HP 3000 Computer Systems

Comprehensive Introduction for The Application Programmer

HEWLETT **hp** PACKARD
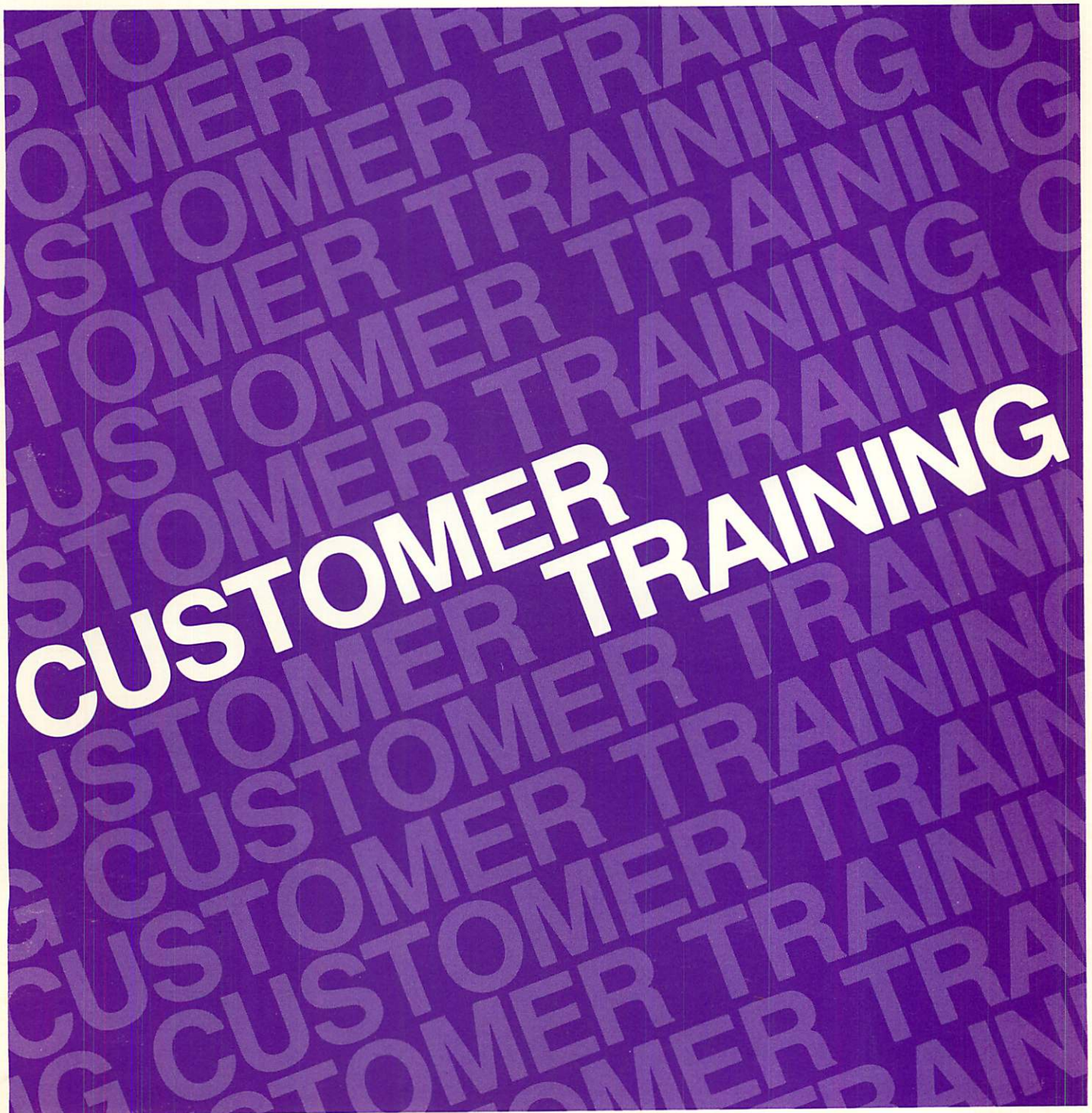
student workbook

CUSTOMER TRAINING

Table of Contents

HP-3000

A COMPREHENSIVE INTRODUCTION

MPE

HEWLETT hp PACKARD

HP-3000 COMPUTER SYSTEM
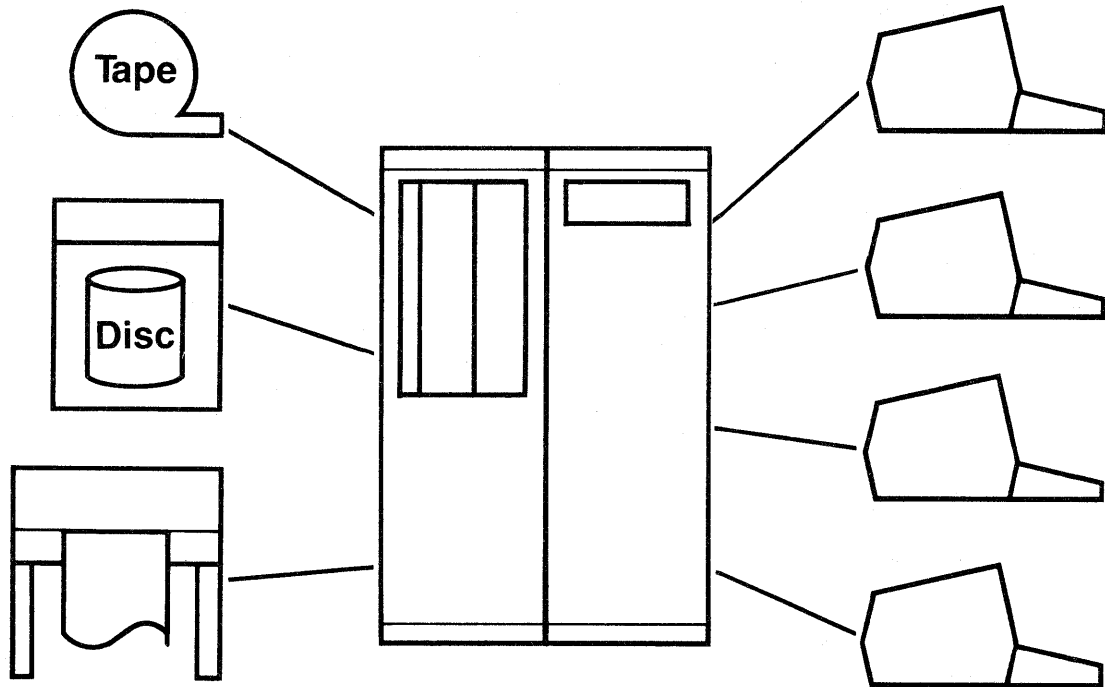
HP-3000 SYSTEM CAPABILITIES
- Data Entry
- Data Management
- Transaction Processing
- On-line Program Development
- Batch Mode Processing
- Data Communications

Ideal for interactive processing, but performs equally well in batch environment.

Flexible design makes it readily adaptable to any business application.

HEWLETT hp PACKARD

DATA MANAGEMENT

APPLICATIONS

DATA COMMUNICATIONS

LANGUAGES

MPE

HARDWARE

BATCH

INTERACTIVE

MULTI USER
ENVIRONMENT
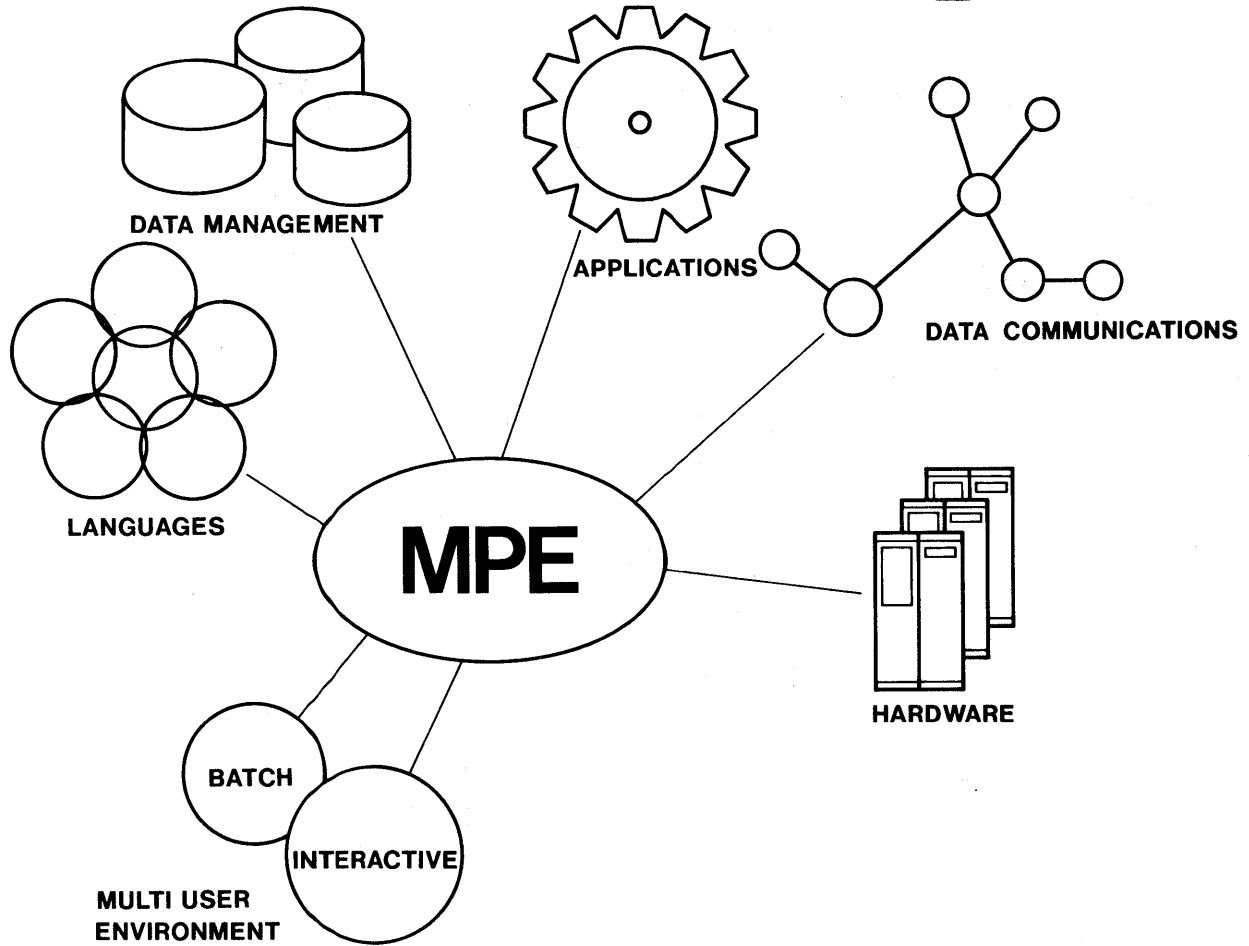
MPE — Multi-Programming Executive

Provides the environment for all the above processes to be
happening at the same time on the HP-3000.

LAB 0(zero) [0.75 hr]

GETTING STARTED

To Log-on the computer, press ⬛ to get the ':'
prompt, then key in:

<u>HELLO user.INTRO/password</u>

You must get your unique username and the Account password
from your instructor.

Do Sections 1 and 2 in the manual "Using the HP-3000; An
Introduction to Interactive Programming". Then read Section 3
in the same manual.

HEWLETT **hp** *PACKARD*

MPE — MULTI-PROGRAMMING EXECUTIVE

THE OPERATING SYSTEM ON THE HP-3000
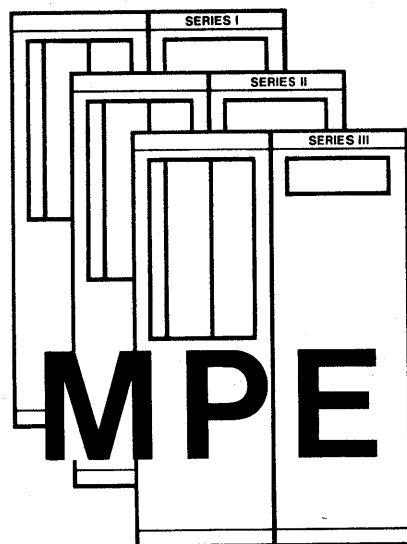
- MPE is the only Operating System for all HP-3000's.

- Comprehensive, yet easy to use.

- Flexible commands work equally well in Interactive SESSION or Batch JOB.

- Today's features of MPE will not be changed.

- New ENHANCEMENTS made available to all users.

HEWLETT hp PACKARD

ON-LINE PROGRAM DEVELOPMENT

- CREATE PROGRAM SOURCE

- MODIFY PROGRAM SOURCE

- COMPILE PROGRAMS

- EXECUTE PROGRAMS

HEWLETT **hp** *PACKARD*

SYSTEM OPERATION

- **FILE SYSTEM**   *DISK*
    DEVICE— not on disk
    - **devices**
    - **security**
- **UTILITIES**
    - **SORT**
    - **FCOPY**
- **JOB STREAMS**

HEWLETT **hp** *PACKARD*

SUBSYSTEM APPLICATIONS

- **Data Entry** — VIEW

- **Data Management** — IMAGE

- **Data Inquiry** — QUERY

- **Data Communications**

MPE

FUNDAMENTALS
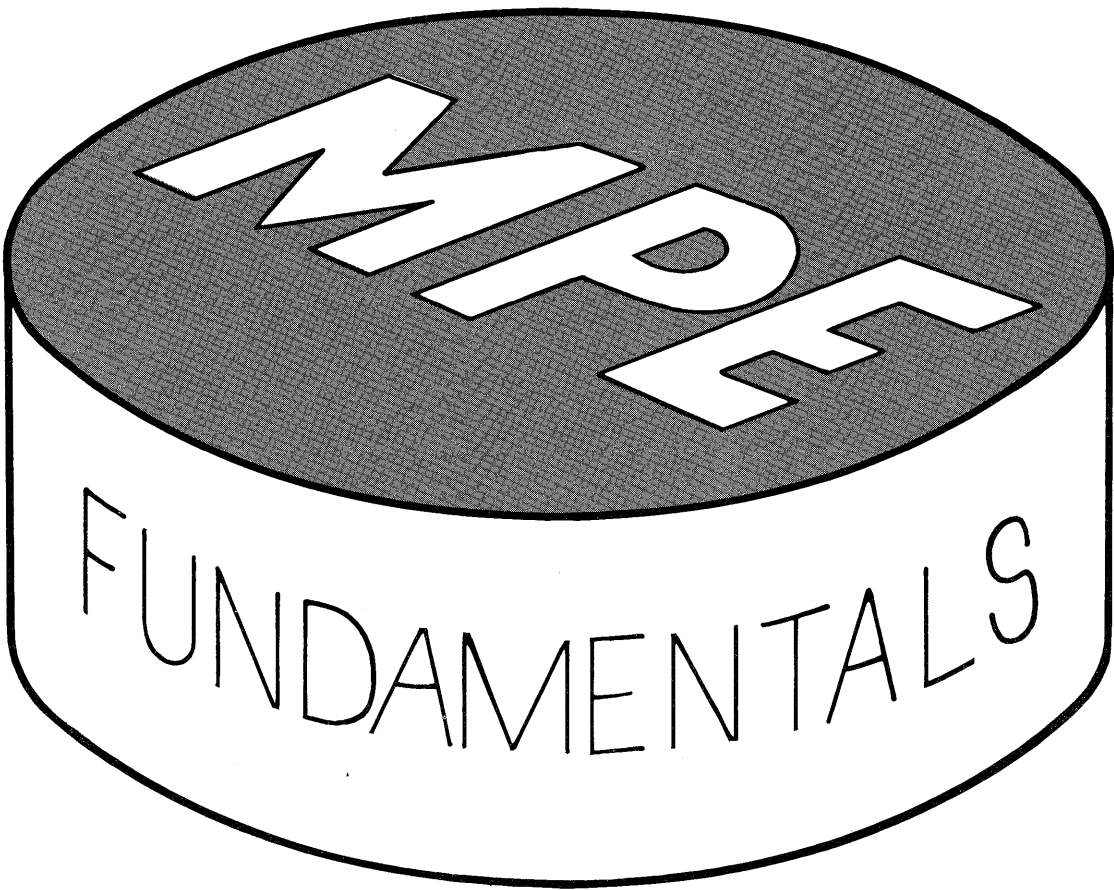
A TYPICAL MPE COMMAND

:RPG [textfile][,[uslfile][,[listfile]]]

examples,

:RPG SOURCE,,LIST

-or-

:RPG SOURCE

COMMAND NAME
      1) Identified as MPE command by ':'
      2) ':' used as prompt in SESSION.
      3) Command-name follows ':'
      4) Parameters separated from command-name by space or
          special character.
POSITIONAL PARAMETERS:
      1) Separated by commas.
      2) Place in list indicates meaning.
      3) Parameters omitted from list must be represented by
          place-holding commas.

A TYPICAL MPE COMMAND

:HELLO username.acctname[;TIME=cpusecs][;PRI={BS}]
{CS}
{DS}
{ES}

examples,

```
:HELLO STUDENT.INTRO
```

-or-

```
:HELLO TEACHER.INTRO;PRI=CS;TIME=10
```

KEYWORD PARAMETERS
    1) Preceded by a semi-colon.
    2) May appear in any order.
    3) May have a positional sub-parameter list.
MIXED PARAMETERS
    All positional parameters must appear before the first
    keyword parameter.

A TYPICAL MPE COMMAND

```
                                              [F][,BINARY]
:FILE formaldesignator[;REC=[recsize][,[blockfactor][,[U][,ASCII   ]]]]
                                              [V]
```

examples,

:FILE XYZ;REC=100,,,ASCII

-or-

:FILE MYFILE;REC=,,V

POSITIONAL SUB-PARAMETER LIST IN KEYWORD PARAMETER:
Same rules as for a positional list.

HEWLETT hp PACKARD

A TYPICAL MPE COMMAND

:RPG [textfile][,[uslfile][,[listfile]]]

examples,

```
:RPG*CARDDECK,USL,*LP
```

-or-

```
:RPG      &
:     *CARDDECK   &
:          ,  USL  ,  *LP
```

COMMAND NAME DELIMITER

        Delimited by space or a special character.

CONTINUATION

    1)   A command may be continued by '&' as the last non-blank character.

    2)   Broken immediately before or after a delimiter.

    3)   Extra spaces may be inserted around delimiters.

    4)   MPE will prompt for each new line with another ':'.

MPE COMMANDS

:HELLO	:FILE LP;DEV=LP	:REPORT
:BYE	:RUN	:LISTF
		:PURGE

:SHOWJOB	:TELL	\<break\>key
:SHOWME	:TELLOP	:RESUME
:HELP	:SETMSG	:ABORT

:HELLO	Initiates an interactive session.
:BYE	Ends an interactive session.
:FILE LP;DEV=LP	File command needed to reference the line printer device file.
:RUN	Executes a prepared program.

:REPORT	displays accounting information for log-on account and group.	
:LISTF	lists description of permanent disc files.	
:PURGE	deletes disc file from system.	

:SHOWJOB	displays status information about jobs/sessions.	
:SHOWME	Displays status of your job/session.	*MPE III only*
:HELP	Displays MPE Command syntax and information.	*MPE III only*

:TELL	send a message to another job or session.
:TELLOP	send a message to the Console Operator.
:SETMSG	disables or enables receipt of messages on your terminal.

\<break\> key	Suspends currently running process. Returns to MPE Command Interpreter.
:RESUME	Resumes execution of a suspended process.
:ABORT	Aborts suspended process.

MPE-C SESSIONS

```
:HELLO SESSIONA,STUDENT.INTRO
ACCT PASSWORD?
PASSWORD
 SESSION NUMBER = #S25
 TUE, FEB 14, 1978,  4:35 PM
 HP32002A.01.2I

WELCOME to your friendly HP-3000...

:SHOWJOB EXEC;JOB=SESSIONA,STUDENT.INTROACCT
ERR 29,5 J
ILLEGAL NAME
:
```

Echo is ON for Passwords!
- 'ESC ;' turns echo off.
- 'ESC :' turns echo on.

Correcting ERRORS.
- MPE displays error number & number of parameter in error,
      then waits for your response.
- 'RETURN' key — no message issued.
- Enter any printing character except ":" & error message
      will be displayed.
- YOU must find the parameter in error.

MPE III COMMAND ERRORS

```
:HELLO MYSESSN,FIELD.SUPPORT,PUB
ACCOUNT PASSWORD (PASS)?

HP3000 III.  TUE, FEB 14, 1978,  4:53 PM
   ** WELCOME TO YOUR FRIENDLY HP-3000 !! **
:SHOWJOB EXEC;JOB=MYSESSN,FIELD.SUPPORTER
                                    ^
ACCOUNT NAME > 8 CHARACTERS LONG. (CIERR 552)
:REDO
SHOWJOB EXEC;JOB=MYSESSN,FIELD.SUPPORTER
                                      DD
SHOWJOB EXEC;JOB=MYSESSN,FIELD.SUPPORT
[RETURN]
JOBNUM  STATE IPRI JIN  JLIST     INTRODUCED  JOB NAME

#S37    EXEC        25  25        TUE  4:53P
        MYSESSN,FIELD.SUPPORT

JOBFENCE= 0; JLIMIT= 8; SLIMIT= 60

:BYE
CPU=1. CONNECT=3. TUE, FEB 14, 1978,  4:56 PM
```

Echo automatically turned off for Passwords and Lockwords.

Correcting Errors:
- MPE Command Interpreter points to parameter in error.
- Error message automatically issued.
- You can now correct portion in error with REDO.

READING ASSIGNMENT — MPE COMMAND SYNTAX

Leading
Colon

Command identifier character, used as prompt in interactive Sessions.

Command
Name

Shown in UPPER-CASE exactly as it must be entered. Contains no blanks. Is delimited by a non-alphanumeric character (usually a blank). 1 to 16 alphanumeric characters; 1-st must be alpha.

Parameters

UPPER-CASE for literal information that must be entered exactly as shown.

LOWER-CASE for variable parameters you supply.

Positional
Parameters

• Position in list determines meaning.
• Separated by commas.
• Adjacent commas indicate omitted parameter (default used).

E X A M P L E S:

:COMMANDNAME P1,,P3   (omit middle)
:COMMANDNAME ,P2,P3   (omit beginning)
:COMMANDNAME P1   (omit end)

Keyword
Parameters

• May appear in any order.
• Preceded by semicolon.
• May have positional subparameter list.

Mixed
Parameters

Positional parameters first; first keyword indicates end of positional list.

Optional
Parameters

[A]   means "A" MAY be included.

[A]
[B]   means "A" or "B" MAY be included.

$\begin{Bmatrix} A \\ B \\ C \end{Bmatrix}$ means one of "A" or "B" or "C" MUST be included.

[A]   [B]   means "A" and/or "B" MAY be included in any order.

$\begin{bmatrix} \begin{Bmatrix} A \\ B \\ C \end{Bmatrix} \end{bmatrix}$ means all MAY be omitted; no more than one of "A" or "B" or "C" MAY be included.

HEWLETT hp PACKARD

READING ASSIGNMENT — MPE COMMAND SYNTAX

NAMES in the System other than MPE Command Names:
(Names of Files, Groups, Accounts, Users, Lockwords, etc.)

- 1 to 8 Alphanumeric characters

- 1-st character must be Alphabetic

NUMBERS as MPE Command Parameters:

- Assumed to be decimal numbers

- Octal numbers preceded by a "%"

CONTINUATION of MPE Commands:

- Ampersand (&) as last non-blank character of a line will
    continue command to next line.

- Commands may be up to 255 characters long (not counting
    ampersands and prompting colons)

- In session, prompting colon will be supplied automatically.

- Must divide command at a delimiter.

EMBEDDED BLANKS:

    Extra blanks allowed between Commandname, Parameters, &
    delimiters.

SPECIAL CHARACTERS that DELIMIT MPE Parameters:
    "," ";" "=" "." and "/"

SPECIAL CHARACTERS that are NOT Delimiters:
    "&" "@" "$" "*" "-" "#" and "%"

ELLIPSIS: (...)
    means that the previous bracketed element may be repeated
    any number of times or an item has been omitted.

UNDERLINING
    User inputs are underlined for clarity.

WORKSESSION — MPE SYNTAX

Check the following statements and see if they are syntactically correct or not. You are not expected to know what would happen when trying to execute the following commands at this point, but you should be able to check their syntax. Use the STANDARD CAPABILITIES section of your "SOFTWARE POCKET GUIDE" to find the syntax for the following commands, circle all errors and write the number of errors you find in the space provided at the right of each example.   [30minutes]

		NO. OF ERRORS
(1)	`:HELLO WORKSESS,STUDENT.INTRO,PUB`	○ _____
(2)	`:HELLO STUDENT,INTRO;TERM=3;&` `:  TIME=TEN`	1 _____
(3)	`:HELLO STUDENT,STUDENT/SECRET.INTRO&` `:  ;PRI=DS`	_____

(4)	`:HELLO &`
	`:  STUDENT.INTRO &`
	`:  ;PRI=HIPRI`
		_____

(5)	`:HELLO STUDENT.INTRO/PASSWORD;HI&` `:  PRI;TERM=10;TIME=10`	_____
(6)	`:HELLO`	_____
(7)	`:LISTF`	○ _____
(8)	`:LISTF;LISTF`	_____
(9)	`:LISTF @.@.SYS,2;LISTFILE`	○ _____

WORKSESSION — MPE SYNTAX

			NO. OF ERRORS
(10)	:LISTF	@.GSTUDENT.@;2	_____
(11)	:LISTF	LISTF	_____
(12)	:SHOWJOB		_____
(13)	:SHOWJOB	JOB=#S	_____
(14)	:SHOWJOB	#J11,#S13,STATUS	_____
(15)	:SHOWJOB	INTRO;JOB=@.INTRO;EXEC	_____
(16)	:SHOWJOB	JOB=@,STUDENT.INTRO	_____
(17)	:SHOWJOB	JOB& : =@.@	_____
(18)	:SHOWJOB	EXEC; JOB=@,@.INTRO	_____

> OPTIONAL — Do this part only if time permits.

Log-on & execute the commands in the last three questions
above. Correct any errors to get them to execute properly.

HEWLETT hp PACKARD

ACCOUNTING STRUCTURE

HP3000 SYSTEM

SYS

PAYROLL

MARKETING

INTRO

SUPPORT

- System file space and resources can be allocated to units called Accounts.
- The system keeps cumulative totals on resource usage at the account level.
- The System Manager defines accounts and file access security for all disc files within each account.

ACCOUNTING STRUCTURE

ACCOUNT (INTRO)

HOME GROUP

USER

STUDENT

G STUDENT GROUP

BRUTUS 39

FINAL

SAVESUB1

F
I
L
E
S
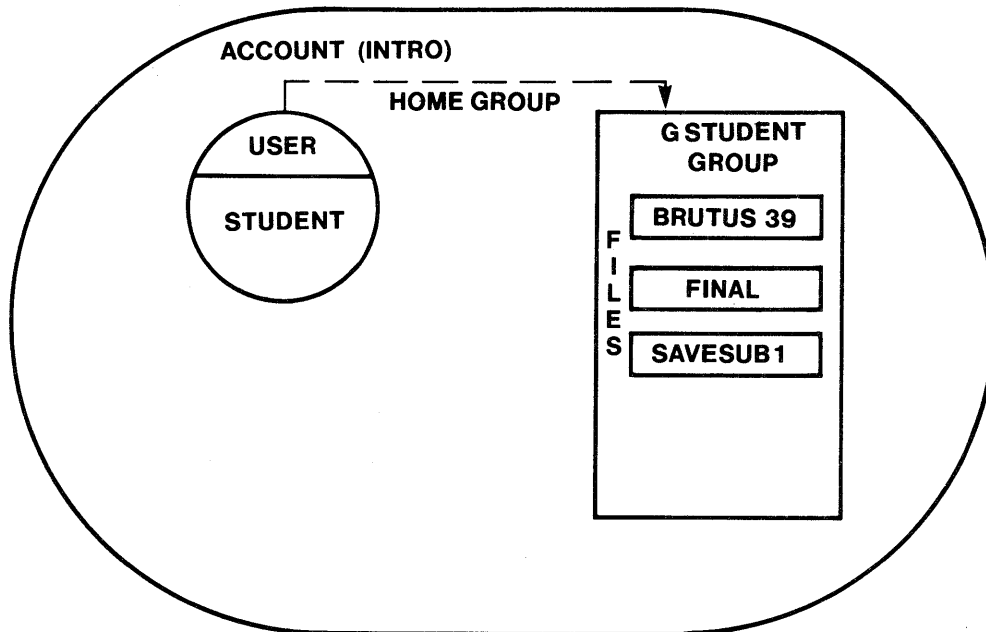
- Groups are defined by each Account Manager and reside within Accounts.
- Users are defined by Account Managers and may be assigned a Home Group.
- Files reside within Groups and are created by each User.

HEWLETT hp PACKARD

ACCOUNTING STRUCTURE

SYS ACCOUNT

HOME GROUP

USER

MANAGER

PUB GROUP

F
I
L
E
S

EDITOR

FCOPY

FORTRAN

ETC.

*can read
Not write*

INTRO ACCOUNT

HOME GROUP

HOME GROUP

USER

MGR

pub

F
I
L
E
S

DEFTABS

LABEDIT

SHORTF

ETC.

USER

STUDENT

GSTUDENT
GROUP

F
I
L
E
S

FINAL

BRUTUS39

SAVESUB1

ETC.
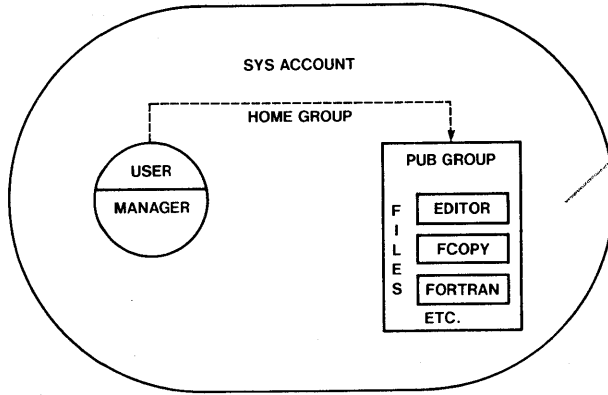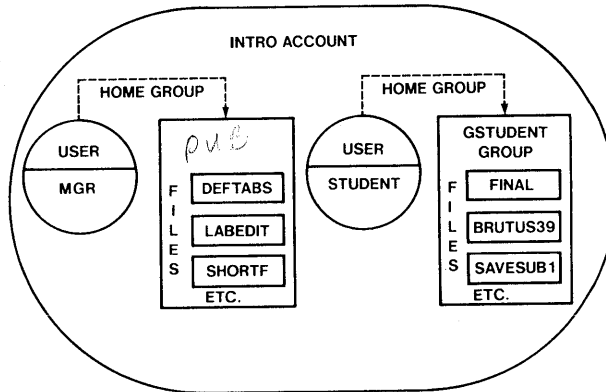
- Permanent disc files reside within groups within accounts.
- A file's whole qualified name is 'filename.groupname.accountname'.
- The groupname and accountname may be implicit.

HEWLETT **hp** *PACKARD*

CLASS WORKSESSION

Use the Accounting Structure diagram from the previous page.
What are the Disc File names each User must use to address the first file in each of the Groups?

To reference file:	For User:		
	STUDENT.INTRO	MGR.INTRO	MANAGER.SYS
BRUTUS39	BRUTUS39	BRUTUS39.GSTUDENT	BRUTUS39.GSTUDENT.INTRO
DEFTABS	DEFTABS.PUB	DEFTABS	DEFTABS.PUB
FCOPY	FCOPY.PUB.SYS	→	FCOPY

TYPES OF FILES

• "FILE" = a collection of similar records
  being processed by the FILE System.

DISC FILES
Permanent
Temporary

SYSTEM DEFINED FILES   *Start with $*

DEVICE FILES

• DISC FILES
—   Permanent Disc Files — Cataloged in a System-wide Directory; Potentially accessible by any user in the system.
—   Temporary Disc Files — Do not count against Permanent Disc space limit; not in System-wide directory.

• SYSTEM DEFINED FILES
—   Uniquely defined for each Job / Session.

• DEVICE FILES
—   All other files in the system (i.e. Mag-tape files, Card files, Line printer files, etc.)

CONTROLLING ACCESS to FILES

DEVICE FILES

Padlock & Key

SYSTEM DEFINED FILE & TEMPORARY DISC FILES

Not known outside a unique Job or Session.

PERMANENT DISC FILES

To access a file, user must pass File Access Security
settings at the ACCOUNT, GROUP, and FILE levels PLUS
present the correct file LOCKWORD. These security
settings are enforced by the FILE SYSTEM.

FILE SECURITY

FILE B

LOCK WORD
FILE SECURITY
GROUP SECURITY
ACCOUNT SECURITY

FILE A

LOCK WORD

FILE SECURITY

GROUP SECURITY
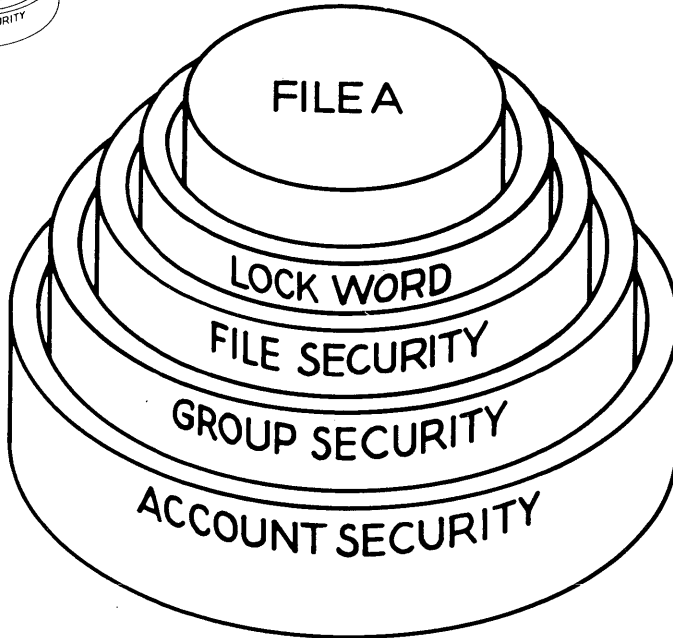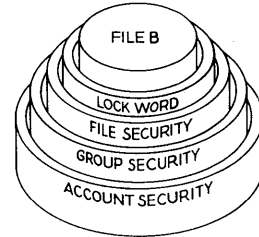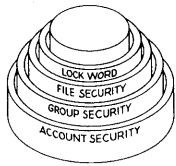
ACCOUNT SECURITY

Every file has 4 levels of security that must be passed each time the file is opened.

- ACCOUNT LEVEL (set by SYSTEM MANAGER)
- GROUP LEVEL (set be ACCOUNT MANAGER)
- FILE LEVEL (set by CREATOR)
- LOCKWORD (set by CREATOR)

HEWLETT **hp** PACKARD

ACCESS TO PERMANENT DISC FILES

ACCESS to Permanent Disc Files under default Security for a Standard user (ANY).

UNLIMITED ACCESS
- All Files in Log-on Group
- All Files in Home Group

READ (R) and EXECUTE (X) ACCESS ONLY
- All Files in PUB Group of Log-on Account
- All Files in PUB.SYS

NO ACCESS
- All Other Files in System

HEWLETT hp PACKARD

ADDRESSING ANY FILE

A filereference may be:

1)   filename [/ lockword] [.group [.account]]

2) $ ...

3)   *formaldesignator

TO REFERENCE ANY TYPE OF FILE FROM AN MPE COMMAND, OR FROM A PROGRAM, USE A "filereference"

A filereference may be:

1) filename[/ lockword][.group[.account]]   *Disc files*

2) $...   *any SYSTEM-DEFINED File*

3) *formaldesignator   *back-reference a previous :FILE command (only way to reference DEVICE files)*

SOME SYSTEM DEFINED FILES

$STDIN $STDINX $STDLIST

ADDRESSING ANY FILE

ADDRESSING A SYSTEM-DEFINED FILE

```
:RPG $STDINX,USL,$STDLIST
```

DEVICE FILE ACCESS via BACK-REFERENCE

```
:FILE LP;DEV=LP
:COBOL MYSOURCE/LOCK1.PUB.SOURCE,MYUSL,*LP
```

DEVICE FILE ACCESS via BACK-REFERENCE

```
:FILE   formaldesignator;DEV=device
:COBOL  textfile,uslfile,listfile
```

NOTE: All of these *files* are *filereferences* .

Example:

```
:FILE LP;DEV=LP
:COBOL MYSOURCE/LOCK1.PUB.SOURCE,MYUSL,*LP
```

(use a back-reference to access a Device file)

TOMBSTONES

```
/K NEWFILE.PUB
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
!  FILE NAME IS NEWFILE.PUB.INTRO               !
!  FOPTIONS: NEW,A,*FORMAL*,F,N,DEQ             !
!  AOPTIONS:  OUTPUT,SREC,NOLOCK,DEF,BUFFER     !
!  DEVICE TYPE: 0      DEVICE SUBTYPE: 8        !
!  LDEV: 2       DRT: 4           UNIT: 1       !
!  RECORD SIZE: 80    BLOCK SIZE: 1280  (BYTES) !
!  EXTENT SIZE: 10    MAX EXTENTS: 1            !
!  RECPTR: 3          RECLIMIT: 3               !
!  LOGCOUNT: 3           PHYSCOUNT: 1           !
!  EOF AT: 3          LABEL ADDR: %00200154007  !
!  FILE CODE: 0      ID IS STUDENT   ULABELS: 0 !
!  PHYSICAL STATUS: 1000000000000001            !
!  ERROR NUMBER: 93    RESIDUE: 640      (WORDS) !        SECURITY
!  BLOCK NUMBER: 1               NUMREC: 16     !        VIOLATION
+------------------------------------------------+
*60*K
FCLOSE FAILURE (93)

/JOIN WRONGONE

+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
!  ERROR NUMBER: 52    RESIDUE: 0             !          FILE NOT
!  BLOCK NUMBER: 0             NUMREC: 0      !          FOUND
+------------------------------------------------+
*51*K
FAILURE TO OPEN JOIN FILE (52)
/
```

HEADERS & TRAILERS

```
#S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
#S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
#S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM



      #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
      #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
      #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM



            #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
            #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
            #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM



            #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
            #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
            #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
```

HEADER

```
MAY 17, 1978, 11:21 PM
MAY 17, 1978, 11:21 PM
MAY 17, 1978, 11:21 PM
```

```
            #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
            #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
            #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM



         #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
         #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
         #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM



   #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
   #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
   #S15; #039  *  STUDENT.INTRO; EDTLIST  *  WED, MAY 17, 1978, 11:21 PM
```

TRAILER

HEWLETT *hp* PACKARD

USING FCOPY IN THE LAB

FCOPY — Standard System Utility to copy files.

```
:RUN FCOPY.PUB.SYS
>FROM=filereference1;TO=filereference2[;NEW]
>EXIT
```

WARNING Messages you may encounter:

200   FROM & TO file records are different lengths.

201   FROM & TO files are different types; one is ASCII,
                       one is BINARY

If either warning occurs, press ⬛ to continue.

FUNDAMENTALS LAB # 1

USING A LANGUAGE [40 minutes]

Choose your favorite language and do the associated chapter in "USING the HP-3000" (chapters 4 through 7).

USING EXISTING FILES [20 minutes]

The utility program 'FCOPY' in PUB.SYS will copy files. Its command syntax is:

> `>FROM=filereference1;TO=filereference2[;NEW]`

where:
filereferences 1 & 2 specify files to be used.

;NEW will create a new Permanent Disc File with exactly
the same attributes as the FROM file.

WARNING Messages you may encounter

'*200*' — Different record lengths in FROM & TO files.

'*201*' — FROM & TO files are different types; one is ASCII,
one is BINARY.

Disregard these warnings. Merely press the RETURN key and by-pass them. Operations will still be performed correctly.

(1) Execute FCOPY by keying in the following command in response to the ':' prompt:

RUN FCOPY.PUB.SYS

(2) In response to the '>' prompt, enter:

FROM=ASCII.PUB;TO=$STDLIST

This will list the contents of file ASCII in the PUB group of the INTRO account on your terminal.

(3) Now list the contents of file BINARY.PUB on your terminal. Expect warning '*201*'.

(4) Copy ASCII.PUB into a NEW file called MYASCII in your group.

HEWLETT hp *PACKARD*

FUNDAMENTALS LAB # 1

(5)   Copy SHORTY.PUB into a NEW file called SHORTY in your group.

(6)   List the contents of SHORTY on your terminal.

(7)   Copy the following from your terminal (use $STDINX) to file SHORTY:

        RECORD #1
        :RECORD #2
        #RECORD #3

You should encounter the end-of-file in 'SHORTY'. You are trying to put 3 records into a two record file.

(8)   Copy the same records to SHORTY but this time use a FROM file of $STDIN. Observe the difference between $STDINX & $STDIN. The ':' in the second record has signalled an end-of-file on FCOPY's input file (for data & commands) so FCOPY has returned to the MPE command interpreter. Call FCOPY again to proceed with the next step.

(9)   List the contents of SHORTY on your terminal.

(10)   List the contents of SHORTY on the line printer. Chances are you do not have an active FILE command pointing to the line printer and chances are this is not the last time you will be in this situation so remember:

        a)   Press the 'BREAK' Key.
        b)   When you receive the ':' prompt enter the command FILE LP;DEV=LP
        c)   Key in 'RESUME'. Another '>' prompt will not be issued, but you are back in FCOPY as indicated by 'READ PENDING', so issue the command to list SHORTY on the line printer (expect '*200*').

(11)   List MYASCII on the line printer. (expect '*200*')

(12)   List BINARY.PUB on the line printer. (expect '*200*' and '*201*')

(13)   Copy from $STDINX to ASCII.PUB. (Look-up resulting error code in your Software Pocket Guide).

(14)   Key in "EXIT" to end FCOPY and log-off the system.

HEWLETT **hp** *PACKARD*

FUNDAMENTALS LAB # 1

OPTIONAL: Proceed only if you have extra time.

You have unlimited file access in both your HOME & LOG-ON group. Log-on with your user name into your lab partner's group (if no partner, use group GTEACHER).

EXAMPLE: For partners JACK & JILL

`:HELLO JACK.INTRO,GJILL`

(1) Copy file SHORTY.PUB into your group as a NEW file called 'X' followed by your User-name (remember max of 8 chars). Specify no group name for the newfile, we'll see where it goes by default.
(2) Exit FCOPY and issue the command to list the attributes of your file (":LISTF Xyour-user,2"). Observe which of the two groups the file went into!
(3) Run FCOPY again and copy that new file into the HOME group as another NEW file 'Cyour-user'.
(4) Use LISTF to ascertain its attributes.
(5) Log-off the system.

HEWLETT *hp* PACKARD

EDITOR WORK FILES

ADD, HOLD
INSERT, HOLD
REPLACE, HOLD

HOLD FILE

(TEMPORARY)

HOLD

WORK FILE

(PERMANENT)

KEEP

TEXT
JOIN

ORIGINAL FILE

(PERMANENT)
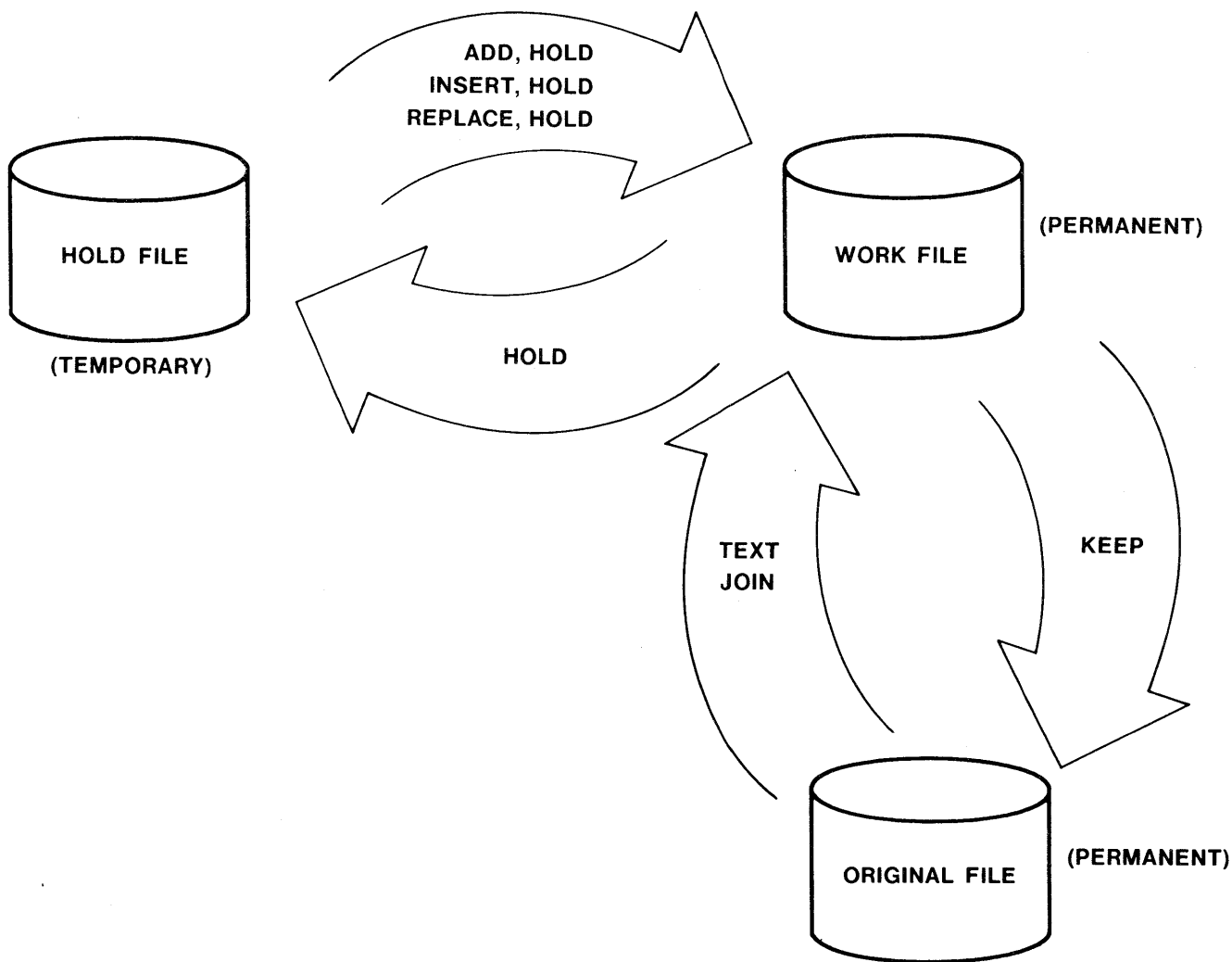
Work File is created by first ADD, TEXT or JOIN. Its name is:

Kdddhhmm

minute (0-59).

hour (0-23).

Julian day (1-365).

DEFINITION OF TERMS

'linenumber'   .001 through 99999.999 (999.999 for FORMAT=COBOL).

'string'   a string of characters enclosed within the same special (non-alphanumeric) character used as delimiters (except apostrophe, comma, semicolon, period, slash, backward slash, parenthesis, plus sign, minus sign, or asterisk). Usually quotes.

EXAMPLES:
"ARRAY" or #A "STRING"# or '7@BELLS@'7

NOTE: Non-printing characters may be represented by their decimal numeric equivalents preceded by an apostrophe (i.e. '7@BELLS@'7 represents the string "BELLS" preceded and followed by bell characters).

'position'

36	line 36 (default is column 1; same as 36(1)).
36(10)	column 10 of line 36.
36(+10)	10-th non-blank character beyond 1-st column of line 36(i.e. 11-th non-blank character in line 36).
FIRST	First line in the Work File.
LAST	Last line in the Work File.
36(LAST)	Last column in line 36.
36+10	The 10-th line after line 36 (not necessarily line 46 !).
LAST-10	The 10-th line before the last line.
"ARRAY"	An occurrence of the string "ARRAY" (internal pointer will be positioned to the 1-st character of "ARRAY").

HEWLETT hp PACKARD

DEFINITION OF TERMS

'position' (cont'd)

"ARRAY"(+15)	The 15-th non-blank character after the first character in "ARRAY".
*	The position currently pointed to by the internal pointer.
*(15)	Column 15 of the current line.

MOST OFTEN USED: linenumber,*,FIRST,LAST, & "STRING"
USEFUL IN RPG: (column)
WORD PROCESSING: (+chars)

'range'

                    "ALL" or characters from one 'position' to another inclusive.

