getting On-line Help

To obtain a summary of Editor commands while in the Editor, press the HELP key. A new window containing the summary is opened. The help file is several pages long; use the Editor commands to move through the file.

To delete the help window, position the pencil cursor within the window and issue the command xd.

8. Thumbing and Scrolling Through a File

Thumbing through a file gives rapid access to any part of the file. The thumb bar is a horizontal bar located in the status bar above the text region. The file is linearly mapped onto the thumb bar, with the part of the file that is visible in the text window represented as a black square within the bar. To move to a different portion of the file, move the arrow to the approximate place on the bar that would represent that portion of the text (the shape of the arrow will change) and press the middle or yellow button.

Scrolling moves you sequentially through a file, either forward or backward. The scroll bar is a vertical bar to the left of the text. Move the arrow into the scroll bar (the shape of the arrow will change) and then do one of the following:

1) To move the line that is now next to the arrow to the top of the window, press the right or green button;

2) To move the line that is now at the top of the window down to the position of the arrow, press the left or white button;

3) To scroll one page at a time, successively press and release hold a button as follows:

   right or middle on a 3-button mouse; yellow or green on a 4-button mouse - to go forward (toward the end of the file).

   left or white - to go backward (toward the beginning of the file).

4) To scroll using the keyboard only, type either:

   CTRL-! (scroll-up-pencil-cursor) - to move the current line (the line containing the pencil cursor) to the top of the window;

   CTRL-x CTRL-! (scroll-down-pencil-cursor) - to move the current line to the bottom of the window.

9. Making Changes

Editing in the Accent Editor involves two steps--moving the pencil cursor to the portion of the text that you want to change and then making the change. The following sections give all of the Editor commands, divided into groups according to their purposes. All commands are summarized at the end of this manual.

Remember that in the Editor:

a) commands are case sensitive (for example, f is a different command than F); and

b) while you issue the command you must press either the CTRL key or a mouse button (middle button on a 3-button mouse or the yellow button on a 4-button mouse).

A few commands include two characters (for example, xf to leave the Editor). In these cases you must press the CTRL key or button while you type both characters.

The PERQ has an auto-repeat feature. If you hold a key down for more than half a second, the key repeats at a rate of about 10 times per second on the PERQ and 16 times per second on the PERQ2. Therefore you can quickly issue the same command several times in succession (handy, for example, in moving forward several characters or down several lines).

If you wish you may change the specific keys that are used to invoke each command; see Section 11.

9.1 Moving the Pencil Cursor

The commands in this section explain how to manipulate the pencil cursor so that you can move through a file in order either to read the file or to make editing changes. (Remember that you can also move the pencil cursor by moving the mouse and pressing a button and by thumbing and scrolling through the file.)

To make the commands easy to remember, they are often the first letter of a word which is meaningful in the context of the command (shown in parentheses below). The commands are:

f (forward-character) move forward one character. If the pencil cursor is at the end of a line, it moves to the beginning of the next line.

b (backward-character) move back one character. If the pencil cursor is at the start of a line, it moves to the end of the previous line.

F (Forward-word) move to the end of the next word.

B (Backward-word) move to the beginning of the previous word.

a (alpha-of-line) move to the beginning of the line.

e (end-of-line) move to the end of the line.

n (next-line) move to the next line. The pencil cursor "remembers" its column location as it moves down.

p (previous-line) move to the previous line (same column). As with the "n" command, the pencil cursor remembers its column location.

v (view-next-page) move forward one page. This command advances you to the next page by scrolling the text up the height of the screen. The pencil cursor remains in the same row and column on the screen. If this action moves the pencil cursor to the end of the file, the text is scrolled only until the pencil cursor is at the end of the file.

V (View-previous-page) move back one page. The command

returns you to the previous page by scrolling the text down the height of the window.

, (top-of-window) The comma command moves the pencil cursor to before the first character of the first line in the current window.

. (bottom-of-window) The period command moves the pencil cursor to before the first character of the last line in the window.

< (top-of-file) The left angle bracket command moves the pencil cursor to before the first character of the first line in the file.

> (bottom-of-file) The right angle bracket command moves the pencil cursor to after the last character in the file.

X! (pencil-cursor-left-column) This command moves the current line of text to the left, so that the column containing the pencil cursor becomes the far left column. (This is useful if you want to see lines that are extending off the right side of the screen.)

9.2 Inserting Text

Insert mode is the normal mode in the Editor. Therefore if you want to insert text, just move the pencil cursor to the position where you want to insert and then type the desired text. The text is inserted before the pencil cursor.

If you do not have the Word Wrap option on, as explained below, lines will extend off the right of the screen if you do not press RETURN before you reach the right edge. As you are inserting, the screen moves to keep up with the pencil cursor. When you finish a line, RETURN or CTRL-A puts you back in the farthest left column, and a black square in the far right column indicates that the line continues off-screen.

There are several parameters you can set when insertng text: right margin, left margin, tab length, and word wrap. To set any of these, first press CTRL-SHIFT-P. You will then be prompted for the parameter. Do the following:

Left margin
: Type l. You will be prompted for a value. You cannot supply a negative number, and the number must be smaller than the right margin. The default is 0.

Right margin
: Type r. You will be prompted for a value. The default is 70.