ALL	All lines in the Work File.
36/45	Beginning of line 36 through the end of line 45.
FIRST/"ARRAY"	From the first column of the first line through the "Y" in the first occurrence of the string "ARRAY".
36(10)/"ARRAY"(+15)	From the 10-th column of line 36 through the 15-th non-blank beyond the last character of "ARRAY".

MOST USED: "ALL" & linenumber/linenumber.

'rangelist'

                    A series of 'ranges' separated by commas.

EXAMPLE: FIRST,36/45,LAST-10/LAST

NOTE: See section 3 of the EDITOR manual for detailed examples.

only need 1st character

suppress listing

HEWLETT **hp** PACKARD

ADD

A[DD][Q] [linenumber][,HOLD[Q][,NOW]]

```
:HELLO STUDENT.INTRO/PASSWORD
HP3000 III.  TUE, FEB  7, 1978,  9:33 AM
:EDITOR
HP32201A.7.0H EDIT/3000   TUE, FEB  7, 1978,  9:34 AM
(C) HEWLETT-PACKARD CO. 1976
/ADD                          create new lines
    1      ORIGINAL LINE 1.
    2      ORIGINAL LINE 2.
    3      //

...
/ADD                          default is add to end
    3      ADDITIONAL LINE 1.
    4      ...                             <CNTL-Y>
/ADD 2                        will NOT overlay existing lines
*15*X COMMAND WILL NOT REPLACE OR INTERLEAVE LINES
/ADD 2.03                     insert new lines
    2.03   INSERT LINE 1.
    2.04   INSERT LINE 2.
    2.05   ...                             <CNTL-Y>
/
```

LIST

L[IST][Q] range [,UNN[UMBERED]] [,OFFLINE]
[,TRANSLATE] [,NOTEXT]

```
/LIST ALL
      1        ORIGINAL LINE 1.
      2        ORIGINAL LINE 2.
      2.03     INSERT LINE 1.
      2.04     INSERT LINE 2.
      3        ADDITIONAL LINE 1.
/L 2/3
      2        ORIGINAL LINE 2.
      2.03     INSERT LINE 1.
      2.04     INSERT LINE 2.
      3        ADDITIONAL LINE 1.
```

OFFLINE LISTINGS

```
/LIST ALL,OFFLINE                    automatically goes to DEV=LP
*** OFF LINE LISTING BEGUN. ***
/
```

OFFLINE LISTINGS to other than DEV=LP...

```
:FILE ABC;DEV=FASTLP      By issuing a file command and back-referencing it, you
:EDITOR *ABC              may direct offline listings to any file (or device).
```

```
/                         <BREAK> pressed.
:FILE EDTLIST;DEV=FASTLP   If already within the EDITOR,
:RESUME                    BREAK out and reference the
READ PENDING               formaldesignator EDITOR will
LIST ALL,OFFLINE           use.
*** OFF LINE LISTING BEGUN. ***
```

MODIFY

M[ODIFY][Q] rangelist

```
/MODIFY 2
MODIFY      2
ORIGINAL LINE 2.
  DDD
ORNAL LINE 2.                              Delete
  D    D
ORNE 2.
RLINE                                      Replace
LINE 2.
IDUPLICATE OF                              Insert
DUPLICATE OF LINE 2.
//                                         Restore Original
MODIFY      2
ORIGINAL LINE 2.
              DITWO                        Delete then Insert
ORIGINAL LINE TWO.
[RETURN]

/L 1/2
     1      ORIGINAL LINE 1.
     2      ORIGINAL LINE TWO.
/
```

HEWLETT hp PACKARD

GATHER

G[ATHER][Q] range TO linenumber [BY increment]

```
/GATHER 2.03/2.1 TO 2.99 BY .002
     2.03   =>       2.99              move lines to another
     2.04   =>       2.992            location in work file
/L ALL
     1        ORIGINAL LINE 1.
     2        ORIGINAL LINE TWO.
     2.99   INSERT LINE 1.
     2.992  INSERT LINE 2.
     3        ADDITIONAL LINE 1.
/G 1/2 TO 100
     1        =>    100
     2        =>    101
/L ALL
     2.99   INSERT LINE 1.
     2.992  INSERT LINE 2.
     3        ADDITIONAL LINE 1.
   100       ORIGINAL LINE 1.
   101       ORIGINAL LINE TWO.
/G ALL                                renumber work file
/L ALL
     1        INSERT LINE 1.
     2        INSERT LINE 2.
     3        ADDITIONAL LINE 1.
     4        ORIGINAL LINE 1.
     5        ORIGINAL LINE TWO.
/
```

CHANGE

C[HANGE][Q] {col[/col]} TO string IN rangelist

{ string }

```
/L ALL
      1        INSERT LINE 1.
      2        INSERT LINE 2.
      3        ADDITIONAL LINE 1.
      4        ORIGINAL LINE 1.
      5        ORIGINAL LINE TWO.
/CHANGE "TWO" TO "2" IN 5                    strings
      5        ORIGINAL LINE 2.
/C 1/9 TO "" IN 5                            cols / null string
      5        LINE 2.
/C 1 TO #THE "ORIGINAL" # IN 5               insert before col
      5        THE "ORIGINAL" LINE 2.
/C @I@ TO @X@ IN ALL                         changes ALL occurrences
      1        XNSERT LXNE 1.
      2        XNSERT LXNE 2.
      3        ADDXTXONAL LXNE 1.
      4        ORXGXNAL LXNE 1.
      5        THE "ORXGXNAL" LXNE 2.
/CQ "X" TO "I" IN ALL                        'quiet' change
/C "LINE 2" TO "LINE 2" IN ALL               'find' all occurrences
      2        INSERT LINE 2.
      5        THE "ORIGINAL" LINE 2.
/
```

KEEP

KEEP [filereference [(range)] [,UNN[UMBERED]]]
- or -
KEEPQ filereference

```
/L ALL
      1       INSERT LINE 1.
      2       INSERT LINE 2.
      3       ADDITIONAL LINE 1.
      4       ORIGINAL LINE 1.
      5       THE "ORIGINAL" LINE 2.
/KEEP KEEPNUM
/K KEEPNUM
KEEPNUM ALREADY EXISTS - RESPOND YES TO PURGE OLD AND THEN KEEP
PURGE OLD?N
PURGE OF OLD FILE NOT CONFIRMED - TEXT NOT KEPT
/K KEEPUNN,UNNUMBERED
/K KEEPUNN,UNN
KEEPUNN ALREADY EXISTS - RESPOND YES TO PURGE OLD AND THEN KEEP
PURGE OLD?Y
/                                       BREAK key pressed
:LISTF,2
ACCOUNT=  INTRO         GROUP=  GSTUDENT

FILENAME  CODE    ------------LOGICAL RECORD-----------   ----SPACE----
                  SIZE   TYP         EOF       LIMIT R/B  SECTORS #X MX

K0381034* EDITQ   112B   FA           7        2000   9      896 16 16
KEEPNUM           80B    FA           5           5  16       10  1  1
KEEPUNN           72B    FA           5           5  16       10  1  1

:RESUME
READ PENDING
L 1
      1       INSERT LINE 1.
/
```

DELETE

D[ELETE][Q] rangelist

```
/L ALL
      1      INSERT LINE 1.
      2      INSERT LINE 2.
      3      ADDITIONAL LINE 1.
      4      ORIGINAL LINE 1.
      5      THE "ORIGINAL" LINE 2.
/DELETE 5
      5      THE "ORIGINAL" LINE 2.
/D 1/2,LAST                               'rangelist'
      1      INSERT LINE 1.
      2      INSERT LINE 2.
      4      ORIGINAL LINE 1.
/L ALL
      3      ADDITIONAL LINE 1.
/DELETE ALL                               clear work file
IF IT IS OK TO CLEAR RESPOND "YES"        EDITOR will double-check
CLEAR? N
CLEAR NOT CONFIRMED - TEXT IS UNCHANGED
/L ALL
      3      ADDITIONAL LINE 1.
/D ALL
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
/A
      1      LINE NOS. START OVER.
      2      ...
/L ALL
      1      LINE NOS. START OVER.
/
```

TEXT

T[EXT] filereference[{   (linenumber/linenumber)   }]
{   (#recnum/#recnum)   }

[,UNN[UMBERED]]

```
/TEXT KEEPUNN                                    TEXT does not list
 FILE UNNUMBERED
/L ALL
      1         INSERT LINE 1.
      2         INSERT LINE 2.
      3         ADDITIONAL LINE 1.
      4         ORIGINAL LINE 1.
      5         THE "ORIGINAL" LINE 2.
/T KEEPNUM(2/4)                                  TEXT clears Work File
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
/L ALL
      2         INSERT LINE 2.
      3         ADDITIONAL LINE 1.
      4         ORIGINAL LINE 1.
/
```

> TO TEXT IN A CARD DECK — Back-reference a file equation pointing to the card reader. Your card deck must be prefaced with a ':DATA' card and end with a ':EOD' card.

```
:FILE CARDIN;DEV=CARD
:EDITOR
   .  .  .
/TEXT *CARDIN
```

JOIN

J[OIN][Q] filereference [{ (linenumber[/linenumber]) }]
{ (#recnum/#recnum) }

[TO linenumber] [BY increment] [[,]UNN[UMBERED]]

```
/D ALL
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
/JOIN SHORTF.PUB                                    JOIN will list
        1     THIS IS LINE ONE
        2     THIS IS LINE TWO
        3     THIS IS LINE THREE
        4     THIS IS LINE FOUR
/JQ SHORTF.PUB(2/3) BY .001                         subset if numbered
NUMBER OF LINES JOINED = 2
/J SHORTF.PUB(#1/#4) TO .1                           numbered or unnumbered
       .1     THIS IS LINE ONE
       .2     THIS IS LINE TWO
       .3     THIS IS LINE THREE
       .4     THIS IS LINE FOUR
/L ALL
       .1     THIS IS LINE ONE
       .2     THIS IS LINE TWO
       .3     THIS IS LINE THREE
       .4     THIS IS LINE FOUR
        1     THIS IS LINE ONE
        2     THIS IS LINE TWO
        3     THIS IS LINE THREE
        4     THIS IS LINE FOUR
        5     THIS IS LINE TWO
    5.001     THIS IS LINE THREE
/
```

HOLD

H[OLD][Q] [range [,APPEND]]

```
/T SHORTF.PUB
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
/HOLD FIRST
THIS IS LINE ONE
/HQ 3/4,APPEND
HOLD FILE LENGTH IS 3 RECORDS
/ADD .01,HOLD,NOW                        with ',NOW'
        .01   THIS IS LINE ONE
        .02   THIS IS LINE THREE
        .03   THIS IS LINE FOUR

...
/ADD,HOLD                                without ',NOW'
        5     *** WITHOUT ",NOW" YOU ARE GIVEN A CHANCE TO
        6     *** ENTER LINES BEFORE DATA ADDED FROM HOLD.
        7     //

...
        7     THIS IS LINE ONE
        8     THIS IS LINE THREE
        9     THIS IS LINE FOUR

...
/L ALL
        .01   THIS IS LINE ONE
        .02   THIS IS LINE THREE
        .03   THIS IS LINE FOUR
        1     THIS IS LINE ONE
        2     THIS IS LINE TWO
        3     THIS IS LINE THREE
        4     THIS IS LINE FOUR
        5     *** WITHOUT ",NOW" YOU ARE GIVEN A CHANCE TO
        6     *** ENTER LINES BEFORE DATA ADDED FROM HOLD.
        7     THIS IS LINE ONE
        8     THIS IS LINE THREE
        9     THIS IS LINE FOUR
/HOLD                                    clear hold file
CLEAR HOLD? Y
/A,HOLD
WARNING - HOLD IS NULL
/
```

REPLACE

R[EPLACE][Q] rangelist [,HOLD[Q] [,NOW]]

```
/T SHORTF.PUB
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
/M 1
MODIFY       1
THIS IS LINE ONE
RLINE #1.
LINE #1.LINE ONE                        Replaces characters
RETURN

/REPLACE 1,LAST
      1      LINE #1.LINE ONE
      1      NO. 1                       Replaces whole line
      4      THIS IS LINE FOUR
      4      NO. 4
/L ALL
      1      NO. 1
      2      THIS IS LINE TWO
      3      THIS IS LINE THREE
      4      NO. 4
/
```

XPLAIN & END

X[PLAIN]{command} [,OFFLINE]   &   {E[ND] }
{ ALL }   {E[XIT]}

```
/XPLAIN
XPLAIN:  TO OBTAIN AN EXPLANATION OF THE COMMANDS
EXAMPLE:  XPLAIN A,SET,F

/X M,R
MODIFY:  TO MODIFY TEXT IN THE TEXT FILE USING THREE OPERATIONS OF
DELETE(D), INSERT(I), AND REPLACE(R)
EXAMPLE:  MODIFY 50/100

REPLACE:  TO REPLACE LINES IN THE TEXT FILE
EXAMPLE:  REPLACE 10/20,HOLD,NOW

/X ALL,OFFLINE
*** OFF LINE LISTING BEGUN. ***
/EXIT
IF IT IS OK TO CLEAR RESPOND "YES"        only if Text file altered
CLEAR? N
CLEAR NOT CONFIRMED - TEXT IS UNCHANGED
/END
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? N
CLEAR NOT CONFIRMED - TEXT IS UNCHANGED
/E
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
                                          Work File space freed
 END OF SUBSYSTEM
:
```

EDITOR LAB # 1 [1.0 hour]

1) Log-on the terminal and invoke the EDITOR anticipating output to the line printer. Issue FILE commands if necessary.
2) TEXT in the file LABEDIT1.PUB.
3) Produce an offline listing for reference & go get it off the line printer.
4) Change "COAL" to "GOAL" in line 7.
5) Add a line after line 21 containing just "EFFICIENTLY".
6) Insert two blank lines following line 22.
7) Insert "FOR" in front of "THEM" in line 33.
8) Insert the missing line: "FOR EXAMPLE, IN A PROGRAM PREPARED TO SOLVE THE INVENTORY PROBLEM" before line 34.
9) Change "NEGSJBIH" to "NEGATIVE" in line 41.
10) Change "T-G-G" to "STER" in line 46.
11) Delete "(DELETE)" from line 53.
12) Insert a period after "CARD" in line 57 and delete the following part of the line.
13) Add your name to line 59.
14) Insert the whole paragraph contained in disc file PARA1.PUB in front of line 16. Do NOT affect numbers of lines already in the WORK file. You should be able to do all of this with just one command.
15) Lines 49 and 50 are out of place. With one command move them in front of line 56.
16) Renumber the file.
17) Obtain another listing on the line printer to double-check your changes.
18) Save the file under the name EDLAB1 in your log-on group

The remainder of this LAB is optional. Only do it if you have extra time. Otherwise proceed to step "END)".

19) Try to keep another copy as file EDLAB1 in PUB (from where you originally read your file for this exercise; you can read from PUB but cannot save a file into it).
20) The filename specified in the KEEP command is a "filereference". So keep another copy as file EDLOCK1 with a LOCKWORD.
21) TEXT EDLOCK1 back in specifying an incorrect lockword. Look up the resulting error code in your pocket guide. TEXT it in again supplying the correct lockword.
22) Keep it again as EDLOCK1 but supply a different lockword.
23) Obtain an XPLAIN listing of all commands offline on the line printer. (Be sure to pick up all of your listings from the line printer.)
END) Exit from the EDITOR and log-off.

<< End >>

EDITOR — ADDITIONAL TOPICS.

V[ERIFY] [{optionlist}]
{ ALL }

Note: 'optionlist' is any combination of the options displayed under VERIFY ALL separated by commas.

```
/VERIFY                                        list pointer position
      9      THIS IS LINE FOUR
             ^(1  )
/V ALL                                         plus parameter settings
      9      THIS IS LINE FOUR
             ^(1  )
POLL = TRUE  (I.E. BATCH = FALSE)
REAR = TRUE  (I.E. FRONT = FALSE)
DELTA =      1
CURRENT DEPTH = 0, THE DEPTH LIMIT = 10
RIGHT =72
LENGTH = 72
LONG = TRUE  (I.E. SHORT = FALSE)
TIME = 50
TOTAL NUMBER OF CURRENT LINES = 9
FROM =     1
LEFT =   1
FIXED = TRUE  (I.E. VARIABLE = FALSE)
SIZE = 0
DISPLAY = TRUE  (I.E. QUIET = FALSE)
FORMAT=DEFAULT
NO TABS USED
FILES:
     WORK: K0391741
     KEEP: SHORTF.PUB.INTRO            WED, FEB  8, 1978,  5:41 PM
     TEXT: SHORTF.PUB.INTRO            WED, FEB  8, 1978,  5:41 PM
           TEXT FILE HAS BEEN ALTERED
     JOIN: SHORTF.PUB.INTRO            WED, FEB  8, 1978,  5:41 PM
/V FORMAT,LEFT,RIGHT,SIZE
FORMAT=DEFAULT
LEFT =   1
RIGHT =72
SIZE = 0
/
```

SET

S[ET] optionlist

Note: 'optionlist' is list of options separated by commas. See "Software Pocket Guide" for options.

```
/SET TABS
/V TABS
TABS = ( 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 65, 71)
/SET FORMAT=COBOL
WARNING - 'LENGTH' 72 RESET TO 74
WARNING - 'RIGHT' 72 RESET TO 74
*** WARNING *** COBOL VALUES SET FOR LENGTH, RIGHT, FROM, DELTA, FRONT
/SET TABS,SIZE=500,RIGHT=65
/VERIFY ALL
     9      THIS IS LINE FOUR
^(1  )
POLL = TRUE (I.E. BATCH = FALSE)
FRONT = TRUE (I.E. REAR = FALSE)
DELTA =      .1
CURRENT DEPTH = 0, THE DEPTH LIMIT = 10
RIGHT =65
LENGTH = 74
LONG = TRUE (I.E. SHORT = FALSE)
TIME = 50
TOTAL NUMBER OF CURRENT LINES = 9
FROM =     1
LEFT =   1
FIXED = TRUE (I.E. VARIABLE = FALSE)
SIZE = 500
DISPLAY = TRUE (I.E. QUIET = FALSE)
FORMAT=COBOL
TAB CHARACTER = '9                    TAB Key on HP-264x
TABS = ( 6, 10, 14, 18, 22, 26, 30, 34, 38, 46, 54, 67)
FILES:
   WORK: K0391741
   KEEP: SHORTF.PUB.INTRO           WED, FEB 8, 1978, 5:41 PM
   TEXT: SHORTF.PUB.INTRO           WED, FEB 8, 1978, 5:41 PM
         TEXT FILE HAS BEEN ALTERED
   JOIN: SHORTF.PUB.INTRO           WED, FEB 8, 1978, 5:41 PM
/
```

HEWLETT hp PACKARD

USE & Q

U[SE] [filereference]   Q string

```
/D ALL
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
/A
    1       Q"This message was generated by a 'Q' command."
    1.1     Q"SET FORMAT=COBOL,TABS"
    1.2     Q"VERIFY FORMAT,TABS";<<Use file may have comments>>
    1.3     S FORMAT=COBOL,TABS;V FORMAT,TABS
    1.4     //
  ...
/K USEFILE,UNN
USEFILE ALREADY EXISTS - RESPOND YES TO PURGE OLD AND THEN KEEP
PURGE OLD?Y
/U USEFILE
This message was generated by a 'Q' command.
SET FORMAT=COBOL,TABS
VERIFY FORMAT,TABS
WARNING - 'RIGHT'  65 RESET TO 74
*** WARNING *** COBOL VALUES SET FOR LENGTH, RIGHT, FROM, DELTA, FRONT
FORMAT=COBOL
TABS = (  6, 10, 14, 18, 22, 26, 30, 34, 38, 46, 54, 67)
/
```

EDITOR LAB #2 [1.0 hour]

To use tabbing with EDIT/3000 on an HP-264x terminal, you must: 1) Enable the TABCHAR and TAB stops in the EDITOR with the SET command & 2) Set the TAB stops in the terminal either physically or with escape characters.

There are predefined USE files in PUB.INTRO that do all this. They are: 1) For COBOL, COBTABS 2) For RPG, RPGTABS & 3) For default formats, DEFTABS.

As an example of using TABs we are going to set tabs for COBOL programs then modify a COBOL program. This demonstrates TABs and does not require any knowledge of COBOL.

1) Log-on & invoke the EDITOR.
2) USE COBTABS.PUB. This will enable the EDITOR program to recognize the TAB Key and set corresponding TAB stops in the program and in the terminal. An extra goody is it also locks a picture of record positions and tab stops at the top of your terminal screen (notice MEMORY LOCK is on).
3) VERIFY format settings and tab settings with VERIFY ALL.
4) TEXT in COBTEST1.PUB (This is a copy of COBOL program from "Using the HP-3000").
5) List the program. We want to indent line 3.6 ("IF Y-N = "N" GO TO ENTER-ROUTINE.") to the next tab stop. Try to use MODIFY to insert 4 additional spaces by pressing the TAB Key and keying in I followed by 4 spaces. You get the 'INVALID' message because the 1-st character encountered is the TABCHAR, not 'R', 'D', nor 'I'. The TAB key is only recognized by the ADD and REPLACE commands.
6) The logical thing to do would be to insert 4 spaces within MODIFY without using the TAB key, but let's practice using the REPLACE command. So REPLACE line 3.6 with its same contents but indented to the next tab stop.
7) Now ADD line 3.61. Also indent it to the second tab stop and enter the contents "IF Y-N = "n" GO TO ENTER-ROUTINE.".
8) KEEP the file both numbered and unnumbered in your group as EDLAB2 & EDLAB2U.
9) Press the <BREAK> Key and when you receive the ":" prompt use LISTF,1 to look at the record sizes of EDLAB2 and EDLAB2U. Observe that 6 additional characters are added to each line of the file when it is kept numbered. Also observe the record sizes of files TRY1 & TRY1UNN in PUB. These are numbered and unnumbered versions of the FORTRAN program from chapter 3 of "Using the HP-3000".
10) Key in "RESUME". You are now back in the EDITOR (remember the "/" prompt will NOT be re-issued). Exit the EDITOR.

EDITOR LAB #2

11) RUN FCOPY.PUB.SYS and list all four files to your terminal screen. Observe where line numbers are put for the different formats and the length of the line numbers.

12) EXIT FCOPY and Log-off the system.

OPTIONAL — Proceed only if you have time.

Obtain a listing on the line printer of any of these tab use files in PUB you are interested in transferring to your system. All contain CNTL characters and lower case characters that will not be represented correctly on the line printer. Before listing it, text it into your Editor work file and change the ESCAPE character to '!' everywhere it is found in the file. List it on the line printer. If the line printer does not have lower case characters, all lower case characters will print as upper case. Get your listing from the line printer, list the work file on your terminal screen (you must have DISPLAY FUNCTIONS on or your terminal will attempt to execute ESCAPE characters rather than list them) & circle the characters on your printed listing that should be lower case.

<< End >>

DEFINITION OF A FILE SYSTEM

- All devices are treated as 'FILES'.

- ALL access to files is accomplished through a portion of MPE called the FILE SYSTEM.

- This includes ALL Input and Output:
    —I/O to Disc Files.
    —I/O to Device Files.
    —I/O to System Defined Files.

- The :FILE command can alter the way the FILE SYSTEM performs I/O for a particular program at run time.

- A program refers to a file with the 'formaldesignator'.

- MPE knows a file by the 'actualdesignator'.

DEFINITION

`:FILE formaldesignator[=filereference][;DEV=device]`

- Have a program use another DISC file.

      `:FILE FILE1=ANOTHER`
      `:RUN MYPROG`

- Have a program use a different DEVICE (DEVICE INDEPENDENCE).

      `:FILE FILE1;DEV=LP`
      `:RUN MYPROG`

```
                    :FILE
                   CHANGES
                    PARMS
                                                  ┌──────────┐
                                              ──▶ │   LINE    │
                                                  │  PRINTER  │
                                                  └──────────┘
┌──────────┐            ┌──────────┐
│          │  OUTPUT    │   FILE   │
│ PROGRAM  │ ─────────▶ │  SYSTEM  │ ─ ─ ─ ─▶    ┌──────────┐
│          │            │          │             │   DISC   │
└──────────┘            └──────────┘             └──────────┘
```
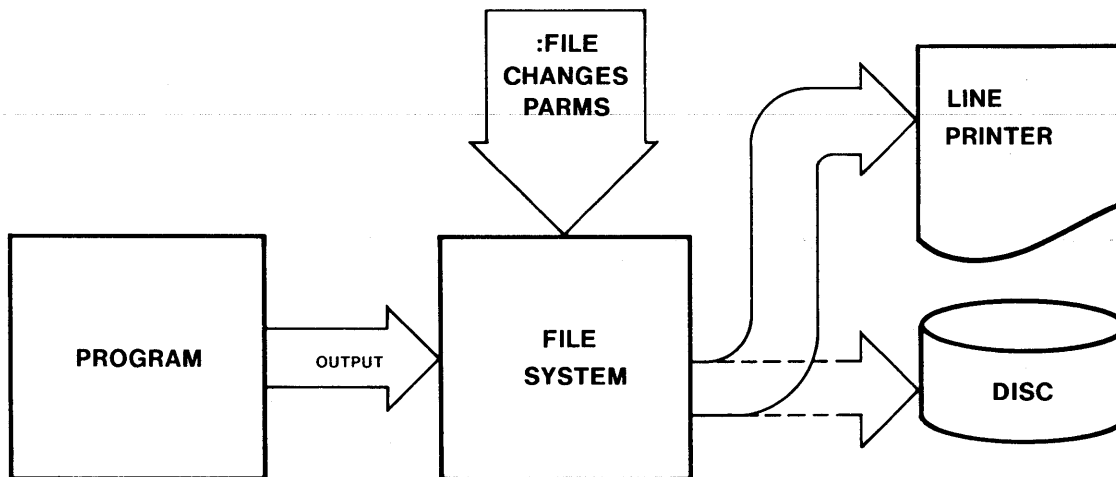
DISC FILES IN GENERAL

logical device *(priv)*

— *FREE.PUB.SYS*

- One system-wide directory on ldev = 1.

- FREE SPACE table on each disc volume.

- DISC Files in any domain may extend over all discs of
  device class 'DISC'.

- A disc pack must always reside on the same ldev.

- All discs must be on-line when the system is running.

DISC FILE DOMAINS

```
                                    [,NEW]
    :FILE formaldesignator[[=filereference][,OLD]    ]
                                    [,OLDTEMP]
```

PERMANENT DISC FILES: [,OLD]
- Known System-wide.
- Directory on Idev=1 points to each Permanent file.
- Space used is applied against any limits set for the Group and Account.

TEMPORARY DISC FILES: [,OLDTEMP]
- Known only to creating Session or Job.
- Exist only for duration of Session or Job.
- Occupy space taken from free space tables.
- Space is NOT applied against any limits set for Group and Account until file is made permanent.

NEW DISC FILES: [,NEW]
- Known only to creating process.
- Exist only for duration of Process.
- Occupy space taken from free space tables.
- Space is NOT applied against any limits set for Group and Account until file is made permanent.

DISPOSITION UPON CLOSE

```
                              [;DEL ]
                  :FILE  ...  [;SAVE]
                              [;TEMP]
```

Disc FILES may be moved to a more permanent domain upon being CLOSEd .

:SAVE Y
  -OR-
:FILE Y, OLDTEMP; SAVE

FCLOSE
(WITH PERM OPTION)
    -OR-
:FILE X, NEW; SAVE

PERMANENT

TEMPORARY

NEW

FCLOSE
(WITH TEMP OPTION)
  -OR-
:FILE Z, NEW; TEMP

ALL SYSTEM-DEFINED FILES

INPUT	OUTPUT	
		terminal in
$STDIN[X]	$STDLIST	a Session
$OLDPASS	$OLDPASS	default
	$NEWPASS	work files
$NULL	$NULL	'bit-bucket'

CAN ONLY APPEAR ON RIGHT SIDE OF A :FILE EQUATION.

- $STDIN[X] & $STDLIST—your terminal in SESSION mode.
- $OLDPASS—a TEMPORARY work file unique to your Session or Job that lasts for the duration of that Session or Job.
- $NEWPASS—a virgin TEMPORARY work file that is renamed to $OLDPASS when closed.
- $NULL Input—instant end-of-file.
     Output—Process operates normally but output records go nowhere (into 'bit-bucket').

FILE ACCESS / SECURITY

MODE   ## USER

R	Read		ANY	Any user (Standard user)
L	Lock		AC	Account Member
A	Append		AL	Account Librarian
W	Write		GU	Group User
X	Execute		GL	Group Librarian
S	Save		CR	user who CReated file

[Append implies Lock]

[Write implies Append (and Lock)]

IMPLICIT FILE SECURITY RULES

- All users may only create files (SAVE) in their own accounts.
- Only the USER who originally created file (CR) may alter its security settings (at the File level).
- If a LOCKWORD is present, it is always required.
- Account Manager (AM) has unlimited access to files within his/her account.
- System Manager (SM) has unlimited access to all files in the system but may only create files (SAVE) in his/her account.
- :RELEASE allows unlimited file access.
- :RELEASE suspends but does not modify security settings at the Account, Group and File levels (Lockword still required).

NOTE: AM and SM implicitly have their file access capabilities; Security settings at the Account, Group, and File levels have no affect on their capabilities. They must still supply file Lockwords if present, however!

FILE ACCESS & SECURITY

A combination of 4 separate levels of settings:

YOU

ACCOUNT
LEVEL

GROUP
LEVEL

FILE
LEVEL

LOCK WORD

YOUR FILE

SET BY	SM	AM	CR	CR
DEFAULT	(R,L,A,W,X:AC)	(R,L,A,W,X,S:GU)	(R,L,A,W,X:ANY)	—
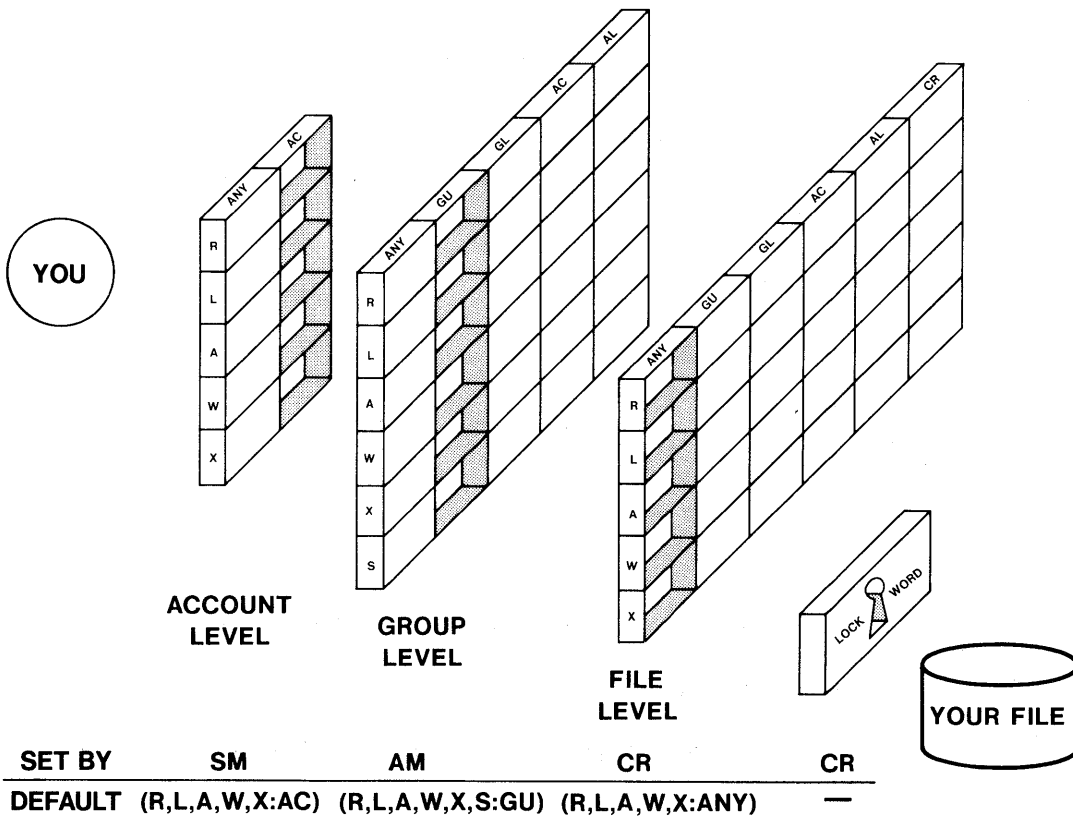
MPE COMMANDS TO USE THE WHOLE SYSTEM

:RENAME oldfilereference,newfilereference[,TEMP]

- Used by CR only.
- Change filename and / or lockword.
- Move file to a new group (same domain).
- Permanent or Temporary files.

:RENAME PROG1,NEWNAME/LOCKWORD

:RENAME PROG2,PROG2.NEWGROUP

```
        {tempfilereference           }
:SAVE {$OLDPASS,newfilereference}
```

Move a Temporary file into the Permanent domain (cataloged).

:RELEASE filereference

Suspend file security (CR only). Lockword still required.

:SECURE filereference

Restore security settings on a released file (CR only).

MPE COMMANDS TO USE THE WHOLE SYSTEM

```
        {@                    }
:RESET {formaldesignator}
```

Remove :FILE command for one or all (@) formaldesignators for your Session / Job.

```
:ALTSEC filereference[;([modelist:userlist[;...]])]
```

Modify security settings at the file level (CR only).

<u>:ALTSEC MYFILE;(R,X:ANY;W:CR)</u>

Allows any user who can pass Account & Group level security settings to READ or EXECUTE the file. Only the CReating user may have WRITE, APPEND, or LOCK access (W implies A & L).

<u>:ALTSEC MYFILE</u>

Restores default security at the file level to the file (i.e. (R,X,L,A,W:ANY)).

MPE COMMANDS TO USE THE WHOLE SYSTEM

:RUN LISTEQ2.PUB.SYS

List a Session's / Job's active file commands and Temporary disc files.

NOTE: The suffix '2' is omitted if run on Series I, CX or pre-CX 3000's.

:RUN LISTDIR2.PUB.SYS

List a file's security settings and your access to it.

:PREP uslfile,progfile

Prepare an object file (USL) into a runnable program.

:PREPRUN uslfile

Do a PREP then execute the resulting program in $OLDPASS.

COMPILERS

- BASICOMP   the BASIC compiler
- BASICPREP   compile & prep
- BASICGO   compile, prep, & run

- COBOL   the COBOL compiler
- COBOLPREP   compile & prep
- COBOLGO   compile, prep & run

- FORTRAN   the FORTRAN compiler
- FORTPREP   compile & prep
- FORTGO   compile, prep, & run

- RPG   the RPG compiler
- RPGPREP   compile & prep
- RPGGO   compile, prep, & run

- SPL   the SPL compiler
- SPLPREP   compile & prep
- SPLGO   compile, prep, & run

COBOLPREP : v ttne

COBOL

PREP

RUN

COMPILERS

All Compilers but BASICOMP Merge Source Files during Compile

TEXT FILE   MASTER FILE   NEW FILE
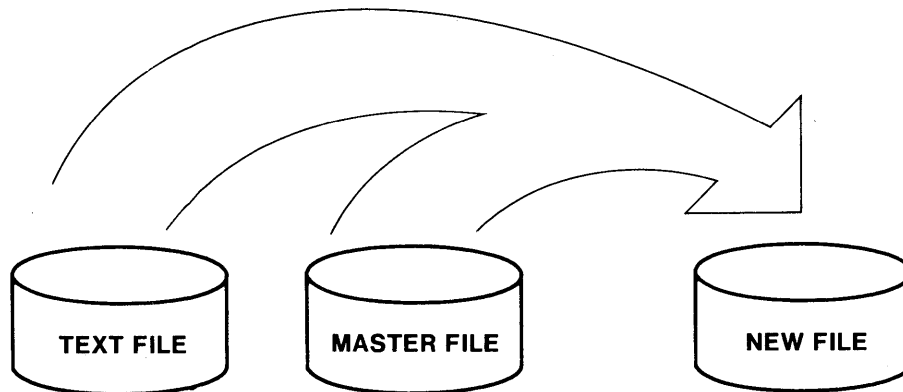
MORE TYPICALLY — Specify one complete source file as 'textfile'

`:RPG  MYSOURCE,MYUSL,*LP`

RULES FOR COMPILERS

- 'files' specified are all 'filereferences'.
- Compilers and PREP create files.
  - —If you specify an optional filereference:
    An existing file with correct file code will be re-used.
    OR a new file with correct file code will be created.
  - —If you do NOT specify an optional filereference — default work files will be used.

HEWLETT **hp** PACKARD

COMPILERS

:SPLGO TEXT FILE

① TEXT FILE → ② COMPILE → $OLDPASS -OR- $NEWPASS → USL ③

USL → ④ PREP → ⑤ → $NEWPASS → PROG ⑥ ------→ $OLDPASS

PROG → $OLDPASS → ⑦ → RUN ⑧
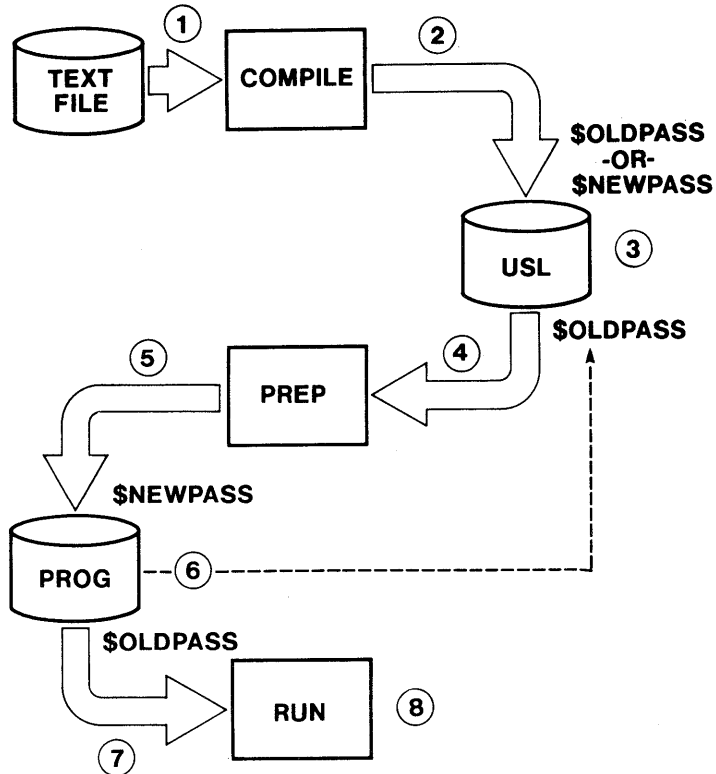
1) Source ('textfile') compiled.

2) Object written to $OLDPASS if it exists as a USL file, else it goes into $NEWPASS.

3) Name will be $OLDPASS after close.

4) Object prepared from $OLDPASS.

5) Program written to new $NEWPASS.

6) $NEWPASS closed. Name changed to $OLDPASS. Old $OLDPASS deleted.

7) Run $OLDPASS.

8) $OLDPASS (program file) not changed nor deleted after run.

FILES LAB # 1 [1.0 hour]

Remember unlimited file access is granted in both your log-on and home group. So log-on as your lab partner into your group. For 'YOU' and 'PARTNER':

:HELLO PARTNER.INTRO/PASSWORD,GYOU

If you don't have a lab partner use 'TEACHER':

:HELLO TEACHER.INTRO/PASSWORD,GYOU

Proceed with the lab filling in the questions as you go. Don't dwell too long on any question; they will be covered later.
See your "Software Pocket Guide" for complete command syntax.

1) Compile the COBOL program COBTEST1 in PUB. Enter 'COBOL COBTEST1.PUB' and use the default USL and list files.

??? By default the listing is on ??? _____

2) Issue ':LISTF,1' to see if the USL file is a permanent file. ':RUN LISTEQ2.PUB.SYS'; it's not listed among the normal temporary files either. The result is in file $OLDPASS. The only way we can determine the attributes of $OLDPASS is save it then LISTF. Enter ':SAVE $OLDPASS,XYZ' , if XYZ already exists, rename it. Now do a ':LISTF XYZ,2' and notice which group you're using.

??? Into what file does object output go by default??? _____

3) The compilers and :PREP will create files. Re-compile COBTEST1.PUB into a uslfile of 'USL' and use a listfile of '$NULL' (this is one way to inhibit listings). Now enter ':PREP USL' and re-examine the :PREP command's syntax in your pocket guide carefully; the progfile is required! You can use the default work file in the following manner, enter: ':PREP USL,$NEWPASS'. As $NEWPASS is closed its name is changed to $OLDPASS. Now run the program file that has resulted from the :PREP. Rename $OLDPASS to 'PROG1' (remember its a TEMP file). RUN LISTEQ2 to see it has taken affect. Make PROG1 a permanent file.

??? In the :PREP command you must specify both
        'uslfile' and 'progfile' (Yes/No) ??? _____
??? What System-defined file can be used to eliminate
        compiler listings ??? _____

HEWLETT **hp** PACKARD

FILES LAB #1

4) Compile and PREP COBTEST1.PUB in one step with COBOLPREP, use defaults for 'uslfile' and 'listfile'. Rename $OLDPASS to XYZ (remember TEMP). We have XYZ, a USL file in the Permanent domain from step 1) and XYZ, a program file in the Temporary domain. Now RUN XYZ. Rename temporary file XYZ to XYZ2. Try to run XYZ again.

??? RUN searches which domain first for progfiles??? _____temp._____

??? In COBOLPREP what is default for uslfile??? _____$NEWPASS_____ *when opened &*

??? In COBOLPREP what is default for progfile??? _____$OLDPASS_____ *after purged*

??? Where is uslfile at completion of COBOLPREP??? _____

5) XYZ2 is your progfile in the Temporary domain. Make it a Permanent file named 'PROG2'. Enter: ':RENAME PROG2,PROG2 / LOCKWORD'. Now RUN PROG2 and see that a lockword has been put on it. Remove the lockword. Both files referenced in the :RENAME command are filereferences so rename PROG2 into your home group thusly:

> ':RENAME PROG2,your-user-name.Gpartner'
> (or use group GTEACHER if 'TEACHER' is your partner)

Do a LISTF and see that PROG2 is no longer in the log-on group. Now do a LISTF @.Gpartner,1 and find it; it has been moved to another group! Rename it back into its original group.

6) Run FCOPY and copy COBTEST1.PUB into your group as 'COBTEST'. RUN LISTDIR2.PUB.SYS. This is a program that will let you see all security settings you are allowed to see. When you receive the '>' prompt, key in 'LISTSEC COBTEST;PASS'. Notice who the CREATOR is! Exit LISTDIR2.

??? Who is the CReating user of COBTEST & PROG2??? _____Partner_____

FILES LAB # 1

OPTIONAL—Proceed only if you have extra time.

7) Now log-on under your user id (i.e., 'HELLO you.INTRO'). Try to rename COBTEST in your group to 'CONFLICT'. You have experienced "CREATOR C O N F L I C T". Try to ':RELEASE' PROG2 and the same thing will happen. Now RUN LISTDIR2 again and LISTSEC for COBTEST as above. It won't show you who the CREATOR is. This information is only given out to the CR, the AM or the SM. Exit LISTDIR2.

??? A file may only be ':RELEASED', ':RENAMED', or ':SECUREd'
         by the ??? _____ *Creator* _____

8) Log-on as your partner or as TEACHER, but use his/her HOME group this time (the default). Run FCOPY and list the contents of COBTEST in your own HOME group on your terminal. Display the tombstone and look-up the error codes in your pocket guide. Press the BREAK key and when you receive the ':' prompt, 'RELEASE' COBTEST in your own HOME group.
         Key in 'RESUME', remember FCOPY's prompt will not be re-issued. Try to list COBTEST from the other group on your terminal again. Success! Now write from your terminal ($STDINX) to COBTEST in the other group (;NEW option not needed since file already exists). Just key in 2 trivial records that strike your fancy, press <CNTL-Y> to shut off entry, and EXIT FCOPY. Use LISTF ,2 to check the current length of COBTEST. PURGE COBTEST from the other group. Look for it with LISTF but you will not find it. Log-off system.

??? When a file is RELEASEd, what restrictions are made as to what users may access it in any way they choose??? _____ *None* _____

<< End >>

PHYSICAL ASPECTS OF FILES

I/O System transfers data between a physical device and buffers a Block at a time under control of the File System.

File System transfers data between buffers & program a logical record at a time.
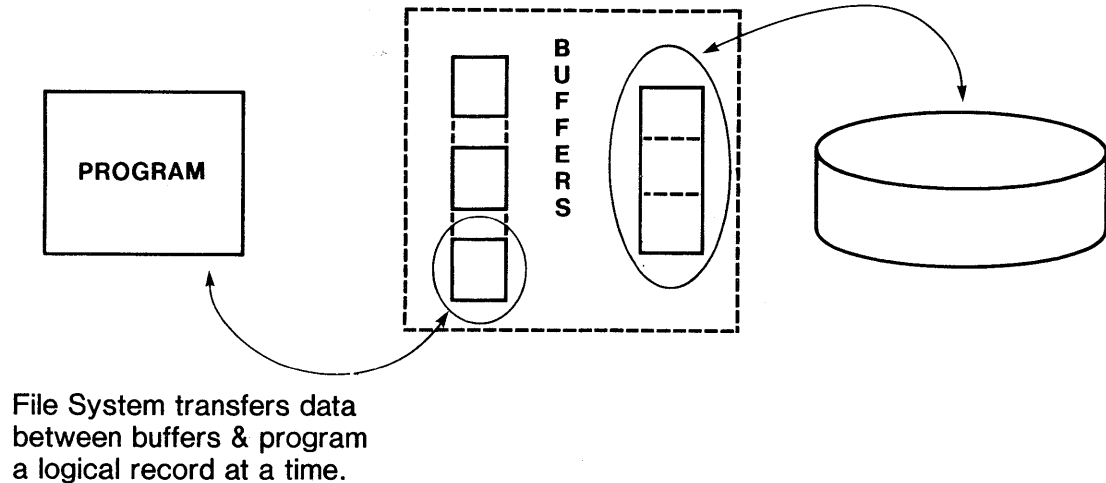
- For Unit Record devices (cards, line printer, etc.) a block is always one record long.

- For Disc and Tape, Blocks may contain more than one Record. Blocking and De-blocking are performed by the File System.

SPECIFYING PHYSICAL ASPECTS OF FILES

```
:FILE ... [;DEV=[device] [,[outpriority] [,numcopies]]]

    [;REC= [recsize] [,[blockfactor] [,[F] [,BINARY]]]] [;NOCCTL]
                                       [U] [,ASCII ]      [;CCTL  ]
                                       [V]

    [;DISC=[numrec] [,[numextents] [,initalloc]]] [;CODE=filecode]

    [;BUF [=numbuffers]]
    [;NOBUF            ]
```

CREATING PERMANENT OR TEMPORARY DISC FILES

```
:BUILD  filereference [;DEV=device]  [;TEMP]

    [;REC= [recsize] [,[blockfactor] [,[F] [,BINARY]]]] [;NOCCTL]
                                       [U] [,ASCII ]      [;CCTL  ]
                                       [V]

    [;DISC=[numrec] [,[numextents] [,initalloc]]] [;CODE=filecode]
```

DEFAULT VALUES

```
 ... ;REC=128,1,F,BINARY;DISC=1023,8,1;CODE=0;NOCCTL;DEV=DISC,8,1
```

HEWLETT hp *PACKARD*

EXAMPLES OF :BUILD AND :FILE COMMANDS

```
:BUILD MYFILE;REC=-80,16,F,ASCII;DISC=10000,10,1;CODE=987
```

A file 'MYFILE' will be built immediately on device=DISC
(Default). It will have 80 byte records, 16 to a block.
It will be able to grow to 10,000 records; space for 1000
is reserved now. The file has a code of '987'.

```
:FILE MYFILE,NEW;REC=-80,16,,ASCII;DISC=10000,10;CODE=987;SAVE
```

Issuing this command merely places an image of it in the
Session/Job table. To actually create a file, a program
must subsequently reference this ':FILE' command:

```
:RUN FCOPY.PUB.SYS
>FROM=$STDINX;TO=*MYFILE
        ... etc.
            <CNTL-Y>
>EXIT
```

DISC FILE CONSIDERATIONS

- Disc physically divided into sectors that are 256 bytes long.
- Each block begins on a disc sector boundary.
- Disc file label occupies 1-st block.
- Disc file may be divided into extents.
- Blocks not ending on a sector boundary WASTE disc space.