Tab
: Type t. You will be prompted for a value. The default is 8. (If you set the tab at 4, for example, a tab is set every 4 spaces across the screen. After you set this parameter, each time you press the TAB key the pencil cursor will move to the next setting.)

Word wrap
: Typing w reverses the ON/OFF setting. The default is OFF. (If you set Word Wrap ON, when the pencil cursor reaches the right margin it will go to the next line, taking with it the last whole "word" (all the characters typed since the last space).

Status
: To see the values that are in effect for the above parameters, type s.

There are some special insert commands, listed below, which you may want to use occasionally.

o
: Insert open space. This command splits the current line into two lines, but does not advance the pencil cursor (leaving you in position to insert text at the end of the first line).

q<character>
: Insert quote character. This command tells the Editor to insert the next character you type after q as a command to be inserted into the file. For example, to insert a control l (page break) in your file, you would type CTRL-q CTRL-l. The control character (the letter "l" in this example) will be inserted into the file.

y
: Insert yank-buffer. This command inserts the contents of the yank buffer at the current position of the pencil cursor, therefore providing a way to undo a delete command or to transfer or copy text from another place. See further discussion in Sections 9.4 and 9.9.

In addition to the above commands, the following special keys may be used:

TAB key
: Insert tab. This command inserts the number of spaces needed to move the pencil cursor forward 8 spaces. (The number can be changed, as explained above.)

RETURN key
: Insert new line. This command inserts a carriage return (splitting the line at the current position)

LF (linefeed)
: Insert carriage return and indent. This command inserts a carriage return, as explained above, and indents the second line as many spaces as the first line is indented.

9.3 Deleting Text

The commands shown below delete text.

d (delete-character) This command deletes the character to the right of the pencil cursor.

D (delete-word) This command deletes the word to the right of the pencil cursor. To delete both a space and a word, position the pencil cursor on the space before the word and give the command.

k (kill) kill to end of line. This command deletes from the pencil cursor to the end of the line. The carriage return is deleted only if the pencil cursor is at the endof the line when you give this command.

h This command deletes the previous character (same as BACKSPACE).

H This command deletes the previous word (the word to the left of the cursor. To delete both a space and a word, position the cursor on the space after the word and give the command.

In addition to the above commands, the BACKSPACE and OOPS keys perform their usual functions:

BACKSPACE - deletes the previous character (the character to the left of the pencil cursor) and moves the pencil cursor back one space.

OOPS - deletes from the pencil cursor to the beginning of the line. If the pencil cursor is at the beginning of the line, the carriage return is deleted.

If you wish to delete a large chunk of material, you will probably find it more convenient to use the select-and-delete procedure explained in Section 9.9 than to delete characters or lines individually.

9.4 Retrieving Deleted Text

The text that is deleted with any of the commands given in the previous section goes into a "kill ring." The kill ring consists of eight buffers, each of which contains a chunk of material removed from the text with a delete command. Thus, only the last eight deletes are saved. The amount of text stored as one "delete" is determined by whether you move the pencil cursor after a delete command. If you do successive deletes without moving the pencil cursor, the deleted text is appended to the previously deleted text (in the same buffer). If you move the pencil cursor, the next text that is deleted will replace the previously deleted text in the top buffer, the previously deleted material is pushed around the ring into the next buffer, and the contents of the oldest buffer are eliminated.

The "yank-buffer" is the newest buffer in the kill ring; its contents are shown in the prompt window in the space after "Kill." If the deleted text is too long to be shown in its entirety, the beginning and end are shown with an ellipsis (...) in the middle.

To retrieve deleted text, position the pencil cursor where you want to insert text and then give one of the following commands:

y retrieve-the-yank-buffer.

Y pop-the-kill-ring. This command rotates the kill ring, moving the newest (displayed) buffer into last position. The second buffer is then displayed in the prompt window and can be retrieved with the "y" command. Issuing eight Y commands returns the ring to its original state, with the last deleted text in the yank buffer.

You cannot retrieve text that was deleted with the delete-window commands (xd or xf) or the replace-text command (R).

9.5 Replaying Transcripts

If you lose the work you have done in an editing session due to a machine crash or another reason where the Editor was not exited successfully, the Editor provides a safeguard in the form of a transcript of the last editing session. It will contain all of the commands you issued (except perhaps the last few which didn't get written to the transcript). The transcript is saved in a file which has the same name as the original file, but with a plus sign added. For example, if you had been editing the file Chap1.txt, the transcript is contained in Chap1.txt+.

If you want to replay the transcript, DO NOT START TO EDIT THE ORIGINAL FILE. If you entered the Editor in the usual manner, a new transcript would be started. Instead do the following:

1) Enter the Editor by typing ed <filename> -replay. Use the original filename--in the above example, Chap1.txt.

2) As the Editor reads the transcript, it will display each change in the prompt window before it makes the change. Respond in one of the following ways:

   To make the change shown, press the INS key. The next change will then be displayed.

   To stop the transcript without making the change shown, press the DEL key. (The transcript will then be deleted.)

   To make the change shown and then stop the transcript, press the TAB key.

   To instruct the Editor to make all of the changes in the transcript without any further action from you, type an exclamation point (!).

3) After the transcript has replayed, if you wish to save the changes issue the save command (xs).

If you wish you can keep a file on-screen and replay a transcript in another window. Open another window for the transcript by giving the xv command and then typing <filename> -replay.