Program "BLOCK" from contributed library can help you
choose blocking factors for Fixed files:

```
:RUN BLOCK.PUB
RECORD SIZE ? 40                               length in WORDS
LOWER AND UPPER BLOCK FACTORS ? 1,30
PERCENT UTILIZATION LIST CUT OFF ? 98

BLOCK           BLOCK           LEFT OVER       PERCENT UTILIZATION
FACTOR          SIZE WORDS
 16             640             0               100
 19             760             8               98.9583
 22             880             16              98.2143
MORE ? NO
 END OF PROGRAM
:
```

HEWLETT hp PACKARD

FIXED LENGTH RECORDS (DISC)

CASE I

BLOCK SIZE = RECORD SIZE X BLOCK FACTOR
(BYTES)   (BYTES)

RECORD SIZE   120

BLOCK FACTOR   1

BLOCK SIZE   120   ←—120—→

←——— 256 BYTES ——→
(SECTOR)

CASE II

RECORD SIZE   120

BLOCK FACTOR   2

BLOCK SIZE   240   ←——— 240 ——→

←——— 256 BYTES ——→
(SECTOR)
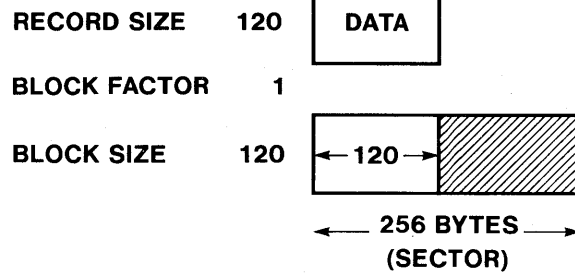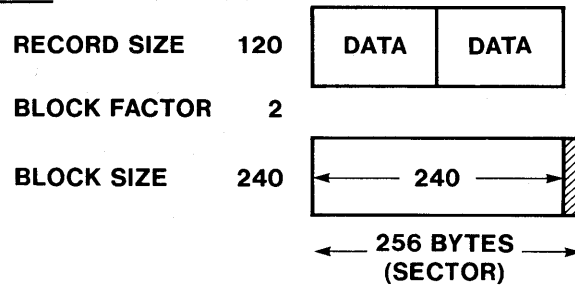
FIXED LENGTH RECORDS (DISC)

CASE III

RECORD SIZE   320
BLOCK FACTOR   1
BLOCK SIZE   320

DATA

| ◄——— 256 ———► | ◄64► |
| 256 BYTES (SECTOR) | 256 BYTES (SECTOR) |
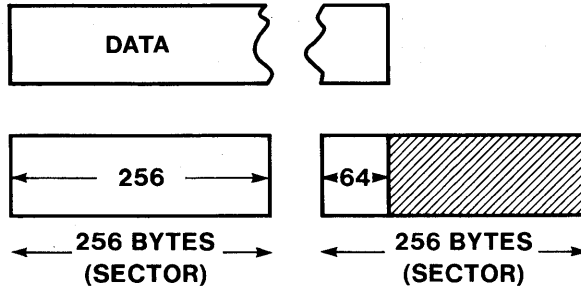
CASE IV

RECORD SIZE   320
BLOCK FACTOR   2
BLOCK SIZE   640

DATA   DATA

| ◄——— 256 ———► | ◄——— 256 ———► | ◄— 128 —► |
| 256 BYTES (SECTOR) | 256 BYTES (SECTOR) | 256 BYTES (SECTOR) |
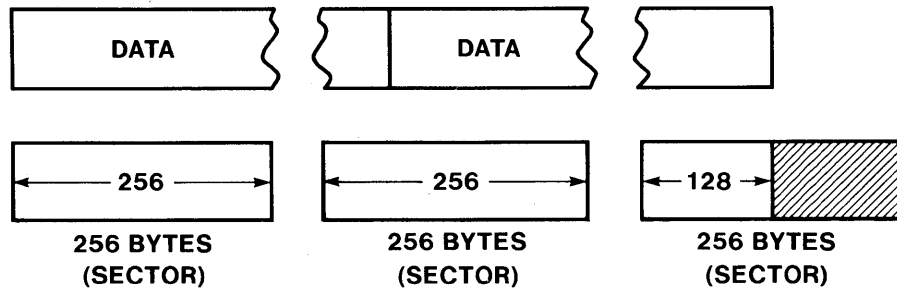
VARIABLE LENGTH RECORDS (DISC)

CASE I

RECORD SIZE   120 BYTES
BLOCK FACTOR   1 BYTES
BLOCK SIZE   124 BYTES

```
┌───┬──────┐
│1  │      │
│0  │ DATA │
│0  │      │
└───┴──────┘
```

|←BLOCK 1→| |←BLOCK 2→|

```
┌──────┬──────┐     ┌─────────────┐
│      │//////│     │/////////////│
└──────┴──────┘     └─────────────┘
```

←—— 256 BYTES——→   ←——256 BYTES——→
   (SECTOR 1)   (SECTOR 2)

BLOCK SIZE = RECORD SIZE X BLOCKFACTOR + 4
   (BYTES)   (BYTES)   (BYTES)
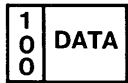
MAXIMUM RECORD THAT CAN BE WRITTEN IS
(BLOCK SIZE – 4)

FOR BEST DISC UTILIZATION BLOCK SIZE
SHOULD BE A MULTIPLE OF 256 (BYTES)
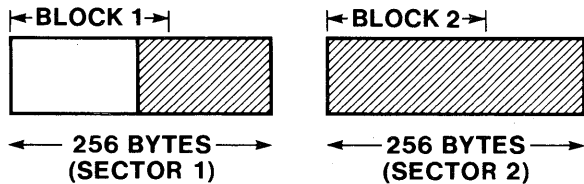
CASE II

RECORD SIZE   320
BLOCK FACTOR   2
BLOCK SIZE   644

```
┌───┬─────────┬───┐ ⎫ ⎧ ┌────────┐
│2  │         │1  │ ⎬ ⎨ │        │
│2  │  DATA   │4  │ ⎭ ⎩ │  DATA  │
│0  │         │0  │     │        │
└───┴─────────┴───┘     └────────┘
```

|←————————— BLOCK 1 —————————→| |←——BLOCK 2——→|

```
┌──────────┐ ┌──────┬──────┐ ┌──────────┐ ┌──────────┐
│          │ │      │//////│ │//////////│ │//////////│
└──────────┘ └──────┴──────┘ └──────────┘ └──────────┘
```

←—256 BYTES—→   ←—256 BYTES—→   ←—256 BYTES—→   ←— 256 BYTES—→
  (SECTOR 1)   (SECTOR 2)   (SECTOR 3)   (SECTOR 4)

HEWLETT hp PACKARD

UNDEFINED RECORDS (DISC)

RECSIZE = 512

SECTOR A	SECTOR B

RECORD

UNINITIALIZED

↑

**CONTENT OF BYTES 255
AND 256 ARE DUPLICATES
OF BYTES 253 AND 254**

RECSIZE = 512

SECTOR A	SECTOR B

RECORD

**CONTENT OF BYTES 257 AND 258
PROPAGATED THROUGH SECTOR B**

BYTES WRITTEN = 258
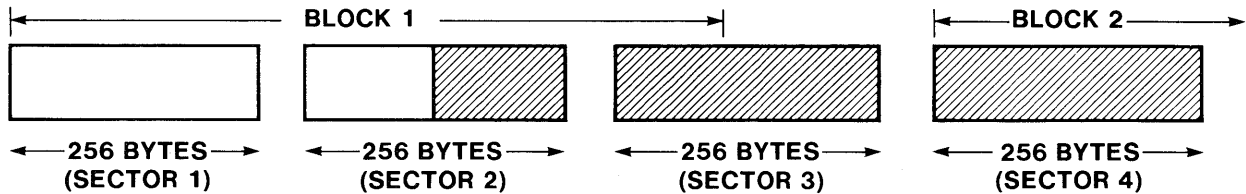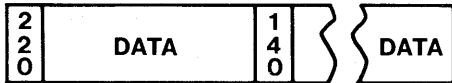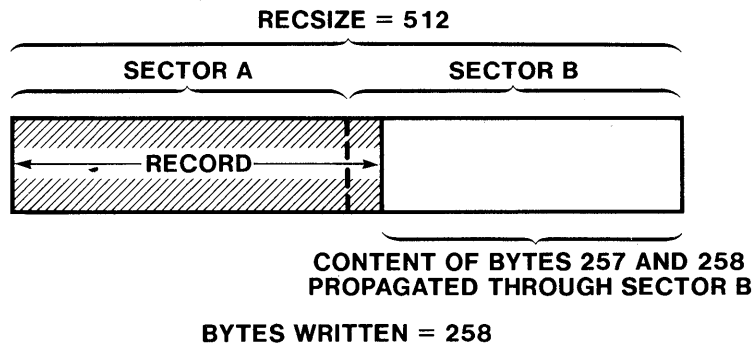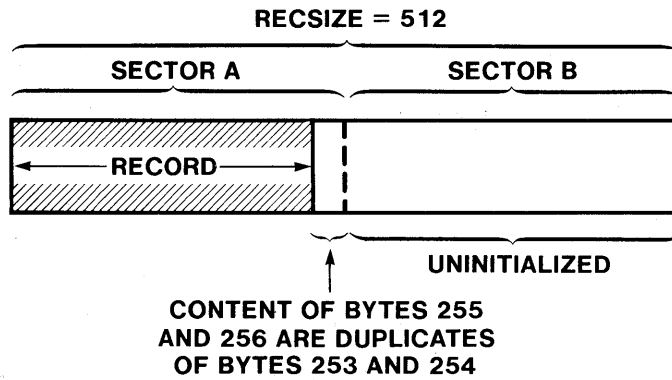
BACK-REFERENCING ANOTHER :FILE COMMAND

```
:FILE   formaldesignator1=*formaldesignator2

        :FILE LISTOUT;DEV=LP,1,3;CCTL
        :FILE FTN06=*LISTOUT
```

SPECIFYING ACCESS TO DATA WITHIN A FILE

```
                    {IN      }
                    {OUT     }
                    {INOUT   }
:FILE   . . .   [;ACC={OUTKEEP}]
                    {APPEND  }
                    {UPDATE  }
```

CONTROLLING SIMULTANEOUS ACCESS TO DISC FILES

```
:FILE   . . .   [;EXC]
                [;SHR]
                [;EAR]
```

HEWLETT **hp** *PACKARD*

ACCESS TO TYPES OF DEVICES

TYPE OF DEVICE					
ACCESS	Input only (card rdr)	Output only (LP)	Input & Output (terminal)	Mag-tape	Disc
IN	X		X	X	X
OUT		X	X	X	X
INOUT			X	X	X
APPEND				X	X
OUTKEEP					X
UPDATE					X

HEWLETT *hp* PACKARD

DISC FILE LABEL

LOCAL FILE NAME
GROUP NAME
ACCOUNT NAME
CREATING USER NAME
LOCKWORD
SECURITY MATRIX
CREATE DATE
LAST ACCESS DATE
LAST MOD DATE
FILE CODE
MAX # LOGICAL REC.
RECORD SIZE (BYTES)
BLOCK SIZE (WORDS)
OF EXTENTS — 1
EXTENT SIZE
OF LOGICAL REC.
USER LABELS
USER LABELS ↓ MAX 255

SECTOR #1 OF FILE

EACH USER
LABEL TAKES
ONE SECTOR

HEWLETT *hp* PACKARD

HIERARCHY OF FILE PARAMETERS

```
            ┌────────────────────────────┐
            │      DISC FILE LABEL        │
            │    (OLD DISC FILE ONLY)     │
            │           -OR-              │
            │   DEVICE CHARACTERISTICS    │
            └────────────────────────────┘
OVERRIDES                   │
                            ▼
            ┌────────────────────────────┐
            │          :FILE             │
            │    COMMAND PARAMETERS       │
            └────────────────────────────┘
OVERRIDES                   │
                            ▼
            ┌────────────────────────────┐
            │      OPEN PARAMETERS        │
            └────────────────────────────┘
OVERRIDES                   │
                            ▼
            ┌────────────────────────────┐
            │      SYSTEM DEFAULTS        │
            └────────────────────────────┘
```
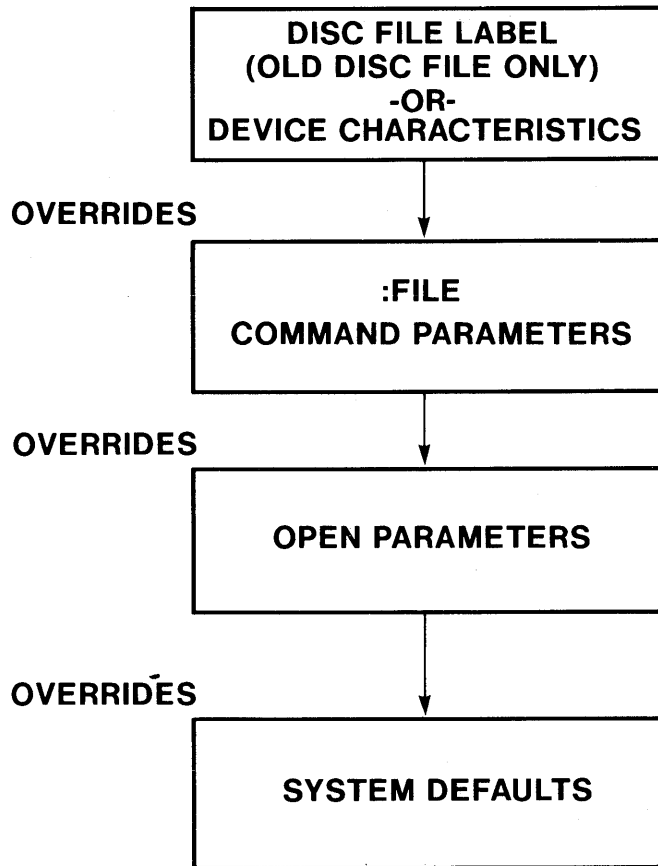
ASSUMPTIONS ABOUT FILES

COBOL & RPG—Try to open an OLD file by that name. If none
        exists, they will open file as NEW.

        :FILE NEWOUT;SAVE   *will save file*

FORTRAN—Opens file as NEW (whether it exists or not!).

        :FILE FTN09=NEWOUT;SAVE   *1-st time*

        :FILE FTN09=NEWOUT,OLD   *Subsequently*

BASIC—Assumes file is OLD

        :FILE NEWOUT,NEW;SAVE   *1-st time*

        :RESET NEWOUT   *Subsequently*

HEWLETT hp PACKARD

BREAKABLE MPE COMMANDS

SUSPENDED

:APL
:BASIC
:BASICGO
:BASICOMP
:BASICPREP
:COBOL
:COBOLGO
:COBOLPREP
:EDITOR
:FORTGO
:FORTPREP
:FORTRAN

:HELP
:PREP
:RJE
:RPG
:PRGGO
:RPGPREP
:RUN
:SEGMENTER
:SPL
:SPLGO
:SPLPREP

ABORTED

:( ) command log on
:BYE
:HELLO
:LISTF
:LISTVS
:REDO
:REMOTE HELLO
:REPORT
:RESTORE
:SHOWDEV
:SHOWIN
:SHOWJCW
:SHOWJOB
:SHOWME
:SHOWOUT
:STORE
:STREAM

ABORT first
before starting
new program.

NON-BREAKABLE COMMANDS

:ABORT	:MOUNT
:ALTSEC	:PTAPE
:BUILD	:PURGE
:COMMENT	:RELEASE
:CONTINUE	:RENAME
:DATA	:RESET
:DEBUG	:RESETDUMP
:DISMOUNT	:RESUME
:DSLINE	:SAVE
:DSTAT	:SECURE
:EOD	:SETCATALOG
:EOF	:SETDUMP
:EOJ	:SETJCW
:FILE	:SETMSG
:FREERIN	:SHOWTIME
:GETRIN	:SPEED
:IF	:TELL
:JOB	:TELLOP
	:VSUSER

MPE INTRINSICS

- System routines are called 'intrinsics'.

- They reside in SL.PUB.SYS and may be called by Users to
      perform system functions programmatically.

- Written in SPL. Documented in "Intrinsics" manual.

- May be called directly from SPL or FORTRAN programs.

READY REFERENCE

READ THIS BEFORE DOING LABS

- RECORDS—Always begin and end on word boundaries. (odd byte length records padded out to a word boundary)

- RECORD FORMATS

  FIXED
  — All records a fixed length; blocks contain a fixed no. of records.
  — Record Length & Blocking Factor known to File system.
  — Records consist of data only.

  VARIABLE
  — Record length varies; blocks contain a variable no. of records.
  — File attribute defined to File system is Maximum Record Length = Record Length X Blocking Factor (this is the largest data record file can accommodate).
  — Block length is Maximum Record Length plus 2 words.
  — Each record consists of data plus a field containing length of that record in bytes (1 word long).
  — Each block contains an end-of-block indicator (1 word long).

  UNDEFINED
  — Block Length set to Record Length defined to File System.
  — Blocking factor assumed to be 1.
  — Blocking and de-blocking of records is the user's responsibility.
  — Records shorter than Block Length have their last word of data repeated to end of block.

- BUFFERING
  ```
  :FILE    ... [{;BUF[=numbuffers]}]
               [{;NOBUF          }]
  ```

  — Block Length is Buffer Size.
  — Default is two buffers.
  — May be overridden at run time with :FILE command.
  — NOBUF specifies no buffers allocated for this file. Blocks transferred directly into users STACK. File system performs no blocking/de-blocking.

READY REFERENCE

PARAMETERS COMMON TO THE ':FILE' AND ':BUILD' COMMANDS.

```
                       [F]
[;REC=[,recsize][,[blockfactor][,[U][,BINARY]]]]
                       [V][,ASCII ]
```

      recsize— + for words; - for bytes.
      BINARY—pad longer records or new disc extents with binary zeroes (%0).
      ASCII—pad longer records or new disc extents with spaces (%40).

```
[;DISC=[numrec][,[numextents][,initalloc]]]
```

      numrec—max no. of records to allow in file. Default=1023.
      numextents—1 thru 32 (16 max for MPE-C). default is 8.
      initalloc—no. of extents initially allocated. Default=1.

```
[;CODE]
```
      — User codes 0 thru 1023
      — Minus numbers indicate accessed only by Privileged Mode.
      — 1024+ or mnemonic are System defined:

1024	USL	a USL file
1025	BASD	a BASIC data file.
1026	BASP	a BASIC program file.
1027	BASFP	a BASIC fast program file.
1028	RL	a Relocatable Library.
1029	PROG	a Program file.
1031	SL	a Segmented Library.
1040	XLSAV	Cross-loader ASCII file (SAVE).
1041	XLBIN	Cross-loader relocatable BINARY file.
1042	XLDSP	Cross-loader ASCII file (DISPLAY).
1050	EDITQ	EDIT non-COBOL KEEPQ.
1051	EDTCQ	EDIT COBOL KEEPQ
1052	EDTCT	EDIT COBOL TEXT file.
1060	RJEPN	RJE punch file.
1070	QPROC	a QUERY procedure file.
1071		QUERY work file.
1072		QUERY work file.
1080	KSAMK	a KSAM key file.

READY REFERENCE

[;CCTL]
[;NOCCTL]

CCTL—an additional character is added to the beginning of each record containing carriage control information, in addition to record length. Valid for ASCII files only.

NOCCTL—no additional char reserved for carriage control. (Default=NOCCTL).

[;TEMP]   *Applies to :BUILD only*

:BUILD command can only create a file in the Permanent or Temporary domain. Default is Permanent.

REFERENCING DISC FILE DOMAINS

```
            [,NEW     ]    [;DEL ]
:FILE ...  [,OLD     ]    [;SAVE]
            [,OLDTEMP]    [;TEMP]
```

NEW—create a disc file in the NEW domain.
OLD—find a disc file that already exists in the OLD (PERMANENT) domain.
OLDTEMP—find a disc file that already exists in the TEMPORARY domain
Default—search TEMPORARY domain then PERMANENT domain.

DEL—delete file upon close.
SAVE—move this file to PERMANENT domain upon close.
TEMP—make this NEW file TEMPORARY upon close.
Default—upon close file assumes same disposition as at open.

:FILE BACK-REFERENCE—FORM 2

`:FILE formaldesignator1=*formaldesignator2`

Here 'formaldesignator1' takes on all the same attributes as
'formaldesignator2' from a previous ':FILE' command.

CONTROLLING SIMULTANEOUS ACCESS TO DISC FILES

```
            [;EXC]
:FILE ...  [;EAR]
            [;SHR]
```

EXC—Exclusive access; no other users will be allowed to
　　　　access this file while you have it open. You will not
　　　　be allowed EXC access if someone is already using the
　　　　file.
EAR—Exclusive Allowing Read; other users may open file
　　　　but only for read-only access (ACC=IN). You will only
　　　　be granted this access if no one else is using file or
　　　　it is opened for read-only access. (see discussion on
　　　　"Simultaneous Access of Files" in Section 6 of the MPE
　　　　commands manual for a detailed discussion).
SHR—Shared access. Allow concurrent use by other users.
　　　　You will not be granted access to file if someone has
　　　　it opened with EXC access.

SPECIFYING ACCESS

```
                {IN      }
                {OUT     }
                {UPDATE  }
:FILE ...  [;ACC={OUTKEEP}]
                {APPEND  }
                {INOUT   }
```

IN—read-only.
OUT—write-only. Original contents of file overlayed.
        File cannot be read!
INOUT—any operation but update allowed. (you can still
        read then write the same record).
OUTKEEP—write-only access. Original contents kept and
        you are allowed to write both before and after end-of-
        file. File cannot be read!
APPEND—records may only be written beyond end-of-file.
        File cannot be read nor can you write into original
        extent.
UPDATE—update access; all operations may be performed
        on file.
Default—IN access for devices that can perform input;
        otherwise OUT for output-only devices.

SPECIALIZED PARAMETERS OF :FILE NOT COVERED IN INTRO

MULTI—requires Process Handling capability.

MR—disregards record boundaries.

NOWAIT—requires Privileged Mode capability.

FILES LAB #2 [0.5 hour]

OBSERVING CHARACTERISTICS OF F, V, AND U FORMAT FILES.

1) Log-on to the system and enter the following file commands:

```
:FILE FIXED,NEW    ;REC=-128,2,F,ASCII;DISC=4,1,1;SAVE
:FILE VARIABLE,NEW;REC=-128,2,V,ASCII;DISC=4,1,1;SAVE
:FILE UNDEFINE,NEW;REC=-128,2,U,ASCII;DISC=4,1,1;SAVE
```

2) Double-check to make sure you have entered them correctly by running LISTEQ2.

3) Now use FCOPY and back-references to create each of these files with 3 records in each one. Following this scenario exactly:

```
>FROM=$STDINX;TO=*FIXED
*200*J
WARNING: FROMFILE RECSIZE IS 80 BYTES, TOFILE RECSIZE IS 128
BYTES.
CONTINUE OPERATION (Y OR N) ?Y
RECORD 1 -- FIXED FILE
RECORD 2 -- FIXED FILE
END-OF-FILE FIXED
 < CONTROL Y >          displayed when CNTL-Y pressed

4 RECORDS PROCESSED *** 0 ERRORS

>FROM=$STDINX;TO=*VARIABLE
*200*
RECORD 1 -- VARIABLE FILE
RECORD 2 -- VARIABLE FILE
END-OF-FILE VARIABLE
 < CONTROL Y >

4 RECORDS PROCESSED *** 0 ERRORS

>FROM=$STDINX;TO=*UNDEFINE
*200*
RECORD 1 -- UNDEFINED FILE
RECORD 2 -- UNDEFINED FILE
END-OF-FILE UNDEFINED
 < CONTROL Y >

4 RECORDS PROCESSED *** 0 ERRORS
```

FILES LAB #2

4) Now list the contents of each file on your terminal. Expect warning '*200*' and merely press <RETURN> to proceed. You should see that the last word of undefined records is indeed propogated through the remainder of the record.

5) Exit FCOPY and do a 'LISTF ,2' for each file individually. Observe what record length has been set for file VARIABLE in its disc file label.

6) Fill in the following table from information obtained from the LISTF displays:

For file:	Record Length (in bytes)	Blocking Factor	Block Length (& Buffer len) (in bytes)
FIXED	128	2	256
VARIABLE	256	1	256
UNDEFINE	128	1	128

(go on to the next page)

FILES LAB #3 [0.5 hour]

1) Use the EDITOR to create a file. Enter three records to your liking and keep it as a file called 'TEMP' with default of numbered. Exit the Editor.

2) Now :BUILD a temporary file called 'TEMP' with 80 byte records blocked 16, fixed and ASCII and DISC=100.

3) Enter the following :FILE command:

```
:FILE TEMP,NEW;REC=-80,16,F,ASCII;DISC=100;SAVE
```

4) Run FCOPY and copy from $STDINX to '*TEMP'. Do NOT specify this as a ;NEW file, as that has already been done in the :FILE command we are using. Enter several records that meet your high standards and signal the end of your file by pressing <CNTL-Y>. You will get an error number. Enter a printing character to get a tombstone and use your pocket guide to interpret the error. Exit FCOPY.

5) Undeterred, enter the following :FILE command and use LISTEQ2 to make sure you do it right:

```
':FILE TEMP,NEW;REC=-80,16,F,ASCII;DISC=100;TEMP'
```

6) Do exactly the same process as in step 4, even down to the interpretation of the error. Exit FCOPY.

7) Now attempt to :SAVE 'TEMP'. Interpret the error.

8) List file 'TEMP' on your terminal with FCOPY. Exit FCOPY. From which file domain did it come by default?
_____?

9) Log-off then log-on the system. What happens to Temporary files at the end of a Job or Session? Verify your hypothesis with LISTEQ2.

10) Now list file 'TEMP' on your terminal with FCOPY. Which file domain did this one come from?
_____?

Exit FCOPY and log-off the system. So... we have just seen that by default files are searched for in the Temporary domain first, if not there then the Permanent domain is searched.

So... we have seen that files of the same name can exist simultaneously in each of the three domains. This works just fine until we try to move one into another domain. Then we lose the latest file we have been working on! This is a lesson to keep in mind. It means, if you are creating a NEW output file from a 3 hour job and only make it permanent upon close, make sure a duplicate permanent file does not exist at the beginning of those 3 hours.

<< End >>

HEWLETT hp PACKARD

ENTRY POINTS INTO MPE

INTERACTIVE SESSION

```
:HELLO [sessionname,]username[/userpasw].acctname[/acctpasw]

    [,groupname[/grouppasw]][;TERM=termtype][;TIME=cpusecs]

    [;HIPRI                     ]        {BS}
    [;INPRI=inputpriority][;PRI={CS}]    {DS}
                                {DS}
                                {ES}

:BYE
```

default

Limit _____   Jobs, Sessions

Job fence (0-13)

PRI — queuing system; dispatcher + timeslice

A =
100   linear queue
B5

C5
160
DS
ES   255
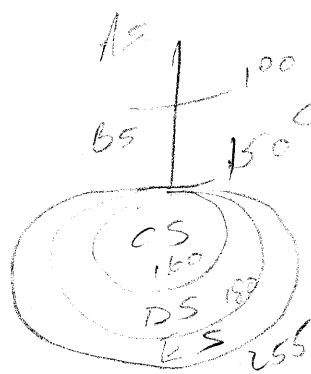
DS — default . Batch

ENTRY POINTS INTO MPE

BATCH JOB

```
:JOB [jobname,]username[/userpasw].acctname[/acctpasw]

    [,groupname[/grouppasw]][;TERM=termtype][;TIME=cpusecs]

                                {BS}
    [;HIPRI                   ] {CS}
    [;INPRI=inputpriority][PRI={DS}][;RESTART]
                                {ES}

    [;OUTCLASS=[device][,[outputpriority][,numcopies]]]
```

`:EOJ` — Bye

`:CONTINUE` — if next command has error, continue to next command

`:COMMENT text`

`:EOD`

`:EOF`

Restart — if interrupted, then warmstart, job will restart from beginning.

outclass — must have higher # than source.

ENTRY POINTS INTO MPE

ENTERING DATA INTO THE SYSTEM

`:DATA [sjname,]username[/userpasw].acctname[/acctpasw]`

`[;filename]`

`:EOD`

	$STDIN[X]	$STDLIST
SESSION	terminal	terminal
JOB	card-reader	line printer
STREAMed JOB	stream file	line printer

HEWLETT hp PACKARD

WHY SPOOL?

- Users can establish Priorities for I/O.

- Priority may be changed dynamically by Console Operator.

- Reduces contention for a device.

- A SPOOLed Device is more efficiently utilized.

- Console Operator initiates/terminates SPOOLing on a device transparent to users.

- More than 1 device of same class can share load. (i.e. 2-nd line printer can be utilized automatically).

- Recover from paper jams without re-running program.

- Handles Special Forms in conjunction with FOPEN parameter. (features provided in COBOL and RPG).

DEVICE FILES

SPOOLED
Immediately accessible by all users.

NOT SPOOLED
Accessed by one Session / Job at a time.

REAL DEVICE FILES
:HELLO / :JOB / :DATA ACCEPTING
- Terminal
- Card Reader
- Card Reader / Punch
- Mag-tape
- Paper Tape Reader

OPERATOR ASSIGNED
- Mag-tape
- Card Reader
- Card Reader / Punch
- Plotter

OUTPUT ONLY
- Line Printer
- Card Reader / Punch
- Paper Tape Punch

SPOOLED DEVICE FILES
:JOB / :DATA ACCEPTING
- Card Reader
- Card Reader / Punch
- Mag-tape
- Paper Tape Reader

OUTPUT ONLY
- Line Printer
- Card Reader / Punch
- Plotter (Not over terminal interface)
- Paper Tape Punch

DEVICEFILE ACCESS

(REAL DEVICEFILES)

REAL INPUT DEVICE

INPUT SPOOLEE

INPUT SPOOLER

REAL

SPOOLED

I/O SYSTEM

READ

INPUT

SPOOLING

USER FILE

USER :STREAM

SPOOLING DISC

(INPUT SPOOLFILES)

JOB/SESSION

(REAL DEVICEFILES)

OUTPUT REAL DEVICE

REAL

I/O SYSTEM

WRITE

SPOOLED

OUTPUT SPOOLEE

SPOOLING

OUTPUT SPOOLER

OUTPUT

SPOOLING DISC
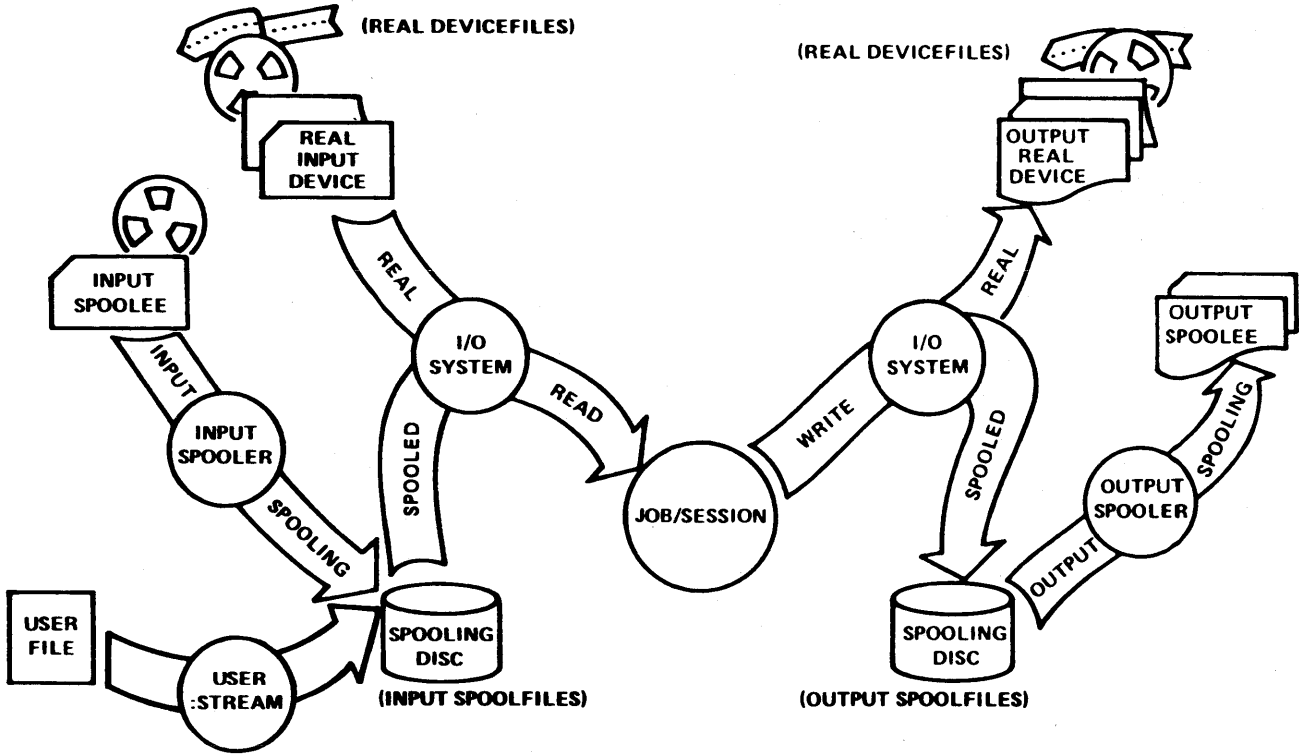
(OUTPUT SPOOLFILES)
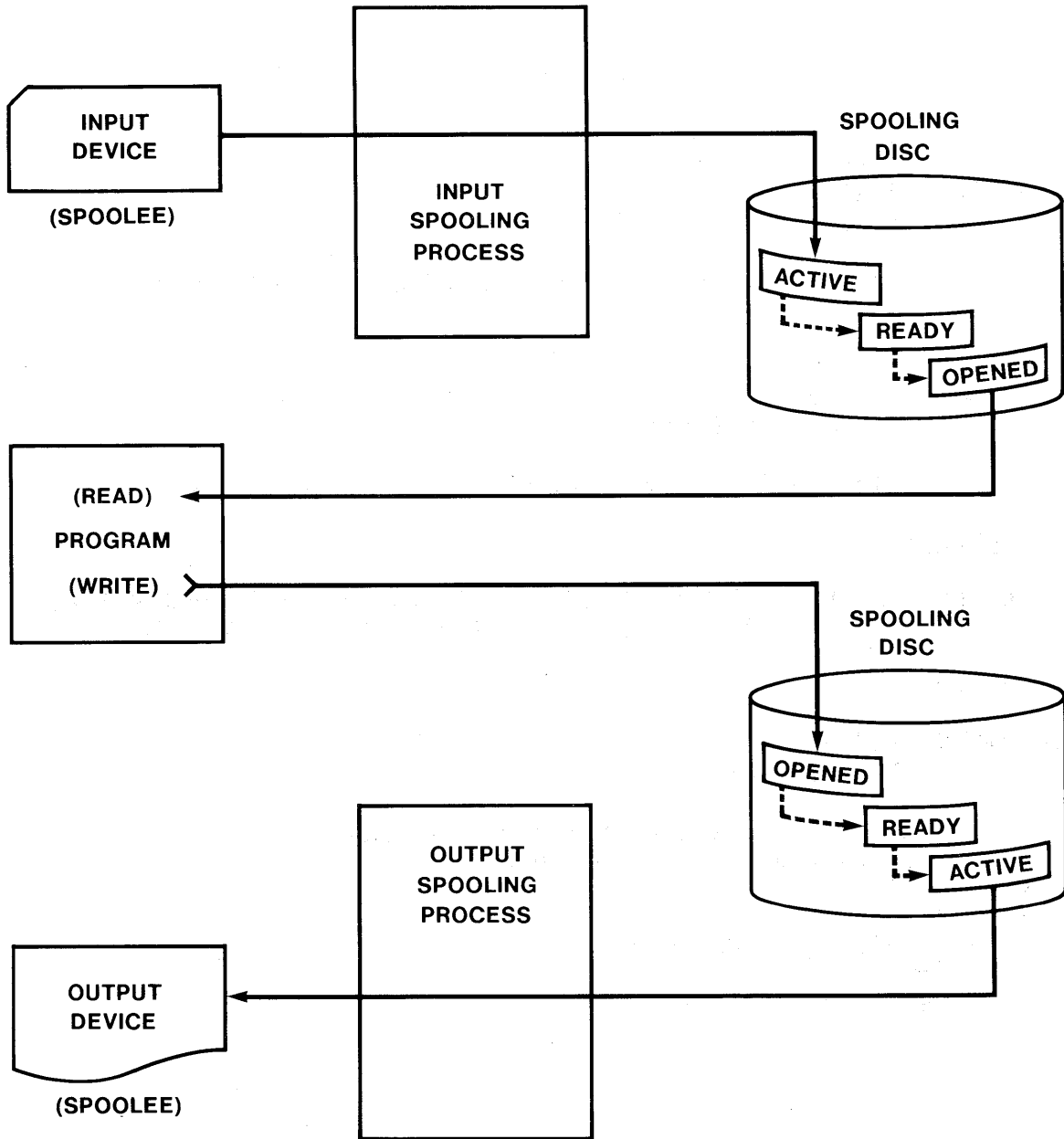
SPOOLFILE STATES

MPE COMMANDS FOR THE SPOOLER

```
            [SP                    ]
            [#Innn                 ]
:SHOWIN     [STATUS                ]
            [item[;item[;item]]]
```

List $STDIN[X] for your Session / Job and system input Spool files.

```
            [SP                    ]
            [#Onnn                 ]
:SHOWOUT    [STATUS                ]
            [item[;item[;item]]]
```

List $STDLIST for your Session / Job and system output Spool files.

:<u>SHOWOUT SP</u>

```
DEV/CL    DFID      JOBNUM    FNAME      STATE FRM SPACE RANK PRI #C
LP        #064      #S54      LP         READY            8   D 1  1

OUTFENCE = 2
```

```
:SHOWDEV   [ldev      ]
           [classname]
```

List status of one device or all devices in a class.
Default is all devices in all classes.

```
:SHOWDEV 6
LDEV  AVAIL     OWNERSHIP      VOLID

  6   SPOOLED   SPOOLER OUT
```

SOME CONSOLE OPERATOR COMMANDS

```
          [SP                    ]
          [#Innn                 ]
=SHOWIN   [STATUS                ]
          [item[;item[;item]]]
```

List status of $STDIN[X] for all Sessions and Jobs and all input Spool files in system.

```
          [SP                    ]
          [#Onnn                 ]
=SHOWOUT  [STATUS                ]
          [item[;item[;item]]]
```

List $STDLIST for all Sessions and Jobs and status of all output Spool files in system.

=SESSION

Allows a Session to be run on the System Console.

HEWLETT *hp* PACKARD

SOME CONSOLE OPERATOR COMMANDS

=RECALL

>List all pending I/O requests that require Console Operator intervention.

```
              {YES }
=REPLY  pin,{NO  }
              {ldev}
```

>Allocate an operator assigned device to a process or refuse request.

```
=RECALL
REPLY(S) PENDING:
?11:13/#S54/23/LDEV# FOR "STUDENT" ON TAPE (NUM)?
?11:13/#S59/26/IS "TEACHER" ON LDEV#7(Y/N)?
=REPLY 23,8
=REPLY 26,Y
```

SOME CONSOLE OPERATOR COMMANDS

=SPOOL ldev {,'etc.'}   *Refer to Pocket Guide*

      Control SPOOLer for a particular device.

=LIMIT [numberjobs][,numbersessions]

      Dynamically set limit of Jobs or Sessions that may execute
      on the system at any one time.

=TELL {'someone'};message

      Send someone or everyone a message. Those with SETMSG OFF
      (QUIET) will NOT receive it.

=WARN {'someone'};message

      Send someone or everyone a message. Overrides SETMSG OFF.

STREAM

To initiate a Job INDEPENDENT of the originating Session or Job.

Does NOT 'LINK' to a subordinate JOB; control CANNOT be returned to the creating Session or Job.

```
:STREAM [inputfile][,character]
```

- Default 'character' for ':' prompt replacement is '!'.

- MPE-C stream files MUST be unnumbered. MPE-III—either numbered or unnumbered stream files.

- '=STREAMS' enables streaming.

HEWLETT **hp** PACKARD

STREAM FROM A SESSION

```
:EDITOR

HP32201A.7.00 EDIT/3000   MON, FEB 20, 1978,  4:51 PM
(C) HEWLETT-PACKARD CO. 1976
/A
    1       !JOB BACKUP,TEACHER.INTRO/PASSWORD
    2       !COMMENT DAILY BACKUP OF INTRO ACCOUNT
    3       !TELLOP MOUNT INTRO BACKUP TAPE FOR 'TEACHER'
    4       !FILE TEACHER;DEV=TAPE
    5       !STORE @.PUB.INTRO,@.GTEACHER.INTRO;*TEACHER;SHOW
    6       !EOJ
    7       ...
/K DAILYJOB,UNN
/E
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y

 END OF SUBSYSTEM
:STREAM DAILYJOB                        STREAM an Editor file
 #J27
:STREAM                                 Create STREAM on-line
>!JOB TEACHER.INTRO/PASSWORD
>!COBOL MYNEWPRG
>!PREP $OLDPASS,NEWPRG
>!SAVE NEWPRG
>!EOJ
 #J28
>:EOD       — to get out of stream
:
```

STREAM FROM A JOB

```
:JOB CLEANUP,MGR/SECRET.INTRO/PASSWORD
:COMMENT -- CLEANUP INTRO ACCOUNT.
      . . .
:COMMENT -- BACKUP WHOLE ACCOUNT.
:STREAM DAILYJOB
                                        #Jnn on $STDLIST listing
:EOJ
```

-OR-

```
:JOB DAILYRUN,MGR.PRODUCTN/PASSWORD
:RUN EDITPROG
:COMMENT  CREATE A PARALLEL JOB TO PRINT ERROR REPORT.
:STREAM
!JOB ERRORS,MGR.PRODUCTN/PASSWORD
!FILE LP=$STDLIST
!RUN ERRORRPT
!EOJ
                                        #Jnn on $STDLIST listing
:EOD
:COMMENT  RUN REPORT OF VALID TRANSACTIONS.
:FILE LP;DEV=LP,,3
:RUN TRANSRPT
:EOJ
```

HEWLETT **hp** PACKARD

STORE / RESTORE CAPABILITIES

- Allows Back-up Storage of DISC files on MAG-TAPE.
- STORE writes tape labels; checked by RESTORE (not ANSI standard).
- STORE writes a complete directory of all files being stored
    at the beginning of each tape volume.

MPE III only

FOR THE USER WITH STANDARD CAPABILITIES

- With :STORE you can back-up any Permanent Disc file to which you have READ access. You must supply any LOCKWORDs in effect. Files opened with other than READ-ONLY access will be bypassed.
- With :RESTORE you can Restore:
    —Any File in your Log-on Account into a Group in which you have SAVE access.
    —Individual Files from a 'SYSDUMP' tape.
    —Files must go back into the same Group and Account from which each was :STORED and User
        who created file must exist within that Account.
    —LOCKWORDs must be provided.

:STORE

:STORE [fileset] [, ...] ;storefile [;SHOW] [;FILES=maxfiles]

where:

'fileset'   is   { file[.group[.account]] }
             { @ [.group[.account]] }
             { @[. @ [.account]] }
             { @[. @ [. @]] }

default is @.

GENERIC Specification may be used for
any File, Group, or Account Name.   *MPE III only*

'storefile'   is back-reference to a :FILE command for:
                           1) Mag-tape.
                           2) Serial Disc   *MPE III only*

'SHOW'   means list results of operation for all files named
                           in STORE command.

'maxfiles'   is Maximum number of files that may be STORED.
                           Default is 4000.

:STORE EXAMPLE

```
:HELLO STUDENT.INTRO
HP3000 III.  TUE, MAR  8, 1978,  6:33 PM
:FILE STUDENT;DEV=TAPE
:STORE @,FCOPY.PUB.SYS,EMPLOY01.PUB,DEFTABS.PUB;*STUDENT;SHOW
LOCKWORD: LOCKED1.GSTUDENT.INTRO?
```
                                        MPE III turns echo off

```
TUE, MAR  8, 1978,  6:34 PM

  FILES STORED = 4

     FILE     .GROUP    .ACCOUNT    LDN   ADDRESS   VOLUME

     BRUTUS39.GSTUDENT.INTRO        4     %1475       1
     LOCKED1 .GSTUDENT.INTRO        5     %14562      1
     FCOPY   .PUB      .SYS         4     %364766     1
     DEFTABS .PUB      .INTRO       4     %54501      1


  FILES NOT STORED = 2

     FILE     .GROUP    .ACCOUNT FILESET       REASON

     OPENFILE.GSTUDENT.INTRO        1      BUSY
     EMPLOY01.PUB      .INTRO       3      FILE CODE<0 AND NO PRIV MODE


:BYE

CPU=2. CONNECT=3. TUE, MAR  8, 1978,  6:35 PM
```

:RESTORE

:RESTORE restorefile [; [fileset] [, ...] [;KEEP] [;DEV=device]

[;SHOW] [;FILES=maxfiles]]

where:

'restorefile'	is	back-reference to a :FILE command for: 1) Mag-tape. 2) Serial Disc *MPE III only*
'fileset'		is identical to the one for :STORE but Default is @.@.@.
'KEEP'		means do NOT restore existing files; keep disc file.
'device'	is	Any specification for a Disc device. Default is 'DISC'. May specify PRIVATE VOLUMES *MPE III only*
'SHOW'		means produce a detailed listing of results for all files named in :RESTORE command.
'maxfiles'	is	Maximum number of files to be restored. Default is 4000.

:RESTORE EXAMPLE

```
:HELLO STUDENT.INTRO
HP3000 III.  TUE, MAR  8, 1978,  6:36 PM
:FILE STUDENT;DEV=TAPE
:PURGE BRUTUS39
:RESTORE *STUDENT;;SHOW;KEEP
LOCKWORD: LOCKED1.GSTUDENT.INTRO?
```
                                        MPE III turns echo off

```
TUE, MAR  8, 1978,  6:37 PM

  FILES RESTORED = 1

    FILE     .GROUP    .ACCOUNT   LDN  ADDRESS

    BRUTUS39.GSTUDENT.INTRO        3    %7722


  FILES NOT RESTORED = 2

    FILE    .GROUP    .ACCOUNT FILESET      REASON

    LOCKED1 .GSTUDENT.INTRO       1     ALREADY EXISTS
    DEFTABS .PUB     .INTRO       1     ALREADY EXISTS


:BYE

CPU=3. CONNECT=4. TUE, MAR  8, 1978,  6:39 PM
```

JOB STREAM LAB # 1 [0.5 hour]

Please read the entire lab before proceeding!

A). Write out below, then create a Job Stream file with the Editor to compile, prep and execute the COBOL source file 'LABJOB1.PUB'. Use three separate MPE commands; do not use COBOLGO. Use $NEWPASS and $OLDPASS for your USL and program files where applicable. Keep the Job Stream file then Stream it. Remember, STREAM'ing will initiate a job independent of the current Job or Session.

Key Points:	Fill In:
What commands delimit a job?	JOB EOJ
What character is the STREAM command expecting?	!
Where will your compilation listing and any program output appear and why?	LP ($STDLIST)
Are there any special considerations when Keeping STREAM files with the Editor (Pre-MPE III)?	UNN
What command can you use to find out the status of your job created by STREAM?	SHOW JOB

B). Create the same Job Stream from your Session without using the Editor.

JOB STREAM LAB #2 [0.1 hour]

As a class exercise we will construct one Job Stream to compile, prep, :STORE and :RESTORE files in the INTRO Account. List any ideas you would like included below, then if time permits, continue with JOB STREAM LAB #3.

JOB STREAM LAB #3 [0.5 hour]

Proceed with this lab only if you have extra time.

Please read the entire lab before proceeding!
Part I—STREAM'ing DATA

1). We are going to submit a source disc file to the COBOL compiler as if it were a deck of cards read through the card reader. If it were a deck of cards, we would have to preface it with a :DATA card containing our User and Account names plus any associated passwords to enable MPE to identify this deck as ours and pass it to our Job or Session. That card deck would be read through the card reader by the input spooler into an input spoolfile where it would remain in the 'Ready' state until referenced by our session.

Instead, we are going to STREAM a disc file containing an image of this card deck with a !DATA command on the front and a !EOD command on the back. This !DATA command must contain the User and Account names of the session that will reference the data exactly like a :DATA card. This !DATA command must contain all associated passwords to be accepted by the system.