9.6 Searching for Text

The search commands allow you to do a search through the file for a particular character or a string of characters, including spaces. The string can contain carriage returns or line feeds, but you must hold down the CTRL key while specifying the key (or else type CTRL-q and then the key, to quote the key). The search is not case sensitive; that is, searching for "program" will locate both "program" and "Program." Therefore you can find a word regardless of its use (first word in a sentence or not, proper name or not, etc.).

To do a search, first issue one of the following commands:

s (search-forward) Searches from the current position of the pencil cursor to the end of the text.

r (reverse-search) Searches from the current position of the pencil cursor to the beginning of the text.

After you have given one of the above commands, either:

a) type the string of characters you are looking for and press the INS or RETURN key; or

b) if the string you previously searched for (shown in the space after "Search for" at the top of your screen) is the same string you want to search for again, just press the INS or RETURN key. If it's almost the same string, at this point you may edit the string using the commands you'd use on any other text and then press the INS or RETURN key. (However, you cannot add characters to the string without first giving an editing command, as the Editor assumes that when you begin typing characters you are typing a new string.)

The Editor will then start the search. The thumb bar shows how far the search has advanced. If the Editor finds the string, it will position the pencil cursor at the first occurrence of the string (in a forward search, at the end of the string; in a reverse search, at the beginning of the string). If the Editor does not find the string, the message "Not Found" appears in the prompt area and the pencil cursor remains where it was when the search command was given. If an occurrence of the string is found and you want to continue the search, press the INS key and the Editor will proceed to the next occurrence, if any.

If you want to cancel a search that is in progress, type CTRL-DEL-c.

The next section tells how to search for a string and replace it with another string in one operation.

9.7 Replacing Text

With the Editor you can search for a string and optionally replace it with another string. The procedure is as follows:

1) To search-and-replace, position the pencil cursor at the beginning of the file to search the whole file or otherwise at the first occurrence of the string to be replaced and give the command R (Replace-text).

2) The prompt area will ask for the search string. Type the string and then press the INS or RETURN key. (The string can include a carriage return or a line feed, but it must be handled as explained in Section 9.6.) The string is not case sensitive; that is, searching for "company" will locate both "company" and "Company.")

3) The prompt area will ask for the replacement string. Type the new string and then press the INS or RETURN key.

4) The pencil cursor will move to the first occurrence, and the prompt area will ask you to press one of the following keys:

   INS - to replace the string and continue the search;

   SPACE BAR - to leave the string as is but continue the search-and-replace;

   DEL - to leave the string as is and cancel the search-and-replace;

   TAB - to replace the string and cancel the search-and-replace;

   ! - to change all occurrences without further prompting.

9.8 Miscellaneous Commands

The following commands help you correct common typing errors:

t (twiddle-characters) To transpose two characters, position the pencil cursor after the second character and issue this command.

C (Case-of-character) To change one letter from lower- to upper-case or vice-versa, position the pencil cursor on the character and issue this command.

U (Upper-case-word) To change a word from lower- to upper-case characters, position the pencil cursor on the first character of the word and issue this command. A "word" is any group of characters surrounded by a space or a punctuation mark.

L (Lower-case-word) To change a word from upper- to lower-case characters, position the pencil cursor on the first character of the word and issue this command.

The following commands allow you to set "bookmarks" at a particular spot on the screen and at particular places in the file and to return to them:

@ (insert-mark) This command is used to set both types of bookmarks, as follows:

To set a nameless bookmark on the screen at the current position of the pencil cursor, just issue this command. As a nameless mark pertains to the screen and not the file, there can be only one nameless mark. When you set a new nameless mark, the previous one is replaced.

To set a named bookmark in the file, first use the u command described below; then issue this command. There can be many named marks in a file.

u (enter-argument) To specify a name for a bookmark (either to set the mark with the @ command or to go to the mark with the xx command), issue this command. Then supply a name and press RETURN. This command also performs another function. See the additional description below.

xx (go-to-mark) To move the pencil cursor to the nameless bookmark, just issue this command. To specify a named bookmark, issue the u command described above, type the

mark name and press RETURN, and then issue this command.

You can go to bookmarks in other windows as well as the current window.

The following commands allow you to save and replay keystrokes, including Editor commands (useful if you need to insert a particular string into the text several times or create a command macro):

((save-keystrokes) The left parenthesis command instructs the Editor to save the following set of keystrokes.

) (end-save-keystrokes) The right parenthesis instructs the Editor to stop saving keystrokes.

* (replay-keystrokes) The asterisk instructs the Editor to insert the characters that you typed between the (and) commands, at the current position of the pencil cursor.

The following command allows you to repeat a keyboard command some number of times you specify:

u (enter-argument) Execute the following command n times, where n is a number that you enter. For example, u 10 CTRL-n moves the pencil cursor down 10 lines. The default argument is 4 lines, so u CTRL-p moves the pencil cursor up 4 lines. Not all commands accept multiple arguments.

The following commands allow you to stop a previously issued but uncompleted command:

g,G (abort-previous-command) Aborts previously issued command that has not yet been completed.

The following command allows you to center a line of text:

xc (center-line) Position the pencil cursor anyplace within the line and issue this command.

9.9 Selecting Text to be Deleted, Moved, or Copied

Selecting text allows you to move, copy, or delete text. This is the only way to copy text, and although there are other ways to move and delete text, this method is especially convenient for handling chunks of text that are more than one line long.

Text can be moved or copied from one place to another within the same file or from one file to another.

To select the desired chunk of material, position the pencil cursor at the desired location and issue the following commands:

[(start-selection) To start making a selection (the character after the pencil cursor becomes underlined to show the selection).

] (end-selection) To end the selection (the underlining will be extended to this point).

As you make the selection you can go forward or backward in the file.

Or, if you prefer to mark the location using the mouse and the arrow, do the following:

left or white button - (select) If the button is pressed once, the character pointed to by the arrow is selected. If the button is pressed twice, the word is selected. If the button is pressed three times, the line is selected. (To extend the selection farther, use the modify procedure discussed below.)

To modify the selection, use the following:

right or green button - (modify selection) To change the length of the selected chunk (to enlarge or reduce it), position the arrow (not the pencil cursor) at the desired location and press the button. The end of the selection closest to the arrow will be changed.

After you have selected the text, you can use any of the following commands:

" (delete) To delete the selected text, issue the quotation command ("). The deleted text is placed in the kill buffer as usual.

" and y To move the selected text, delete it with the " command. Then position the pencil cursor where you want to insert the text and issue the y (yank-buffer) command.

' (copy) To copy the selected text, position the pencil cursor where you want to insert the text and issue the ' command. The selected text remains in its original position and can be copied as many times as you wish.

xj (fill) To fill the lines of the currrent selection within the present margin setting. You can issue this command from anywhere in the file; the selection will fill regardless of where the cursor is.

You can also select text, go to another part of the file, and then return to the selected text by using the following commands:

x[(beginning-of-selection) When you give this command, the pencil cursor is moved to the beginning of the selected text.

x] (end-of-selection) When you give this command, the pencil cursor is moved to the end of the selected text.

10. Using Multiple Windows

The Editor allows you to examine and edit up to nine files at one time and to exit any file at any time. Each file will be displayed in a separate window. (However, even when more than one window is present, there is only one prompt window and one kill ring.)

Commands such as y (yank-buffer) and select-and-copy work across window boundaries--just move the pencil cursor and issue the command.

The commands are:

xv (visit-file) This command makes a new window by splitting the current window in half horizontally. (The current window is the one in which the pencil cursor is located when you give the command.) The Editor will then prompt you for the name of the file to be put into the new window.

There are times when it is convenient to be able to read or edit different parts of the same file at the same time. To open another window for the file you are working on, give the xv command and then type an equal sign (=) instead of a filename and press RETURN. The current window will be split and both windows will contain the same file. The changes you make in either window will appear in both.

Further, if you wish to open a window for another file and in that window replay the transcript of the previous editing session, give the xv command and then type <filename> -replay. Transcripts are explained in Section 9.5.

xn (next-window) This command moves the pencil cursor to the next window, in the order in which the windows were opened. If the pencil cursor is in the last window opened, it moves to the first window.

xp (previous-window) This command moves the pencil cursor to the previous window, in the order in which the windows were opened. If the pencil cursor is in the first window opened, it moves to the last window.

xd (delete-window) This command removes the current window and allots its space to other windows. Before the window is removed, the Editor turns that window to reverse video (so you are less likely to delete the wrong window); if

you picked the wrong window, cancel the command by pressing the DEL key. At this point the Editor also prompts for the name of a file into which the updated text is to be written. The default name is the original filename. To save the changes, either press RETURN to use the original filename, edit the filename, or type in a new filename. If you don't want to save the changes, press OOPS and RETURN in succession.

If the last window is deleted with this command, a new blank window is created.

xf (leave-the-Editor) This command removes all windows and exits the Editor. As each window is removed, the Editor turns the window to reverse video and lets you decide whether to keep the changes, as explained above for xd.

11. Changing the Keys Associated with Commands

It is possible (but not recommended) to change the keys associated with commands in the Editor. For example, you might prefer that the character to invoke the "forward one character" command be "r" (for "right") rather than "f" (for "forward"). We recommend that you do not make changes unless you are an expert user and are very careful as you make changes. Nearly all the letters are already assigned to a function, so you must change as many commands as necessary in order to have all the letters uniquely represent a command. In the example above, because the letter "r" invokes the "reverse search" command, you would have to also assign a new letter to that command, making sure the new letter isn't being used.

If you do decide to make changes, contact your PERQ Systems representative to request a copy of the file tha allows you to redefine command keys and its accompanying instruction file.

SUMMARY OF EDITOR COMMANDS

Invoking the Editor (Section 2): ed /path/<filename> -{switch}
- -help displays message to hit HELP key;
- -log creates transcript of editing session;
- -nolog does not keep transcript of editing session;
- -replay replays transcript of last session.

Thumbing (Sec. 8): Arrow in Thumb Bar, press middle/yellow button.

Scrolling (Section 8): With arrow in Scroll Area--
- press right or green button - line by arrow to top of window;
- press left or white button - line at top of window to arrow;
- hold right/middle or yellow/green button - scroll forward;
- hold left or white - scroll backward;
- CTRL-! - current line to top; CTRL-x CTRL-! - to bottom.