The COBOL source deck we are going to submit is in the file 'LABJOB3.PUB'; it is unnumbered. Modify the !DATA command to match your session's parameters. Refer to the syntax for the :DATA command in your pocket guide (do NOT use a 'session-name' nor a 'file-name' in your !DATA statement; they would unneccessarily complicate things). Keep the file unnumbered as 'LABJOB3' in your group.

Exit the Editor and STREAM LABJOB3. '#Innn' should be displayed on your terminal; if not, seek aid from your instructor. Issue a :SHOWIN command and you will see an input spool-file in the 'READY' state for your User and Account that contains your COBOL source deck. Notice which device it appears to have been read from (either 5 or 10).

2). We must now reference this input spool-file with a :FILE command. If we had read a card deck through an unspooled card reader, we would have referenced it with the command:
        ':FILE xyz;DEV=CARD'
This would have given us exclusive access to the card reader to read in our deck.

JOB STREAM LAB #3

If we had read our card deck through a spooled card reader, as soon as we placed it in the card reader, the input Spooler would have read it into an input spool-file where it would remain until referenced via this same :FILE command above.

We have submitted a !DATA disc file to the input Spooler with the :STREAM command, but the net effect has been exactly the same as reading a card deck through a spooled card reader. We can reference it with a similar :FILE command. The device a STREAM'd file thinks it was read from depends on how the '=STREAMS' command has been issued from the Operator's Console. Now issue a :FILE command referencing the device number associated with your input spool-file from the :SHOWIN display.

3). Use :COBOLPREP to both compile and prepare your program with one command. Back-reference the file name you used in your :FILE command in step 2 as the 'textfile' and put the resulting program in file 'PGM3'.

4). Change the name of 'PGM3' to 'PROG'. If you get an error, chances are you have forgotten in which domain :PREP places its program files. Now :SAVE 'PROG' as a permanent file.

5). Issue the command to reset all active :FILE commands for your session. Run 'LISTEQ2.PUB.SYS' to make sure no :FILE commands remain active.

6). Using the Editor, create a Job Stream file to:
        a) :STORE all files in your group to mag-tape.
        b) :RESTORE files 'PROG' and 'LABJOB3' from mag-tape with the KEEP option specified.
        c) Obtain a list of all files within your group on the line printer with a detail option of ',1'.

REMEMBER:
- You must supply your Username, Acctname, and Account Password on the JOB command.
- You must KEEP the Stream file unnumbered if operating under pre-MPE III versions.
- :STORE and :RESTORE must back-reference a :FILE command for the tape drive. Use your User name for the file name so you can easily recognize which request is your's on the SYSTEM CONSOLE.
- $STDLIST within a JOB will automatically be assigned to the line printer.

JOB STREAM LAB #3

7). :STREAM your Job Stream file. If you have constructed it correctly, a '#Jnnn' number will be displayed on your terminal. You now have both a JOB and a SESSION running concurrently logged-on under your USER.ACCT. Issue a ':SHOWJOB JOB=@,user.INTRO' to display them both. If your JOB is in the EXEC state and a mag-tape is available, do 'PART II—Using the SYSTEM CONSOLE' now. Otherwise continue with the next step and do PART II later.

8). From within your session, obtain a list of all files within your group on the line printer with a detail option of ',2'.

9). Now make a 'LISTEQ2' listing on the line printer by issuing the following commands:
                    :FILE LIST;DEV=LP
                    :RUN LISTEQ2.PUB.SYS;PARM=1
Specifying ';PARM=1' directs LISTEQ2 to use file 'LIST' in a :FILE command instead of file '$STDLIST'. You could also get a LISTEQ2 listing to the line printer by running it from a JOB, but it would be a listing of the file equations and temporary files active within that JOB not within your SESSION!

End of PART I—Do PART II if you haven't done it yet.

PART II—Using the SYSTEM CONSOLE.

1). The SYSTEM CONSOLE looks like your Session's Terminal, but it operates differently. Sit at the CONSOLE and attempt to key in a command. The CONSOLE will not respond until you enter the code: Press CNTL and upper-case 'A' simultaneously. When the CONSOLE is ready to accept your input, it will prompt you with a '='. You must enter 'CNTL-A' before every command. Refer to the CONSOLE OPERATOR section of your Software Pocket Guide.

2). Enter a '=RECALL' command to display all pending I/O requests. If a request for your tape file is there, continue with the next step and mount your mag-tape. If your request is not there, either your Job is not yet into execution or there was some error in your Job Stream file. In either case, give your tape to anyone ready to use it and take corrective action.

JOB STREAM LAB #3

3). Mount your mag-tape on the drive. It must have a write-ring to be written on. The drive's hubs do turn, if somewhat reluctantly, so put the tape reel on the top hub, thread it according to the diagram on the drive, then press the LOAD button, followed by the ON-LINE button. The tape should advance to the LOAD point and signal ON-LINE. If you get to this point by yourself, congratulations. If not, seek consolation from your instructor.

4). The fourth item in your I/O request on the CONSOLE is your PIN (Process Identification Number). You must enter a '=REPLY' on the CONSOLE referencing your PIN and the logical device number (Idev) of the tape drive your tape is mounted on. On the drive, if 0 (zero) is lighted, the 'Idev' is 7. If '1' is lighted, the 'Idev' is 8; '2' lighted is Idev 9. Don't use 3 (Idev=10) as this is usually configured to be :JOB and :DATA accepting. If you just want to skip the whole operation, entering an Idev of '0' (zero) or 'N' will abort the I/O request.

5). While the tape is being written, enter a =SHOWIN command on the CONSOLE. Observe that here it lists $STDIN for all users on the system by default.

6). When the tape has been written, it will be rewound and the RESET and LOAD lights will be illuminated. At this point another I/O request should appear on the CONSOLE for your :RESTORE operation. Put the tape drive ON-LINE and reply to this new I/O request. At the completion of this operation the tape drive will again be RESET and at LOAD point. Press REWIND, then dismount the tape, give it to the next team, and get the listing from your JOB off the line printer to double check the results.

<< End >>

MPE PHILOSOPHY

- MPE is never CHANGED; MPE is continually ENHANCED.

- Two levels of MPE now exist; MPE-C and MPE III.

- MPE-C is a subset of MPE III; All features of MPE-C exist in MPE III.

- This section describes features unique to MPE III; in other sections MPE III features are so noted.

- MPE III will NOT be installed on Series I, CX and pre-CX 3000's.

UNCL

UNified Command Language

• Friendlier User Interface with more Helpful Error Messages.

• :HELP Subsystem.

• :REDO command.

• User Defined Commands (UDC's).

• Conditional Job Control.

• Users and Subsystems can create their own message catalogs and have them output by the MPE message system.

:HELP SUBSYSTEM

	[HELP	]
	[tablecontents	]
:HELP	[command[,keyword]	]
	[ALL	]
	[EXIT	]

-or-

:HELP udc-command   *Immediate mode only*

Messages reside in CICAT.PUB.SYS. HELP Subsystem can be disabled by purging this file.

tablecontents: keyword:
       SESSIONS   PARMS
       JOBS   OPERATION
       PROGRAMS
       FILES
       MANAGE
       UTILITY

Lengthy descriptions may be frozen on screen by stopping output to terminal with CNTL-S; output resumes with CNTL-Q.

:REDO SUBCOMMANDS

R   Replace (default)   I   Insert
D   Delete   U   Undo

USE :REDO TO CORRECT AN MPE COMMAND IN ERROR

```
:LISFT FCOPY.PUB.SX,2
 ^

UNKNOWN COMMAND NAME.    (CIERR 975)
:REDO
LISFT FCOPY.PUB.SX,2
    TF                          REPLACE by default
LISTF FCOPY.PUB.SX,2
              DIYS              like MODIFY in EDITOR
LISTF FCOPY.PUB.SYS,2
U                               UNDO level 1
LISTF FCOPY.PUB.SX,2
U                               UNDO level 2
LISFT FCOPY.PUB.SX,2
[RETURN]                        RETURN = execute command
 ^

UNKNOWN COMMAND NAME.    (CIERR 975)
```

USE :REDO TO MODIFY A VALID COMMAND

```
:LISTF FCOPY.PUB.SYS,2
ACCOUNT=  SYS          GROUP=  PUB

FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE----
                 SIZE  TYP         EOF      LIMIT R/B  SECTORS #X MX

FCOPY     PROG   128W  FB          173        173   1     174  1  1

:REDO
LISTF FCOPY.PUB.SYS,2
      DDDDDIEDITOR
LISTF EDITOR.PUB.SYS,2
[RETURN]
ACCOUNT=  SYS          GROUP=  PUB

FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE----
                 SIZE  TYP         EOF      LIMIT R/B  SECTORS #X MX

EDITOR    PROG   128W  FB          284        284   1     285  1  1

:
```

USER DEFINED COMMANDS (UDC)

```
udc-name   required-parm,optional-parm=default
[OPTION  [LIST][,LOGON][,NOBREAK][,NOHELP] ]
mpe-command !optional-parm
mpe-command-2 !required-parm
***********************************************
Q
RUN LISTEQ2.PUB.SYS
*

BUP FILE=@.@
OPTION LIST,NOBREAK
FILE BACKUP;DEV=TAPE
STORE !FILE;*BACKUP;SHOW
```

definition
options
body

delimiter

ENABLING UDC'S

To enable all UDC Commands for the Log-on User.Account contained in the specified UDC Catalog Files:

:SETCATALOG [catfilename[,catfilename[, ...]]] [;SHOW]

-where-

> 'catfilename' UDC Catalog File name.

> 'SHOW' Lists UDC Commands as Directory is built.

- First, all UDC Commands in effect for this User.Account are disabled.

- Then a sequential list of UDC Commands and the Files in which they reside is constructed in the Directory maintained in file COMMAND.PUB.SYS. A separate list is maintained for each different User.Account.

To disable all UDC Commands defined for Log-on User.Account:

:SETCATALOG

EXECUTING UDC'S

- :SETCATALOG establishes sequence in which UDC Commands appear in Directory.

- When the User enters an MPE Command, this Directory list is scanned sequentially from the beginning (i.e. the first occurrence of the UDC Command will be used).

- One UDC may call another UDC, but only forward in the Directory list.

- An MPE Command may be disabled for a User.Account by enabling a UDC command with the same name.

- User must have READ and LOCK Access to a UDC Catalog File.

- At Log-on or ':SETCATALOG with parms', all UDC Catalog Files associated with that User.Account are opened with ACC=IN;EAR.

- At Log-off or ':SETCATALOG without parms', all UDC Catalog Files for Log-on User.Account are closed.

EXECUTING UDC'S

- Disable UDC's system-wide by purging Directory file COMMAND.PUB.SYS.

- UDC calls may be nested. Each called UDC returns to the calling UDC upon completion. (JOB STREAM)

- An error at any nested level within a UDC will return immediately to the Command Interpreter.

- Within nested UDC's, once NOBREAK is encountered, it is in effect until control is returned to the Command Interpreter.

- Subsystems may be called by a UDC but commands cannot be passed to them from the UDC file.

EXECUTING UDC'S

PASSING PARAMETERS TO A UDC

- Parameters passed to a UDC either all positional or all keywords.

- A parameter containing embedded blanks or special characters must be enclosed in quote marks.

- LIST OPTION UDC—will be listed on $STDLIST just prior to execution. Shows the UDC after replacement by passed parms.

- If OPTION NOHELP is NOT specified, the UDC may be listed by the HELP command.

UDC EXAMPLES

```
:SETCATALOG UDC
:HELP Q
USER DEFINED COMMAND:

Q REQ,OPT=@
OPTION LIST
COMMENT !REQ
COMMENT !OPT
:Q
Q REQ,OPT=@
  ^

REQUIRED UDC PARAMETER IS MISSING.  (CIERR 1948)
:Q X                                    OPTIONAL parms may be omitted
COMMENT X
COMMENT @
:Q Y,Z                                  all positional
COMMENT Y
COMMENT Z
:Q A,OPT=B                              can't mix positional & keyword
      ^

FOUND MORE PARAMETERS THAN IN UDC DEFINITION.  (CIERR 1946)
:Q REQ=C,OPT=D                          all keyword
COMMENT C
COMMENT D
:Q A,"B,C,D"                            embedded delimiters
COMMENT A
COMMENT B,C,D
:
```

JOB CONTROL WORD (JCW)

0   1   2   15

		user defined values
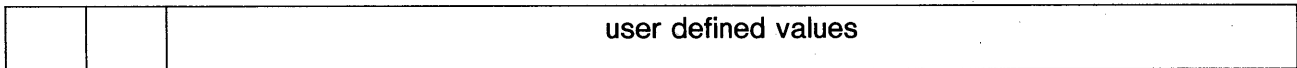
set by
system

- If bit 0 is set, system will terminate JOB / SESSION.

- One system-defined JCW maintained per JOB / SESSION.

- Subsystems automatically post completion codes to JCW.

- Predefined values for JCW:

```
OK     =   0
WARN   =   16384    (%40000)
FATAL  =   32768    (%100000)
SYSTEM =   49152    (%140000)
```

JCW

MPE COMMANDS:

```
:SETJCW   jcwname{delimiter}value

:SHOWJCW  [jcwname]

:IF  [(] logical expression [)] THEN
:ELSE
:ENDIF
```

MPE INTRINSICS:

```
jcw := GETJCW

PUTJCW (jcwname, jcwvalue, status)

SETJCW (word)
```

HEWLETT **hp** PACKARD

JCW EXAMPLE

```
:JOB STUDENT.INTRO/PASSWORD
:RUN UPDATE1
:IF JCW < WARN  THEN
:     SETJCW UPDATE1JCW := JCW
:     RUN UPDATE2
:     IF JCW < WARN  THEN
:            SETJCW UPDATE2JCW := JCW
:            RUN DBSTATUS
:            IF (UPDATE1JCW<50) AND (UPDATE2JCW<50) THEN
:                  RUN DAILYRPT
:            ENDIF
:            IF ((UPDATE1JCW = 1) OR (UPDATE2JCW = 1)) THEN
:                  RUN WEEKRPT
:            ENDIF
:     ELSE
:            RUN FIXUP
:     ENDIF
:ELSE
:     TELLOP UPDATE1 ABORTED
:     TELL STUDENT.INTRO;UPDATE1 ABORTED
:ENDIF
```

GENERIC NAMES

GENERIC NAMES may be used with the following Commands in MPE III:

LISTF	@.@.@	fileset
STORE	@.@.@	fileset
RESTORE	@.@.@	fileset
LISTVS	@.@.@	volume set
REPORT	@.@	groupset (AM & SM only)
LISTDIR2		

@	all strings of any length (including null string).
?	a single alphanumeric character.
#	a single numeric character.

EXAMPLES:

:LISTF @K@	will list files 'DISK', 'K', 'KK','K0381440', and 'QKZ'.
:LISTF LAB#??	will list files 'LAB234','LAB22Z', 'LAB2YZ', and 'LAB9Q9'.
	but will NOT list files 'LAB', 'LABXYZ', nor 'LAB2'.

MPE III MISCELLANEOUS

:STORE / :RESTORE

- :STORE writes whole directory at the beginning of every tape.

- Restore last tape first. :RESTORE reads directory first. Only if a file to be restored is on that tape does it read further; otherwise it tells you to mount previous tape.

MEMORY SIZE

- Maximum memory size increased to 2 Mega-bytes.

NUMBERED STREAMS

- MPE III streams either numbered or unnumbered.
    - —Sequence numbers must be at Rear of each record.
    - —Record length not restricted to 80 characters.

- Pre-MPE III systems will still only accept UNNumbered STREAM files !!

HEWLETT hp PACKARD

MAG-TAPE LABELS

- automatic volume recognition.

- automatic reel switching.

- multifile per volume capability.

- multivolume per file capability.

can READ	can WRITE
ANSI	ANSI
IBM	no-labels
no-labels	

ADDITIONS MADE—To :FILE Command

```
:FILE formaldesignator[ =filename.groupname [/lockword] ]

    [;NOLABEL                                         ]
    [;LABEL[=volid],[type],[exp-date],[seq] ]
```

To Intrinsics:

FOPEN, FCONTROL, FREADLABEL, & FWRITELABEL

HEWLETT hp PACKARD

DISC NOW DIVIDED INTO TWO DOMAINS

SYSTEM DISC DOMAIN

- Ldev=1 contains:
  —Directory for all Files in System Domain.
  —Accounting structure for all Groups, Accounts, and
     Users in the whole system.

- All types of discs supported.

- Each disc volume always mounted on same ldev.

- All packs must be on-line for system to run.

- System = 1 to 8 drives.

TWO DISC DOMAINS

PRIVATE DISC DOMAIN

- Directory for all Files in Volume Set on Master Volume.
- A copy of the Accounting structure for Groups and Accounts on the Volume Set are "spanned" to the Master Volume.
- Only drives with removable packs may be included in the Private Domain.
- Each disc volume may be mounted on any drive (ldev) in the Private Domain. Volumes automatically recognized.
- Whole Private Volume Class must be mounted.
- Volume Set = 1 to 8 volumes.
- Private Domain defined at system reload (1 to 8 drives).
- Private packs may be moved from system to system or stored off-line.

VOLUME SET

MASTER VOLUME

VOLUME CLASS 1

VOLUME CLASS 2

VOLUME CLASS 3
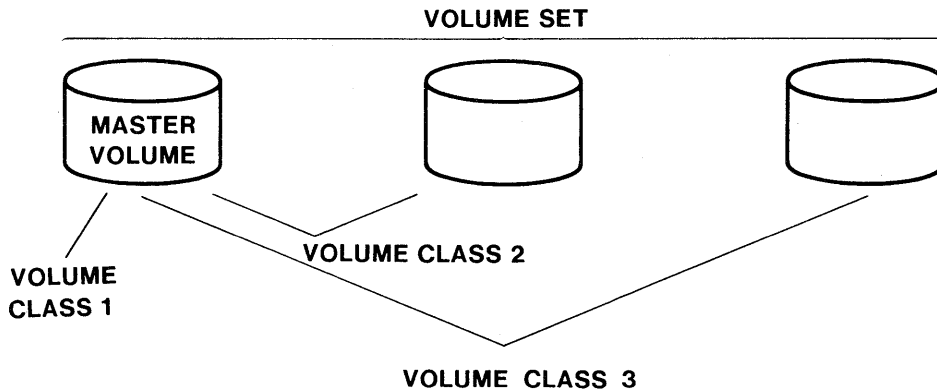
PRIVATE VOLUMES

- Private volumes initialized on-line with VINIT.

- To be able to access Private Volumes:
  - —User must have 'CV' or 'UV' capability.
  - —User's Account structure must have been 'spanned' to Volume Set by SM.
  - —User's Group structure must have been 'spanned' to Volume Set by AM.
  - —Needed volumes must be mounted.
  - —Access allowed by Console Operator.

- All Files in a Group must reside on the same Volume Class (or all in System Domain). This is Group's HOME VOLUME SET.

- Console Operator can specify 3 levels of access:
  - —Automatic recognition.
  - —User :MOUNT requests must be answered.
  - —No User :MOUNT requests will be honored.

- More private disc packs than private disc drives. Not all users will be able to access their files at all times!

PRIVATE VOLUMES

VINIT

- Gives each Private volume a name.

- Conditions private volume packs on-line (drive must be downed by Console Operator).

- Can make an existing volume a 'SCRATCH' pack.

- Disc condense—regain unusable space on-line by packing files together (either system volume or private volume).

- Disc-to-disc copy (must consider defective tracks).

SERIAL DISC

- Treats whole Private disc pack as mag-tape volume.

- "High Speed Mag-Tape".

- SYSDUMP or :STORE possible to Serial Disc.

- System Reload possible from Serial Disc.

MPE III LAB # 1 [1.0 hour]

1) Construct a JOB stream file to do a LISTF of the following sets of files in PUB.INTRO.
  a) All files beginning with 'K' (Use default detail for all LISTF's in this lab).
  b) All files beginning with 'LAB'.
  c) All files with at least one number in their name.
  d) All files with the number '1' as the 4-th character in their name.
  e) All files with any number as the 4-th character in their name.
Now STREAM the file. If a #Jnnn number is not returned, there is a problem with your Stream file. Upon completion, go pick up your output from the line printer.

2) Build a file with the following UDC's in it:
  a) A UDC to issue a :FILE command for the line printer and ':SETMSG OFF' for you automatically at log-on. Also have it list messages on your terminal. This can be accomplished with the :COMMENT command if the LIST OPTION is specified. Make the message something meaningful like "UDC has assumed control—say 'UNCL'".
  b) Place another UDC called 'Q' in the same file that will run LISTEQ2.PUB.SYS. Use an OPTION so the UDC will not be listed as it is executed. Keep the file then issue a :SETCATALOG to invoke it.

3) Use the UDC just created by entering 'Q'. Your Temp files and File Commands should be listed. Notice—Your line printer File Command is not among them. Now issue a :HELLO command for the same User.Account you are currently logged-on under. Your log-on messages should appear. Use 'Q' again and see that your line printer File command now exists... Huzzah.

MPE III LAB # 1

4)   Now add several more UDC's to the same file:
    a)   Add a UDC 'F' that runs FCOPY.PUB.SYS. Have it list the UDC as it is executed.
    b)   Add another UDC called 'L' that will call :LISTF with a detail parameter of ',2'. Construct
          it so a fileset or a different detail may be specified if the User chooses. Use the OPTION
          to inhibit BREAK.
    Now keep this file under its previous name. You must have removed this file from the Catalog or
it will still be open with ACC=EAR and you will not be able to purge it. If you are in this situation, keep the new
file under a different name, remove your UDC from the catalog, purge the old one and rename the new one.
    Invoke your latest copy of your UDC file. Test the new commands by entering an 'L'. You should
get a level '2' detail listing of all files in your group. Try to interrupt the listing with the <BREAK> Key.
Specify several different filesets and different detail levels. Try passing positional parameters and keyword
parameters. Enter 'F' just to make sure it works. Exit FCOPY immediately.

5)   Write out the :STORE command to store all files beginning with 'LAB' in all PUB groups of all
      Accounts (this could actually be done only by the System Manager).

_____

6)   Write out a :RESTORE command to restore all files beginning with 'LAB' into all groups of all
      accounts in the system from a SYSDUMP tape. Only restore files that did not previously exist in any
      group.

_____

7)   Write out below a Job Stream to Run 'LABJCW1', show the JCW setting, abort the Job if JCW is
      WARN or greater. Then if JCW is greater than 'OK' plus 100, run 'LABJCW2' then 'LABJCW3'. Else,
      just run 'LABJCW2'.

_____

MPE III LAB # 1

OPTIONAL—Proceed with the remainder of the lab only if time permits.

8)   Enhance your Log-on messages with some imaginative Terminal Display Enhancements. Some of interest are:

ESC & d J	Half-bright Inverse Display Enhancement
ESC & d @	Reset Display Enhancements
ESC H	Home Cursor
ESC J	Clear Display
ESC M	Delete Line
ESC S	Roll Up
ESC T	Roll Down
ESC z	Terminal Self-Test

NOTE: 'ESC' represents the ESCAPE key.

<< End >>

FCOPY CAPABILITIES

- **Copy Files.**

- **Function on File Subsets.**

- **Manipulate Multifile Tape Volumes.**

- **Perform Code Translation.**

- **File Dumps.**

- **Create New Files.**

- **Lower-case to Upper-case Conversion.**

- **File Error Handling Techniques.**

- **Perform Copy Verification.**

- **Perform File Compares.**

HEWLETT **hp** PACKARD

FCOPY

> **FROM={filereference1} ;TO={filereference2}[;options]**
>
>   **{   *   }   {   *   }**
>
>   **{ 'empty' }   { 'empty' }**

```
:RUN FCOPY.PUB.SYS
>FROM=;TO=          ; ECD
>EXIT
```
'empty' — from $STDIN to $STDLIST

```
:FILE STUDENT;DEV=TAPE;&
:  REC=-80,16,F,ASCII
:RUN FCOPY.PUB.SYS
>FROM=DISCFILE;TO=*STUDENT
```
filereference (back-reference required for device files)

```
>FROM=FILE2;TO=*
>EXIT
```
* (internal back-reference)
Concatenation of output

HEWLETT hp PACKARD

CREATE A NEW DISC FILE

>FROM=filereference1;TO=filereference2;NEW

```
:HELLO STUDENT.INTRO/PASSWORD
SESSION NUMBER = #S77
THU, MAY 18, 1978,  2:07 PM
HP32002A.01.MR

THIS IS THE TRAINING SYSTEM.
:RUN FCOPY.PUB.SYS

HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976

>FROM=DEFTABS.PUB;TO=NEWTABS;NEW          Copies attributes exactly
EOF FOUND IN FROMFILE AFTER RECORD 13

14 RECORDS PROCESSED *** 0 ERRORS

>EXIT

 END OF PROGRAM
:
```

FCOPY ;SUBSET

Select Whole Records by matching patterns

;SUBSET="characterstring"[,[column][,EXCLUDE]]

    ;SUBSET="ANSWER ""YES"""
                   (above string is ANSWER "YES")

;SUBSET=#patternlist#[,[column][,EXCLUDE]]

    ;SUBSET=#%7,27,%110#,11
                 (above pattern is BELL ESC H)

- only 1 subset may be specified.
- columns start with 1.
- 'string' or 'pattern' represents up to 35 characters.
- in continued commands, a 'string' may NOT be split between lines; a 'pattern' may only be split at commas separating character values.

FCOPY — ;SUBSET

Select Whole Records by Ranges of Record Numbers.

```
              {range              }
  ;SUBSET={(range[;range]...)}
```

where 'range' is either:

[starting-record-number][,number-of-records]

-or-

[starting-record-number][:last-record-number]

- records start with 0.
- up to 255 ranges may be specified.

'from-file' is on Unlabeled Mag-Tape

<default>   Copy entire tape.

;SUBSET   Copy current file only. Tape will be positioned
                                            at the beginning of the current tape file.

FCOPY — ;SUBSET EXAMPLES

```
:RUN FCOPY.PUB.SYS

HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
```

LIST ALL RECORDS WITH 'C' IN COLUMN 53

```
>FROM=LAB1DATA.PUB;TO=;SUBSET="C",53
*200*X
WARNING: FROMFILE RECSIZE IS 71 BYTES, TOFILE RECSIZE IS 80 BYTES.
CONTINUE OPERATION (Y OR N) ?Y
OLIVER     TEETHOUT   867-0138 20085 UPPER PLATE PL    CUPERTINO      95053
ARMAND     HAMMER     298-4988 1350  ALKALI AV         CAMBRIAN PARK  95131
TRUDY      TEKTIFF    255-1005 17155 POIROT PL         CAMPBELL       95121
JOSE       CANUSI     214-5566 2485  ANTHEM WY         CAMPBELL       95129
ANDY       LUCIAN     264-4169 1119  IBERIAN CT        CUPERTINO      95070
BUZZ       SAWYER     259-3434 1850  FOREST DR         CUPERTINO      95023
EOF FOUND IN FROMFILE AFTER RECORD 25

            6 RECORDS PROCESSED *** 0 ERRORS
```

LIST ALL RECORDS WITH PATTERN 'CAM' IN COLUMN 53

```
>FROM=LAB1DATA.PUB;TO=;SUBSET=#%103,65,%115#,53
*200*[RETURN]
ARMAND     HAMMER     298-4988 1350  ALKALI AV         CAMBRIAN PARK  95131
TRUDY      TEKTIFF    255-1005 17155 POIROT PL         CAMPBELL       95121
JOSE       CANUSI     214-5566 2485  ANTHEM WY         CAMPBELL       95129
EOF FOUND IN FROMFILE AFTER RECORD 25

3 RECORDS PROCESSED *** 0 ERRORS
```

FCOPY — ;SUBSET EXAMPLES

LIST FIRST FOUR RECORDS OF FILE

```
>FROM=LAB1DATA.PUB;TO=;SUBSET=,4        — four records
*200*RETURN
NEIL       DU PREE    246-1112 4097   PRIE DIEUX DR    SAN JOSE     95013
OLIVER     TEETHOUT   867-0138 20085  UPPER PLATE PL   CUPERTINO    95053
AMANDA     RECKONWITH247-9142 2474    MACHO ST         SANTA CLARA  95020
AMOS       QUITO      243-8171 1467   ANOPHELES AV     NEW ALMADEN  95143
4 RECORDS PROCESSED *** 0 ERRORS
```

LIST RECORDS 0(ZERO) THROUGH 4

```
>FROM=LAB1DATA.PUB;TO=;SUBSET=:4        — copy thru 4
*200*RETURN
NEIL       DU PREE    246-1112 4097   PRIE DIEUX DR    SAN JOSE     95013
OLIVER     TEETHOUT   867-0138 20085  UPPER PLATE PL   CUPERTINO    95053
AMANDA     RECKONWITH247-9142 2474    MACHO ST         SANTA CLARA  95020
AMOS       QUITO      243-8171 1467   ANOPHELES AV     NEW ALMADEN  95143
ARTHUR     MOMITER    443-5346 1554   MERCURY ST       MILPITAS     94173
5 RECORDS PROCESSED *** 0 ERRORS
```

LIST FILE FROM RECORD 23

```
>FROM=LAB1DATA.PUB;TO=;SUBSET=23
*200*RETURN
BUZZ       SAWYER     259-3434 1850   FOREST DR          CUPERTINO    95023
PHIL       ARBUSTER   997-1040 672    CONSTITUTION DR    SANTA CLARA  95110
CLARA      NETTE      243-4493 2667   GOODMAN DR         ALVISO       95143
EOF FOUND IN FROMFILE AFTER RECORD 25

3 RECORDS PROCESSED *** 0 ERRORS
```

FCOPY — TRANSLATION FUNCTIONS

```
;EBCDICIN
;EBCDICOUT    { field               }
;BCDIN        [={(field[;field]...)} [,EXCLUDE] ]
;BCDOUT
```

WHERE:

EBCDICIN	translates from EBCDIC to ASCII
EBCDICOUT	translates from ASCII to EBCDIC
BCDIN	translates from BCD to ASCII
BCDOUT	translates from ASCII to BCD

'field'   is a pair of unsigned integers of the
form 'a,b' or 'a:c' where:
'a' is starting column (first is 1)
'b' is number of columns
'c' is ending column.

EXCLUDE   specifies referenced fields are NOT to be converted,
otherwise they are the only fields converted.

NOTE: EBCDIKIN and EBCDIKOUT are special functions for the Katakana
character set (modern Japanese alphabet).

COMPARING FILES

;COMPARE[=number-of-errors]

- Compare the data in two files on any devices.
- Neither file is altered.
- Record characteristics identical (Block factor may differ).
- Data compared record by record.
- The record number & the number of the byte that do not compare are listed on $STDLIST & error count is incremented.
- Operation terminates when 'number-of-errors' has been reached.
- Default for 'number-of-errors' is 1.

;VERIFY[=number-of-errors]

- Verification pass associated with making a new copy.
- Like compare except files may only be on DISC or MAG-TAPE.

FCOPY OPTIONS

BYPASSING MAG-TAPE ERRORS

`;IGNERR=number-of-errors`

- Applies to MAG-TAPE read errors only.
- 'number-of-errors' indicates number to be tolerated; one more than that terminates operation.
- Default for 'number-of-errors' is 0 (zero).

SPECIAL FILE NAMES FOR PERIPHERALS ON HP TERMINALS

$CTUL addresses left tape cartridge on HP-264x terminal.
$CTUR addresses right tape cartridge on HP-264x terminal.
$HARD addresses hard copy printer attached to HP-264x terminal.

- Program will prompt for DUPLEX and PARITY switch settings.
- TERMTYPE must be 10 or 12.
- No internal back-references allowed.
- Not supported under MPE-C.

FCOPY OPTIONS

USING PERIPHERALS WITH NO LOWER-CASE

;UPSHIFT Lower-case characters will be upshifted to upper-case.

COPYING MULTIPLE FILES FROM UNLABELED MAG-TAPES

```
;FILES={number-of-files}
       {ALL            }
```

FCOPY CAN DEBLOCK ODD BYTE LENGTH RECORDS

```
;DEBLOCK=logical-record-length
```

where: + 'logical-record-length' = words.
           − 'logical-record-length' = bytes.

• MPE Records and Blocks begin & end on word boundaries.

• ';DEBLOCK' allows record deblocking to be done by FCOPY.

FCOPY — ;SKIPEOF

FILE POSITIONING FOR CARTIDGES AND UNLABELLED MAG-TAPES

```
             {+}                    {+}
 ;SKIPEOF=[{{-} from-eofs   }] [,{{-} to-eofs   }]
          [{from-file-number}] [,{to-file-number}]
```

where:

'+' or '—'   specify forward or backward positioning.

'from-eofs'   number (integer) of files to be skipped on 'from' tape. O(zero) re-reads current file.

'from-file-number' absolute file number (integer) on 'from' tape. Begins with 1 for first file.

'to-eofs'   number (integer) of files to be skipped on the 'to' tape. O(zero) = current file.

'to-file-number'   absolute file number (integer) on 'to' tape. Begins with 1 for first file.

FCOPY — USER LABELS

;NOUSERLABELS

Users may create their own 'USER LABELS' on disc or labeled mag-tape.

- on disc user labels are 256 bytes long.

- on labeled tape user labels are 80 bytes long.   *MPE III ONLY*

USER LABELS are defined either through using MPE Intrinsics or via the 'USE' statement in Declarative portion of COBOL Procedure Division.

If both 'from' and 'to' files are declared to have user labels, FCOPY will copy USER LABELS by default.

Specifying ';NOUSERLABELS' will create a 'to' file without USER LABELS.

HEWLETT hp PACKARD

FCOPY — PROCESSING TAPES

UNLABELED MAG-TAPE

- One file through whole tape may be processed in 1 FCOPY operation.

- File attributes specified via :FILE.

LABELED MAG-TAPE   *MPE III ONLY*

- Process one file at a time.

- File attributes, volume ID, file location on volume, and file name
    defined via :FILE command.

HP-264x TAPE CARTRIDGES

- Process one file at a time.

- Attributes assumed to be ;REC=-256,1,F,ASCII.

- CNTL chars and ESCAPE sequences that would normally be executed by
    the terminal CAN be written to and read from tape cartridges.

HEWLETT hp PACKARD

FCOPY 'TAPE' SUMMARY

FCOPY 'TAPE' OPTIONS VALID FOR EACH TYPE OF 'TAPE'

	UNLABELED MAG-TAPE	LABELED MAG-TAPE	TAPE CARTRIDGES
;FROM=* / ;TO=* (internal back-reference)	X		
;DEBLOCK=	X	X	
;FILES=	X		
;NOUSERLABELS		X	
;SKIPEOF=	X		X

FCOPY — DUMP FORMATS

```
{CHAR }    {HEX  }
[;{CLEAR}] [;{OCTAL}] [;NORECNUM] [;TITLE="string"]
{KANA }
```

where:

CHAR   produces a character dump of printable ASCII characters (CNTL chars are represented by decimal points).

CLEAR   produces a character dump of all ASCII characters (CNTL chars will NOT be modified; a device like an HP-264x will execute CNTL characters).

KANA   Katakana character set (modern Japanese alphabet).

HEX   produces a hexadecimal dump.

OCTAL   produces an octal dump (same character will be different octal values in low-order & high-order bytes of a word).

NORECNUM   omit record number & word number captions from dump.

TITLE   specifies a heading (70 chars max). If output is to line printer string will appear at top of each page; otherwise appears once at beginning of file. If command is continued, string may not be split between lines.

FCOPY — DUMP FORMATS

```
>                              < BREAK key pressed >
:BUILD CNTLCHAR;REC=-40,32,F,ASCII;DISC=10
:RESUME
READ PENDING
FROM=;TO=CNTLCHAR
*200*[RETURN]          'DISPLAY FUNCTIONS' on when ESCAPE sequences entered
<2 BELLS>bb<ESC S>eS<ESC M>eMXYZ.
 < CONTROL Y >                    where: 'b' represents BELL.
                                        'e' represents ESCAPE key.

2 RECORDS PROCESSED *** 0 ERRORS

>FROM=CNTLCHAR;TO=;CHAR                  ';CHAR' lists CNTL chars
                                        as decimal points

CNTLCHAR RECORD 0 (%0)

000000:  <2 BELLS>..<ESC S>.S<ESC M>.MXYZ.
EOF FOUND IN FROMFILE AFTER RECORD 0

1 RECORD PROCESSED *** 0 ERRORS

>FROM=CNTLCHAR;TO=;HEX;CHAR

CNTLCHAR RECORD 0 (%0)

000000:  3C32 2042 454C 4C53 3E07 073C 4553 4320 <2 BELLS>..<ESC
000010:  533E 1B53 3C45 5343 204D 3E1B 4D58 595A S>.S<ESC M>.MXYZ
000020:  2E20 2020 2020 2020
EOF FOUND IN FROMFILE AFTER RECORD 0

1 RECORD PROCESSED *** 0 ERRORS
```

FCOPY — DUMP FORMATS

OTHER LISTINGS PASS CNTL CHARS TO TERMINAL

>FROM=CNTLCHAR;TO=;CLEAR

CNTLCHAR RECORD 0 (%0)

```
000000:  <2 BELLS><ESC S>
XYZ.
EOF FOUND IN FROMFILE AFTER RECORD 0

1 RECORD PROCESSED *** 0 ERRORS
```

>FROM=CNTLCHAR;TO=
```
*200*RETURN
<2 BELLS><ESC S>
XYZ.
EOF FOUND IN FROMFILE AFTER RECORD 0

1 RECORD PROCESSED *** 0 ERRORS
```

>EXIT

```
END OF PROGRAM
:
```

FCOPY — CARRIAGE CONTROL

Combination of FCOPY option and :FILE commands for 'to' and 'from' files define use of carriage control character.

';CCTL' specified in :FILE command or disc label for:		CCTL FCOPY OPTION specified:		
'from' file	'to' file	-neither-	;CCTL	;NOCCTL
NO	NO	1	1	1
NO	YES	2	3	2
YES	NO	1	1	1
YES	YES	4	4	2

where:

1 signifies entire 'from' record copied as data.
2 signifies 1 char for single spacing added to each 'to' record.
3 signifies 1-st char of 'from' file used for carriage control.
4 signifies exact copy of data and control char.

FCOPY — EXIT

>EXIT   terminate FCOPY; close 'from' and 'to' files.

NOTES ON FCOPY

• At the end of each FCOPY operation both the 'to' and 'from' files are left OPEN anticipating concatenation of output. (EOF has not yet been written on unlabeled mag-tape 'to' file)

• If subsequent FCOPY operation is 'EXIT' or is NOT an internal back-reference, the file from the previous operation is CLOSEd at that time.

• Features relating to KSAM (;KEY= / ;NOKSAM / ;TO=(dfile,kfile)) are covered in the KSAM section of this course.

• Features NOT supported under MPE-C:
        —KSAM features (;KEY= / ;NOKSAM / ;TO=(dfile,kfile)).
        —;NOUSERLABELS
        —Katakana features.
        —peripherals connected to terminals ($xxxx).

FCOPY LAB #1

From a Session, concatenate into one listing on the line printer all of the following:

1) All records in LAB1DATA.PUB with '951' beginning in column 67.

2) All records in LAB1DATA.PUB that DO NOT have '951' beginning in column 67.

3) A 'CHAR' dump of DEFTABS.PUB.

4) A 'CHAR' & 'HEX' dump of DEFTABS.PUB.

5) An 'OCTAL' & 'CHAR' dump of DEFTABS.PUB.

FCOPY LAB #2

Do all steps in FCOPY LAB #1 from a Job Stream. Output all listings into the same file but don't use $STDLIST.

> OPTIONAL — Proceed only if time permits.

FCOPY LAB #3

Modify your Job Stream from FCOPY LAB #2 to concatenate all output in 1 disc file, then list it on the line printer honoring carriage control characters. Execute your Job Stream.

HEWLETT **hp** *PACKARD*

SORT/MERGE

SORT ANY FILE / MERGE ANY SORTED FILES.

- Any I/O media type.

- Fixed or Variable Length records.

- SORT Output may be records, sort keys, record numbers, or sort keys & record numbers.

SORT KEYS MAY BE ANY 3000 DATA TYPE.

- Each key may be either ascending or descending.

- Keys may be contiguous, separated, or overlapped.

MAY BE RUN STAND-ALONE OR ACCESSED VIA INTRINSICS.

- FORTRAN and SPL can easily access intrinsics.

- COBOL has SORT verb.

HEWLETT **hp** PACKARD

SORT — INPUT

SPECIFY WHERE TO READ RECORDS TO BE SORTED

```
:FILE INPUT=filereference
```

-or-

```
>INPUT {filename [,number-of-records]          }
       {* [,number-of-records [,record-size]]  }
```

where:

filename	is any 'filereference' you have READ access to.
*	input will be from $STDINX.
number-of-records	file from DISC — SORT will calculate total number of records to be read from file label. file non-DISC —you should specify total no. of records to be read; else assumes 10,000.
record-size	record size if different than that of $STDINX.

HEWLETT hp PACKARD

SORT — OUTPUT

SPECIFY FILE WHERE OUTPUT IS TO BE PLACED

:FILE OUTPUT=filereference

-or-

```
>OUTPUT   {filename}  [,NUM]  [,KEY]
          {    *    }
```

where:

filename	filereference of file you have WRITE access to.
*	output listed on $STDLIST (no other output is saved).
-default-	output is input records in sorted sequence.
NUM	output is double-word record numbers of records in the input file. These pointers can be used to access the original input file in sorted sequence (ADDROUT Sort).
KEY	output is sort keys in sorted sequence. Sort keys are concatenated together with most major first.
NUM & KEY	output will be an 'index' each record of which contains a double-word integer record number followed by the concatenated sort keys.

HEWLETT hp PACKARD

SPECIFY SORT KEYS

```
                        {[,BYTE]  }
             ,length {,DISPLAY }
                        {,PACKED  }
                        {,PACKED* }
  KEY position {                         } [;DESC]   [;position ...]
                        {,INT    }
             [,length] {,DOUBLE}
                        {,REAL   }
                        {,LONG   }
```

SORT KEYS MAY BE SPECIFIED WITH SEVERAL PER KEY COMMAND, ONE PER KEY
COMMAND, OR A MIXTURE OF BOTH. 1-ST IS MOST MAJOR; LAST IS MOST MINOR.

where:

position	character position of record where key begins (1 is first position).
length	length of key in bytes.
DESC	descending sequence for this key. Default is ascending.

SORT KEYS

'type'

BYTE   (default) logical sort by collating sequence. Can sort ASCII, EBCDIC, BCD, or LOGICAL data.

DISPLAY   Punched card or COBOL Display format (signs 'overpunched' in units position of numeric fields).

0	=	%60	+0	=	{ (%173)	-0	=	}	(%175)
1	=	%61	+1	=	A (%101)	-1	=	J	(%112)

...   .

9	=	%71	+9	=	I (%111)	-9	=	R	(%122)

PACKED   packed decimal with an odd number of digits.

WORD

d	d	d	d	d	d	d	S

BYTE   BYTE   BYTE
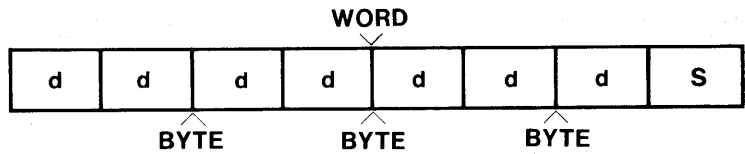
PACKED*   packed decimal with even number of digits.

INT   2's complement integer — default length is 2 bytes.

DOUBLE   2's complement double-word integer — default is 4 bytes.

REAL   short floating point — default length is 4 bytes.

LONG   long floating point — default length is 8 bytes. (on Series I, CX and pre-CX 3000's, default is 6 bytes).

SORT — REMAINING COMMANDS

RESET   erases KEY specifications so they may be corrected (from sessions only).

VERIFY   list specifications in effect on file 'LIST'.

CNTL-Y   pressing CNTL-Y while running stand-alone SORT from session will display current status of the sort on file 'LIST'.

END   end of commands; start sort. (if INPUT is * from a session, '?' will prompt for input on $STDINX. Terminate this data with ':EOD'.)

HEWLETT **hp** *PACKARD*

SORT EXAMPLE

PRESSING CNTL-Y DURING EXECUTION OF STAND-ALONE SORT FROM A SESSION,
THE FOLLOWING WILL BE DISPLAYED ON 'LIST'.

```
        INPUT PHASE:  xxxx RECORDS HAVE BEEN INPUT
 -or-
        INTERMEDIATE SORT PHASE:  PASS x OF y
 -or-
        OUTPUT PHASE:  xxxx RECORDS HAVE BEEN OUTPUT
```

SAMPLE SORT FROM A SESSION

```
:FILE INPUT=DATA2
:RUN SORT.PUB.SYS

HP32214B.01.05 SORT/3000  THU, APR 20, 1978,  3:30 PM
(C) HEWLETT-PACKARD CO. 1976

>OUTPUT SORTFILE
>KEY 67,1,DESC;68,4
>KEY 11,10
```

SORT EXAMPLE

```
>VERIFY

INPUT FILE = INPUT
OUTPUT FILE = SORTFILE
KEY POSITION    LENGTH      TYPE      ASC/DESC
        67           1      BYTE        DESC    (MAJOR KEY)
        68           4      BYTE        ASC
        11          10      BYTE        ASC

>END

                                        < CNTL Y pressed >
OUTPUT PHASE:  4 RECORDS HAVE BEEN OUTPUT
PURGE OLD OUTPUT FILE SORTFILE.GSTUDENT.INTRO ? YES


                        STATISTICS

NUMBER OF RECORDS =                              15
NUMBER OF INTERMEDIATE PASSES =                   0
SPACE AVAILABLE (IN WORDS) =                 12,197
NUMBER OF COMPARES =                             63
NUMBER OF SCRATCHFILE IO'S =                     10
CPU TIME (MINUTES) =                            .01
RECORD SIZE (IN BYTES) =                         71
SCRATCH FILE SIZE (# SECTORS) =                  93

 END OF PROGRAM
:
```

MERGE COMMANDS

MERGE FILES WITH IDENTICAL KEY SPECIFICATIONS SORTED IN SAME SEQUENCE

        INPUT   filename1[,filename2] ...

        OUTPUT   {filename} [,number-of-records]   [,KEY]
                 {$STDLIST}

                 where:

filename   filereference (READ access required for INPUT files; WRITE access required for OUTPUT file).

$STDLIST   output is $STDLIST listing (no other output is saved).

number-of   if all INPUT files DISC — MERGE determines number of records from DISC file labels.
-records   if any INPUT file non-DISC — you should specify total number of records; otherwise 10,000 is assumed.

KEY   output is records consisting only of sort keys, most major first.

All other commands are identical with SORT commands:
        KEY   RESET   VERIFY   END   CNTL-Y

MERGE EXAMPLE

:<u>RUN MERGE.PUB.SYS</u>

HP32214B.01.05 MERGE/3000 THU, APR 20, 1978, 3:34 PM
(C) HEWLETT-PACKARD CO. 1976

›<u>INPUT SORTFILE,DATA1</u>
›<u>OUTPUT MERGFILE</u>
›<u>KEY 67,1,DESC;68,4;11,10</u>
›<u>VERIFY</u>

INPUT FILES = SORTFILE,DATA1
OUTPUT FILE = MERGFILE

KEY POSITION	LENGTH	TYPE	ASC/DESC	
67	1	BYTE	DESC	(MAJOR KEY)
68	4	BYTE	ASC	
11	10	BYTE	ASC	

›<u>END</u>

PURGE OLD OUTPUT FILE MERGFILE.GSTUDENT.INTRO ? <u>YES</u>

                    STATISTICS

NUMBER OF INPUT FILES =   2
NUMBER OF RECORDS =   26
SPACE AVAILABLE (IN WORDS) =   12,214
NUMBER OF COMPARES =   25
CPU TIME (MINUTES) =   .01
ELAPSED TIME (MINUTES) =   .09
RECORD SIZE (IN BYTES) =   71

 END OF PROGRAM
 :

FORMAL DESIGNATORS

input comes from   'INPUT'

sorted or merged records are output to   'OUTPUT'

commands read from   'TEXT'

messages, prompts, status, and statistics listed on   'LIST'

sort internal work file (on DISC)   'SORTSCR'

NOTE: Be careful that you do not have :FILE commands active that may re-direct I/O to another file. (e.g. ':FILE LIST;DEV=LP').

SUPPORTED UTILITIES

LISTEQ2   Lists temporary files and active file commands for the Session or Job it is run from.

LISTDIR2   Lists security settings at the Account, Group and File levels and lists access any User has to any file.

FREE2   Lists all space available on discs.

SPOOK   Maintenance program for SPOOL files. Can store a spool file to mag-tape, then load it back to that or another system for printing at a later time.
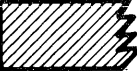
NOTE 1: Omit '2' from above for MPE-C.