Replay Transcript (Section 9.5): ed <filename> -replay

Press either the CTRL key or mouse button (middle or yellow) while you issue all of the following commands, except those marked *:

Moving the Pencil Cursor (Sec. 9.1)
- f - forward one character
- b - back one character
- F - forward one word
- B - back one word
- a - alpha of line
- e - end of line
- n - next line
- p - previous line
- v - view next page
- V - view previous page
- , - top of window
- . - bottom of window
- < - top of file, > - bottom
- X! - far left column
- x[- beg. of selection;
- x] - end of selection

Inserting Text (Section 9.2)
- o - insert open space
- q<char.> - insert quoted char. into text as a command
- y - insert yank-buffer
- *TAB - insert tab
- *RETURN - insert new line
- *LF - insert line/indent

Deleting Text (Section 9.3)
- d - delete next character
- D - delete next word
- k - kill to end of line
- h or *BACKSPACE-delete prev. char.
- *OOPS - kill to beginning of line
- H - delete previous word

Retrieving Text (Section 9.4)
- y - insert yank-buffer
- Y - pop kill ring

Misc. Commands (Sect. 9.2,9.8)
- C - invert char. case
- U - word to upper case
- L - word to lower case
- u - enter bookmark name OR enter argument
- @ - insert bookmark
- xx - go to bookmark
- (- start saving keystrokes
-) - stop saving keystrokes
- * - replay keystrokes
- t - twiddle prev. 2 chars.
- P - set parameters
- xc - center line
- g,G- abort previous command

Searching (Section 9.6)
- s - search forward
- r - reverse search
- DEL-c - cancel search

Replacing Text (Section 9.7)
- R - replace globally

Selecting Text (Section 9.9)
- [- start selection
-] - end selection
- " - delete selection
- ' - copy selection
- xj - fill selection

Saving Changes (Section 6)
- xs - save changes

Using Windows (Section 10)
- xv - visit file (new window)
- xn - go to next window
- xp - go to previous window

Closing Windows (Section 3)
- xd - close current window
- xf - close all windows/exit

APPENDIX A

COMMAND SUMMARY

June 8, 1984

2600 Liberty Avenue
P. O. Box 2600
Pittsburgh, PA 15230
(412) 355-0900

Accent is a trademark of Carnegie-Mellon University.

Accent and many of its subsystems and support programs were originally developed by the CMU Computer Science Department as part of its Spice Project.

The information in this document is subject to change without notice and should not be construed as a commitment by PERQ Systems Corporation. The company assumes no responsibility for any errors that may appear in this document.

PERQ Systems Corporation will make every effort to keep customers apprised of all documentation changes as quickly as possible. The Reader's Comments card is distributed with this document to request users' critical evaluation to assist us in preparing future documentation.

COMMANDS BY FUNCTIONAL GROUPS

INITIALIZATION COMMANDS

Alias
ChangeUser
Define
Launch
Login
Path
SetSearch
Shell
Unalias
UserControl

PROGRAM DEVELOPMENT COMMANDS

Asm (assemble for C)
Compile (Pascal)
cpp (C)
Link (Pascal)
Lisp
Lnk (C)
Make
Matchmaker
Pascal
PasMac
pcc (C)
PRQMic
PRQPlace
QDis

INFORMATIONAL COMMANDS

Details
DirTree
Help
IconWallclock
Show
Statistics
Version
Wallclock

PROCESS CONTROL COMMANDS

Bye
Debug
Kill
Mace
Pause
Priority
Quit
Resume
Run
Suspend
Verbose

DEVICE MANAGEMENT COMMANDS

Dismount
Mount
Partition
Scavenger

COMMUNICATIVE COMMANDS

Chatter
FTP
Listen
On
Remote
Speak

FILE MANAGEMENT COMMANDS

Append
Compare
Copy
Delete
Direct
Edit
FindString
MakeDir
Patch
Print
Rename
Stut
TypeFile

SYSTEM MAINTENANCE COMMANDS

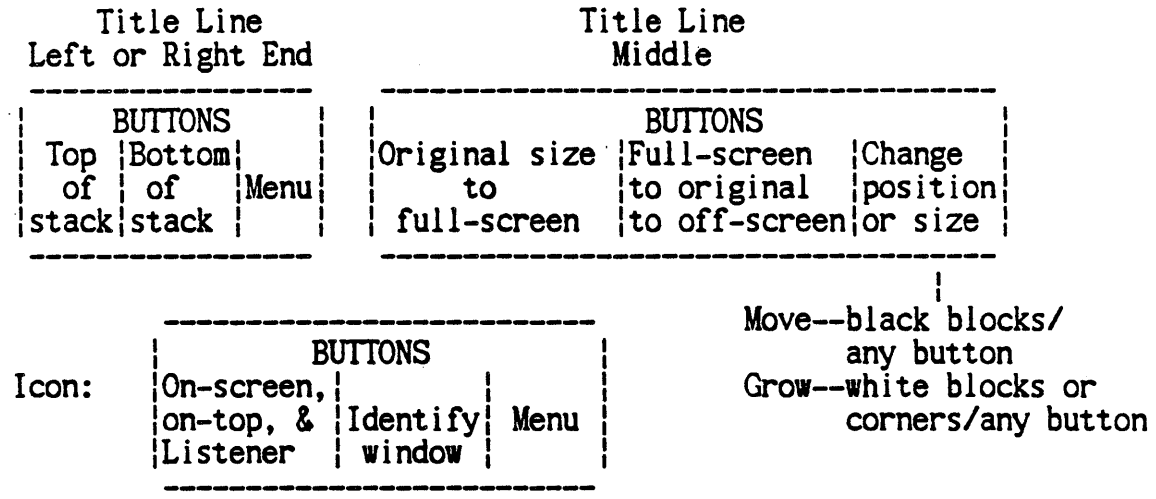
Bindboot
ExpandTabs
Floppy
KeyTranCom
SetBaud
SetProtect
SetTime