NOTE 2: These Utilities are documented in the MPE System Utilities Reference Manual (32000-90008 for MPE-C; 30000-90044 for MPE-III) which is handed out in the System Manager Course.

HEWLETT **hp** PACKARD

SORT LAB # 1 [0.6 hour]

1-5	6-25	26-27	28	29-30	31	32-34	35-80
EMPL. NO	NAME	AGE	SEX (1-M, 2=F)	YEARS SERVICE	X	JOB CODE	////////

1) Two Employee data files exist with the above record layout, EMPDATA.PUB which is already sorted into the desired sequence and EMPCARD.PUB which is unsorted.

2) Find the attributes of these files and :BUILD a permanent disc file 'MFILE' big enough to hold both of them but otherwise with the same attributes as the two files.

3) Using FCOPY, make an exact copy of EMPCARD.PUB in your group called DFILE.

4) Sort DFILE by years of service (longest first) and put the output back into the same file.

5) Merge DFILE with EMPDATA.PUB and put the output in MFILE.

6) Using FCOPY, make a listing of MFILE on the line printer, deleting the fields from job code through the end of the record.

OPTIONAL — Proceed only if time permits.

7) Build and execute a Job Stream to accomplish the above lab. From a Job the output of Sort is not allowed to go into the input file, so create a temporary file to contain the output of step 4) and be the input for step 5).

HEWLETT hp PACKARD
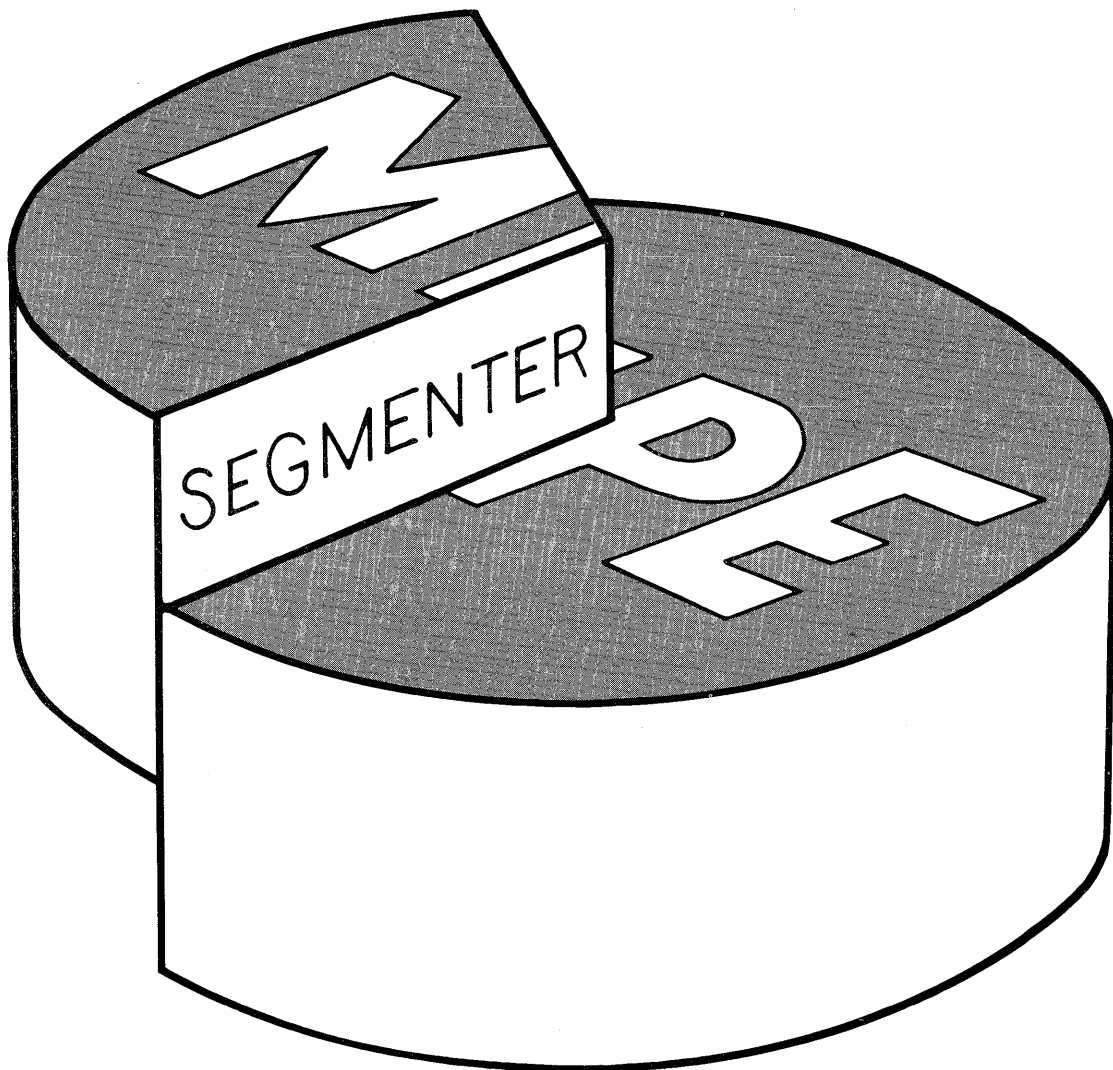
HP-3000 HARDWARE ARCHITECTURE

CENTRAL DATA BUS
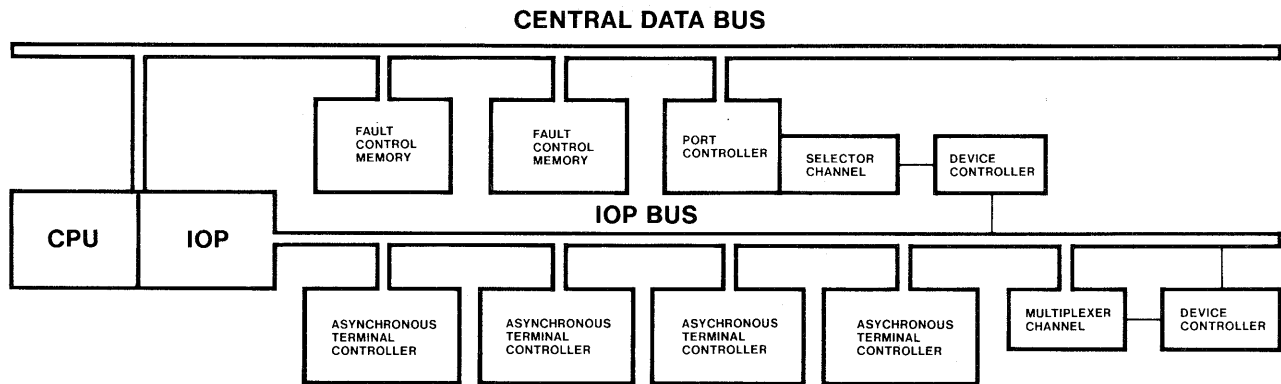
- Bus architecture
- 2 Processors
- Hardware implemented stack
- Byte manipulation
- 16- and 32-bit integer arithmetic
- 32- and 64-bit floating point arithmetic
  (48-bit for Series I)
- 28-digit packed decimal arithmetic

The following apply to Series III only:
- 209 Unique Instructions
- Automatic restart after power failure
- 175-nanosecond microinstruction time
- Pipelining of microinstructions
- 700-nanosecond cycle time for semi-conductor memory
- Automatic fault detection & single bit fault correction
- Rechargeable battery packs maintain memory data for a
  minimum of 40 minutes in event of power failure

(See Price/configuration guide (5953-0521 for Series III ;
5953-0518 for Series I) for more information.)

STACK ARCHITECTURE

**CODE SHARING
BY
PROCESSES**
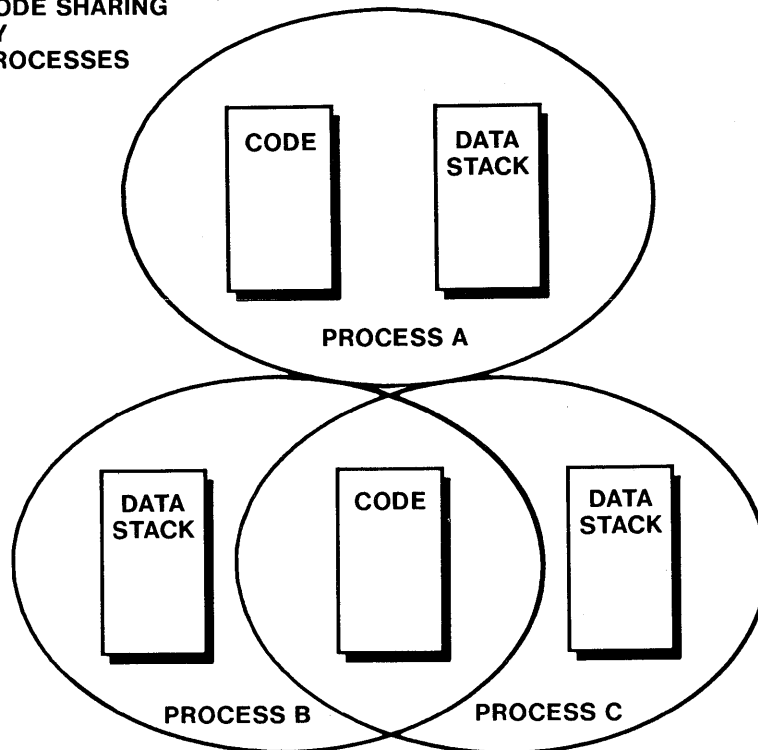
- Separation of Code & Data
- Code automatically sharable & re-entrant

MEMORY SEGMENTATION

ADVANTAGES OF STACK ARCHITECTURE

- 38 Specific Purpose Registers.

- 1 Set of registers points to Code Segment & Stack of the currently active Process.

- User Protection by automatic hardware detection of out of bounds addressing.

- Efficient Expression Evaluation.

- Efficient Subroutine Linkage.

- Rapid Interruption & Restoration of user Environments.

- Automatically Re-entrant Code.

- Recursion.

HEWLETT hp PACKARD

INTERNALS

CODE SEGMENT POINTING REGISTERS

DATA SEGMENT POINTING REGISTERS

PB BANK

PB-REGISTER
(PROGRAM BASE)

CODE SEGMENT

P-REGISTER
(PROGRAM COUNTER)

OPTIMALLY 4K WORDS

PL-REGISTER
(PROGRAM LIMIT)

STT

MAXIMUM 16K WORDS

STACK BANK

DL-REG
(DATA LIMIT)

DB BANK

DB-REG
(DATA BASE)

DATA SEGMENT

DYNAMIC AREA

GLOBAL DATA

STACK MARKER

Q-REG
(STACK MARKER)

SR-REG

DISPLACEMENT 0, 1, 2, 3, 4

(TOP-OF-STACK IN MEMORY)

SM-REG

S-POINTER

(LOGICAL TOP-OF-STACK)

MAXIMUM 32K WORDS

EXTRA DATA SEGMENTS

DB BANK

DB-REG

FILE SYSTEM BUFFER

MAXIMUM 8K WORDS

AVAILABLE (RESERVED)

Z-REG
(STACK LIMIT)

DYNAMIC EXPANSION

**1 PER FILE PER PROCESS
(BLOCK SIZE X NO. BUFFERS)**

STACK MARKER

| X-REG |
| P-REG + 1 |
| STATUS |
| Δ Q |

CODE SEGMENT # CC ETC.
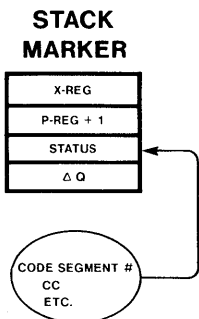
INCREASING ADDRESSES

VIRTUAL MEMORY

VIRTUAL MEMORY—DATA is swapped; CODE is only read.

FLOW OF PROGRAM CONTROL

DB

DYNAMIC

PRIMARY (256)

GLOBAL DATA AREA

SECONDARY

FIRST Q

MAINLINE

TEMPORARY STORAGE

PROCEDURE PARAMETERS

STACK MARKER

PREVIOUS Q

PROCEDURE A

TEMPORARY STORAGE

PROCEDURE PARAMETERS

STACK MARKER

PREVIOUS Q

PROCEDURE B

TEMPORARY STORAGE

ALLOCATIONS DUE TO CALLING PROCEDURE C

S

PROCEDURE PARAMETERS

STACK MARKER

Q

LOCAL VARIABLES

ALLOCATIONS LOCAL TO PROCEDURE C

TEMPORARY STORAGE

S

AVAILABLE

Z

CODE SEGMENT 22

PB

BEGIN PROC C

BEGIN

END

5
4 0 U PB REL ADDRESS
3
2
1
0 PL

STACK MARKER

SEGMENT TRANSFER TABLE

CODE SEGMENT 21

PB

MAINLINE PROGRAM BEGIN MOVE PARMS (CALL "A") PCAL (1)

END

	STT #	SEG #
2		
1	0	SEG 23
0		

PB L

SEGMENT TRANSFER TABLE

CODE SEGMENT 23

PB

BEGIN PROC A

CALL "B"

PCAL (4)
BEGIN

BEGIN PROC B

PCAL (5)

CALL "C"

END

STT ENTRY NUMBER

8			
7			
6			
5	1	STT #	SEG #
4	0 U	PB REL ADDRESS	
3			
2			
1			
0	LAST SST # = 8		PL

SEGMENT TRANSFER TABLE

ALREADY IN MEMORY

MPE MEMORY MANAGER

NOT IN MEMORY

ENTRY POINTS
EXTERNAL REFERENCE

TO RUN ANY PROGRAM

COBOL
SOURCE
FILE
MYSOURCE → **COMPILER** → **MYUSL**
USER
SUBPROGRAM
LIBRARY
FILE
(USL)

:COBOL MYSOURCE,MYUSL

PMAP

SEGMENTER → **MYPROG**
USER
PROGRAM
FILE

:PREP MYUSL,MYPROG

LMAP

SYSTEM
SEGMENTED
LIBRARY (SL)
SEARCHED
BY DEFAULT
SL.PUB.SYS → **LOADER**

ALL EXTERNAL
REFERENCES
MUST BE
RESOLVED FOR
A PROGRAM
TO RUN

**VIRTUAL
MEMORY**

:RUN MYPROG;LMAP

RESOLVING EXTERNAL REFERENCES

UTILIZING AN RL & SEVERAL SL'S

COBOL
SOURCE
FILE
MYSOURCE → **COMPILER** → **MYUSL**
USER
SUBPROGRAM
LIBRARY
FILE
(USL)

:COBOL MYSOURCE,MYUSL

RELOCATABLE
LIBRARY
FILE   (RL)
MYRL → **SEGMENTER** ⇢ **PMAP**

MYPROG
USER
PROGRAM
FILE

:PREP MYUSL,MYPROG;PMAP;RL=MYRL

HIERARCHY OF
3 SEGMENTED
LIBRARY
FILES
(SL'S)

SL ①
SL.PUB ②
SL.PUB.SYS ③
→ **LOADER** ⇢ **LMAP**

**VIRTUAL
MEMORY**

ALL EXTERNAL
REFERENCES
MUST BE
RESOLVED FOR
A PROGRAM
TO RUN

:RUN MYPROG;LMAP;LIB=G

HEWLETT hp PACKARD

LMAP

LMAP—Inventory of external references resolved at :RUN time.

```
:COBOLPREP COBTEST1.PUB,$NEWPASS,$NULL
  ...
:RUN $OLDPASS;LMAP

PROGRAM FILE $OLDPASS..

TERMINATE'        PROG 0   11    1 SSL  0    2   43
C'ACCEPT          PROG 0   10    1 SSL  0   31  165
C'DISPLAY'FIN     PROG 0    7    1 SSL  0   20  165
C'DISPLAY'L       PROG 0    6    1 SSL  0   15  165
C'DISPLAY'INIT    PROG 0    5    1 SSL  0   16  165
C'DISPLAY'ID      PROG 0    4    1 SSL  0   17  165
C'EDITMOVEN       PROG 0    3    1 SSL  0   15  164
C'DISPLAY'FC      PROG 0    2    1 SSL  0   21  165
C'GOTO            PROG 0    5    0 SSL  0   11  164
COBOLTRAP         PROG 0    3    0 SSL  0    6  166
DEBUG             PROG 0    2    0 SSL  0    1   57

301 302

ENTER COST OF SALE (BEFORE TAX) NO DECIMAL PT
             ...                etc.
```

Annotations pointing to columns:
- **FROM**
- **TO**
- **STT#**
- **CODE SEGMENT#**

PMAP

PMAP—Inventory of STT entries (external references and entry points) and which were resolved by :PREP
and which will be resolved at :RUN time.

```
:COBOL COBTEST1.PUB,$NEWPASS,$NULL

:PREP $OLDPASS,MYPROG;PMAP

PROGRAM FILE MYPROG.GSTUDENT.INTRO

COBOLTEST1          0
    NAME            STT   CODE ENTRY SEG
    COBOLTEST1'      1      0    0
    DEBUG            2                ?
    COBOLTRAP        3                ?
    ENTERROUTINE00'  4                1
    C'GOTO           5                ?
    SEGMENT LENGTH         (40)
ENTERROUTINE00'     1
    NAME            STT   CODE ENTRY SEG
    ENTERROUTINE00'  1      0    0
    C'DISPLAY'FC     2                ?
    C'EDITMOVEN      3                ?
    C'DISPLAY'ID     4                ?
    C'DISPLAY'INIT   5                ?
    C'DISPLAY'L      6                ?
    C'DISPLAY'FIN    7                ?
    C'ACCEPT         10               ?
    TERMINATE'       11               ?
    SEGMENT LENGTH         (414)
```

SEGMENT SIZE octal

```
PRIMARY DB          0   INITIAL STACK  1440   CAPABILITY       600
SECONDARY DB      341   INITIAL DL        0   TOTAL CODE       454
TOTAL DB          341   MAXIMUM DATA      ?   TOTAL RECORDS     11
ELAPSED TIME   00:00:08.769              PROCESSOR TIME   00:00.605

END OF PREPARE
```

DEFINITION OF TERMS

CODE SEGMENT

- smallest part of 'programs' kept track of by MPE at :RUN time.

- each is in 'PREPARED' form.

- similar to an 'overlay', but automatically managed by the operating system.

RELOCATABLE BINARY MODULE (RBM)

- smallest entity accessible by the Segmenter at :PREP time.

- each is in object form.

- each has at least one entry point.

HEWLETT hp PACKARD

GENERATING RBM'S AND CODE SEGMENTS

Language	Statement	Starts new RBM	Starts new Code Segment
COBOL	Main Program	Yes	Yes
	SECTION unnumbered	No	No
	SECTION nn*	Yes	Yes
	SUBPROGRAM	Yes	Yes
	$CONTROL SUBPROGRAM	Yes	Yes
	$CONTROL DYNAMIC	Yes	Yes
RPG	Main Program	Yes	Yes
	SUBROUTINE	Yes	No
	$CONTROL SEG= {1}{2}{3}{4}	---	---

NOTE: * Contiguous Sections with the same number go into the Same Code Segment (& RBM), otherwise a new Code Segment is generated.

HEWLETT hp PACKARD

GENERATING RBM'S AND CODE SEGMENTS

Language	Statement	Starts new RBM	Starts new Code Segment
FORTRAN	Main Program	Yes	Yes
	SUBROUTINE	Yes	No
	FUNCTION	Yes	No
	$CONTROL SEGMENT=name*	Yes	Yes
BASICOMP	Main Program	Yes	Yes
	$CONTROL SUBPROGRAM	Yes	Yes
	$CONTROL SEGMENT=name*	Yes	Yes
SPL	Main Program	Yes	Yes
	PROCEDURE	Yes	No
	$CONTROL SUBPROGRAM	Yes	Yes
	$CONTROL SEGMENT=name*	Yes	Yes

NOTES: * Must immediately precede a statement that generates an RBM. In effect for next RBM only, then RBMs revert to going into Main Code Segment.

NET EFFECT

RPG -or- BASICOMP   User has little control over Segmentation with the Segmenter or at the source level.

FORTRAN -or- SPL   Many RBMs generated per Code Segment. Maximum opportunity to perform 'tuning' by moving RBM's between Code Segments with the Segmenter.

COBOL   One additional Code Segment is generated for each Compilation to perform initialization of variables. One RBM generated per Code Segment. Tuning limited to combining Code Segments into bigger Code Segments with the Segmenter.

$CONTROL

LIST / NOLIST

All kinds of lines will be written to 'listfile'.

SOURCE / NOSOURCE

Lists all source records when LIST is also in effect. (Except: COBOL from a Session with 'listfile' to terminal defaults to NOSOURCE)

WARN / NOWARN

List all warning messages. NOLIST does NOT suppress warning messages.

MAP / NOMAP

Prints Symbol Table Map if LIST is in effect.

CODE / NOCODE

Prints a listing of Object Code if LIST is in effect.

LINES=nn

Specifies lines to be printed on one page in 'listfile' (default = 32767 for terminal; else 60).

USLINIT

Initializes the 'uslfile' to empty status prior to generation of object code.

LIBRARY FILES

- Includes USL (User Subprogram Library), RL (Relocatable Library), and SL (Segmented Library) files.
- Each has a DIRECTORY.
- About 10% of space is reserved for the directory.
- File statistics stated in octal words by Segmenter.

USL (User Subprogram Library)

—Contains RBMs in object form.

—Several compiles may put object output into the same USL.

—A '$CONTROL USLINIT' statement in any compilation will initialize whole USL.

—Versions of RBMs maintained (you get latest active by default).

—A compilation will deactivate previous RBMs with the same names as the ones it creates. Version numbers will also be updated.

LIBRARY FILES

RL (Relocatable Library)
- —Contains RBMs in object form.
- —All code included from a RL during :PREP goes into the same Code Segment.
- —Multiple Users of same program will share this Code Segment.
- —Other programs using routines in this RL Segment will NOT be able to use that Code Segment. They will each have their own distinct copy of the routine in their own Code Segment.

SL (Segmented Library)
- —Contains Code Segments in 'PREPARED' form.
- —Many programs may share the same Code Segment.
- —Parameters must be passed on the 'Top of STACK' since it is the only area common to all routines that may potentially use SL Code Segment.

USES OF THE SEGMENTER

- Primary use is to maintain RL and SL files.
- Adding code to RLs or SLs can only be accomplished by the Segmenter and only from USL files.
- Segmenter may also be used to re-combine RBMs within Code Segments for 'tuning' of programs.

USING THE SEGMENTER

- To operate on a library file, user must first point to it or build it with the Segmenter.
- An operation using a RL file and a USL file will automatically use the ones most recently pointed to.
- Notice no provision in command syntax to specify files to be used in Copy, List or Purge operations.

USING THE SEGMENTER

```
:COBOL COBTEST1.PUB,$NEWPASS,$NULL
   . . .
:SEGMENTER

SEGMENTER SUBSYSTEM (C.0)
-USL $OLDPASS
-BUILDRL MYRL,30,1
-LISTUSL

USL FILE $OLDPASS..
```

```
►COBOLTEST1
     COBOLTEST1'◄      30   OB A C N
►ENTERROUTINE00'
     ENTERROUTINE00'◄  377  P  A C N R
```

< *Code Segment*

< *RBM*
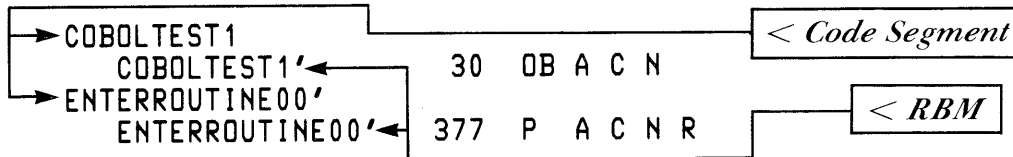
```
FILE SIZE         377600
DIR. USED            133        INFO USED            676
DIR. GARB.             0        INFO GARB.             0
DIR. AVAIL.        37245        INFO AVAIL.       337102
-LISTRL

RL FILE MYRL.GSTUDENT.INTRO

* ENTRY POINTS *

* EXTERNALS *

USED                 400        AVAILABLE           7000
-EXIT

 END OF SUBSYSTEM
:
```

SEGMENTER MAINTENANCE FUNCTIONS

	U S L	R L	S L
Build a library file	BUILDSL	BUILDRL	BUILDSL
Point to an existing library file	USL, AUXUSL	RL	SL
C O P Y: RBM's from AUXUSL to USL RBM's from USL to RL Code Segments from USL to SL	COPY	ADDRL	ADDSL
List library files	LISTUSL	LISTRL	LISTSL
P U R G E: Entry Point RBM Code Segment	----- PURGERBM UNIT PURGERBM SEGMENT	PURGERL ENTRY PURGERL UNIT -----	PURGESL ENTRY ----- PURGESL SEGMENT

COPY OPERATIONS ARE ONE-WAY

AUXUSL → COPY → USL

USL → ADDRL → RL

USL → ADDSL → SL
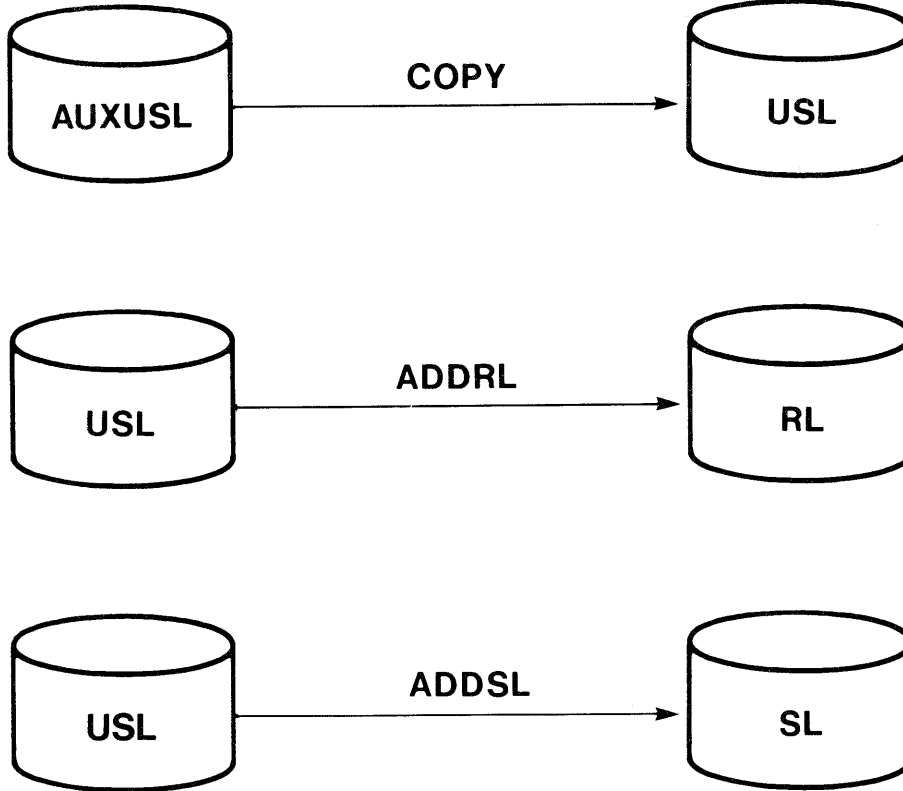
MINIMIZING THE IMPACT OF CHANGE

- Subroutines that are going to change can be separate compilations. Just re-compile them & re-PREP (assuming you have saved USL).

- Routines common to MOST programs can be put in a SL.

- Routines in a SL may be changed without having to re-compile or re-PREP programs that use them.

- Routines common to MANY programs can be put in either a RL or a SL.

- Routines in a RL may be changed without having to re-compile programs that use them; they will have to be re-PREP'd, however.

READING ASSIGNMENT

software tips
about the HP 3000

FCOPYing Files to Magnetic Tape

When FCOPYing a file to magnetic tape, the tape device does not rewind until the next FCOPY command is entered. If the next command does not append to the current tape file, FCOPY writes an EOF on the tape and rewinds it. Do not manually rewind or dismount the tape before entering another FCOPY command. If you do, the tape will not contain a proper EOF, and your Job/Session will wait for the tape drive to become ready so that FCOPY can write the EOF. While your Session is waiting, the terminal is locked out. If someone else mounts a tape with a write ring on your tape unit, they may find to their dismay that FCOPY has written an EOF on their good tape.

To free the terminal, mount a scratch tape with a write ring on the tape unit owned by FCOPY. If you have already entered another FCOPY command, or attempted to abort your Session, FCOPY will write an EOF on the scratch tape and rewind it. Your terminal should become available for further use. To obtain a tape with a valid EOF, re-do the previous FCOPY function(s) and allow FCOPY to rewind the tape for you.

Sam Boot
HP General Systems

SEGMENTATION FOR MAXIMUM EFFICIENCY OF SYSTEM-TYPE PROGRAMS

The purpose of this article is to describe, for the benefit of system programmers, some guidelines for the optimum design of programs for the 3000; in particular, attention will be given to the questions of segmentation.

The 3000 is a process oriented machine, incorporating the separation of code and data, and stack architecture. This permits easy design of re-entrant code. The purpose here is to discuss ways of making a particular process

a. Run as fast as possible

b. Have minimum effect on other processes in the system.

As more and more load is applied to a machine like the 3000, a point is reached where all users experience a very rapid deterioration of service. This corresponds to a kind of 'overload' condition where the system is working harder to switch from job to job than running your programs. The size of memory is the primary determinant of this point, but given a fixed memory size, the size of your programs and the quality of this segmentation have a strong influence on the work the machine will accept before overloading.

Process Environment

When you write a program, it is executed by MPE in the form shown in Figure 1. The process has a single data segment (or "stack") and a variable number of code segments of varying sizes. When you write your program you can control:

a. the size of the stack

b. the number of your code segments

c. the size of each segment

d. which code goes into which segment.
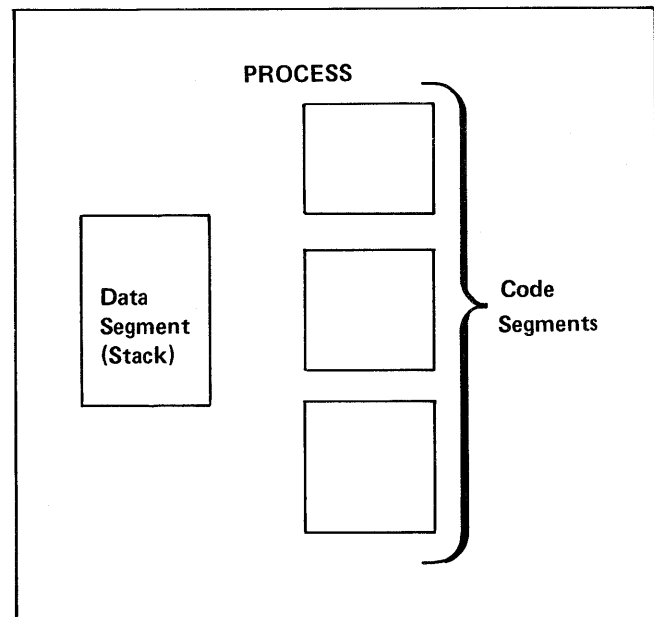
Figure 1.

The diagram shown above is actually a simplification since it does not show the externals referenced by your program (see Figure 2). If for example, your SPL-written program calls FOPEN, then a link will be created from your code to an MPE segment containing the FOPEN intrinsic code. Most of these intrinsics and all the Compiler Library routines are not in memory permanently, thus they are viewed by MPE as code segments identical to your own even though they were not written by you. For programs written in SPL, you are in control of which external procedures are called, since the calls are made explicitly. For other languages, the compiler will implicitly create in your program calls to external routines in order to perform, for example, a Fortran WRITE or a COBOL DISPLAY. The environment of a non-SPL program is harder to control because it requires a knowledge of when the compiler will

READING ASSIGNMENT

PROCESS

Data Segment (Stack)

Your

Code

Segments

Code Segments belonging to MPE. (Intrinsics, Library Routines, Language, Run-time Routines, etc.)
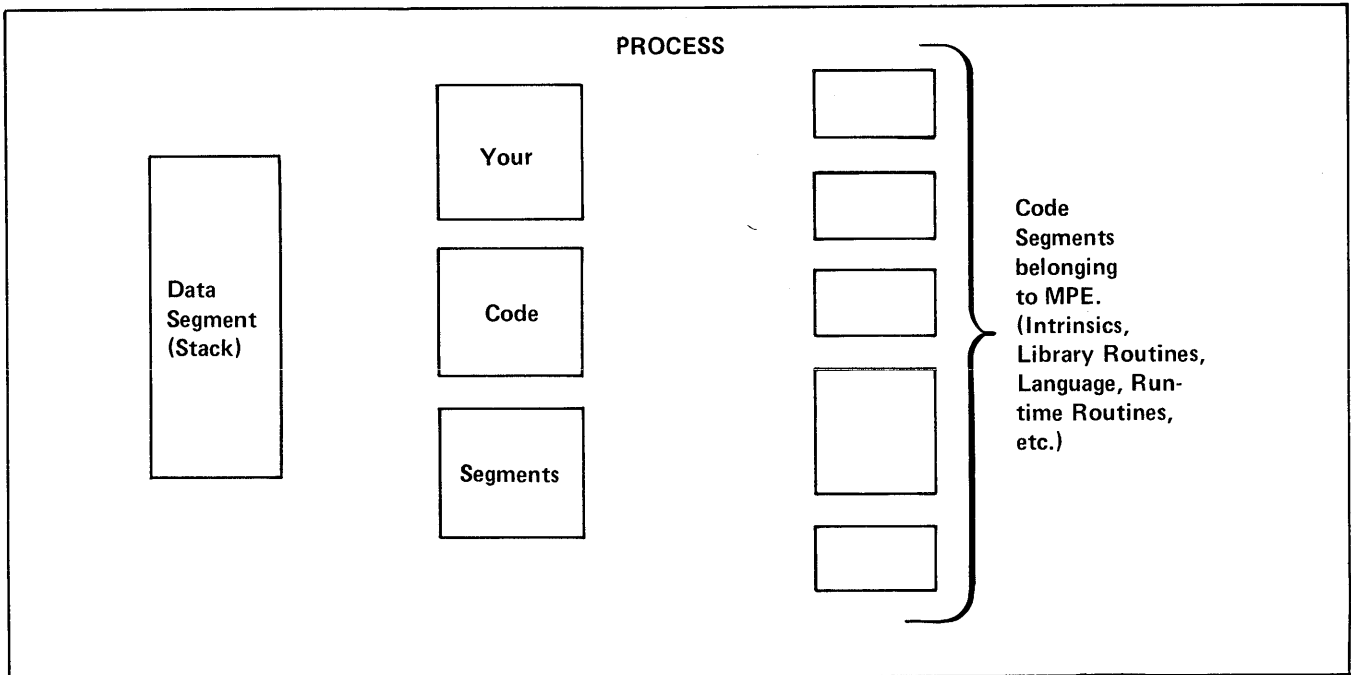
Figure 2.

emit those external calls. We will limit this discussion to those areas over which you have primary control: your own program code and data stack. Given any language, there are some fundamental principles to follow which will decrease the run-time of a process and its impact on system load.

How to Determine a Program Environment

When you prepare your program the PMAP option will show you the size of each segment, which procedures are in which segment, and the names of externals called by each segment. The MPE manual describes the format of the PMAP in detail.

How MPE Runs Your Program

There are two MPE modules concerned here — the dispatcher and the memory management system. The dispatcher is responsible for the allocation of CPU time to all the executing processes. The memory management system has the job of fitting code and data segments into memory as they are required, this operation often requiring the decision of which segment(s) to delete to make space. When your time-slice starts, the stack is made present in memory and control is passed to the program. As the program proceeds, it will call procedures which are not in the current segment. At this point your program is suspended while MPE arranges to make the required segment present. This can take from 20 to 100 milliseconds since a disc access is involved. While this is going on the dispatcher tries to run the process with the next highest priority which is already resident in memory. When the destination segment has been made present, control is passed to the procedure originally called.

The point to note here is that calling a procedure in an absent code segment is a time-consuming job.

How Do I Tell If A Segment Will Be Present?

You can't for sure. The memory management system will attempt to keep the most popular segments in memory, and the system is aware, using an internal table, which segments you use most in your process. Using this information the system will postpone for as long as possible disrupting your process, but in a busy system it is very difficult to predict the state of memory.

Rules for Segmenting Your Program

Rule No. 1

Minimize the number of times the program crosses a segment boundary. In other words, stay within a segment for as long as possible. When you leave it, stay out for as long as possible.

Design of Programs is Important

Do not leave segmentation to the last minute. As will be shown below, it is possible to write programs that cannot be correctly segmented.

Any procedure or outer block Relocatable Binary Module (RBM) must reside wholly within a segment. Thus if it proves necessary to move a block of code into a separate segment, it will only be possible if the code is a procedure. You cannot take an arbitrary set of instructions and place them into a named segment — the whole RBM must be moved. Therefore, the way you divide your program into procedures is vitally important in the design phase.

Concept of Locality

The locality of a program is the degree to which control

READING ASSIGNMENT

remains in the same general area of code. A high locality means that control remains in the same area for a long period of time. Poor locality means the program branches wildly and rapidly all over the place. The 3000 needs programs that have good segment locality but does not care about the degree of locality within any given segment. That is to say, it does not want programs that jump from segment to segment continuously but once inside any given segment, it doesn't matter what the locality is like.

Functional vs. Temporal Segmentation

Intuitively, one segments according to the function of the procedures. That is, all the command decoding routines are put together, the command executors are put together, etc. This is wrong. Wrong. Segmentation is a speed-enhancing operation thus time, not function, is the critical dimension. Since Rule No. 1 says stay inside a segment for as long as you can, control must pass smoothly from segment to segment as the program progresses.

As an example, consider a small utility program which dumps a file to the line printer in some special format. Let

us suppose that the operator can choose the name of the file and which of three possible formats to use. The program is written with four procedures: A, B, C, and D.

Let us further suppose that each dump routine has a procedure to fetch a record from its file and a procedure to format a print line:

It would be tempting to put all the formatting routines in one segment, and the record fetching routines in another. This would cause a segment boundary to be crossed twice for every record dumped — perhaps a thousand times. The correct way is to put B1B2 together, C1C2, etc. If A is in its own segment then only three segment boundaries are crossed for a whole dump. In a busy system this simple change could make large differences in the run time of your program.

To sum up, estimate the number of times a segment boundary is crossed in your program and multiply this by 40 milliseconds (12 msec if you have a swapping disc and your program resides on it). This is the time your program will be doing no useful work and other processes will be disrupted.

Ask user for file name and dump format (A)

Open file

OK? — NO error msg

Choose dump routine

Procedure A talks to operator (might be the outer block).

Procedures B,C,D, produce the dumps.

(B) Format 1

(C) Format 2

(D) Format 3
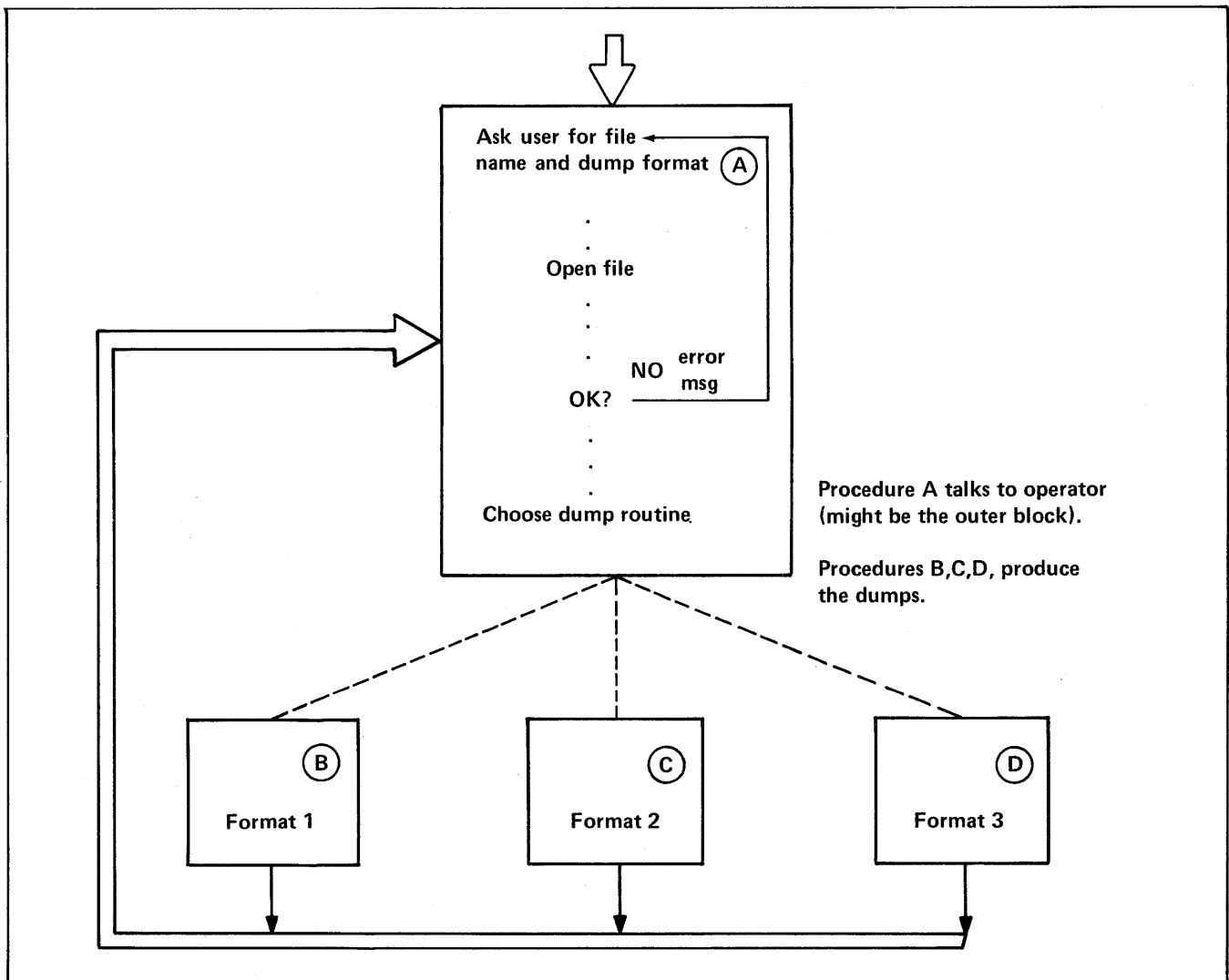
Figure 3.

READING ASSIGNMENT
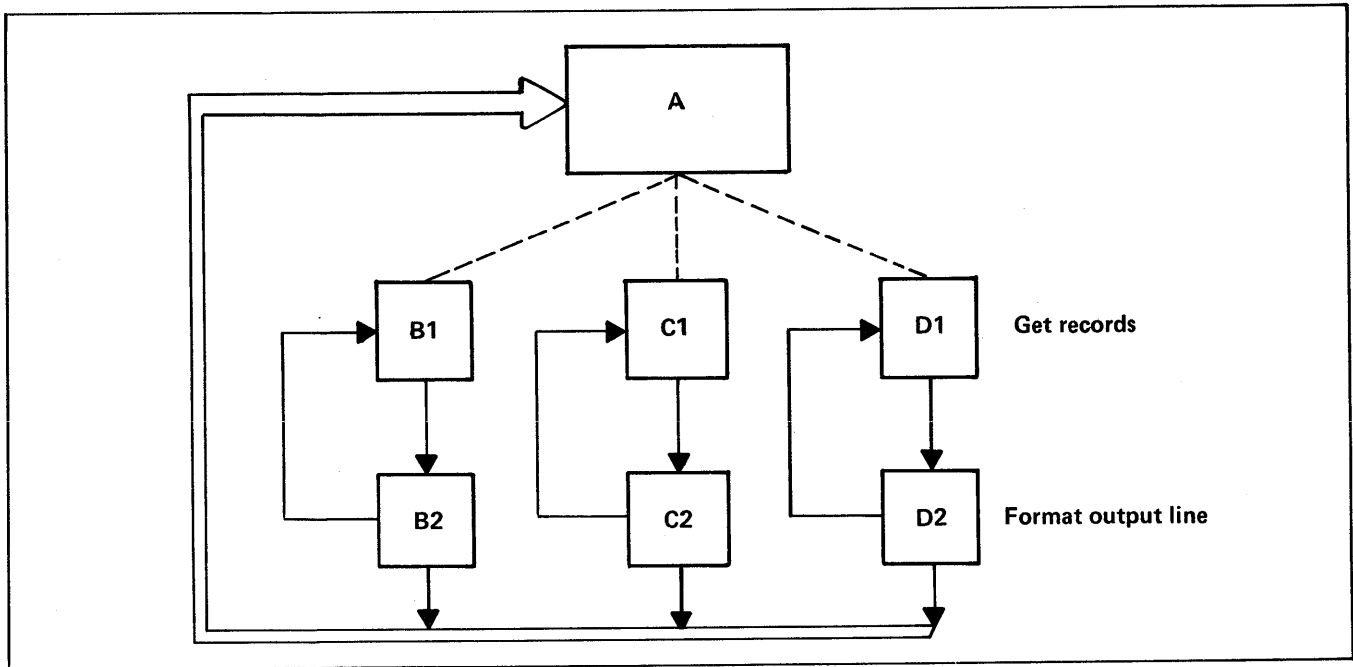
Figure 4.

Rule No. 2

Do Not Burden Your Working Set With Infrequently Used Code

Let us suppose that you have arrived at some segmentation scheme using the above rule so that you have good segment locality. The next step is to reduce the size of the 'working set'.

Frequency of Code Use

The 'working set' of segments is the set that consumes most of the CPU time. For example in the program above the working set is the code that executes the main loop such as C1C2. Let us assume that C1C2 are in a segment of their own called CSEG. The system may spend minutes in this segment for a large dump. It is important therefore to minimize its size in order to reduce contention for the scarce memory resource.

To do this, examine the codes in the working set and remove any code that exectues infrequently. Very often, this applies to code which does error-handling. When your program detects an error, do not handle it in-line. Write an error-message generating procedure and call it with a parameter indicating which message to output. This can be put in a separate segment and thus not clutter up memory while doing normal error-free processing. As another example, suppose that in the program mentioned above, after doing an FWRITE, you check the condition code for end-of-file and, if required, execute a somewhat elaborate sequence to extend the file by building a new one and copying the old into it and then purging the old file. If this condition is likely to occur once in every 500 runs, why hold it in precious memory with the working set? Banish it to some auxiliary segment and let MPE bring it in only when needed. Remember that you can only move this code if it is a procedure.

WRONG	RIGHT
FWRITE(. . .);	FWRITE(. . .);
IF>THEN	IF>THEN EXTEND'
BEGIN	FILE;
.	
.	
<<LENGTHEN FILE>>	Procedure EXTEND'FILE
	is put in another segment
.	
.	
END;	

Segment Sizes

This is a trade-off. If you segment into many small segments, each one has to be separately read into memory before your program can begin execution at the start of a time-slice. (Segments are in fact only read in when actually referenced, but a program with dozens of small segments is likely to need several of them before any real work can be done). This leaves less of the time-slice for useful work.

At the other end of the spectrum, a program with a few large segments will take up a lot of memory — perhaps unnecessarily. Segments should be typically around 3K decimal words, but if you have lots of memory and are nowhere near the machine overload point, larger segments may enhance throughput slightly. Such a strategy may cause trouble later however when machine load increases.

How Many Segments

As a guide, a program is getting large at 10 segments or so. A typical compiler has around 25 while a small utility like SORT has about 3. There is a hardware limitation of 63 code segments in a process.

READING ASSIGNMENT

Rule No. 3

Make segment as small as possible with a maximum of about 3K decimal words.

Rule No. 4

If Rule 3 has to be violated in order to reduce the number of segments, keep principal working sets small and make infrequently used segments large.

If Your Code is Shared

If your program is going to be run from multiple terminals then the code segments will automatically be shared by the multiple processes. Each process will have its own stack of course. If your program design requires data which is never altered such as error messages, look-up tables, etc., then by placing them in the code rather than the stack, only one copy is required for all processes.

WRONG

```
BEGIN
BYTE ARRAY MESSG (0:22):="TOO
MANY TIMES ENTERED";        Global Declara-
                            tions


.
.
.


PROCEDURE MESSOUT;   Procedure to print error
BEGIN                message
PRINT (MESSG,–23,0);
```

```
:
END;
:
END.
```

WHY WRONG?   The array MESSG is present in the stack perpetually. Each process running this program carries the message string around in its stack.