Alias <CommandName> <Definition> {-Switch} [Documentation]
Append [file][,file2][,...filen] [-Help]
Asm -O -o <filename.o> <filename.s>
Bindboot {-switch}
Bye
ChangeUser
Chatter
Compare <FilenameA> <FilenameB> [OutputFile] {-Switch}
Compile [inputfile] [] [outputfile] {-switch}
Copy [SourceFile][][DestinationFile] {-switch}
CPP <filename.c> <filename.i>
Debug [Process]
Define <name> [name_switch]... [[element [, element]...
Delete <Filename>[,file2][...,filen] {-switch}
Details {-switch[=switch value]}[outputfile]
Direct [dirSpec/][fileSpec] {-switch}[][outputfile]
DirTree [RootDirectory] {-switch}
Dismount
Edit [filename]
Edit <filename> -replay
ExpandTabs <SourceFile> <DestinationFile> [-Help]
FindString string,filelist {-switch}[outputfile]
Floppy [command] {-switch}
FTP [command] {-switch}
Help [Command]
Wallclock
KeyTranCom [filename.KTEXT]
Kill [Process] [-switch]
Launch <filename> {-switch}
Link {-switch} Source[,Source2,...SourceN] [RunFile]
Lisp
Listen [Name] {-switch[=value]}
Lnk -o <filename.EXE> CTRO {x.o, y.o z.o} <filename.O>
Login [UserName]
Mace
Make {-switch} [TargetFileName]
MakeDir [FileSpecification] [-Help]
Matchmaker FileName -Lang1=Opt1 ... -LangN=OptN {switch}
Mount
On <MachineName> <CommandString>
Partition [-help] [-debug]
Pascal [inputfile] [] [outputfile] {-switch}
PasMac <InputFile> <OutputFile>
Path [pathname]
Pause [message]
Pcc <filename.i> <filename.s>

```
Print <Filename>[,filename2,...filenameN] {-switch}
Priority [<Process> [<LevelNo.>]] [-Help]
PRQMic <InputFile> [-Help]
PRQPlace <RootFile> <ListingFile> {-switch}
QDis <SegFile> <ListFile> {-switch}
Quit [yes|no]
Remote [-Help]
Rename <SourceFile>[]<DestinationFile> {-switch}
Resume [Process] [-Help]
Run <ProgramName> [argument1 [argument 2]]
Scavenger [partition]
SetBaud [baudrate] {switch}
SetProtect <directory or filename> <-switch>
SETSEARCH [pathname][,pathname] {-switch}
SetTime [DD MMM YY] [HH:MM[:SS] {-switch}
Shell [-quit] [command] [-Help]
Show [<Name>] {-switch}
Speak [UserName] [-switch]
Statistics [Yes|No] [-switch]
Stut [-Help]
Suspend [Process] [-Help]
TypeFile [FileSpecification][,file2][...,fileN] {-switch}
Unalias [command]
UserControl [command] {-switch}
Verbose [Boolean] [-switch]
Version
```

SUMMARY OF THE WINDOW MANAGER COMMANDS

Mouse Commands



Menu Commands	Keyboard Commands*
Shell (create new window)	<prefix> s
Eradicate window (kill process and delete window)	<prefix> e
Help	HELP or <prefix> h
Listener menu (to designate new Listener)	<prefix> l
Bottom (send to bottom of stack)	<prefix> b
Top (send to top of stack)	<prefix> t
>Bigger (from off-screen to original to full-screen)	<prefix> >
<Smaller (from full-screen to original to off-screen)	<prefix> <
Grow window (using corners or white blocks)	<prefix> g
Move window (using black blocks)	<prefix> m
Reshape window (redefine upper left & lower right corners)	<prefix> r
Icon window (bring on-screen, on-top)	<prefix> i
Organize icons (compact the icons)	<prefix> o
Find icon (identify corresponding window)	<prefix> f
Debug process (send window to debugger)	<prefix> d
Cancel process	<prefix> c
Kill process (including command files)	<prefix> k
Suspend process	<prefix> z
Resume process	<prefix> q

Keyboard Commands Not Available from Menu

Obtain menu	<prefix> ?
Forward in ring	<prefix> +
Backward in ring	<prefix> -
Exit window manager and return to application	<prefix> x

*Prefix is CTRL-DEL on the PERQ and CTRL-DEL or SETUP on the PERQ2.

SUMMARY OF EDITOR COMMANDS

Invoking the Editor (Section 2): ed /path/<filename> -{switch}
- -help displays message to hit HELP key;
- -log creates transcript of editing session;
- -nolog does not keep transcript of editing session;
- -replay replays transcript of last session.