RIGHT

BEGIN   MESG only exists while MESSOUT
.   executes. SPL will store the string
.   in quotes in the code segment –
.   effectively making it shared. The
.   stack is now smaller.
.

.
```
PROCEDURE MESSOUT;
BEGIN
BYTE ARRAY MESG(0:22);
MOVE MESG:="TOO MANY VALUES ENTERED";
PRINT (MESG,–23,0);
END;
.

.

.
END.
```

Rule No. 5

In SPL, keep initialized variables, especially arrays, out of the GLOBAL DECLARATIONS.

In Fortran, infrequently used variables and arrays should not be initialized in DATA statements.

HEWLETT hp PACKARD

SEGMENTER LAB # 1 [1.0 hour]

Please read the entire lab before proceeding!

Create an SL

1)   Compile the COBOL program VALIDNO.PUB into a USL file by itself. You can guarantee this by specifying a 'uslfile' of '$NEWPASS'.
2)   Invoke the Segmenter.
3)   Point to $OLDPASS as the 'uslfile'.
4)   Build an SL file called 'SL' in your group 20 records long in 1 extent.
5)   List the SL file—it should be empty (about 10% of its space will be reserved for the directory, however).
6)   List the USL file.
7)   Use 2 ADDSL commands to copy both Code Segments in the USL into the SL (COBOL generates an additional initialization code segment for each code segment normally generated).
8)   List the SL to make sure both segments are there.
9)   Exit the Segmenter.

Create an RL
10)   Compile the COBOL program DISCIO.PUB into $NEWPASS (specify $NEWPASS to make sure the object output of this compile goes into a different USL file than the previous one).
11)   Invoke the Segmenter.
12)   Point to $OLDPASS as the USL file.
13)   Build an RL file called 'SEGRL' in your group 30 records long in 1 extent.
14)   List the RL file—it should be empty but have space reserved for the directory.
15)   List the USL then use 2 ADDRL commands to copy RBM's DISCIO' and DISCIO into the RL file.
16)   List the RL file to make sure both RBM's are there.
17)   Exit the Segmenter.

HEWLETT **hp** *PACKARD*

SEGMENTER LAB # 1

Use the RL and SL

18)   Compile the COBOL program 'INVUPD.PUB' into $NEWPASS.
19)   Invoke the Segmenter.
20)   Point to $OLDPASS as the USL file.
21)   Prepare this USL creating a program file 'SEGRUN'. Obtain a PMAP, use MAXDATA=10000, and use the RL file you created in steps 10 through 17.
22)   Exit the Segmenter.
23)   Run 'SEGRUN'. You should get a LOAD ERROR 201,27 Unresolved Prog External VALIDNO.
24)   Run 'SEGRUN' again, this time obtaining an LMAP and specifying LIB=G to use your group SL you created in steps 1 through 9.
25)   Enter a '/' to exit program.

Problems you may encounter.

          If you repeat steps, be careful. When you build a library file with the Segmenter, it builds it as 'NEW' with a close disposition of 'SAVE'. This means it is possible to create a second library file with the same name and you will only learn of the conflict when you close the newly created file, normally as you exit the Segmenter. If you already have a file of that name, point to it, don't create another one.
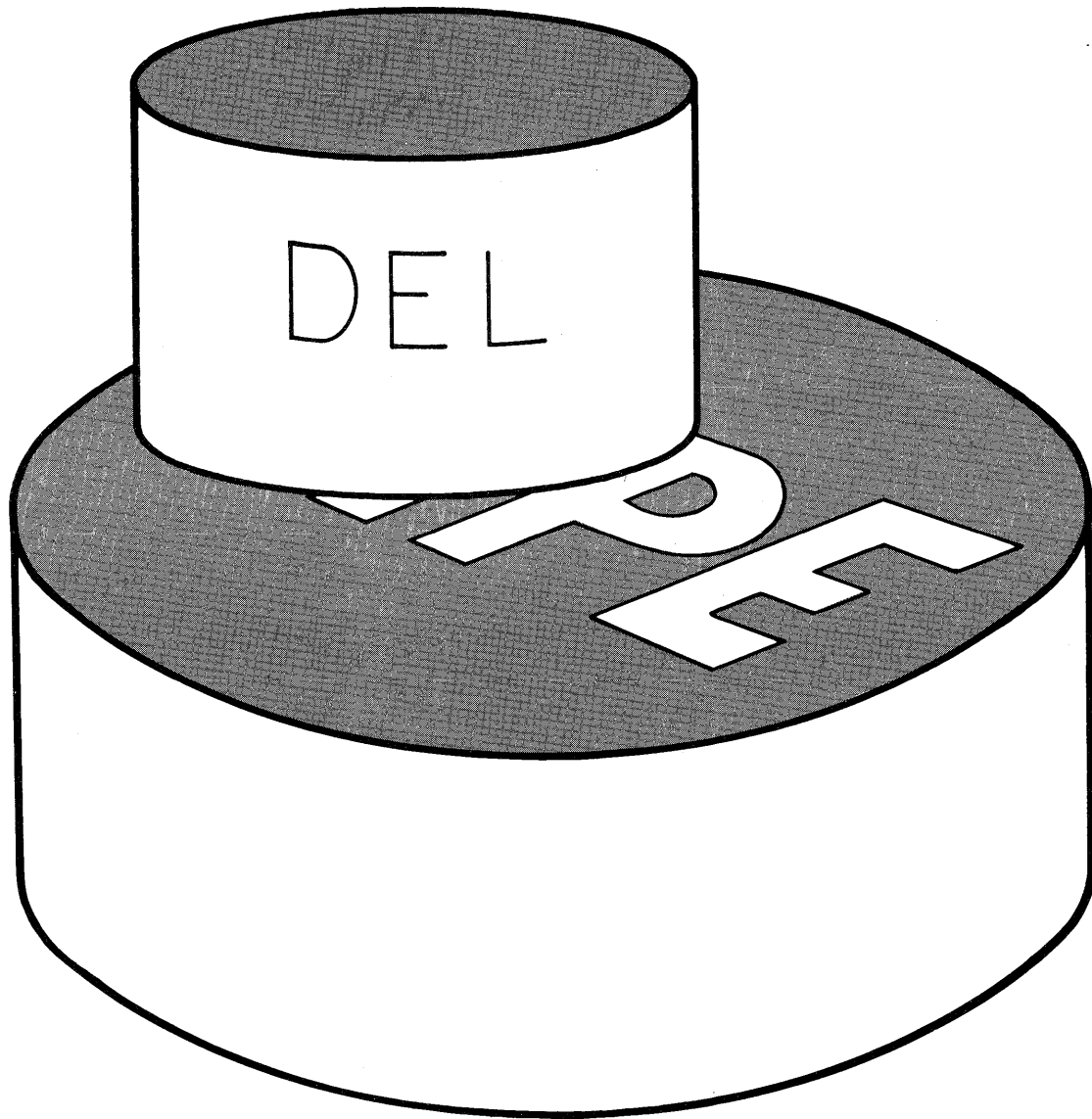
DEL/3000

- DEL is a set of software tools to ease the writing of data entry programs to utilize the HP-264x family of terminals.

- Frees programmer from having to hard code forms into programs.

- Forms are created and maintained independently of Entry Programs.

- DEL is N O T a complete data entry system by itself.
  —Programmer must supply Error Correction & Batch Total logic.
  —Programmer must supply Logic for Validation against Master files and Posting to Data Base.
  —No interface provided to use tape cartridges.

- Accessible from COBOL, FORTRAN, BASIC or SPL.

- Available on all 3000's but CX and prior models.

DEL FORM FILE

• DEL Form File contains many forms and a Directory.

• Each form in the form file contains:
 —Your form as you constructed it on the terminal screen.
 —Control Characters to establish proper environment when form is written to terminal.
 —Optionally — Edit Specifications the program may use to call DEL Edit Procedures to edit input
 data.

```
DIRECTORY

FORM1

              FORM AS READ FROM TERMINAL SCREEN

   FORM

     CONTROL CHARS          EDIT SPECIFICATIONS —
     ADDED BY DEL           AT LEAST ONE PER FIELD
                               (72 CHARS EACH)
```
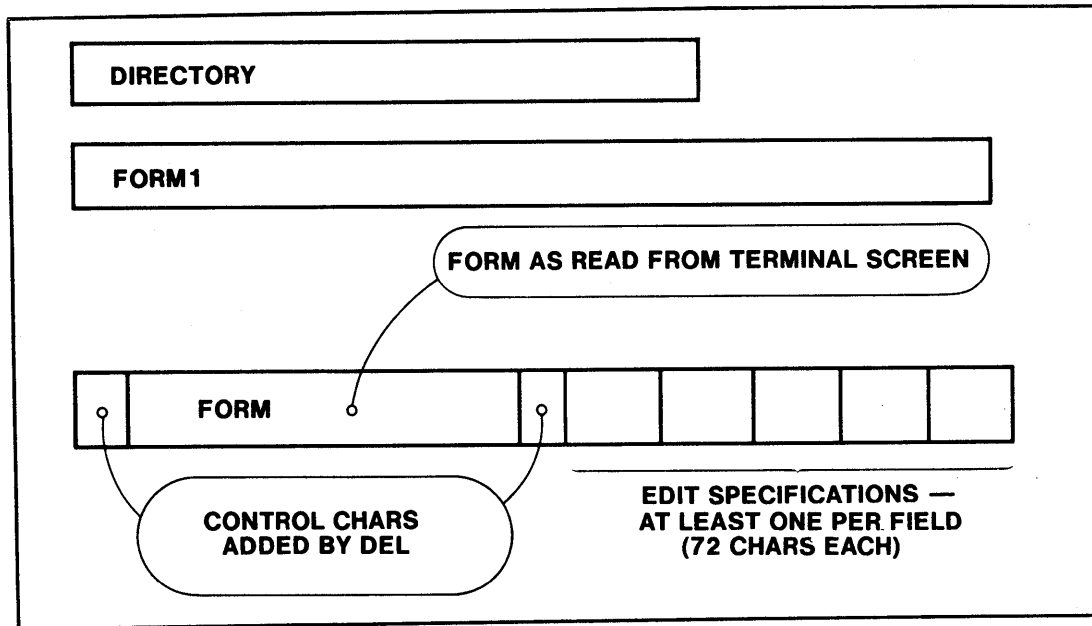
ENVIRONMENT

- DEL Requires a minimum of 4K of terminal memory.

- FORMAT MODE — all data assumed to be protected (i.e. may NOT be altered by the operator). Only fields specified as 'UNPROTECTED' may have data entered into them & are transferred between terminal and computer. TAB to beginning of next unprotected field.

- BLOCK MODE — data read a screen-full at a time, not character by character. 'ENTER' key used instead of 'RETURN' key. May use the EDIT and DISPLAY group keys.

- BLOCK MODE / PAGE — all unprotected fields read with one hardware read; protected areas not read.

- BLOCK MODE / LINE — one unprotected field read with each hardware read; protected areas not read. DEL automatically issues number of reads required to read all unprotected characters on screen. All terminals operate in this mode on Series I.

BLOCK MODE SETTINGS FOR TERMINALS

	2640A	2640B,2644	2645/48/41
Defaults to BLOCK MODE / LINE	X	X	
May be physically 'strapped' to use BLOCK MODE / PAGE		X	
Will be set by DEL to use BLOCK MODE / PAGE			X

HEWLETT *hp* PACKARD

MAINTAINING DEL FORMS

FORMAINT — A utility program to create, display, and modify forms.

:RUN FORMAINT.PUB.SYS

```
HP32206A.01.07.FORM MAINTENANCE  (C) HEWLETT-PACKARD CO. 1976

Enter the name of your form file here  [                              ]
and select one of the following functions by entering an X in front of the
desired function.

 ▨ DEFINE A NEW FORM

 ▨ LIST FORM FILE DIRECTORY

 ▨ MODIFY AN EXISTING FORM

 ▨ DISPLAY AN EXISTING FORM

 ▨ DELETE AN EXISTING FORM

 ▨ DELETE THE FORM FILE

 ▨ EXIT FORMAINT
```

SPECIFYING DEL NAMES

FORM FILE NAME is a MPE filereference. FORMAINT requires Read & Write access. (Each name may be a max of 8 A / N chars; 1-st must be alpha; 35 char max)

FORMS

FINANCE / MONEY.PUB.INTRO

DEL FORM NAME (any 16 characters)

ACCOUNTSREC

2GENERAL—LEDGER

@ #!any-chars%*

CREATING A NEW FORM

Enter the name of your form here [   ]

If this form is a member of a series of forms enter the
name of the next form in the series [   ]

- You will be presented with a blank screen on which to enter your form.

- Indicate end of form with Record Separator (CNTL circumflex).

EDIT SPECIFICATIONS

```
       MY FIRST DEL FORM -- DEL LAB #1
       FIELD1: ░░░░░░░░░░
               ^

  If no editing is required or all edits for this field have been specified
enter an X here ░

  The edit procedure name is ░░░░░░░░░░░░░░░░.

  Test flag #░ before performing edit.  After edit set flag #░ and it must be
the same as flag #░ or opposite flag #░

  For range check editing the low value is ░░░░░░░░░░░
                       and the high value is ░░░░░░░░░░░

  For file look-up procedures the file name is ░░░░░░░░░░░░░░░░░░░░░░░ .

  If the edit is not a range test nor a file look-up you may enter up to 32
characters in this space ░░░░░░░░░░░░░░░░░░░ for use by the
edit procedure.
```

- Press f8 (CNTL f8 on 2640A or B) for first edit if you want no edit specifications included in DEL form file.

- DEL Edit Procedure Names (COBOL versions are prefixed with 'C'):
  - 'ALPHAEDIT'   only alpha chars (A-Z and a-z).
  - 'ALPHAFILL'   alpha with trailing blanks.
  - 'ANEDIT'   alphanumeric & embedded spaces (no special chars).
  - 'NUMRCEDIT'   only numeric (0-9).
  - 'ZEROFILL'   numeric, signs, trailing & leading spaces.
  - 'NRANGE'   first ZEROFILL then check against range of values.
  - 'M11CREATE'   first ZEROFILL then append mod-11 check digit.
  - 'M11VERIFY'   first ZEROFILL then verify last digit.
- EDIT FLAGS — 16 1-bit flags in DEL communications area that can be set and tested by Edit Procedures.
  - —Helpful when multiple edits performed on same field.
  - —Helpful when editing several fields that are logically connected.

LIST FORM FILE DIRECTORY

- Contents of Form File Directory listed on terminal.

FORM NAME	CREATION DATE	TIME	FORM NAME	CREATION DATE	TIME
DELEDIT	6/30/77	10:19	TEST-CNRANGE	12/21/77	16:57
OLDCDELE	6/30/77	10:19	DELEDITS	12/28/77	14:36
CDELEDIT	12/28/77	14:47	!@ANY-16-CHARS'"	12/28/77	14:51
SAMPLE-PROGRAM	12/28/77	15:01	ANYCHARS	12/28/77	14:51
SAMPLE-#007	12/28/77	15:01	NOTHING-REALLY	5/19/78	17:26

HEWLETT hp PACKARD

MODIFY AN EXISTING FORM

- All Edit Specifications are erased and must be re-specified.
- Form presented to you for modification.
- You must re-write Record Separator delimiting end of form.

```
Enter the name of the form to be modified ▓▓▓▓▓▓▓▓▓▓▓▓ .

If the form to be modified is a member of a series of forms and the name of the
next form in the series is to be changed enter the name of the next form in the
series ▓▓▓▓▓▓▓▓▓▓ .
```

DISPLAY AN EXISTING FORM

- If no destination file is specified, only form will be displayed on your terminal.
- If a destination file is specified, Form Descriptive Information, the Form, and the Edit Specifications will be listed.

```
Enter the name of the form to be displayed ▓▓▓▓▓▓▓▓▓▓ .

If you want the edit specifications displayed enter the name of the destination
file ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓ .
```

HEWLETT hp PACKARD

DISPLAY TO A DESTINATION FILE

FORM DESCRIPTIVE INFORMATION

```
   FORM NAME IS DELEDIT              CREATED   6/30/77  10:19
THIS FORM CONTAINS 9 INPUT FIELDS, TOTAL INPUT LENGTH IS  81 BYTES.
THERE ARE 8 EDITS SPECIFIED.
```

FORM

```
   DEMONSTRATE ALL DEL/3000 EDIT PROCEDURES.


                       / = END PROGRAM.
   FUNCTION            ! = RE-DISPLAY PREVIOUS DATA BUFFER AFTER BEING
                           PROCESSED BY EDIT PROCEDURES.
                       ALL OTHERS = READ & PROCESS NEXT RECORD.

ALPHAEDIT                  (A-Z)

ALPHAFILL                  (A-Z WITH TRAILING SPACES)

ANEDIT                     (A-Z, 0-9, SPACES -- IMBEDDED SPACES ALLOWED)

NUMRCEDIT                  (0-9)

ZEROFILL                   (0-9, TRAILING & LEADING SPACES)

NRANGE                     (ZEROFILL THEN RANGE TEST)

M11CREATE                  (ZEROFILL THEN INSERT CHECK DIGIT AT END)

M11VERIFY                  (ZEROFILL THEN CHECK RIGHT-MOST DIGIT)
```

INPUT EDIT SPECIFICATIONS

```
   FIELD LOCATION    FIELD    NUMBER
  ROW  COL IN INPUT  LENGTH  OF EDITS
   4    13      1       1       0

   FIELD LOCATION    FIELD    NUMBER
  ROW  COL IN INPUT  LENGTH  OF EDITS
   8    13      2       10      1

PROCEDURE NAME   FLAGS TEST SET SAME OPPOSITE PROCEDURE DATA
ALPHAEDIT                0    0   0      0

   FIELD LOCATION    FIELD    NUMBER
  ROW  COL IN INPUT  LENGTH  OF EDITS
  10    13      12      10      1

PROCEDURE NAME   FLAGS TEST SET SAME OPPOSITE PROCEDURE DATA
ALPHAFILL               0    0   0      0

   FIELD LOCATION    FIELD    NUMBER
  ROW  COL IN INPUT  LENGTH  OF EDITS
  12    13      22      10      1

PROCEDURE NAME   FLAGS TEST SET SAME OPPOSITE PROCEDURE DATA
ANEDIT                  0    0   0      0

   FIELD LOCATION    FIELD    NUMBER
  ROW  COL IN INPUT  LENGTH  OF EDITS
  14    13      32      10      1

PROCEDURE NAME   FLAGS TEST SET SAME OPPOSITE PROCEDURE DATA
NUMRCEDIT               0    0   0      0

   FIELD LOCATION    FIELD    NUMBER
  ROW  COL IN INPUT  LENGTH  OF EDITS
  16    13      42      10      1

PROCEDURE NAME   FLAGS TEST SET SAME OPPOSITE PROCEDURE DATA
ZEROFILL                0    0   0      0

   FIELD LOCATION    FIELD    NUMBER
  ROW  COL IN INPUT  LENGTH  OF EDITS
  18    13      52      10      1

PROCEDURE NAME   FLAGS TEST SET SAME OPPOSITE PROCEDURE DATA
NRANGE                  0    0   0      0        0000000000    0000009999
```

HEWLETT hp PACKARD

OTHER FORMAINT FUNCTIONS

DELETE AN EXISTING FORM

Enter the name of the form to be deleted▓▓▓▓▓▓▓▓ .

DELETE THE FORM FILE

Is form file FORMS to be deleted?
Enter YES or NO ▓▓

EXIT FORMAINT

You may return to the function 'menu' by pressing CNTL f8 on
2640A or B; f8 by itself on all other models.

DEL LAB # 1 [0.4 hour]

Issue a ':REPORT' command and check the amount of disc space available in your group. The Form file you are about to create will occupy about 300 sectors; make sure there is room for it.

Using FORMAINT, create a very simple form containing a title and two 10-char fields. Call the first field 'FIELD1:' and specify 'ANEDIT' for it. Call the second 'FIELD2:' and specify 'NRANGE' of 1 to 99999 for it. Use a Display Enhancement of 'J' for all unprotected fields. After you have built your form, :RUN CRUNFORM.PUB which will actually run your form and allow you to test the edits. If you have problems, the hints on the next page may help. Build your form following these directions:

To create an UNPROTECTED field:
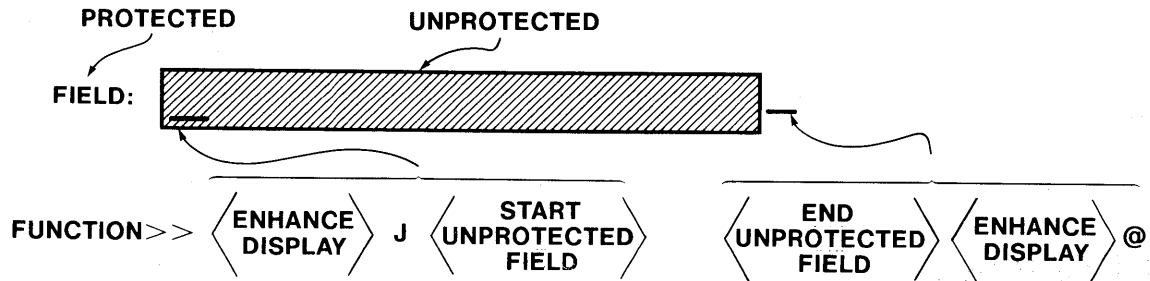
FUNCTION>>	<ENHANCE> DISPLAY	J	START <UNPROTECTED> FIELD	END <UNPROTECTED> FIELD	<ENHANCE> DISPLAY	@
2640A or B	f1	J	f2	f3	f1	@
other 264x	CNTL f1	J	CNTL f2	CNTL f3	CNTL f1	@
Escape Sequence	ESC &d	J	ESC [	ESC]	ESC &d	@

To correct an incorrectly entered unprotected field:
    1) Backspace into the previous protected area.
    2) Press ESC K (erase to end of line).
    3) Re-enter remainder of line.

Indicate end of your form with a Record Separator (CNTL circumflex).

DEL HINTS

To UNLOCK a terminal after an abort out of FORMAINT or a DEL program:

2640 & 2644

1) Unlatch BLOCK MODE
2) RESET TERMINAL
3) ESC : (turn echo on)

other 264x

1) Hard TERMINAL RESET (press RESET TERMINAL twice within 1/2 second).
2) ESC : (turn echo on).

Display Enhancements:	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Half-Bright									X	X	X	X	X	X	X	X
Underline					X	X	X	X					X	X	X	X
Inverse Video			X	X			X	X			X	X			X	X
Blinking		X		X		X		X		X		X		X		X
End Enhancement	X															

DEL LAB # 2 [0.6 hour]

Create the following form in the same form file as DEL LAB # 1 using FORMAINT. Then use CRUNFORM.PUB to check your edits. Remember: If you have to go back and MODIFY your form or edits, you will have to re-specify ALL edits!

PETTY CASH REQUEST

TO: [   ]   DATE: [ ] / [ ] / [ ]

DESC: [   ]   ACCT: [ ]   LOC: [ ]   AMOUNT: [ ].[ ]

EDIT SPECIFICATIONS FOR DEL LAB # 2

FIELD	LENGTH	TYPE OF EDIT	RANGE
TO	20	ANEDIT	
DATE			
MONTH	2	NRANGE	1 to 12
DAY	2	NRANGE	1 to 31
YEAR	2	NRANGE	78 to 84
DESC	26	no edit	
ACCT	4	ZEROFILL	
LOCATION	4	ZEROFILL	
AMOUNT			
DOLLARS	2	NRANGE	0 to 49
CENTS	2	NRANGE	0 to 99

DEL LAB #3 [0.5 hour]

OPTIONAL — Proceed only if time permits.

:RUN CRUNFORM.PUB and use FORMS.PUB for your forms file. There are two forms to be used, DELEDITS which contains an example of all non-COBOL edits & CDELEDIT which contains an example of COBOL edits. Enter data into each to become familiar with the operation of all the standard DEL Edit Procedures. Enter some signed numbers into the 'ZEROFILL' & 'NRANGE' fields and re-display valid entries to see how they are processed internally. You can re-display the previous good record by pressing the f1 key or by obtaining a listing of file 'DELOUT' if you have made it a permanent file in your group. Explicit instructions can be listed out as CRUNFORM starts into execution; they may be helpful.

If you have free time you may create your own forms and run them with CRUNFORM. The source for CRUNFORM is in file CRUNSRCE.PUB; it is a COBOL program.

MPE

USER SUPPORT
SERVICES

HEWLETT hp PACKARD
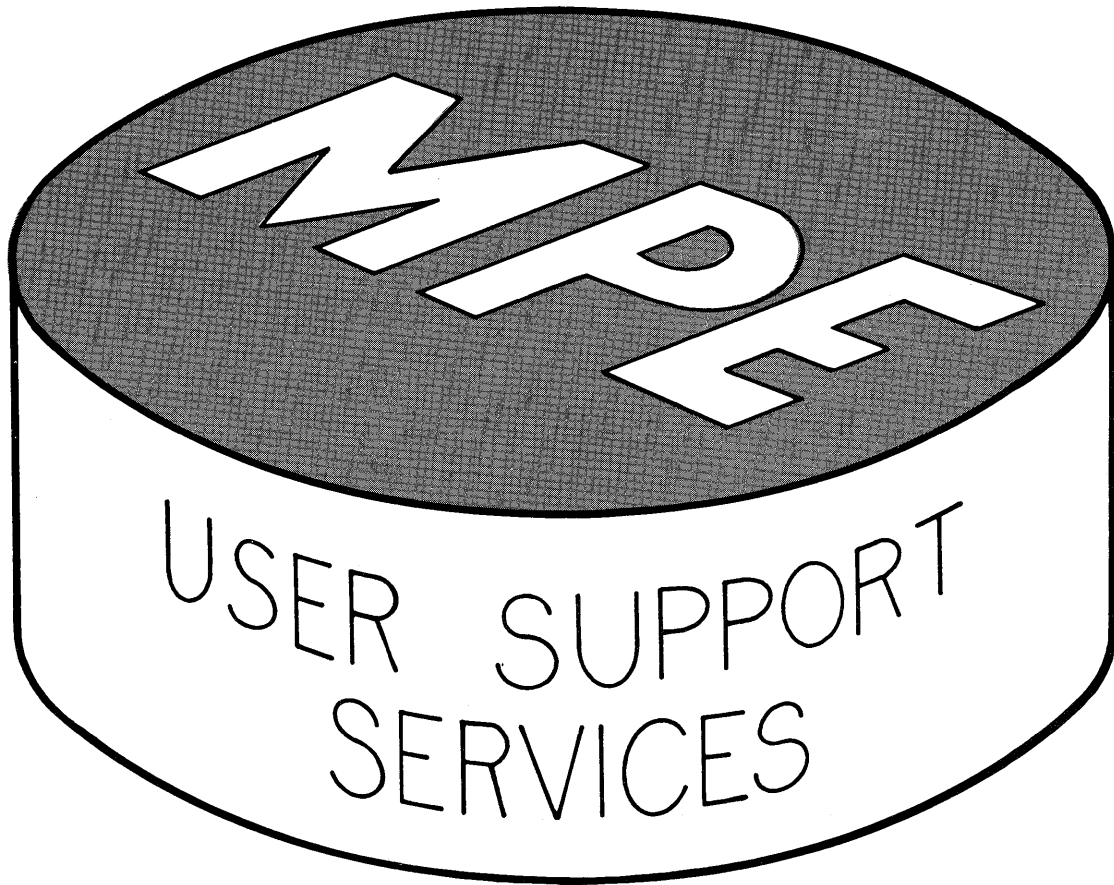
HP OVERVIEW

HP'S COMPUTATION BUSINESS

1966   ⬤ $203 MILLION
0

1972   ◐ 32.6% $479 MILLION
$156 MILLION

1977   ◐ 42.1% $1360 MILLION
$572 MILLION

HEWLETT **hp** *PACKARD*

HP COMPUTER SYSTEMS GROUP

COMPUTER SYSTEMS GROUP

GROUP COORDINATION

R & D
FINANCE
LEGAL
ADMINISTRATION
MARKETING
SALES

DIVISIONS

PERIPHERALS

GENERAL SYSTEMS

BUSINESS DATA PROCESSING

DATA SYSTEMS

INDUSTRIAL SCIENTIFIC PROCESSING

SERVICE DIVISION

SERVICES

DATA TERMINALS
DISC MEMORY
DATA COLLECTION TERMINALS
PRINTERS MAG-TAPE DRIVES
SMALL BUSINESS SYSTEMS & MINI-PERIPHERALS
LSI OPERATIONS

WORLDWIDE SALES FORCE

SYSTEMS ENGINEERING ORGANIZATION (SEO)

HEWLETT hp PACKARD

USER SUPPORT SERVICES

I)   SUPPORT SERVICES
    A)   Responsibilities of your System Manager.
        1)   All interaction with HP done through the System Manager.
        2)   System Manager responsible for isolating and fully documenting bugs and reporting them to HP on SMR form (Software Maintenance Request).
        3)   Only your System Manager can call 'PICS' (Phone-In Consulting Service).
    B)   Customer Support Brochure.
    C)   SSB (Software Status Bulletin).
    D)   Software Releases.
        1)   Master Installation Tape (MIT)
        2)   COMMUNICATOR.

II)   FOLLOW-ON COURSES AVAILABLE.

III)   USER'S GROUP
    A)   Membership (site or individual).
    B)   Regular meetings.
    C)   Contributed Library.

HEWLETT hp PACKARD

SELECTIONS FROM THE CONTRIBUTED LIBRARY

GALLEY   create formatted text from EDIT/3000 files. Features include: paging, headings, text compaction, text justification, centering.

DELAID   DEL/3000 formfile utility. Rename, copy, move forms or modify edits in DEL formfile.

GETFILE   Restores files into log-on group & account from STORE or SYSDUMP tape.

LISTCRET   lists file names and creators on STORE or SYSDUMP tape.

RPGDEL   RPG/DEL interface.

STAR   HP's Statistical Analysis Routines.

IDEA   Image Data Base Evaluation Analyzer.

BLOCK   calculates fixed disc file blocking factors.

-GAMES-   ANIMAL
              BLKJCK
              CRAPS
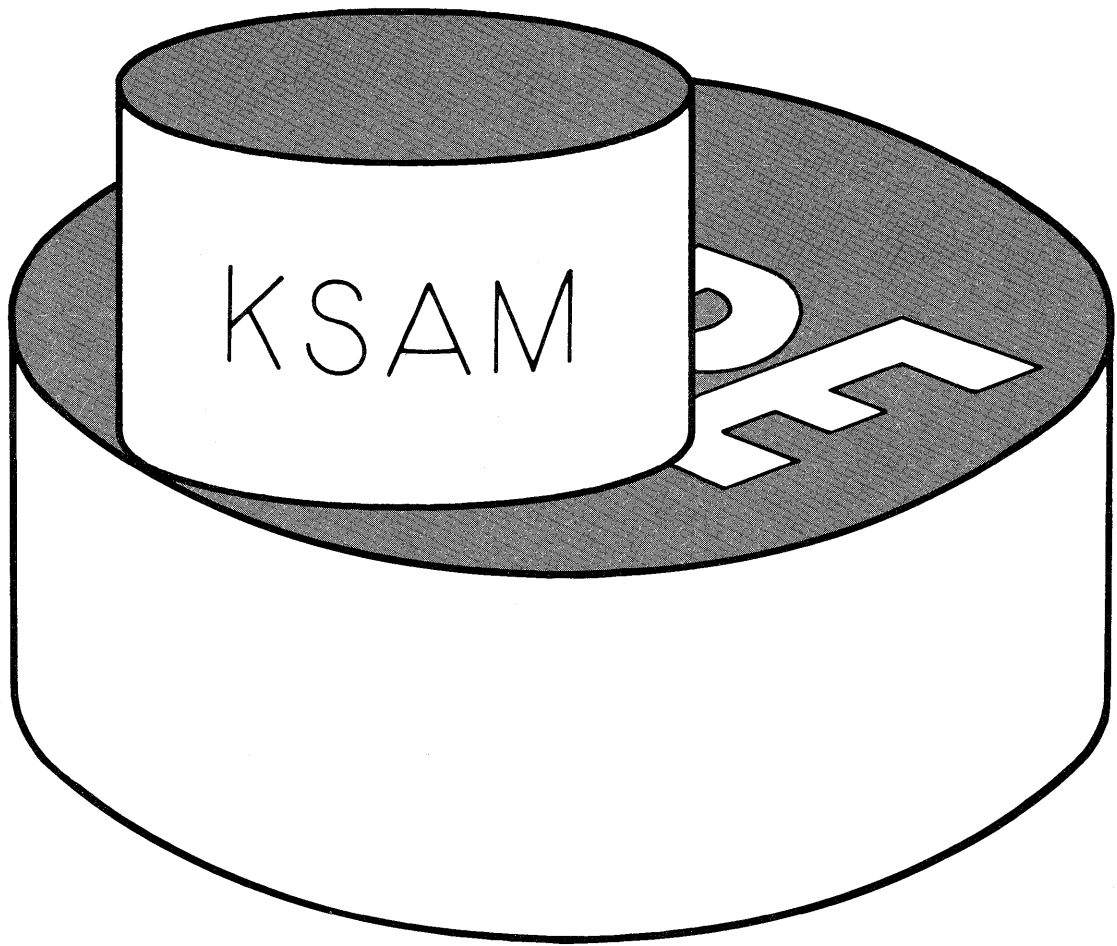
KSAM FEATURES

- Random Access by Key Value and Sequential Access.

- Fixed or Variable Length Records.

- 1 to 16 Keys. Each may be any of 8 data types available on HP-3000.

- Duplicate Key Values Permitted (stored chronologically).

- Concurrent Access by multiple Users or Programs Supported.

- From RPG, COBOL, BASIC, FORTRAN, or SPL.

- Generation of Simple Reports with FCOPY.

- Easy Conversion from Existing 'ISAM' Applications.

- KEY and DATA Files are separate MPE files (May be on different discs).

- Security and Access Restrictions provided by MPE.

SERIES III DATA MANAGEMENT

MPE FILE MANAGEMENT SYSTEM
- Sequential Access (F, U, V).
- Direct Access by record number (F).
- Standard Software on all 3000's.

KSAM
- Sequential Access (F or V).
- Random Access by Key Value (F or V).
- Direct Access (F or V).
- Not available under MPE-C.

IMAGE / QUERY
- DATA BASE Management System.
- On-Line English Language Inquiry Facility.
- Available on all 3000's.

KSAM ACCESS METHODS

SEQUENTIAL ACCESS
- Sequentially by any Key Value (F or V).
- Chronologically (F or V).

RANDOM ACCESS BY KEY VALUE
- Exact Match (F or V).
- Generic Match (F or V).
- Approximate Match (F or V).

DIRECT ACCESS
- Fixed — Relative Record Number.
- Variable — Byte displacement from beginning of file.

COMBINATION
- Processing between limits.

HEWLETT **hp** *PACKARD*

SEQUENTIAL ACCESS

(Simplified KSAM File Structure)

	DATA FILE				KEY FILE	
PHYSICAL RECORD NUMBER	HIRE DATE	NAME	EMPLOYEE NUMBER		PRIMARY KEY ENTRIES	FIRST SET OF ALTERNATE KEY ENTRIES
1	630228	FOX, JIM	0023		ABRAMS, BEN	0023
2	630715	CLARK, JANE	1195		BENTON, ANN	0057
3	640110	BENTON, DON	0057		BENTON, DON	0101
4	640418	DEAN, JOHN	0837		CLARK, JANE	0432
5	640520	ABRAMS, BEN	0101		DEAN, JOHN	0837
6	650217	BENTON, ANN	0432		FOX, JIM	1195
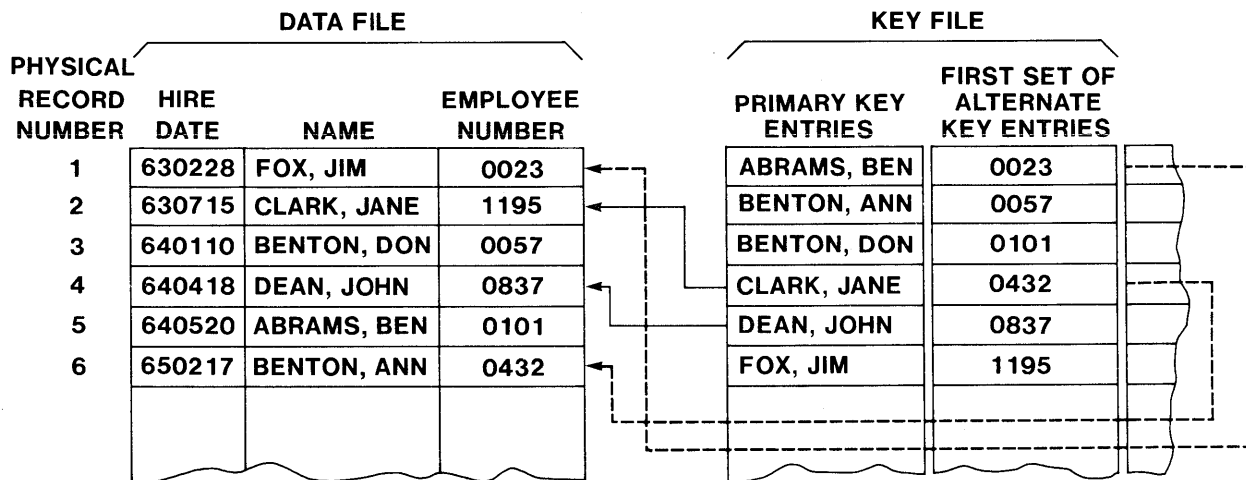
CHRONOLOGICAL ORDER: 1, 2, 3, 4, 5, 6

PRIMARY KEY (LAST NAME) ORDER: 5, 6, 3, 2, 4, 1

FIRST ALTERNATE KEY (EMPLOYEE NUMBER) ORDER: 1, 3, 5, 6, 4, 2

HEWLETT *hp* PACKARD

RANDOM ACCESS BY KEY VALUE

(Simplified KSAM File Structure)

	DATA FILE				KEY FILE		
PHYSICAL RECORD NUMBER	HIRE DATE	NAME	EMPLOYEE NUMBER		PRIMARY KEY ENTRIES	FIRST SET OF ALTERNATE KEY ENTRIES	
1	630228	FOX, JIM	0023		ABRAMS, BEN	0023	
2	630715	CLARK, JANE	1195		BENTON, ANN	0057	
3	640110	BENTON, DON	0057		BENTON, DON	0101	
4	640418	DEAN, JOHN	0837		CLARK, JANE	0432	
5	640520	ABRAMS, BEN	0101		DEAN, JOHN	0837	
6	650217	BENTON, ANN	0432		FOX, JIM	1195	
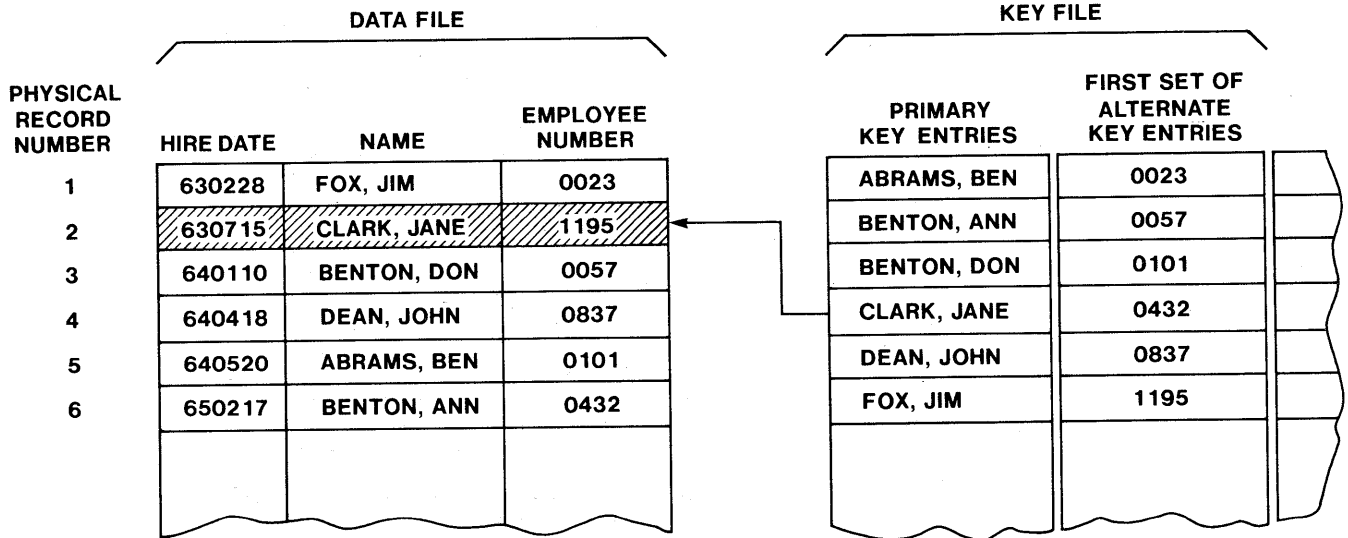
EXAMPLE: READ RECORD WITH EMPLOYEE-NAME = "CLARK, JANE"

TYPES OF RANDOM ACCESS

	NAME	RECORD #
	DRESSER, STEVE	317
EXACT MATCH	DUNN, FRANK	465
"EDWARDS, RICH" →	EDWARDS, RICH	004
	EICHER, MARY	782
GENERIC MATCH	ELDER, MIKE	569
"EP" →	EPPS, JERRY	277
	EPSTEIN, ROGER	854
APPROXIMATE MATCH	ESTOVAN, DON	915
≥ "FA" →	FINDLEY, KEN	417
	FLOYD, RICHARD	904

HEWLETT **hp** PACKARD

FCOPY ENHANCEMENTS FOR KSAM FILES

CREATE A NEW COPY OF A KSAM FILE PAIR

>FROM=KSAMDATA;TO=(NEWDATA,NEWKEY);NEW

PROCESS SEQUENTIALLY BY KEY VALUES

;KEY=nn   (where 'nn' is byte position of key in Data record; first position = 1)

>FROM=KSAMDATA;TO=;KEY=26

;KEY=0 (Chronologically; active records only)

>FROM=KSAMDATA;TO=;KEY=0

No 'KEY=' parameter — Primary Key Sequence by default.

>FROM=KSAMDATA;TO=

CHRONOLOGICALLY (all records in file)
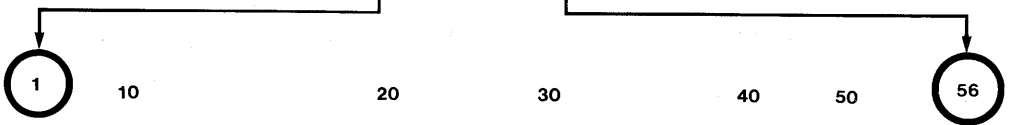
;NOKSAM

>FROM=KSAMDATA;TO=MPEFILE;NOKSAM

REPORTS WITHOUT PROGRAMMING

:RUN FCOPY.PUB.SYS

> FROM = CUSTOMER; TO = ; KEY = 1; SUBSET = "950", 56

(1)   10   20   30   40   50   (56)

ATLAS WIDGET CO.	123	BOULEVARD WAY	SAN JOSE, CA	95068
BAKER MFG. CO.	987	FIRST ST.	SANTA CLARA, CA	95050
BALL DEMOLITION	463	ANTIQUE WAY	SAN JOSE, CA	95061
CABLE TV OF SAN JOSE	295	BROADCAST ST.	SAN JOSE, CA	95062
CENTRAL CITY DISTRIBUTORS	444	MAIN ST.	SAN JOSE, CA	95065
COURTEOUS DELIVERY CO.	6825	STEVENS CREEK BLVD.	SANTA CLARA, CA	95052

KEY SEQUENCE

SUBSET

UTILITY SUPPORT

- File Copying — FCOPY/3000.

- File Backup — MPE :STORE / : RESTORE.

- Utility — KSAMUTIL (operates on both files simultaneously).

```
:RUN KSAMUTIL.PUB.SYS
>HELP
>EXIT
>BUILD        (only way to define KSAM files of new size)
>ERASE
>PURGE
>RENAME       (only way to RENAME KSAM files)
>SAVE
>VERIFY
```

HEWLETT **hp** PACKARD

EASE OF FILE CONVERSION

**WRITE FILE
TO TAPE
SEQUENTIALLY**

PRESENT
COMPUTER
SYSTEM
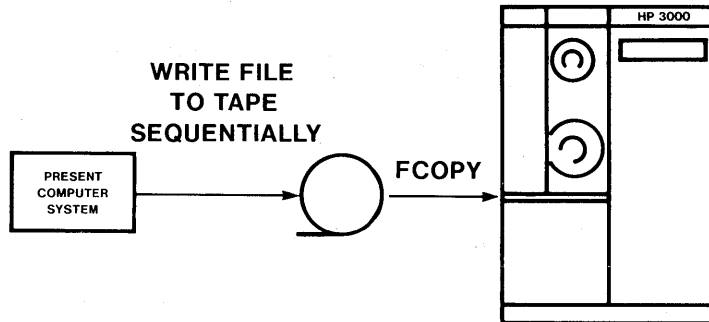
FCOPY

HP 3000

1)   Create Sequential Mag-tape on other Computer System.
2)   Build KSAM file on HP-3000.

```
:RUN KSAMUTIL.PUB.SYS
>BUILD   KSAMFILE  ;KEYFILE=KSAMKEY  ;REC=-80,16,F,ASCII &
>   ;DISC=5000  ;KEY=BYTE,10,4  ;KEY=B,24,6,,DUP
>EXIT
```

3)   Copy Mag-tape to KSAM file with FCOPY.

```
:FILE MYTAPE;DEV=TAPE
:RUN FCOPY.PUB.SYS
>FROM=*MYTAPE;TO=KSAMFILE;SUBSET
>EXIT
```

KSAM KEY FILE

1)   KSAM Key and Data Files are 2 separate MPE files.

2)   Keys dynamically added or deleted from Key File.

3)   Key File dynamically re-structured to maintain a balanced tree structure.
      a)   Blocks split when necessary during addition.
      b)   Blocks re-combined when necessary during deletion.

4)   Each Key has a separate section of the Key File maintained for it.

5)   Enough space reserved for Key File so it will only be a maximum of half full.

6)   Key File must be Allocated in one extent.

7)   Best to let KSAMUTIL calculate blocking factor for keys (may be different for each Key).

SNAP-SHOT OF KSAM KEY FILE

DOUBLE-WORD POINTER TO LOWER LEVEL OF KEY FILE

KEY VALUE

DOUBLE-WORD POINTER TO DATA RECORD

BLK 1

AFTER ADDING A, B, C, D

BLK 1   LEVEL 1

BLK 2   BLK 3   LEVEL 2

ADD E = BLOCK SPLIT

AVAILABLE SPACE
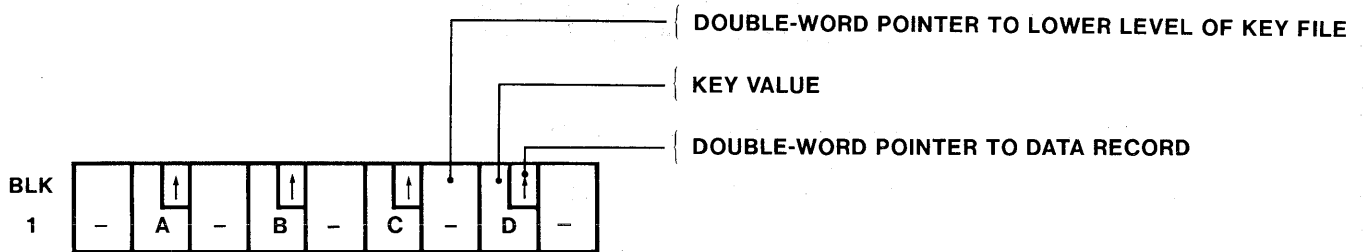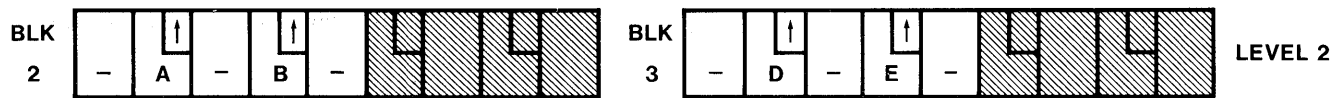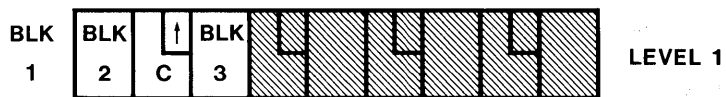
KSAM DATA FILE

1) Data Records stored in chronological order.

2) Deleted records remain in the Data File with their first word set to '-1' as a Delete Flag.

3) Data File may be allocated an extent at a time.

4) Key and Data Files may be on separate Disc Packs.

5) The Key and Data Files each have a user label containing the name of the other.

6) To access a record in the Data File in other than chronological order, the Key File must be accessed to obtain pointer to Data Record.

HEWLETT **hp** PACKARD

KSAM DATA FILE AT LAST SNAP-SHOT

{ DELETE
{ FLAG

KEY

−1		A	
		A	
		B	
		C	
		D	
		E	

AVAILABLE SPACE

KSAM LAB # 1 [1.0 hour]

FIRST NAME	LAST NAME	TELEPHONE #	ADDRESS	CITY	ZIP
9 9 9 9 9 9 9 9 9 9	9 9 9 9 9 9 9 9 9	9 9 9 9 9 9 9 9 9	9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9	9 9 9 9 9 9 9 9 9 9 9 9 9 9	9 9 9 9 9
1 2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17 18 19 20	21 22 23 24 25 26 27 28 29	30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52	53 54 55 56 57 58 59 60 61 62 63 64 65 66	67 68 69 70 71

1   **11**   **21**   **30**   **53**   **67**

A KSAM file with the above format resides in PUB.INTRO. Keys have been defined for positions 1, 11, 53 and 67 of the record. The data file is 'KDATA'; the key file is 'KKEY'.

1) Using FCOPY, list KDATA.PUB on your terminal in chronological sequence (the order in which the file was written).

2) List the file in alphabetical order by last name on your terminal.

3) List the file in order by zip code on your terminal.

4) List the first five records in primary key sequence on your terminal.

5) List all those people who live in San Jose in zip code sequence on your terminal.

6) List 10 records beginning with record number 6 in alphabetical sequence by first name on your terminal.

7) Copy the file in chronological sequence to a non-KSAM file in your group. List this non-KSAM file on your terminal, then purge it.

OPTIONAL — Proceed only if time permits.

8) Build a KSAM file in your group by modifying and running a job stream. From the editor, text in KSAMBILD.PUB unnumbered. Modify the ':JOB' command in record 1 to reference your Username. Keep this job stream unnumbered in a file 'LABKSAM1' in your group. Now ':STREAM' this file. Upon completion of that JOB, inspect the line printer listing and find KSAMDATA and KSAMKEY have been built in your group.

KSAM LAB # 1

9) Load KSAMDATA in your group from LAB1DATA in PUB using FCOPY.

10) Run KSAMUTIL and using 'HELP' list all KSAMUTIL commands on your terminal.

11) Use the VERIFY command to display the attributes of KSAMDATA.

12) Use one ERASE command to delete all entries from both the data and key files but leave the structure intact.

13) Use the VERIFY command to make sure there are no remaining entries in either file.

14) Use FCOPY to copy KDATA.PUB into KSAMDATA using no 'KEY=' parameter. Now list KSAMDATA on your terminal in chronological sequence to see that the file is now stored in order by primary key.

15) Use the PURGE command in KSAMUTIL to purge both KSAMDATA and KSAMKEY at the same time. Using LISTF verify that they have both been purged.

<<< End >>>
(KSAMUTIL BUILD syntax on next page)

KSAMUTIL 'BUILD' COMMAND

```
>BUILD    filereference1      ;KEYFILE=filereference2

 ;KEY=keytype,keylocation,keysize[,[keyblocking][,DUP[LICATE]]]

[;KEY=keytype,keylocation,keysize[,[keyblocking][,DUP[LICATE]]]]...

[;TEMP]  [;KEYENTRIES=numentries]  [;DEV=device]  [;CODE=filecode]

[;REC=[recsize][,[blockfactor][,F][,BINARY]]]
                              [,V][,ASCII ]

[;DISC=[numrec][,[numextents][,initalloc]]]

[;LABELS=numlabels]    [;KEYDEV=device]    [;FIRSTREC={0}]
                                                      {1}


----------------------------------------------------------------

              {B[YTE]   }
              {I[NTEGER]}
              {D[OUBLE] }
    keytype = {R[EAL]   }
              {L[ONG]   }
              {N[UMERIC]}
              {P[ACKED] }
              {*[PACKED]}
                                          << End >>
```
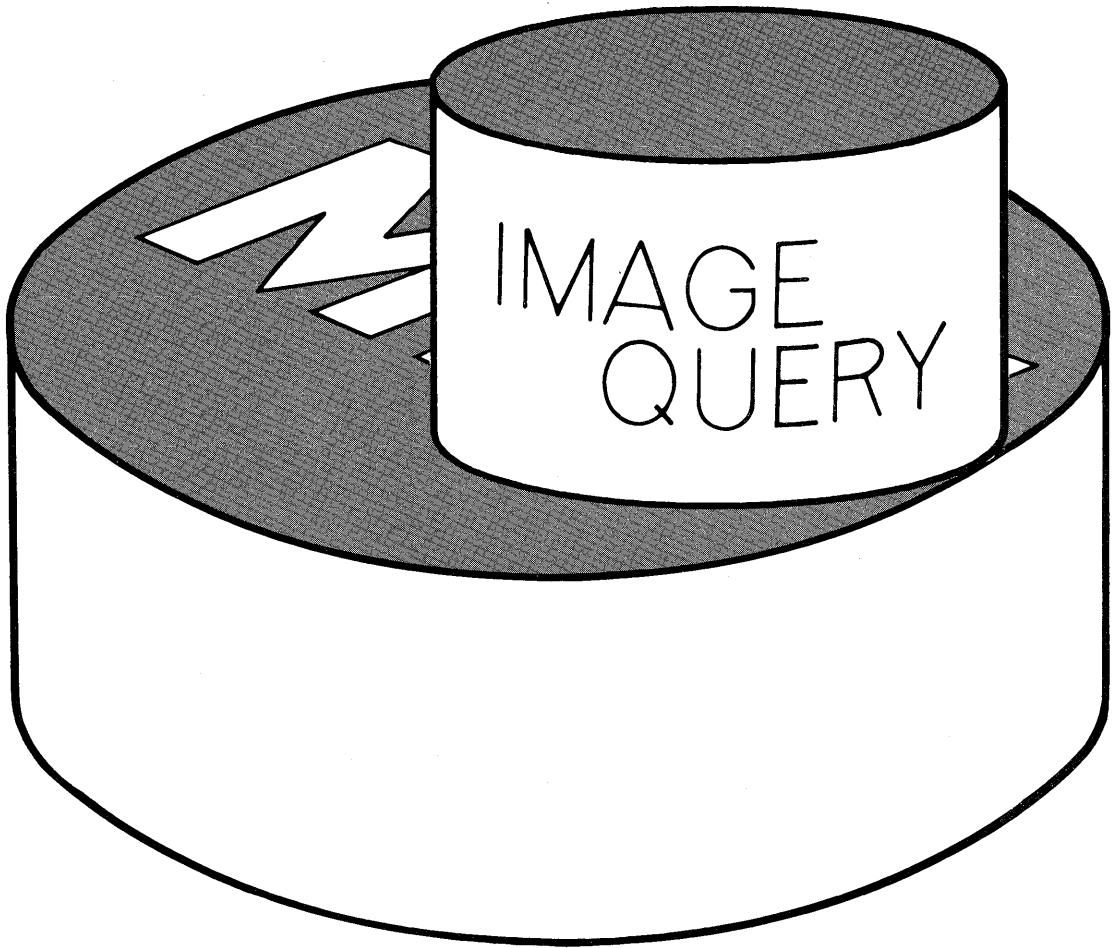
IMAGE
QUERY

FEATURES OF IMAGE/3000

- Provides powerful software tools to define and create a Data Base.

- Network Data Structure allowing cross-referenced access to collections of data.

- Data Sets and interrelationships defined only once.

- Reduces data redundancy.

- Application programmers need not be concerned about details of accessing the Data Base.

- Host languages can be COBOL, RPG, FORTRAN, BASIC or SPL.

- Spontaneous and unanticipated inquiry to the external user through QUERY/3000.

- Flexible SECURITY SYSTEM at the Data Base, Data Set, and Data Item levels.

IMAGE TERMINOLOGY

DATA ITEM	**Smallest accessible element (FIELD).**
DATA ENTRY	**An ordered collection of related data items (RECORD).**
DATA SET	**A collection of data entries sharing a common definition (FILE).**
DATA BASE	**A named collection of related data sets. Data Item names and Data Set names are unique within a Data Base.**
DATA BASE MANAGEMENT SYSTEM	**1) Data Base Definition Language.** **2) Data Manipulation Language.** **3) Utilities.** **4) Inquiry Language.**

IMAGE/3000 SPECIFICATIONS

DATA ITEM NAMES PER DATA BASE	255
DATA ITEM NAMES PER DATA ENTRY	127
DATA SETS PER DATA BASE	99
DETAIL DATA SETS PER MASTER DATA SET	16
SEARCH ITEMS (KEYS) PER DETAIL DATA SET	16
MAXIMUM ENTRY SIZE	4094 bytes
ENTRIES PER DATA SET	8,388,608 ($2^{23}-1$)
ENTRIES PER CHAIN	65,000
CHARACTERS PER DATA BASE NAME	6
CHARACTERS PER LEVEL WORD NAME	8
CHARACTERS PER DATA SET NAME	16
CHARACTERS PER DATA ITEM NAME	16

IMAGE SUBSYSTEMS

• Data Base Definition Subsystem (DBDS)

—Used to define all aspects of the data base (SCHEMA).

—Defines data items, security levels, and relationships between data sets.

• Data Base Management Subsystem (DBMS)

—Provides the means for application programmers to access an Image Data Base.

—Set of stored library routines invoked by call statements in host language application programs.

• Data Base Utility Subsystem (DBUS)

—Stand-alone utility programs used for creating and maintaining Data Bases.

—Used to create, erase, purge, store, restore, load and unload Data Bases.

—Assists in restructuring Data Bases.

IMAGE ACCESS METHODS

. **SERIAL**

. **DIRECTED**

. **CALCULATED** (Master Sets only)

. **CHAINED**

MASTER DATA SETS

- SERVE AS INDEXES TO RELATED DETAIL DATA SET CHAINS.

- CONTAIN ONE SEARCH ITEM AND UNIQUE SEARCH ITEM VALUES.

- MAY BE RELATED TO UP TO 16 DETAIL DATA SETS.

- RELATIVE RECORD LOCATION ASSIGNED TO EACH MASTER ENTRY IS DETERMINED BY PASSING ITS ASSOCIATED SEARCH ITEM (OR KEY VALUE) THROUGH AN ADDRESS CALCULATION ALGO-RITHM.

- TWO TYPES OF MASTER DATA SETS

    1. MANUAL

    2. AUTOMATIC

DATA BASE MANAGEMENT SUBSYSTEM (DBMS)

DEFINITION: A SET OF STORED LIBRARY ROUTINES INVOKED BY CALL STATEMENTS IN HOST LANGUAGE APPLICATION PROGRAMS.

FUNCTIONS: A MEANS FOR APPLICATION PROGRAMMERS TO ACCESS AN IMAGE DATA BASE.

- SERVES AS THE INTERFACE BETWEEN THE DATA BASE AND THE APPLICATION PROGRAMS.

- INITIATES USER ACCESS (OPENING A DATA BASE).

- READS AND UPDATES DATA ITEMS.

- READS, WRITES AND DELETES DATA ENTRIES.

- RETURNS NAME, STRUCTURE, AND ORGANIZATION INFORMATION.

- TERMINATES USER ACCESS (CLOSING A DATA BASE).

ACCESSING DATA BASES (INTRINSICS)

DBOPEN   (base,password,mode,status)

DBLOCK   (base,dset,mode,status)

DBFIND   (base,dset,mode,status,item,arg)

DBGET   (base,dset,mode,status,list,buffer,arg)

DBUPDATE   (base,dset,mode,status,list,buffer)

DBPUT   (base,dset,mode,status,list,buffer)

DBDELETE   (base,dset,mode,status)

DBINFO   (base,qualifier,mode,status,buffer)

DBUNLOCK   (base,dset,mode,status)

DBCLOSE   (base,dset,mode,status)

QUERY/3000

FEATURES AND ADVANTAGES

- Provides a simple method of Data Base access without programming effort.

- Self-contained subsystem interfacing with DBMS.

- Adheres to IMAGE/3000 Security provisions.

- Interactive capability and Batch capability.

- Selects data through compound Logical comparisons (FIND command).

- Permits simple data:
  - —Retrieval
  - —Reporting (formatted or unformatted)
  - —Updating
  - —Addition
  - —Deletion

- Pre-defined QUERY procedures may be executed from a disc file.

- May be used to display the Data Base structure.

QUERY/3000 APPLICATIONS

. **UNANTICIPATED INQUIRY OF THE DATA BASE.**

. **DATA BASE MODIFICATION**

       —Data Entry Addition/Deletion.

       —Data Item Value Modification.

       —Low volume only!

. **REPORT GENERATION**

. **APPLICATION PROGRAM DEBUGGING**

IMAGE OVERVIEW

START

| DATA BASE DIRECTORY ③ | DATA BASE FLOW-CHART ② | PROB DEFINITION ① | APPLICATION FLOW-CHART Ⓐ | APPLICATION PROGRAM PREP Ⓑ |

DATA BASE SCHEMA ④

DBDL ⑤
DATA BASE DEFINITION LANGUAGE PROCESSOR

DBMS
DATA BASE MANAGEMENT SYSTEM LIBRARY

LOAD PROGRAM

DBOPEN
DBPUT
DBCLOSE

APPLICATION PROGRAM INTRINSICS

DBUS
DBUTIL ⑥

DBUS
DBSTORE

DBUS
DBUNLOAD

DBUS
DBRESTORE

DBUS
DBLOAD

DATA BASE ROOT FILES
N
2
1

DETAIL
MASTER
DETAIL
MASTER
DB DATA SETS
2_{99}
2_3
2_2
2_1

DETAIL
MASTER
DETAIL
MASTER
DB DATA SETS
N_{99}
N_3
N_2
N_1

DETAIL
MASTER
DETAIL
MASTER
DB DATA SETS
1_{99}
1_3
1_2
1_1

DBGET
DBFIND
DBUPDATE
DBDELETE
DBINFO
DBLOCK
DBUNLOCK
Ⓒ

QUERY

IMAGE/3000 NFL DATA BASE

PHYSICAL ORGANIZATION OF DATA SETS
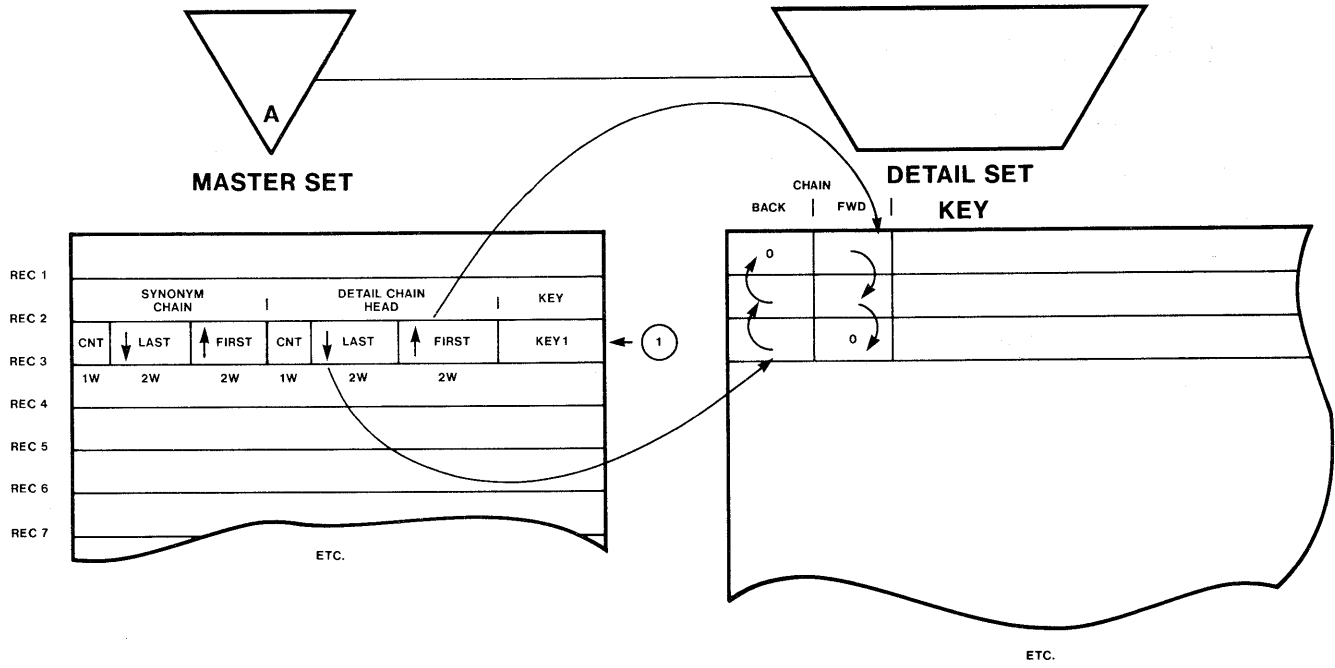
QUERY DEMO

```
:hello mgr.intro

ACCT  PASSWORD?
password
USER  PASSWORD?
secret
 SESSION NUMBER = #S216
 MON, MAR 20, 1978,  3:51 PM
 HP32002A.01.MR

:RUN DBUTIL.PUB.SYS,PURGE

WHICH DATA BASE? EMPLOY
 DATA BASE PURGED

 END OF PROGRAM
:FILE DBSTEXT=EMPSCMAD
:RUN DBSCHEMA.PUB.SYS;PARM=1

PAGE 1       HEWLETT-PACKARD 32215A.04     IMAGE/3000
                MON, MAR 20, 1978,  4:06 PM


BEGIN DATA BASE EMPLOY;
PASSWORDS:
     5 READER;
    10 WRITER;

ITEMS:
    AGE,      Z2(5/10);
    DIV-CODE, X2(5/10);
    DIV-NAME, X20(5/10);
    EMP-NO,   X6(5/10);
    JOB-CODE, X4(5/10);
    NAME,     X20(5/10);
    SEX,      X2(5/10);
    YOS,      Z2(5/10);

SETS:
    NAME:  MSTR-EMPNO,AUTOMATIC(5/10); << MASTER EMPLOYEE NUMBERS >>
    ENTRY: EMP-NO(1);
    CAPACITY: 65;

    NAME:  MSTR-DIV,MANUAL(5/10);       << MASTER DIVISION NUMBERS >>
    ENTRY: DIV-CODE(1),
           DIV-NAME;
    CAPACITY:10;

    NAME:  DET-EMPDATA,DETAIL(5/10);    << DETAIL EMPLOYEE DATA >>
    ENTRY: EMP-NO(MSTR-EMPNO),
```

HEWLETT *hp* PACKARD

QUERY DEMO

```
        NAME,
        AGE,
        SEX,
        YOS,
        JOB-CODE,
        DIV-CODE(MSTR-DIV);
    CAPACITY: 65;
END.
```

DATA SET NAME	TYPE	FLD CNT	PT CT	ENTR LGTH	MED REC	CAPACITY	BLK FAC	BLK LGTH	DISC SPACE
MSTR-EMPNO	A	1	1	3	13	65	36	471	12
MSTR-DIV	M	2	1	11	21	10	10	211	4
DET-EMPDATA	D	7	2	19	27	70	14	379	18

```
                    TOTAL DISC SECTORS INCLUDING ROOT: 43


NUMBER OF ERROR MESSAGES: 0
ITEM NAME COUNT: 8        DATA SET COUNT: 3
ROOT LENGTH: 373    BUFFER LENGTH: 471    TRAILER LENGTH: 256

ROOT FILE EMPLOY CREATED.

 END OF PROGRAM
:RUN DBUTIL.PUB.SYS,CREATE

WHICH DATA BASE? EMPLOY
DATA BASE CREATED

 END OF PROGRAM
:RUN COBLDPRG

DATA BASE OPENED
DBPUT COMPLETED -- DATA BASE EMPLOY LOADED
DBCLOSE OK -- NORMAL TERMINATION

 END OF PROGRAM
:RUN QUERY.PUB.SYS

 HP32216A.03.04 QUERY/3000  MON, MAR 20, 1978,  4:08 PM


 QUERY/3000 READY

>DEFINE
DATA-BASE = >>EMPLOY
PASSWORD = >>READER
MODE = >>3
```

QUERY DEMO

```
DATA-SETS = >>[RETURN]
PROC-FILE = >>PROCFILE
OUTPUT = TERM
OUTPUT = >>[RETURN]
>FIND ALL DET-EMPDATA.AGE
USING SERIAL READ
55  ENTRIES QUALIFIED
>REPORT ALL

EMP-NO            =010002
NAME              =LINDA RAY
AGE               =21
SEX               =02
YOS               =02
JOB-CODE          =0180
DIV-CODE          =20

EMP-NO            =010054
NAME              =SYLVIA COHENSKI
AGE               =21
SEX               =02
YOS               =06
JOB-CODE          =0160
DIV-CODE          =40

< CONTROL Y >
>REPORT EMPREP
```

03/20/78	H E W L E T T - P A C K A R D			PAGE 1
	E M P L O Y E E R E P O R T			
EMP NO.	NAME	AGE	YRS. OF SERV	JOB CODE
010002	LINDA RAY	21	2	0180
010054	SYLVIA COHENSKI	21	6	0160
010048	VICKI LAWRENCE	22	2	0130
010009	PAUL MASSON	23	5	0410
010039	ROZZANNE GANBUZZA	23	1	0260
010056	TERRY KUESTER	23	1	0170
010006	ROCKWELL FLINT	24	3	0030
010012	PHILLIP ANDERSON	24	3	0240
010045	JOHN HARDY	24	2	0070
010049	WILLIAM CLUTTER	25	3	0390

HEWLETT **hp** PACKARD

QUERY DEMO

010044	DEGAN KEPPLER	25	1	0440
010007	DICK BENTZ	25	2	0210
010037	LYNN KUHNHARDT	25	2	0050
010036	PATTY UETZ	25	2	0200
010047	MARGIE BRAKER	26	3	0270
010055	JOYCE MURRIN	27	1	0070
010038	ANDREW O. LARSON	27	2	0040
010053	HARVEY CONNER	29	2	0230
010050	PHILLIP ANDERSON	29	2	0420
010010	TOM JONES	30	11	0060
010018	THOMAS BROTHERS	30	17	0430
010052	PAT MICHAELS	30	10	0190
010003	RICHARD ANDERSON	32	10	0120
010016	TOMMY COLLINS	32	13	0290
010026	PETE HENDRICKSON	32	14	0350
010011	JACK HOWARD	33	14	0150
010064	DOTTIE KEPPLER	33	10	0210
010029	TIMOTHY SHANAHAN	33	12	0100

03/20/78	H E W L E T T - P A C K A R D			PAGE 2
	E M P L O Y E E R E P O R T			
EMP NO.	NAME	AGE	YRS. OF SERV	JOB CODE
010030	STANLEY SHELL	34	11	0360
010001	WILLIAM SICKMILLER	35	11	0380
010013	MARSHA KARNES	36	17	0500

QUERY DEMO

010035	TONY ALEXANDER	37	20	0370
010004	DAVID BUNCH	38	14	0220
010019	JOHN LANGFORD	38	19	0490
010021	MICHAEL SCIMECA	38	11	0340
010025	BARRY TIMM	38	15	0360
010027	MICHAEL KAVANAGH	38	19	0020
010017	LARRY MADISON	40	15	0480
010034	LINDA JONES	40	9	0280
010031	NORM ALEXANDER	42	15	0300
010005	MYRTLE FORTNEY	45	19	0110
010015	PAMELA TURNER	45	25	0080
010066	SUZZANNE SMITH	45	19	0400
010062	JAMES WILLITS	46	22	0250
010042	SWEDE TURNER	47	15	0310
010058	JACK HARBOUR	48	20	0010
010041	ELLIE TUTTLE	50	13	0450
010020	RICHARD OLSON	50	25	0090
010024	DAVE KRAMER	51	18	0330
010040	JANET VAN AMBER	54	23	0190
010060	LAUREL MADSEN	55	23	0140
010032	DAVID KRAMER	56	20	0320
010023	JUDY MARTINI	60	30	0400
010046	ORLANDO LARSONA	67	47	0460
010043	JANET MATTHEWS	87	33	0450

QUERY DEMO

NUMBER EMPLOYEES SELECTED =   55

>REPORT EMPTERM

```
03/20/78              H E W L E T T - P A C K A R D           PAGE 1
                      E M P L O Y E E   R E P O R T
   EMP NO.        NAME                   AGE          YRS. OF SERV     JOB CODE

   010002     LINDA RAY                   21               2            0180
   010054     SYLVIA COHENSKI             21               6            0160
   010048     VICKI LAWRENCE              22               2            0130
   010009     PAUL MASSON                 23               5            0410
   010039     ROZZANNE GANBUZZA           23               1            0260
   010056     TERRY KUESTER               23               1            0170
   010006     ROCKWELL FLINT              24               3            0030
   010012     PHILLIP ANDERSON            24               3            0240
   010045     JOHN HARDY                  24               2            0070
   010049     WILLIAM CLUTTER             25               3            0390
   010044     DEGAN KEPPLER               25               1            0440
   010007     DICK BENTZ                  25               2            0210
   010037     LYNN KUHNHARDT              25               2            0050
   010036     PATTY UETZ                  25               2            0200
   010047     MARGIE BRAKER               26               3            0270
   010055     JOYCE MURRIN                27               1            0070
   010038     ANDREW D. LARSON            27               2            0040
   010053     HARVEY CONNER               29               2            0230
   010050     PHILLIP ANDERSON            29               2            0420


03/20/78              H E W L E T T - P A C K A R D           PAGE 2
                      E M P L O Y E E   R E P O R T
   EMP NO.        NAME                   AGE          YRS. OF SERV     JOB CODE

   010010     TOM JONES                   30              11            0060
   010018     THOMAS BROTHERS             30              17            0430
   010052     PAT  MICHAELS               30              10            0190
   010003     RICHARD ANDERSON            32              10            0120
   010016     TOMMY COLLINS               32              13            0290
   010026     PETE HENDRICKSON            32              14            0350
   010011     JACK HOWARD                 33              14            0150
   010064     DOTTIE KEPPLER              33              10            0210
   010029     TIMOTHY SHANAHAN            33              12            0100
   010030     STANLEY SHELL               34              11            0360
   010001     WILLIAM SICKMILLER          35              11            0380
```

QUERY DEMO

010013	MARSHA KARNES	36	17	0500
010035	TONY ALEXANDER	37	20	0370
010004	DAVID BUNCH	38	14	0220
010019	JOHN LANGFORD	38	19	0490
010021	MICHAEL SCIMECA	38	11	0340
010025	BARRY TIMM	38	15	0360
010027	MICHAEL KAVANAGH	38	19	0020
010017	LARRY MADISON	40	15	0480

```
03/20/78                 H E W L E T T - P A C K A R D              PAGE 3
                         E M P L O Y E E   R E P O R T
    EMP NO.       NAME                    AGE        YRS. OF SERV    JOB CODE
```

010034	LINDA JONES	40	9	0280
010031	NORM ALEXANDER	42	15	0300
010005	MYRTLE FORTNEY	45	19	0110
010015	PAMELA TURNER	45	25	0080
010066	SUZZANNE SMITH	45	19	0400
010062	JAMES WILLITS	46	22	0250
010042	SWEDE TURNER	47	15	0310
010058	JACK HARBOUR	48	20	0010
010041	ELLIE TUTTLE	50	13	0450
010020	RICHARD OLSON	50	25	0090
010024	DAVE KRAMER	51	18	0330
010040	JANET VAN AMBER	54	23	0190
010060	LAUREL MADSEN	55	23	0140
010032	DAVID KRAMER	56	20	0320
010023	JUDY MARTINI	60	30	0400
010046	ORLANDO LARSONA	67	47	0460
010043	JANET MATTHEWS	87	33	0450

```
NUMBER EMPLOYEES SELECTED =   55
>REPORT EMPPAUSE
03/20/78                 H E W L E T T - P A C K A R D              PAGE 1
                         E M P L O Y E E   R E P O R T
    EMP NO.       NAME                    AGE        YRS. OF SERV    JOB CODE
```

010002	LINDA RAY	21	2	0180
010054	SYLVIA COHENSKI	21	6	0160
010048	VICKI LAWRENCE	22	2	0130
010009	PAUL MASSON	23	5	0410
010039	ROZZANNE GANBUZZA	23	1	0260
010056	TERRY KUESTER	23	1	0170
010006	ROCKWELL FLINT	24	3	0030
010012	PHILLIP ANDERSON	24	3	0240
010045	JOHN HARDY	24	2	0070
010049	WILLIAM CLUTTER	25	3	0390

QUERY DEMO

010044	DEGAN KEPPLER	25	1	0440
010007	DICK BENTZ	25	2	0210
010037	LYNN KUHNHARDT	25	2	0050
010036	PATTY UETZ	25	2	0200
010047	MARGIE BRAKER	26	3	0270
010055	JOYCE MURRIN	27	1	0070
010038	ANDREW O. LARSON	27	2	0040
010053	HARVEY CONNER	29	2	0230
010050	PHILLIP ANDERSON	29	2	0420

<< Pause >>

03/20/78   H E W L E T T - P A C K A R D   PAGE 2
                    E M P L O Y E E   R E P O R T

EMP NO.	NAME	AGE	YRS. OF SERV	JOB CODE
010010	TOM JONES	30	11	0060
010018	THOMAS BROTHERS	30	17	0430
010052	PAT MICHAELS	30	10	0190
010003	RICHARD ANDERSON	32	10	0120
010016	TOMMY COLLINS	32	13	0290
010026	PETE HENDRICKSON	32	14	0350
010011	JACK HOWARD	33	14	0150
010064	DOTTIE KEPPLER	33	10	0210
010029	TIMOTHY SHANAHAN	33	12	0100
010030	STANLEY SHELL	34	11	0360
010001	WILLIAM SICKMILLER	35	11	0380
010013	MARSHA KARNES	36	17	0500
010035	TONY ALEXANDER	37	20	0370
010004	DAVID BUNCH	38	14	0220
010019	JOHN LANGFORD	38	19	0490
010021	MICHAEL SCIMECA	38	11	0340
010025	BARRY TIMM	38	15	0360
010027	MICHAEL KAVANAGH	38	19	0020
010017	LARRY MADISON	40	15	0480

<< Pause >>

03/20/78   H E W L E T T - P A C K A R D   PAGE 3
                    E M P L O Y E E   R E P O R T

EMP NO.	NAME	AGE	YRS. OF SERV	JOB CODE
010034	LINDA JONES	40	9	0280
010031	NORM ALEXANDER	42	15	0300
010005	MYRTLE FORTNEY	45	19	0110
010015	PAMELA TURNER	45	25	0080
010066	SUZZANNE SMITH	45	19	0400
010062	JAMES WILLITS	46	22	0250
010042	SWEDE TURNER	47	15	0310
010058	JACK HARBOUR	48	20	0010
010041	ELLIE TUTTLE	50	13	0450

QUERY DEMO

```
010020    RICHARD OLSON          50         25         0090
010024    DAVE KRAMER            51         18         0330
010040    JANET VAN AMBER        54         23         0190
010060    LAUREL MADSEN          55         23         0140
010032    DAVID KRAMER           56         20         0320
010023    JUDY MARTINI           60         30         0400
010046    ORLANDO LARSONA        67         47         0460
010043    JANET MATTHEWS         87         33         0450
NUMBER EMPLOYEES SELECTED =    55
>ADD DET-EMPDATA
ILLEGAL ACCESS
>PASSWORD=WRITER
>ADD DET-EMPDATA
EMP-NO            =>>999999
NAME              =>>JIVE A. EMPLOYEE
AGE               =>>9
SEX               =>>NONE
INPUT TOO LONG - TRUNCATED
YOS               =>>13
JOB-CODE          =>>12345
INPUT TOO LONG - TRUNCATED
DIV-CODE          =>>40


EMP-NO            =>>//
>FIND AGE=9
USING SERIAL READ
1  ENTRIES QUALIFIED
>REPORT ALL

EMP-NO            =999999
NAME             =JIVE A. EMPLOYEE
AGE              =09
SEX              =NO
YOS              =13
JOB-CODE         =1234
DIV-CODE         =40


>DELETE
DELETE ALL RETRIEVED ENTRIES (YES OR NO)?
>>YES
>FIND AGE=40
USING SERIAL READ
2  ENTRIES QUALIFIED
>REPORT EMPTERM
```

```
03/20/78            H E W L E T T - P A C K A R D            PAGE 1
                    E M P L O Y E E   R E P O R T
    EMP NO.      NAME                AGE        YRS. OF SERV   JOB CODE
```

QUERY DEMO

```
010017   LARRY MADISON              40          15          0480
010034   LINDA JONES                40           9          0280
NUMBER EMPLOYEES SELECTED =    2
>REPLACE AGE="39"
>>END
>FIND AGE=39
USING SERIAL READ
2  ENTRIES QUALIFIED
>REPORT EMPTERM

03/20/78           H E W L E T T - P A C K A R D              PAGE 1
                   E M P L O Y E E   R E P O R T
  EMP NO.       NAME                 AGE         YRS. OF SERV   JOB CODE

  010017   LARRY MADISON              39          15          0480
  010034   LINDA JONES                39           9          0280
  NUMBER EMPLOYEES SELECTED =    2
>EXIT

  END OF PROGRAM
:
```

HEWLETT hp PACKARD

KSAM vs. IMAGE

CONSIDERATION	KSAM	IMAGE
Heavy Sequential Processing	Better Suited	---
Unanticipated Inquiries	FCOPY (limited)	YES (QUERY/3000)
Program-Data Independence	NO	YES
Easy File Conversion from ISAM	YES	Can be CUMBERSOME
PRIVACY and SECURITY	FILE LEVEL only	FILE, RECORD, or FIELD
Privileged Files (Protected)	NO	YES
Variable Length Records	YES	NO
Field Access by Name	NO	YES
Generic (partial) Key Retrieval	YES	NO

KSAM vs. IMAGE

CONSIDERATION	KSAM	IMAGE
Duplicate Keys	YES	DETAIL SETS - YES MASTER SETS - NO
Maximum Record Size	8 K bytes	4 K bytes
Buffering Overhead	1 Extra Data Seg / File / Process	1 Extra Data Seg / Data Base / Process
Minimum Field Size	1 byte	2 bytes (1 word)

Disc Space Allocation	Key File
	    All at once
	Data File
	    As required
		Master Set
		    All at once
		Detail Set
		    All at once

Sort Files Directly	YES	NO
Max number of 'Keys'	16	16 Master Sets referencing one Detail Set.
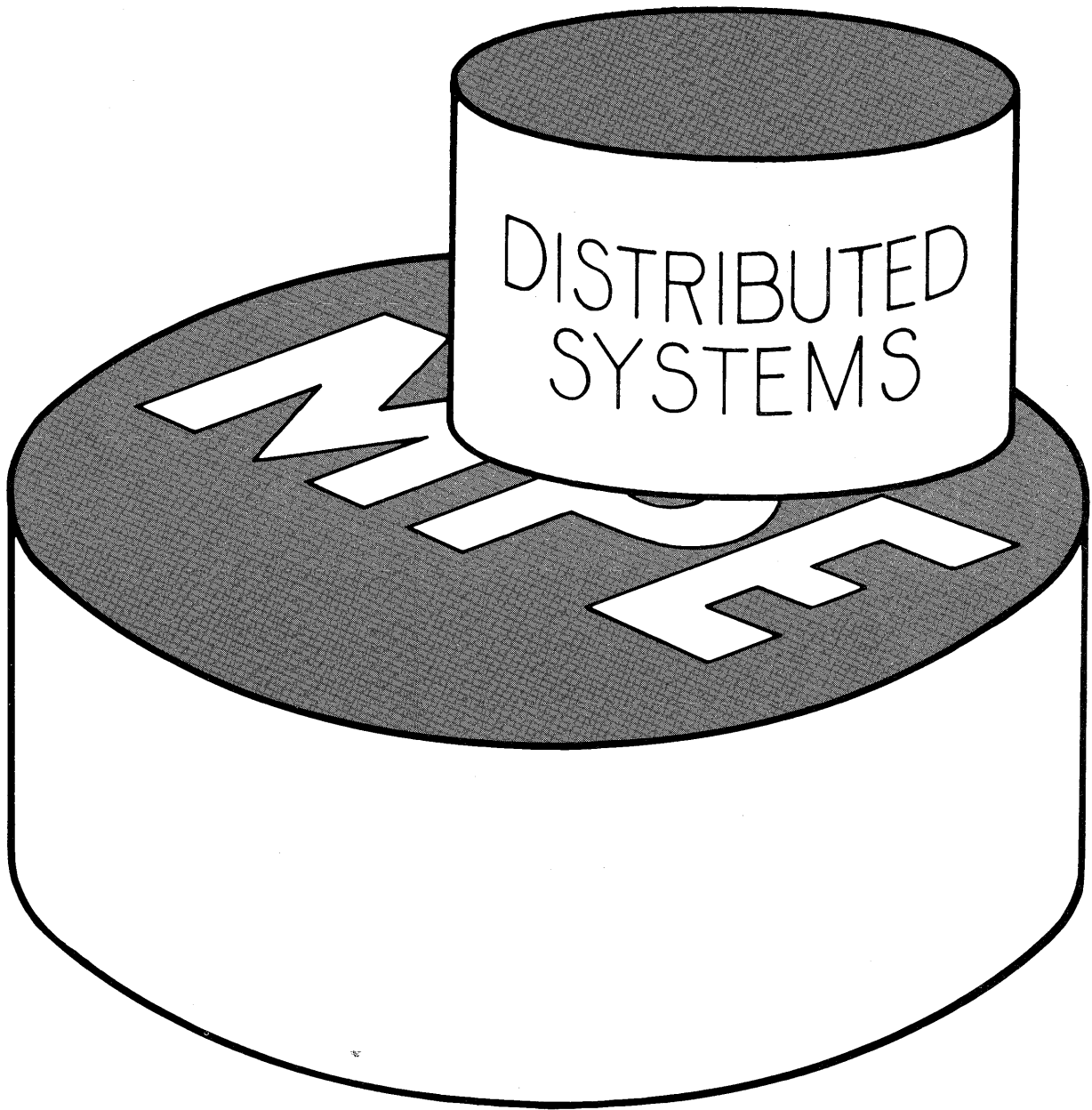
HP COMMUNICATION PRODUCTS

DS/3000

MRJE / RJE

2780
RJE   RJE

DS/1000

Other
mainframes
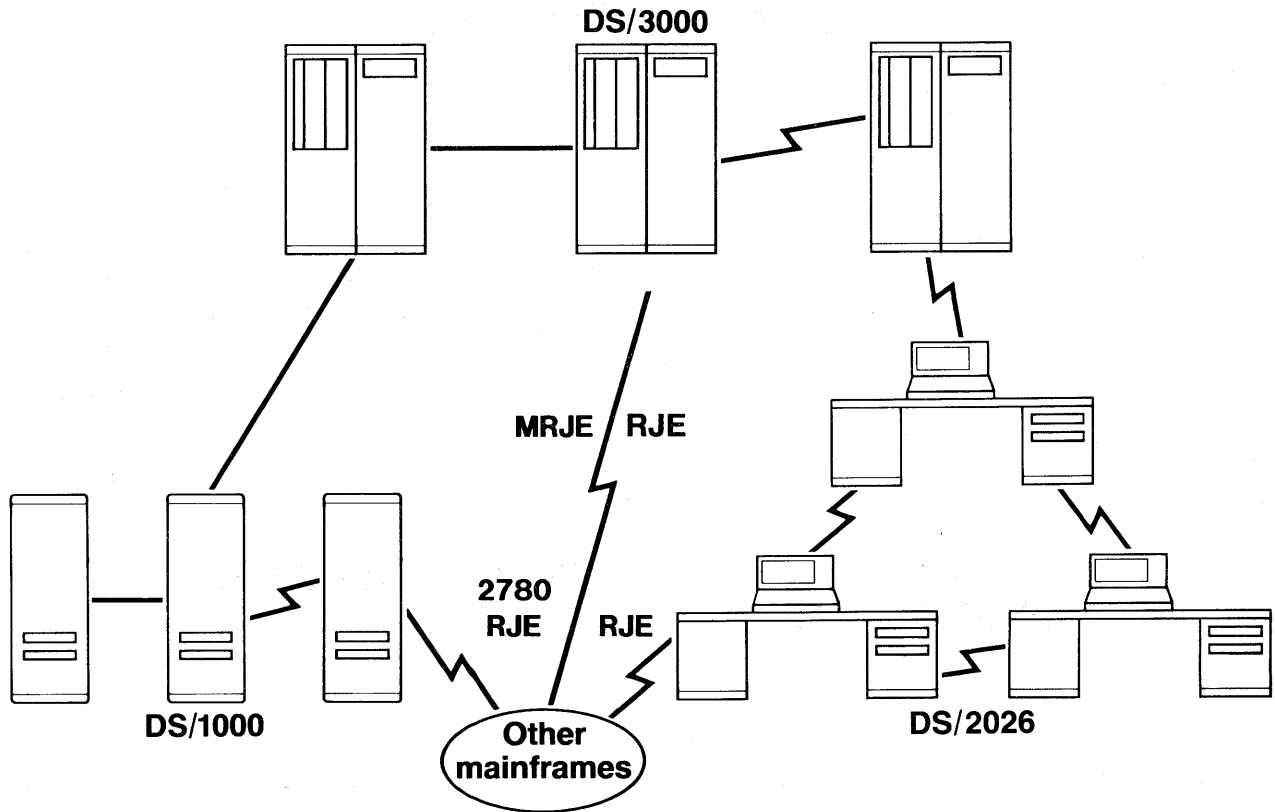
DS/2026

AMONG HP COMPUTERS

Hardwired interface or over telephone lines.

DS/3000   DS/1000   DS/2026

HP TO OTHER MAINFRAMES

Over telephone lines.

MRJE   RJE

HEWLETT hp PACKARD

REMOTE COMMAND PROCESSING

DS 3000

:Hello

:Remote hello
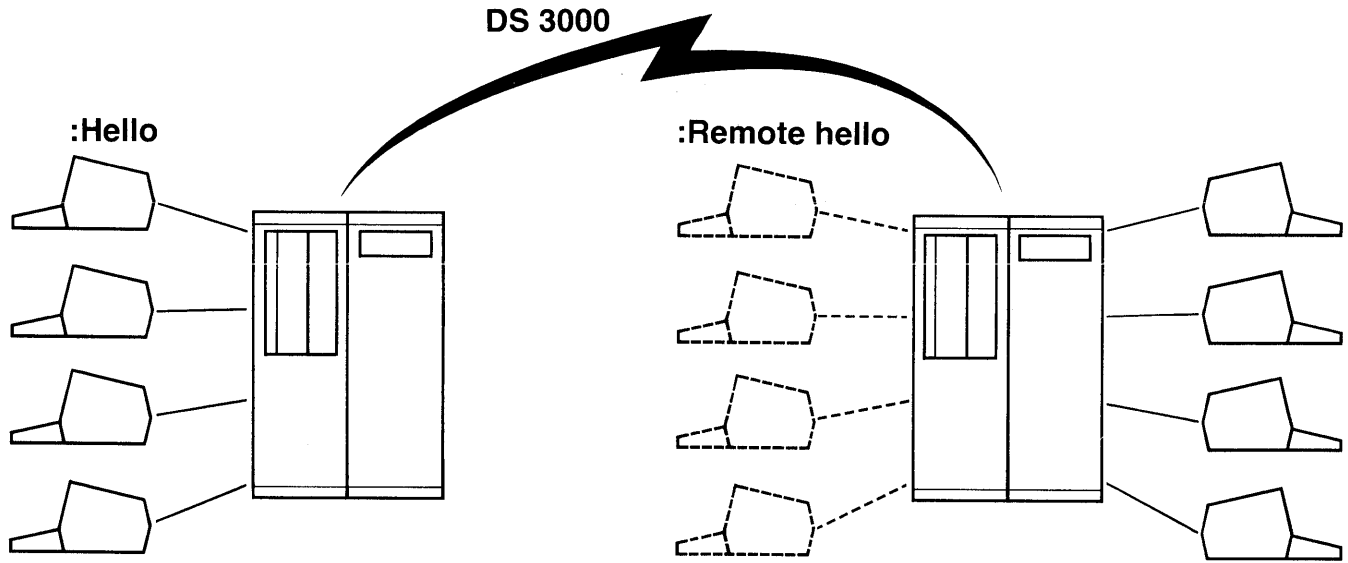
DS/3000 USES VIRTUAL TERMINALS ON REMOTE HP 3000 SYSTEMS

USING DS/3000

LINE MUST BE OPENED AT BOTH ENDS BY CONSOLE OPERATORS

    =DSLINE 50,OPEN   *on local 'TRAINING' System Console*

    =DSLINE 62,OPEN   *on remote 'DEMO' System Console*

TO ACCESS FILES ON A REMOTE SYSTEM, USER MUST IDENTIFY LINE TO BE USED AND LOG-ON TO REMOTE SYSTEM (USER MAY ONLY LOG-ON IF A DS 'VIRTUAL TERMINAL' IS AVAILABLE ON THE REMOTE SYSTEM).

```
    :DSLINE DST2D
    :REMOTE HELLO REMOTE.RINTRO/ANOTHER
  -or-
    :REMOTE HELLO REMOTE.RINTRO/ANOTHER;DSLINE=DST2D
```

ACCESSED VIA NEW OPTION IN :FILE — ';DEV=[[dsdevice]#][device]...'

```
    :FILE REMFILE=FILENAME.GRPNAME.ACCTNAME;DEV=DST2D#DISC
```

:REMOTE COMMAND WILL PUT YOU INTO REMOTE MODE

```
    :REMOTE
    #
```

EXAMPLE 1 — ACCESS A REMOTE FILE

```
:HELLO STUDENT.INTRO/PASSWORD                 Log-on to local system
HP3000 III.  MON, APR  3, 1978,  1:38 PM
*** WELCOME TO YOUR FRIENDLY TRAINING SYSTEM ***
:DSLINE DST2D
DS LINE NUMBER = #L3.
:FILE REMFILE=SHORTF.PUB;DEV=DST2D#           File on remote system
:EDITOR
HP32201A.7.0H EDIT/3000  MON, APR  3, 1978,  1:38 PM
(C) HEWLETT-PACKARD CO. 1976
/T *REMFILE                        must be logged-on to remote
                                   system to access files on it.
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
!  ERROR NUMBER: 216    RESIDUE: 0               !
!  BLOCK NUMBER: 0              NUMREC: 0         !
+------------------------------------------------+
*23*X
FAILURE TO OPEN TEXT FILE    (216)
```

EXAMPLE 1 — ACCESS A REMOTE FILE

```
/                                    < BREAK Key pressed
:REMOTE HELLO REMOTE.RINTRO/ANOTHER
HP3000 III.  MON, APR  3, 1978,  1:40 PM
<<< WELCOME TO THE DEMO SYSTEM >>>
:RESUME
READ PENDING
T *REMFILE                           after remote log-on can
/L ALL                               access remote files.
      1      THIS IS LINE ONE
      2      THIS IS LINE TWO
      3      THIS IS LINE THREE
      4      THIS IS LINE FOUR
/EXIT

END OF SUBSYSTEM
:REMOTE BYE
CPU=1. CONNECT=1. MON, APR  3, 1978,  1:41 PM
:
```

EXAMPLE 2 — RUNNING A REMOTE PROGRAM

```
:REMOTE HELLO REMOTE.RINTRO/ANOTHER
HP3000 III.  MON, APR  3, 1978,  1:39 PM
<<< WELCOME TO THE DEMO SYSTEM >>>
:REMOTE

#FILE IN=SHORTF.PUB;DEV=#
#RUN FCOPY.PUB.SYS

HP32212A.3.03 FILE COPIER (C) HEWLETT-PACKARD CO. 1976

>FROM=*IN;TO=NEWFILE;NEW
EOF FOUND IN FROMFILE AFTER RECORD 3

4 RECORDS PROCESSED *** 0 ERRORS

>EXIT

 END OF PROGRAM
```

EXAMPLE 3 — EXECUTING REMOTE COMMANDS

```
#LISTF NEWFILE,2
ACCOUNT=  RINTRO        GROUP=  GREMOTE

FILENAME   CODE   ------------LOGICAL RECORD-----------   ----SPACE----
                   SIZE  TYP          EOF       LIMIT R/B SECTORS #X MX

NEWFILE           80B  FA            4           4  16      10   1  1
```

`#:`[RETURN]   *Exit remote mode*

```
:LISTF SHORTF.PUB,2
ACCOUNT=  INTRO         GROUP=  PUB

FILENAME   CODE   ------------LOGICAL RECORD-----------   ----SPACE----
                   SIZE  TYP          EOF       LIMIT R/B SECTORS #X MX

SHORTF            80B  FA            4           4  16      10   1  1

:REMOTE BYE
CPU=8. CONNECT=4. MON, APR  3, 1978,  1:43 PM
:BYE

CPU=9. CONNECT=7. MON, APR  3, 1978,  1:44 PM
```

PROGRAM-TO-PROGRAM COMMUNICATION

SPL Procedures — most easily called by SPL or FORTRAN.