Thumbing (Sec. 8): Arrow in Thumb Bar, press middle/yellow button.
Scrolling (Section 8): With arrow in Scroll Area--
press right or green button - line by arrow to top of window;
press left or white button - line at top of window to arrow;
hold right/middle or yellow/green button - scroll forward;
hold left or white - scroll backward;
CTRL-! - current line to top; CTRL-x CTRL-! - to bottom.
Replay Transcript (Section 9.5): ed <filename> -replay

Press either the CTRL key or mouse button (middle or yellow) while you issue all of the following commands, except those marked *:

Moving the Pencil Cursor (Sec. 9.1)
- f - forward one character
- b - back one character
- F - forward one word
- B - back one word
- a - alpha of line
- e - end of line
- n - next line
- p - previous line
- v - view next page
- V - view previous page
- , - top of window
- . - bottom of window
- < - top of file, > - bottom
- X! - far left column
- x[- beg. of selection;
- x] - end of selection

Inserting Text (Section 9.2)
- o - insert open space
- q<char.> - insert quoted char. into text as a command
- y - insert yank-buffer
- *TAB - insert tab
- *RETURN - insert new line
- *LF - insert line/indent

Deleting Text (Section 9.3)
- d - delete next character
- D - delete next word
- k - kill to end of line
- h or *BACKSPACE-delete prev. char.
- *OOPS - kill to beginning of line
- H - delete previous word

Retrieving Text (Section 9.4)
- y - insert yank-buffer
- Y - pop kill ring

Misc. Commands (Sect. 9.2,9.8)
- C - invert char. case
- U - word to upper case
- L - word to lower case
- u - enter bookmark name OR enter argument
- @ - insert bookmark
- xx - go to bookmark
- (- start saving keystrokes
-) - stop saving keystrokes
- * - replay keystrokes
- t - twiddle prev. 2 chars.
- P - set parameters
- xc - center line
- g,G- abort previous command

Searching (Section 9.6)
- s - search forward
- r - reverse search
- DEL-c - cancel search

Replacing Text (Section 9.7)
- R - replace globally

Selecting Text (Section 9.9)
- [- start selection
-] - end selection
- " - delete selection
- ' - copy selection
- xj - fill selection

Saving Changes (Section 6)
- xs - save changes

Using Windows (Section 10)
- xv - visit file (new window)
- xn - go to next window
- xp - go to previous window

Closing Windows (Section 3)
- xd - close current window
- xf - close all windows/exit

PERQ/ACCENT FAULT DICTIONARY:

KEY TO THE DIAGNOSTIC DISPLAY

June 8, 1984

2600 Liberty Avenue
P. O. Box 2600
Pittsburgh, PA 15230
(412) 355-0900

Accent is a trademark of Carnegie-Mellon University.

Accent and many of its subsystems and support programs were originally developed by the CMU Computer Science Department as part of its Spice Project.

The information in this document is subject to change without notice and should not be construed as a commitment by PERQ Systems Corporation. The company assumes no responsibility for any errors that may appear in this document.

PERQ Systems Corporation will make every effort to keep customers apprised of all documentation changes as quickly as possible. The Reader's Comments card is distributed with this document to request users' critical evaluation to assist us in preparing future documentation.

TABLE OF CONTENTS

PERQ/ACCENT FAULT DICTIONARY:

KEY TO THE DIAGNOSTIC DISPLAY

1. Introduction

The document "Basic Operations" in this manual describes how to boot a PERQ workstation. As the workstation goes through the boot sequence, numbers appear on the Diagnostic Display (DDS). On the PERQ workstation the DDS is located on the underside of the keyboard. On the PERQ2 workstation, depending on the model, the DDS is located either on the base of the display or on the front of the processor box. If any step fails, look at the Diagnostic Display to see where in the boot sequence the failure occurred. A list of the DDS values and their meanings is given in the following pages.

The boot sequence has three steps. In the first part of the boot sequence, microcode executes out of a small ROM. This ROM covers the lower 2k of standard control store during this part of the boot sequence. The microcode runs a simple diagnostic on the processor and memory systems. If there are any errors, the microcode halts. The value in the DDS gives the reason that the workstation halted. Once these diagnostics complete, the microcode makes a decision about which device is to be used for booting. Currently, there are three possible boot devices.

1. The first possible boot device is another PERQ or PERQ2 workstation. The microcode determines if there is a PERQ Link Board plugged into the I/O Option slot of the workstation. If the board is plugged in and there is another PERQ or PERQ2 workstation on the other end of the link, the booting workstation waits for commands from the link.

2. The second choice of boot devices is a floppy disk. The microcode checks to see if there is a floppy in the floppy drive. If there is a floppy in the drive and the floppy is a boot floppy, the second part of the boot is done from the floppy.

3. As the third alternative, the microcode tries to boot from the hard disk.

If all of these fail, then the DDS contains an indication of what the error is.

After choosing the boot device, the second part of the boot sequence begins. In this part, the workstation reads 3.75K words of microcode from the selected boot device. This microcode is in two sections--a more extensive diagnostic (VFY) and a system boot loader (SYSB). VFY attempts to verify that all of the CPU and Memory systems are working. VFY displays failures on the DDS.

If no failures are detected, the microcode loads the operating system and transfers control to the third portion of the boot sequence. The operating system that is loaded is either the default (Accent) or the system corresponding to the key you have pressed (a lower-case letter for hard disk, an upper-case letter for floppy). Note that Accent boot floppies are not supported. In the third portion of the boot sequence, the interpreter microcode performs system initialization and then starts to execute the operating system. If you used the default operating system and there were no errors during the boot sequence, the workstation is running Accent and the DDS reads 400 (1M memory) or 450 (2M memory). (If you booted another operating system rather than Accent, the final number will be different. See the documentation for that operating system.)

If the workstation has not booted after two or three minutes, try pressing the Boot button. If pressing the boot button does not cause the workstation to boot, call your system administrator or PERQ Systems.

2. Diagnostic Display

Display	Description
000	Boot never got going, StackReset does not work or other major problem in the processor board (or clock).
001	Simple Branches fail.
002	Main Data Path Failure.
003	Dual Address failure on Registers.
004	Y Ram Failure.
005	Const/Carry Propagate failure.
006	ALU failure.
007	Conditional Branch failure.
008	Looping failure.
009	Control Store (or Write Control Store) failure.
010	Hung in Disk Boot.
011	Memory Data Error.
012	Memory Address Error.
013	Disk never became ready.
014	Could not boot from either disk.
015 - 020	Bad Interrupts Reading Floppy Disk Data.
030	VFY Hung.

050 Bad Error Message from VFY.

051 Empty stack bit not working.

052 Could not load TOS.

053 Push did not work.

054 Stack Empty did not go off.

055 Data error in push.

056 Empty or Full set when that is not the case.

057 Data error in bit 15 of the stack.

058 Stack empty set when the stack is full.

059 Data error on stack.

060 Data error after POP. Bit 14.

061 Data error after POP. Bit 13.

062 Data error after POP. Bit 12.

063 Data error after POP. Bit 11.

064 Data error after POP. Bit 10.

065 Data error after POP. Bit 9.

066 Data error after POP. Bit 8.

067 Data error after POP. Bit 7.

068 Data error after POP. Bit 6.

069 Data error after POP. Bit 5.

070 Data error after POP. Bit 4.

071 Data error after POP. Bit 3.

072 Data error after POP. Bit 2.

073	Empty wrong.
074	Data error after POP. Bit 1.
075	Data error after POP. Bit 0.
076	Empty not set after all pops.
077	Call test failed.
078	Odd did not jump on a 1.
079	Odd jumped on a 0.
080	Byte sign did not jump on 200.
081	Byte sign jumped on 0.
082	C19 did not jump when it should have.
083	BCP[3] did not jump when it should have.
084	C19 jumped when it should not have.
085	BCP[3] jumped when it should not have.
086	GTR did not jump.
087	GTR jumped when it should not have.
088	GEQ did not jump.
089	GEQ jumped when it should not have.
090	LSS did not jump when it should have.
091	LSS jumped when it should not have.
092	LEQ did not jump.
093	LEQ jumped when it should not have.
094	GEQ did not jump on equal.
095	LEQ did not jump on equal.

096	Carry did not jump when it should have.
097	Carry jumped when it should not have.
098	Overflow did not jump when it should have.
099	Overflow jumped when it should not have.
100	And-Not ALU function failed.
101	Or ALU function failed.
102	Or-Not ALU function failed.
103	And ALU function failed.
104	Or-Not ALU function failed.
105	Not-A ALU function failed.
106	Not-B ALU function failed.
107	Xor ALU function failed.
108	Xnor ALU function failed.
109	OldCarry-Add ALU function failed.
110	OldCarry-Sub ALU function failed.
111	OldCarry-Add /w No OldCarry failed.
112	Fetch error on Force Bad Parity.
113	Unexpected Parity error.
114	No parity errors on force bad parity.
115	Wrong address on force bad parity.
116	Upper 4 bit test failed.
117	MDX test failed.
118	Stack upper bits test failed.

119	Store/Fetch test failed.
120	Unexpected refill.
121	BPC test failed.
122	Fetch4 test failed.
123	Fetch4R test failed.
124	Store4 test failed.
125	Fetch2 test failed.
126	Store2 test failed.
127	NextOp test failed.
128	Fetch/Store overlap failed.
129	Bad interrupt Loc 4.
130	Bad interrupt Loc 14.
131	Bad interrupt Loc 20.
132	Bad interrupt Loc 30.
133	Data error on memory sweep.
134	Address error on memory sweep.
135	Field did not work.
136	Dispatch did not jump.
137	Wrong Dispatch target.
138	Data error on inverted memory sweep.
139	Address error on inverted memory sweep.
150	Sysb not loaded correctly.
151	Sysb did not complete.

152	Illegal boot key.
153	Reset error on PERQ workstations; not used on PERQ2 workstations
154	No such boot.
155	No interpreter for that key.
156	Interpreter file is empty.
157	Disk error.
158	Floppy error.
159	Malformed boot file.
160	CheckSum error in microcode.
161	CheckSum error in QCode.
162-168	Bad interrupts.
169	Not used.
170	No ACK from keyboard; on PERQ2 workstations only
171	Wrong disk type for this Sysb; on PERQ2 workstations only
198	QCode interpreter microcode not entered correctly.
199	System not entered - calls or assignments do not work.
200	System entered, InitMemory to be called.
300	System booted (256K memory).
350	System booted (512K memory).
400	System booted (1M memory).
450	System booted (2M memory).