Can initiate 'Slave' process on remote system.

Allows decisions based on data content or pre-processing of data by remote system.

Greater efficiency in data transfers than generalized DS transfers.

MPF

LAB SOLUTIONS

W O R K S E S S I O N -- M P E   S Y N T A X

```
**************************************************************
**    Check the following statements and see if they are    **
** syntactically correct or not.  You are not expected to    **
** know what would happen when trying to execute the follow- **
** ing commands at this point, but you should be able to     **
** check their syntax.  Use the STANDARD CAPABILITIES section**
** of your "SOFTWARE POCKET GUIDE" to find the syntax for the**
** following commands, circle all errors and write the number**
** of errors you find in the space provided at the right of  **
** each example.                              [30 minutes]   **
**************************************************************
```

		no. of errors
(1)	:HELLO WORKSESS,STUDENT.INTRO,PUB	0

(2)	:HELLO STUDENT,INTRO;TERM=3;&
	       ^
	: TIME=TEN
	       ^^^
		2

(3)	:HELLO STUDENT,STUDENT/SECRET.INTRO& : ;PRI=DS	0
(4)	:HELLO & : STUDENT.INTRO & : ;PRI=HIPRI ^^^^^	1
(5)	:HELLO STUDENT.INTRO/PASSWORD;HI& ^^ : PRI;TERM=10;TIME=10	1
(6)	:HELLO ^^^^ ^ ^^^^	3
(7)	:LISTF	__0___
(8)	:LISTF;LISTF	__0___
(9)	:LISTF @.@.SYS,2;LISTFILE	__0___
(10)	:LISTF @.GSTUDENT.@ ;2 X ^	2
(11)	:LISTF LISTF	__0___
(12)	:SHOWJOB	__0___

```
|                                                         NO. OF  |
|                                                         ERRORS  |
|-----------------------------------------------------------------|
|(13) :SHOWJOB    JOB=#S                                    1     |
|                     ^                                           |
|                                                        ------   |
|-----------------------------------------------------------------|
|(14) :SHOWJOB   #J11,#S13,STATUS                          2-4    |
|                    ^  ^^  ^  ^^^^                               |
|                                                        ------   |
|-----------------------------------------------------------------|
|(15) :SHOWJOB INTRO;JOB=@.INTRO;EXEC                      1-2    |
|                               ^ ^^^                             |
|                                                        ------   |
|-----------------------------------------------------------------|
|(16) :SHOWJOB   JOB=@,STUDENT.INTRO                      __0___  |
|-----------------------------------------------------------------|
|(17) :SHOWJOB      JOB&                                    1     |
|     :    =@.@                                                   |
|             ^                                          ------   |
|-----------------------------------------------------------------|
|(18) :SHOWJOB   EXEC;  JOB=@,@.INTRO                     __0___  |
|                                                                 |
|*****************************************************************|
|*         OPTIONAL -- Do this part only if time permits.       *|
|*****************************************************************|
|                                                                 |
|    Log-on & execute the commands in the last three questions    |
|above.  Correct any errors to get them to execute properly.      |
|                                                                 |
|                        << End >>                                |
```

C L A S S   W O R K S E S S I O N

Use the Accounting Structure diagram from the previous page.
What are the Disc File names each User must use to address the first
file in each of the Groups?

To reference file:	For User:		
	STUDENT.INTRO	MGR.INTRO	MANAGER.SYS
BRUTUS39	BRUTUS39	BRUTUS39.GSTUDENT	BRUTUS39.GSTUDENT.INTRO
DEFTABS	DEFTABS.PUB	DEFTABS	DEFTABS.PUB.INTRO
FCOPY	FCOPY.PUB.SYS	FCOPY.PUB.SYS	FCOPY

F U N D A M E N T A L S   L A B   #1

U S I N G   A   L A N G U A G E [40 minutes]

Choose your favorite language and do the associated chapter
in "USING the HP-3000" (chapters 4 through 7).

U S I N G   E X I S T I N G   F I L E S [20 minutes]

The utility program "FCOPY" in PUB.SYS will copy files.
Its command syntax is:

>FROM=filereference1;TO=filereference2[;NEW]

where:

filereferences 1 & 2 specify files to be used.

;NEW will create a new Permanent Disc File with exactly
the same attributes as the FROM file.

*** WARNING Messages you may encounter ***

'*200*' -- Different record lengths in FROM & TO files.

'*201*' -- FROM & TO files are different types; one is ASCII
one is BINARY.
Disregard these warnings. Merely press the RETURN key and
by-pass them. Operations will still be performed correctly.

(1) Execute FCOPY by keying in the following command in response
to the ':' prompt:

RUN FCOPY.PUB.SYS

(2) In response to the '>' prompt, enter:

FROM=ASCII.PUB;TO=$STDLIST

This will list the contents of file ASCII in the PUB group
of the INTRO account on your terminal.
:HELLO STUDENT.INTRO/PASSWORD
SESSION NUMBER = #S228
FRI, MAR 3, 1978, 8:38 AM
HP32002A.01.MR
WELCOME TO THE TRAINING HP-3000.
:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976

>FROM=ASCII.PUB;TO=$STDLIST
  ASCII FILES -- Identify DATA files. (the Editor can only process
ASCII files). As new disc file extents are created or as records
are padded, spaces (%40) are used.
RECORD 4 ... END-OF-FILE.
EOF FOUND IN FROMFILE AFTER RECORD 3
4 RECORDS PROCESSED *** 0 ERRORS
    (3) Now list the contents of file BINARY.PUB on your terminal. |
         Expect warning '*201*'.
>FROM=BINARY.PUB;TO=$STDLIST
201
  BINARY FILES -- Usually identify Program files but data they
contain may be the same as ASCII files. Only real differences:
1) As disc file extents are created or as records are padded, binary
zeroes (%0) are used.
2) BINARY files cannot be processed by the EDITOR.
RECORD 6 ... END-OF-FILE.
EOF FOUND IN FROMFILE AFTER RECORD 5
6 RECORDS PROCESSED *** 0 ERRORS
    (4) Copy ASCII.PUB into a NEW file called MYASCII in your group. |
>FROM=ASCII.PUB;TO=MYASCII;NEW
EOF FOUND IN FROMFILE AFTER RECORD 3
4 RECORDS PROCESSED *** 0 ERRORS
    (5) Copy SHORTY.PUB into a NEW file called SHORTY in your group. |
>FROM=SHORTY.PUB;TO=SHORTY;NEW
EOF FOUND IN FROMFILE AFTER RECORD 1
2 RECORDS PROCESSED *** 0 ERRORS
    (6) List the contents of SHORTY on your terminal. |
>FROM=SHORTY;TO=$STDLIST
SHORTY FILE -- FIRST RECORD.
SHORTY FILE -- LAST RECORD.
EOF FOUND IN FROMFILE AFTER RECORD 1
2 RECORDS PROCESSED *** 0 ERRORS
    (7) Copy the following from your terminal (use $STDINX) to
         file SHORTY:
               RECORD #1
               :RECORD #2
               #RECORD #3

         You should encounter the end-of-file in 'SHORTY'. You are
         trying to put 3 records into a two record file.

```
>FROM=$STDINX;TO=SHORTY
RECORD #1
:RECORD #2
#RECORD #3
*134*X
WARNING: FOUND EOF IN TOFILE
2 RECORDS PROCESSED *** 0 ERRORS
```
    (8)   Copy the same records to SHORTY but this time use a FROM
          file of $STDIN. Observe the difference between $STDINX &
          $STDIN. The ':' in the second record has signalled an
          end-of-file on FCOPY's input file (for data & commands) so
          FCOPY has returned to the MPE command interpreter. Call
          FCOPY again to proceed with the next step.
```
>FROM=$STDIN;TO=SHORTY
RECORD #1
:RECORD #2
EOF FOUND IN FROMFILE AFTER RECORD 0
1 RECORD PROCESSED *** 0 ERRORS
 END OF PROGRAM
:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
```
    (9)   List the contents of SHORTY on your terminal.
```
>FROM=SHORTY;TO=$STDLIST
RECORD #1
EOF FOUND IN FROMFILE AFTER RECORD 0
1 RECORD PROCESSED *** 0 ERRORS
```
    (10) List the contents of SHORTY on the line printer. Chances
          are you do not have an active FILE command pointing to the
          line printer and chances are this is not the last time you
          will be in this situation so remember:
               a)   Press the 'BREAK' Key.
               b)   When you receive the ':' prompt enter the command
                     FILE LP;DEV=LP
               c)   Key in 'RESUME'. Another '>' prompt will not be
```
>                                                    << BREAK pressed >>
:FILE LP;DEV=LP
:RESUME
READ PENDING
FROM=SHORTY;TO=*LP
*200*
EOF FOUND IN FROMFILE AFTER RECORD 0
```

```
1 RECORD PROCESSED *** 0 ERRORS
   |(11) List MYASCII on the line printer. (expect '*200*')            |
>FROM=MYASCII;TO=*LP
*200*
EOF FOUND IN FROMFILE AFTER RECORD 3
4 RECORDS PROCESSED *** 0 ERRORS
   |(12) List BINARY.PUB on the line printer. (expect '*200*' and      |
   |      '*201*')
>FROM=BINARY.PUB;TO=*LP
*200*
*201*
EOF FOUND IN FROMFILE AFTER RECORD 5
6 RECORDS PROCESSED *** 0 ERRORS
   |(13) Copy from $STDINX to ASCII.PUB. (Look-up resulting error      |
   |      code in your Software Pocket Guide).
>FROM=$STDINX;TO=ASCII.PUB
*106*X
CAN'T OPEN TOFILE
DISPLAY FILE INFORMATION (Y OR N) ?Y
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
!  FILE NUMBER -1       IS UNDEFINED.             !
!  ERROR NUMBER: 93    RESIDUE: 0                 !
!  BLOCK NUMBER: 0             NUMREC: 0          !
+------------------------------------------------+
*103*X
CAN'T CLOSE TOFILE
DISPLAY FILE INFORMATION (Y OR N) ?Y
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
!  FILE NUMBER -1       IS UNDEFINED.             !
!  ERROR NUMBER: 93    RESIDUE: 0                 !
!  BLOCK NUMBER: 0             NUMREC: 0          !
+------------------------------------------------+
0 RECORDS PROCESSED *** 1 ERROR
>EXIT
 END OF PROGRAM
:BYE
CPU (SEC) = 8
CONNECT (MIN) = 31
FRI, MAR  3, 1978,  9:08 AM
```

END OF SESSION
```
    (14) Key in "EXIT" to end FCOPY and log-off the system.
    ****************************************************************
    *        OPTIONAL:  Proceed only if you have extra time.        *
    ****************************************************************


        You have unlimited file access in both your HOME & LOG-ON
group.  Log-on with your user name into your lab partner's group
(if no partner, use group GTEACHER).

        EXAMPLE:   For partners JACK & JILL

                   :HELLO JACK.INTRO,GJILL


    (1)  Copy file SHORTY.PUB into your group as a NEW file called
         'X' followed by your User-name (remember max of 8 chars).
         Specify no group name for the newfile, we'll see where it
         goes by default.
:HELLO STUDENT.INTRO,GTEACHER
ACCT  PASSWORD?
PASSWORD
 SESSION NUMBER = #S230
 FRI, MAR  3, 1978,  9:09 AM
 HP32002A.01.MR
WELCOME TO YOUR FRIENDLY TRAINING HP-3000.
:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>FROM=SHORTY.PUB;TO=XSTUDENT;NEW
EOF FOUND IN FROMFILE AFTER RECORD 1
2 RECORDS PROCESSED *** 0 ERRORS
    (2)  Exit FCOPY and issue the command to list the attributes of
         your file (":LISTF Xyour-user,2").  Observe which of the two
         groups the file went into!
>EXIT
 END OF PROGRAM
:LISTF XSTUDENT,2
ACCOUNT= INTRO          GROUP= GTEACHER
FILENAME  CODE  ------------LOGICAL RECORD----------   ----SPACE----
                SIZE  TYP        EOF       LIMIT R/B  SECTORS #X MX
XSTUDENT        80B   FA          2           2  16       10  1  1
    (3)  Run FCOPY again and copy that new file into the HOME group
         as another NEW file 'Cyour-user'.
```

```
:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>FROM=XSTUDENT;TO=CSTUDENT.GSTUDENT;NEW
EOF FOUND IN FROMFILE AFTER RECORD 1
2 RECORDS PROCESSED *** 0 ERRORS
   (4)  Use LISTF to ascertain its attributes.
   (5)  Log-off the system.
>EXIT
 END OF PROGRAM
:LISTF CSTUDENT.GSTUDENT,2
ACCOUNT=  INTRO        GROUP=  GSTUDENT
FILENAME  CODE   ------------LOGICAL RECORD-----------   ----SPACE----
                 SIZE   TYP        EOF      LIMIT R/B  SECTORS #X MX
CSTUDENT         80B   FA           2          2  16       10  1  1
:BYE
CPU (SEC) = 4
CONNECT (MIN) = 16
FRI, MAR  3, 1978,  9:24 AM
END OF SESSION
```

<< End >>

E D I T O R   L A B   #1   [1.0 hour]

1) Log-on the terminal and invoke the EDITOR anticipating
output to the line printer. Issue FILE commands if
necessary.

:HELLO STUDENT.INTRO
ACCT PASSWORD?
PASSWORD
 SESSION NUMBER = #S212
 THU, MAR 2, 1978, 5:47 PM
 HP32002A.01.MR
:EDITOR LINEP
HP32201A.7.00 EDIT/3000 THU, MAR 2, 1978, 5:49 PM
(C) HEWLETT-PACKARD CO. 1976

2) TEXT in the file LABEDIT1.PUB.
3) Produce an offline listing for reference & go get it off
the line printer.
4) Change "COAL" to "GOAL" in line 7.

/T LABEDIT1.PUB
/L ALL,OFFLINE
*** OFF LINE LISTING BEGUN. ***
/M 7
MODIFY   7
SUCCESSFUL, AND THIS COAL IS THE PRIMARY AIM OF A PROGRAMMER.
                     RG
SUCCESSFUL, AND THIS GOAL IS THE PRIMARY AIM OF A PROGRAMMER.

5) Add a line after line 21 containing just "EFFICIENTLY".
6) Insert two blank lines following line 22.
7) Insert "FOR" in front of "THEM" in line 33.
8) Insert the missing line: "FOR EXAMPLE, IN A PROGRAM
PREPARED TO SOLVE THE INVENTORY PROBLEM" before line 34.

/A 21.1
   21.1   EFFICIENTLY.
   21.2   //
...
/A 22.1
   22.1
   22.2
   22.3   ...
/M 33
MODIFY   33
MANY UNUSUAL CASES AS POSSIBLE AND MAKE ALLOWANCES THEM.
                                              I FOR
MANY UNUSUAL CASES AS POSSIBLE AND MAKE ALLOWANCES FOR THEM.

/A 33.9
   33.9   FOR EXAMPLE, IN A PROGRAM PREPARED TO SOLVE THE INVENTORY PROBLEM
   33.91 //
...
/

```
   9)   Change "NEGSJBIH" to "NEGATIVE" in line 41.
  10)   Change "T-G-G" to "STER" in line 46.
  11)   Delete "(DELETE)" from line 53.
/M 41,46,53
MODIFY      41
NEGSJBIH.
RNEGATIVE
NEGATIVE.


MODIFY      46
DO DATA-PROCESSING JOBS FAT-G-G MORE RELIABLY, AND LESS EXPENSIVELY
                              DDDDDISTER
DO DATA-PROCESSING JOBS FASTER MORE RELIABLY, AND LESS EXPENSIVELY


MODIFY      53
THERFORE, TO PLAN HIS PROGRAM PREPARATION CAREFULLY (DELETE).
                                                    D        D
THERFORE, TO PLAN HIS PROGRAM PREPARATION CAREFULLY.


  12)   Insert a period after "CARD" in line 57 and delete the
        following part of the line.
  13)   Add your name to line 59.
/M 57/59
MODIFY      57
THIS IS THE LAST TEXT CARD999999**************######
                         D                       DI.
THIS IS THE LAST TEXT CARD.


MODIFY      58


MODIFY      59
RJIVE A. STUDENT
JIVE A. STUDENT


/
  14)   Insert the whole paragraph contained in disc file PARA1.PUB
        in front of line 16.  Do NOT affect numbers of lines already
        in the WORK file.  You should be able to do all of this with
        just one command.
/J PARA1.PUB TO 15.9
  15.9   ***************PARA1***************
  15.91  THIS IS THE PARAGRAPH TO BE INSERTED IN
  15.92  FRONT OF LINE 16 IN THE TEXT EDITOR I
  15.93  LAB EXERCISE 2.
  15.94  **********************************
  15)   Lines 49 and 50 are out of place.  With one command move
        them in front of line 56.
  16)   Renumber the file.
  17)   Obtain another listing on the line printer to double-check
```

```
        your changes.
    18) Save the file under the name EDLAB1 in your log-on group.
/G 49/50 TO 55.9
    49      =>      55.9
    50      =>      55.91
/G ALL
*70*WARNING: WORK FILE IS TEMPORARY.  INSUFFICIENT SPACE IN GROUP.
        ****************************************************
        *  NOTE: Work File will be a Temporary file if not *
        * enough space in Group or Account.                *
        ****************************************************
/L ALL,OFFLINE
*** OFF LINE LISTING BEGUN. ***
/K EDLAB1
    ***********************************************************************
    *  The remainder of this LAB is optional.  Only do it if you     *
    * have extra time.  Otherwise proceed to step "END)".            *
    ***********************************************************************
    19) Try to keep another copy as file EDLAB1 in PUB (from where
        you originally read your file for this exercise; you can
        read from PUB but cannot save a file into it).
/K EDLAB1.PUB
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
! FILE NAME IS EDLAB1.PUB.INTRO                   !
! FOPTIONS: NEW,A,*FORMAL*,F,N,DEQ                !
! AOPTIONS:  OUTPUT,SREC,NOLOCK,DEF,BUFFER        !
! DEVICE TYPE: 0       DEVICE SUBTYPE: 8          !
! LDEV: 2          DRT: 4          UNIT: 1        !
! RECORD SIZE: 80     BLOCK SIZE: 1280  (BYTES)   !
! EXTENT SIZE: 30     MAX EXTENTS: 1              !
! RECPTR: 70          RECLIMIT: 70               !
! LOGCOUNT: 70          PHYSCOUNT: 5              !
! EOF AT: 70          LABEL ADDR: %00200263017    !
! FILE CODE: 0     ID IS STUDENT    ULABELS: 0   !
! PHYSICAL STATUS: 1000000000000001               !
! ERROR NUMBER: 93    RESIDUE: 640       (WORDS)  !
! BLOCK NUMBER: 5               NUMREC: 16        !
+-------------------------------------------------+
*60*J
FCLOSE FAILURE (93)
    20) The filename specified in the KEEP command is a
        "filereference".  So keep another copy as file EDLOCK1 with
        a LOCKWORD.
    21) TEXT EDLOCK1 back in specifying an incorrect lockword.  Look
        up the resulting error code in your pocket guide.  TEXT it
        in again supplying the correct lockword.
/K EDLOCK1/LOCKWORD
/T EDLOCK1
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
```

LOCKWORD: EDLOCK1.GSTUDENT.INTRO?
WRONGONE
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
! ERROR NUMBER: 92   RESIDUE: 0   !
! BLOCK NUMBER: 0   NUMREC: 0   !
+--+
*23*J
FAILURE TO OPEN TEXT FILE   (92)
/T EDLOCK1/LOCKWORD
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
/
    22) Keep it again as EDLOCK1 but supply a different lockword.
    23) Obtain an XPLAIN listing of all commands offline on the line
         printer. (Be sure to pick up all of your listings from the
         line printer)
/K EDLOCK1/ANOTHER
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
! ERROR NUMBER: 92   RESIDUE: 0   !
! BLOCK NUMBER: 0   NUMREC: 0   !
+--+
*41*J
FAILURE TO OPEN KEEP FILE   (92)
/X ALL,OFFLINE
*** OFF LINE LISTING BEGUN. ***
/EXIT
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
 END OF SUBSYSTEM
:BYE
CPU (SEC) = 33
CONNECT (MIN) = 51
THU, MAR 2, 1978, 6:57 PM
END OF SESSION
    END) Exit from the EDITOR and log-off.

                                    << End >>

        E D I T O R   L A B   #2 [1.0 hour]

        To use tabbing with EDIT/3000 on an HP-264x terminal, you
must: 1) Enable the TABCHAR and TAB stops in the EDITOR with

the SET command & 2) Set the TAB stops in the terminal either
physically or with escape characters.
     There are predefined USE files in PUB.INTRO that do all
this. They are: 1) For COBOL, COBTABS 2) For RPG, RPGTABS
& 3) For default formats, DEFTABS.
     As an example of using TABs we are going to set tabs for
COBOL programs then modify a COBOL program. This demonstrates
TABs and does not require any knowledge of COBOL.

1) Log-on & invoke the EDITOR.
2) USE COBTABS.PUB. This will enable the EDITOR program to
    recognize the TAB Key and set corresponding TAB stops in
    the program and in the terminal. An extra goody is it also
    locks a picture of record positions and tab stops at the
    top of your terminal screen (notice MEMORY LOCK is on).
3) VERIFY format settings and tab settings with VERIFY ALL.
4) TEXT in COBTEST1.PUB (This is a copy of COBOL program from
    "Using the HP-3000").
5) List the program. We want to indent line 3.6 ("IF Y-N =
    "N" GO TO ENTER-ROUTINE.) to the next tab stop. Try to
    use MODIFY to insert 4 additional spaces by pressing the
    TAB Key and keying in I followed by 4 spaces. You get the
    'INVALID' message because the 1-st character encountered is
    the TABCHAR, not 'R', 'D', nor 'I'. The TAB key is only
    recognized by the ADD and REPLACE commands.
6) The logical thing to do would be to insert 4 spaces within
    MODIFY without using the TAB key, but let's practice using
    the REPLACE command. So REPLACE line 3.6 with its same
    contents but indented to the next tab stop.
7) Now ADD line 3.61. Also indent it to the second tab stop
    and enter the contents "IF Y-N = "n" GO TO ENTER-ROUTINE.".
8) KEEP the file both numbered and unnumbered in your group
    as EDLAB2 & EDLAB2U.
9) Press the <BREAK> Key and when you receive the ":" prompt
    use LISTF,1 to look at the record sizes of EDLAB2 and
    EDLAB2U. Observe that 6 additional characters are added to
    each line of the file when it is kept numbered. Also
    observe the record sizes of files TRY1 & TRY1UNN in PUB.
    These are numbered and unnumbered versions of the FORTRAN
    program from chapter 3 of "Using the HP-3000".
10) Key in "RESUME". You are now back in the EDITOR (remember
    the "/" prompt will NOT be re-issued). Exit the EDITOR.
11) RUN FCOPY.PUB.SYS and list all four files to your terminal
    screen. Observe where line numbers are put for the
    different formats and the length of the line numbers.
12) EXIT FCOPY and Log-off the system.

****
*   OPTIONAL -- Proceed only if you have time.   *
****

EDITOR LAB #2 (cont'd)

Obtain a listing on the line printer of any of these tab use files in
PUB you are interested in transferring to your system. All contain
CNTL characters and lower case characters that will not be represented
correctly on the line printer. Before listing it, text it into
your Editor work file and change the ESCAPE character to '!'
everywhere it is found in the file. List it on the line printer.
If the line printer does not have lower case characters, all
lower case characters will print as upper case. Get your listing
from the line printer, list the work file on your terminal screen
(you must have DISPLAY FUNCTIONS on or your terminal will attempt
to execute ESCAPE characters rather than list them) & circle the
characters on your printed listing that should be lower case.

<< End >>

LAB SOLUTIONS 16

F I L E S   L A B   #1 [1.0 hour]

> Remember unlimited file access is granted in both your log-on and home group. So log-on as your lab partner into your group. For 'YOU' and 'PARTNER':
>
> :HELLO PARTNER.INTRO/PASSWORD,GYOU
>
> If you don't have a lab partner use 'TEACHER':
>
> :HELLO TEACHER.INTRO/PASSWORD,GYOU
>
> Proceed with the lab filling in the questions as you go. Don't dwell too long on any question; they will be covered later.
>
> See your "Software Pocket Guide" for complete command syntax.
>
> 1) Compile the COBOL program COBTEST1 in PUB. Enter 'COBOL COBTEST1.PUB' and use the default USL and list files.
>
> ??? By default the listing is on ??? $STDLIST
>   ------------

```
:HELLO TEACHER.INTRO/PASSWORD,GSTUDENT
 SESSION NUMBER = #S53
 MON, MAR  6, 1978,  2:45 PM
 HP32002A.01.MR
WELCOME TO YOUR FRIENDLY HP-3000.
:COBOL COBTEST1.PUB
PAGE 0001   HP32213C.02.00  (C) HEWLETT-PACKARD CO. 1977
        001100 IDENTIFICATION DIVISION.
        001200 PROGRAM-ID.  COBOL-TEST1.
        001300 AUTHOR.  STUDENT.INTRO.
        001400 ENVIRONMENT DIVISION.
        001500 DATA DIVISION.
        001600 WORKING-STORAGE SECTION.
        001700 77  EDIT-FIELD      PIC $Z,ZZ9.99.
        001800 77  TOTAL-COST      PIC 999V99.
        001900 77  COST-OF-SALE    PIC 99V99.
        002000 77  TAX             PIC 99V99.
        002100 77  Y-N             PIC X.
        002200
        002300 PROCEDURE DIVISION.
        002400 ENTER-ROUTINE.
        002500     MOVE ZEROS TO TOTAL-COST.
        002600     DISPLAY SPACE.
        002700     DISPLAY "ENTER COST OF SALE (BEFORE TAX) NO DECIMAL PT".
```

```
002800        DISPLAY "(4 DIGITS MAX) INCLUDE LEADING ZEROS!".
002900        ACCEPT COST-OF-SALE.
003000        COMPUTE TAX = COST-OF-SALE * .06.
003100        ADD COST-OF-SALE, TAX TO TOTAL-COST.
003200        MOVE TOTAL-COST TO EDIT-FIELD.
003300        DISPLAY "TOTAL COST OF PURCHASE = " EDIT-FIELD.
003400        DISPLAY "ARE YOU FINISHED? (Y OR N)".
003500        ACCEPT Y-N.
003600        IF Y-N = "N" GO TO ENTER-ROUTINE.
003700        STOP RUN.
   DATA AREA IS %000341 WORDS.
   CPU TIME = 0:00:01.  WALL TIME = 0:00:20.
END COBOL/3000 COMPILATION.  NO ERRORS.  NO WARNINGS.
 END OF COMPILE
```

2)   Issue ':LISTF,1' to see if the USL file is a permanent file.
':RUN LISTEQ2.PUB.SYS'; it's not listed among the normal temporary
files either. The result is in file $OLDPASS. The only way we
can determine the attributes of $OLDPASS is save it then LISTF.
Enter ':SAVE $OLDPASS,XYZ' , if XYZ already exists, rename it.
Now do a ':LISTF XYZ,2' and notice which group you're using.

???   Into what file does object output go by default??? $OLDPASS

```
:LISTF,1
:RUN LISTEQ2.PUB.SYS
LISTEQ2 A01.01 (C) HEWLETT-PACKARD CO., 1976
***NO TEMP FILES
***NO FILE EQUATIONS
 END OF PROGRAM
:SAVE $OLDPASS,XYZ
:LISTF XYZ,2
ACCOUNT= INTRO          GROUP= GSTUDENT
FILENAME  CODE   ------------LOGICAL RECORD-----------  ----SPACE----
                 SIZE  TYP        EOF       LIMIT R/B  SECTORS #X MX
XYZ       USL    128W  FB         131        1023  1      512  1  2
```

3)   The compilers and :PREP will create files. Re-compile
COBTEST1.PUB into a uslfile of 'USL' and use a listfile of
'$NULL' (this is one way to inhibit listings). Now enter
':PREP USL' and re-examine the :PREP command's syntax in your

pocket guide carefully; the progfile is required! You can
use the default work file in the following manner, enter:
':PREP USL,$NEWPASS'. As $NEWPASS is closed its name is changed
to $OLDPASS. Now run the program file that has resulted from
the :PREP. Rename $OLDPASS to 'PROG1' (remember its a TEMP
file). RUN LISTEQ2 to see it has taken affect. Make PROG1 a
permanent file.

??? In the :PREP command you must specify both   YES
    'uslfile' and 'progfile' (Yes/No) ???   ----------
??? What System-defined file can be used to eliminate $NULL
    compiler listings ???   ----------

```
:COBOL COBTEST1.PUB,USL,$NULL
PAGE 0001   HP32213C.02.00  (C) HEWLETT-PACKARD CO. 1977
    DATA AREA IS %000341 WORDS.
    CPU TIME = 0:00:01.  WALL TIME = 0:00:09.
END COBOL/3000 COMPILATION.  NO ERRORS.  NO WARNINGS.
 END OF COMPILE
:PREP USL
ERR 21,2 J
PARAMETER NOT OPTIONAL
:PREP USL,$NEWPASS
 END OF PREPARE
:RUN $OLDPASS
ENTER COST OF SALE (BEFORE TAX) NO DECIMAL PT
(4 DIGITS MAX) INCLUDE LEADING ZEROS!
1000
TOTAL COST OF PURCHASE = $     10.60
ARE YOU FINISHED? (Y OR N)
Y
 END OF PROGRAM
:RENAME $OLDPASS,PROG1,TEMP
:RUN LISTEQ2.PUB.SYS
LISTEQ2 A01.01 (C) HEWLETT-PACKARD CO., 1976
***TEMP FILES
PROG1.GSTUDENT.INTRO
***NO FILE EQUATIONS
```

```
END OF PROGRAM
:SAVE PROG1
│ 4)    Compile and PREP COBTEST1.PUB in one step with COBOLPREP,
│ use defaults for 'uslfile' and 'listfile'.  Rename $OLDPASS to
│ XYZ (remember TEMP).  We have XYZ, a USL file in the Permanent
│ domain from step 1) and XYZ, a program file in the Temporary
│ domain.  Now RUN XYZ.  Rename temporary file XYZ to XYZ2.  Try
│ to run XYZ again.
│
│ ???  RUN searches which domain first for progfiles???  Temporary
│                                                        ------------
│ ???  In COBOLPREP what is default for uslfile???         $OLDPASS
│                                                        ------------
│ ???  In COBOLPREP what is default for progfile???        $NEWPASS
│                                                        ------------
│ ???  Where is uslfile at completion of COBOLPREP???      deleted
│                                                        ------------
:COBOLPREP COBTEST1.PUB
PAGE 0001    HP32213C.02.00   (C) HEWLETT-PACKARD CO. 1977
           001100 IDENTIFICATION DIVISION.
           001200 PROGRAM-ID.  COBOL-TEST1.
           001300 AUTHOR.  STUDENT.INTRO.
           001400 ENVIRONMENT DIVISION.
           001500 DATA DIVISION.
           001600 WORKING-STORAGE SECTION.
           001700 77  EDIT-FIELD      PIC $Z,ZZ9.99.
           001800 77  TOTAL-COST      PIC 999V99.
           001900 77  COST-OF-SALE    PIC 99V99.
           002000 77  TAX             PIC 99V99.
           002100 77  Y-N             PIC X.
           002200
           002300 PROCEDURE DIVISION.
           002400 ENTER-ROUTINE.
           002500     MOVE ZEROS TO TOTAL-COST.
           002600     DISPLAY SPACE.
           002700     DISPLAY "ENTER COST OF SALE (BEFORE TAX) NO DECIMAL PT"
           002800     DISPLAY "(4 DIGITS MAX) INCLUDE LEADING ZEROS!".
           002900     ACCEPT COST-OF-SALE.
           003000     COMPUTE TAX = COST-OF-SALE * .06.
           003100     ADD COST-OF-SALE, TAX TO TOTAL-COST.
           003200     MOVE TOTAL-COST TO EDIT-FIELD.
           003300     DISPLAY "TOTAL COST OF PURCHASE = " EDIT-FIELD.
           003400     DISPLAY "ARE YOU FINISHED? (Y OR N)".
```

```
       003500      ACCEPT Y-N.
       003600      IF Y-N = "N" GO TO ENTER-ROUTINE.
       003700      STOP RUN.
```
    DATA AREA IS %000341 WORDS.
    CPU TIME = 0:00:01. WALL TIME = 0:00:12.
END COBOL/3000 COMPILATION. NO ERRORS. NO WARNINGS.
 END OF COMPILE
 END OF PREPARE
:RENAME $OLDPASS,XYZ,TEMP
:RUN XYZ
ENTER COST OF SALE (BEFORE TAX) NO DECIMAL PT
(4 DIGITS MAX) INCLUDE LEADING ZEROS!
2000
TOTAL COST OF PURCHASE = $ 21.20
ARE YOU FINISHED? (Y OR N)
Y
 END OF PROGRAM
:RENAME XYZ,XYZ2,TEMP
:RUN XYZ
ERR 208 J
INVALID PROGRAM FILE

> 5)   XYZ2 is your progfile in the Temporary domain. Make it a
> Permanent file named 'PROG2'.
> Enter: ':RENAME PROG2,PROG2/LOCKWORD'. Now RUN PROG2 and see
> that a lockword has been put on it. Remove the lockword. Both
> files referenced in the :RENAME command are filereferences so
> rename PROG2 into your home group thusly:
>
>   ':RENAME PROG2,your-user-name.Gpartner'
>   (or use group GTEACHER if 'TEACHER' is your partner)
>
> Do a LISTF and see that PROG2 is no longer in the log-on group.
> Now do a LISTF @.Gpartner,1 and find it; it has been moved to
> another group! Rename it back into its original group.

:SAVE XYZ2
:RENAME XYZ2,PROG2
:RENAME PROG2,PROG2/LOCKWORD
:RUN PROG2
LOCKWORD: PROG2.GSTUDENT.INTRO?
WRONG
ERR 217,92 J
PROGRAM FILE ACCESS ERROR
:RENAME PROG2,PROG2
LOCKWORD: PROG2.GSTUDENT.INTRO?

```
LOCKWORD
:RUN PROG2
ENTER COST OF SALE (BEFORE TAX) NO DECIMAL PT
(4 DIGITS MAX) INCLUDE LEADING ZEROS!
3000
TOTAL COST OF PURCHASE = $    31.80
ARE YOU FINISHED? (Y OR N)
Y
 END OF PROGRAM
:RENAME PROG2,PROG2.GTEACHER
:LISTF
FILENAME
PROG1        USL           XYZ
:LISTF @.GTEACHER,1
ACCOUNT=  INTRO         GROUP=  GTEACHER
FILENAME   CODE    -----------LOGICAL RECORD--------
                   SIZE  TYP       EOF       LIMIT
PROG2      PROG    128W  FB         9           9
:RENAME PROG2.GTEACHER,PROG2
:LISTF,1
ACCOUNT=  INTRO         GROUP=  GSTUDENT
FILENAME   CODE    -----------LOGICAL RECORD--------
                   SIZE  TYP       EOF       LIMIT
PROG1      PROG    128W  FB         9           9
PROG2      PROG    128W  FB         9           9
USL        USL     128W  FB       131        1023
XYZ        USL     128W  FB       131        1023
```

6) Run FCOPY and copy COBTEST1.PUB into your group as
'COBTEST'. RUN LISTDIR2.PUB.SYS. This is a program that will let
you see all security settings you are allowed to see. When you
receive the '>' prompt, key in 'LISTSEC COBTEST;PASS'. Notice who
the CREATOR is!   Exit LISTDIR2.

??? Who is the CReating user of COBTEST & PROG2???   partner

```
:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>FROM=COBTEST1.PUB;TO=COBTEST;NEW
EOF FOUND IN FROMFILE AFTER RECORD 27
```

```
28 RECORDS PROCESSED *** 0 ERRORS
>EXIT
 END OF PROGRAM
:RUN LISTDIR2.PUB.SYS
LISTDIR2 A01.01 (C) HEWLETT-PACKARD CO., 1976
TYPE 'HELP' FOR AID
>LISTSEC COBTEST;PASS
*******************
FILE: COBTEST.GSTUDENT.INTRO
SECURITY--READ:      AC
 (ACCT)   WRITE:     AC
          APPEND:    AC
          LOCK:      AC
          EXECUTE:   AC
SECURITY--READ:      GU
(GROUP)   WRITE:     GU
          APPEND:    GU
          LOCK:      GU
          EXECUTE:   GU
          SAVE:      GU
SECURITY--READ:      ANY                 FCODE: 1052
 (FILE)   WRITE:     ANY                 CREATOR: TEACHER
          APPEND:    ANY                 LOCKWORD: **
          LOCK:      ANY               **SECURITY IS ON
          EXECUTE:   ANY
FOR TEACHER.INTRO: READ,WRITE,APPEND,LOCK,EXECUTE
>EXIT
 END OF PROGRAM
   ******************************************************************
   *      OPTIONAL -- Proceed only if you have extra time.         *
   ******************************************************************

   7)    Now log-on under your user id (i.e., 'HELLO you.INTRO').  Try
   to rename COBTEST in your group to 'CONFLICT'.  You have experienced
   "CREATOR   C O N F L I C T".  Try to ':RELEASE' PROG2 and the
   same thing will happen.  Now RUN LISTDIR2 again and LISTSEC for
   COBTEST as above.  It won't show you who the CREATOR is.  This
   information is only given out to the CR, the AM or the SM.  Exit
   LISTDIR2.

   ???  A file may only be ':RELEASED', ':RENAMED', or ':SECUREd'
```

by the ???	creator

```
:HELLO STUDENT.INTRO/PASSWORD
CPU (SEC) = 16
CONNECT (MIN) = 27
MON, MAR  6, 1978,  3:12 PM
END OF SESSION
 SESSION NUMBER = #S61
 MON, MAR  6, 1978,  3:12 PM
 HP32002A.01.MR
WELCOME TO YOUR FRIENDLY HP-3000.
:RENAME COBTEST,CONFLICT
ERR 120 X
CREATOR CONFLICT
:RELEASE COBTEST
ERR 120 X
CREATOR CONFLICT
:RUN LISTDIR2.PUB.SYS
LISTDIR2 A01.01 (C) HEWLETT-PACKARD CO., 1976
TYPE 'HELP' FOR AID
>LISTSEC COBTEST;PASS
********************
FILE: COBTEST.GSTUDENT.INTRO
SECURITY--READ:      AC
 (ACCT)   WRITE:     AC
          APPEND:    AC
          LOCK:      AC
          EXECUTE:   AC
SECURITY--READ:      GU
(GROUP)   WRITE:     GU
          APPEND:    GU
          LOCK:      GU
          EXECUTE:   GU
          SAVE:      GU
SECURITY--READ:      ANY              FCODE: 1052
 (FILE)   WRITE:     ANY             CREATOR: **
          APPEND:    ANY             LOCKWORD: **
          LOCK:      ANY          **SECURITY IS ON
          EXECUTE:   ANY
FOR STUDENT.INTRO: READ,WRITE,APPEND,LOCK,EXECUTE
>EXIT
```

END OF PROGRAM
:

8)   Log-on as your partner or as TEACHER, but use his/her HOME
group this time (the default). Run FCOPY and list the contents of
COBTEST in your own HOME group on your terminal. Display the tomb-
stone and look-up the error codes in your pocket guide. Press the
BREAK key and when you receive the ':' prompt, 'RELEASE' COBTEST
in your own HOME group.
    Key in 'RESUME', remember FCOPY's prompt will not be re-
issued. Try to list COBTEST from the other group on your
terminal again. Success! Now write from your terminal ($STDINX)
to COBTEST in the other group (;NEW option not needed since file
already exists). Just key in 2 trivial records that strike your
fancy, press <CNTL-Y> to shut off entry, and EXIT FCOPY. Use
LISTF ,2 to check the current length of COBTEST. PURGE COBTEST
from the other group. Look for it with LISTF but you will not
find it. Log-off system.

???   When a file is RELEASEd, what restrictions are made as to
what users may access it in any way they choose???   none!!!

HELLO TEACHER.INTRO
CPU (SEC) = 2
CONNECT (MIN) = 2
MON, MAR 6, 1978, 3:24 PM
END OF SESSION
ACCT PASSWORD?
PASSWORD
 SESSION NUMBER = #S64
 MON, MAR 6, 1978, 3:25 PM
 HP32002A.01.MR
WELCOME TO YOUR FRIENDLY HP-3000.
:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>FROM=COBTEST.GSTUDENT;TO=$STDLIST
*105*X
CAN'T OPEN FROMFILE
DISPLAY FILE INFORMATION (Y OR N) ?Y
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
! FILE NUMBER -15333 IS UNDEFINED.   !
! ERROR NUMBER: 93   RESIDUE: 0   !
! BLOCK NUMBER: 0   NUMREC: 0   !

```
+---------------------------------------------------+
*102*J
CAN'T CLOSE FROMFILE
DISPLAY FILE INFORMATION (Y OR N) ?Y
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
! FILE NUMBER -15333 IS UNDEFINED.               !
! ERROR NUMBER: 93    RESIDUE: 0                 !
! BLOCK NUMBER: 0            NUMREC: 0           !
+---------------------------------------------------+
0 RECORDS PROCESSED *** 1 ERROR
>                                      << BREAK pressed >>
:RELEASE COBTEST.GSTUDENT
:RESUME
READ PENDING
FROM=COBTEST.GSTUDENT;TO=$STDLIST
001000$CONTROL USLINIT,SOURCE
001100 IDENTIFICATION DIVISION.
001200 PROGRAM-ID.  COBOL-TEST1.
001300 AUTHOR.  STUDENT.INTRO.
001400 ENVIRONMENT DIVISION.
001500 DATA DIVISION.
001600 WORKING-STORAGE SECTION.
001700 77  EDIT-FIELD    PIC $Z,ZZ9.99.
001800 77  TOTAL-COST    PIC 999V99.
001900 77  COST-OF-SALE  PIC 99V99.
002000 77  TAX           PIC 99V99.
002100 77  Y-N           PIC X.
002200
002300 PROCEDURE DIVISION.
002400 ENTER-ROUTINE.
002500     MOVE ZEROS TO TOTAL-COST.
002600     DISPLAY SPACE.
002700     DISPLAY "ENTER COST OF SALE (BEFORE TAX) NO DECIMAL PT".
002800     DISPLAY "(4 DIGITS MAX) INCLUDE LEADING ZEROS!".
002900     ACCEPT COST-OF-SALE.
003000     COMPUTE TAX = COST-OF-SALE * .06.
003100     ADD COST-OF-SALE, TAX TO TOTAL-COST.
003200     MOVE TOTAL-COST TO EDIT-FIELD.
003300     DISPLAY "TOTAL COST OF PURCHASE = " EDIT-FIELD.
003400     DISPLAY "ARE YOU FINISHED? (Y OR N)".
003500     ACCEPT Y-N.
003600     IF Y-N = "N" GO TO ENTER-ROUTINE.
003700     STOP RUN.
EOF FOUND IN FROMFILE AFTER RECORD 27
28 RECORDS PROCESSED *** 0 ERRORS
>FROM=$STDINX;TO=COBTEST.GSTUDENT
```

1 RECORD THAT STRIKES MY FANCY
RECORD 2 THAT STRIKES MY FANCY
< CONTROL Y >
3 RECORDS PROCESSED *** 0 ERRORS
>EXIT
 END OF PROGRAM
:LISTF COBTEST.GSTUDENT,2
ACCOUNT= INTRO   GROUP= GSTUDENT
FILENAME   CODE   ------------LOGICAL RECORD----------   ----SPACE----
                   SIZE TYP   EOF   LIMIT R/B SECTORS #X MX
COBTEST   1052   80B   FA   2   28 16   15 1 1
:PURGE COBTEST.GSTUDENT
:LISTF COBTEST.GSTUDENT,2
ERR 108 J
NON-EXISTENT FILE
:BYE
CPU (SEC) = 3
CONNECT (MIN) = 5
MON, MAR 6, 1978, 3:29 PM
END OF SESSION
                                          << End >> |

          F I L E S   L A B   #2 [0.5 hour]

OBSERVING CHARACTERISTICS OF F, V, AND U FORMAT FILES.

1)   Log-on to the system and enter the following file commands:

:FILE FIXED,NEW   ;REC=-128,2,F,ASCII;DISC=4,1,1;SAVE
:FILE VARIABLE,NEW;REC=-128,2,V,ASCII;DISC=4,1,1;SAVE
:FILE UNDEFINE,NEW;REC=-128,2,U,ASCII;DISC=4,1,1;SAVE

2)   Double-check to make sure you have entered them correctly
by running LISTEQ2.

:HELLO STUDENT.INTRO
ACCT PASSWORD?
PASSWORD
 SESSION NUMBER = #S66
 MON, MAR 6, 1978, 3:51 PM

```
 HP32002A.01.MR
WELCOME TO YOUR FRIENDLY HP-3000.
:FILE FIXED,NEW     ;REC=-128,2,F,ASCII;DISC=4,1,1;SAVE
:FILE VARIABLE,NEW;REC=-128,2,V,ASCII;DISC=4,1,1;SAVE
:FILE UNDEFINE,NEW;REC=-128,2,U,ASCII;DISC=4,1,1;SAVE
:RUN LISTEQ2.PUB.SYS
LISTEQ2 A01.01 (C) HEWLETT-PACKARD CO., 1976
***NO TEMP FILES
***FILE EQUATIONS
:FILE FIXED,NEW;REC=-128,2,F,ASCII;DISC=4,1,1;SAVE
:FILE VARIABLE,NEW;REC=-128,2,V,ASCII;DISC=4,1,1;SAVE
:FILE UNDEFINE,NEW;REC=-128,2,U,ASCII;DISC=4,1,1;SAVE
 END OF PROGRAM
   3)    Now use FCOPY and back-references to create each of these
   files with 3 records in each one.  Following this scenario
   exactly:

   >FROM=$STDINX;TO=*FIXED
   *200*J
   WARNING: FROMFILE RECSIZE IS 80 BYTES, TOFILE RECSIZE IS 128
   BYTES.
   CONTINUE OPERATION (Y OR N) ?Y
   RECORD 1 -- FIXED FILE
   RECORD 2 -- FIXED FILE
   END-OF-FILE FIXED
    < CONTROL Y >          << displayed when CNTL-Y pressed >>

   4 RECORDS PROCESSED *** 0 ERRORS

   >FROM=$STDINX;TO=*VARIABLE
   *200*
   RECORD 1 -- VARIABLE FILE
   RECORD 2 -- VARIABLE FILE
   END-OF-FILE VARIABLE
    < CONTROL Y >

   4 RECORDS PROCESSED *** 0 ERRORS

   >FROM=$STDINX;TO=*UNDEFINE
   *200*
   RECORD 1 -- UNDEFINED FILE
   RECORD 2 -- UNDEFINED FILE
   END-OF-FILE UNDEFINED
    < CONTROL Y >
```

```
4 RECORDS PROCESSED *** 0 ERRORS

:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>FROM=$STDINX;TO=*FIXED
*200*X
WARNING: FROMFILE RECSIZE IS 80 BYTES, TOFILE RECSIZE IS 128 BYTES.
CONTINUE OPERATION (Y OR N) ?Y
RECORD 1 -- FIXED FILE
RECORD 2 -- FIXED FILE
END-OF-FILE FIXED
 < CONTROL Y >
4 RECORDS PROCESSED *** 0 ERRORS
>FROM=$STDINX;TO=*VARIABLE
*200*Y
WARNING: FROMFILE RECSIZE IS 80 BYTES, TOFILE RECSIZE IS 256 BYTES.
CONTINUE OPERATION (Y OR N) ?Y
RECORD 1 -- VARIABLE FILE
RECORD 2 -- VARIABLE FILE
END-OF-FILE VARIABLE
 < CONTROL Y >
4 RECORDS PROCESSED *** 0 ERRORS
>FROM=$STDINX;TO=*UNDEFINE
*200*Z
WARNING: FROMFILE RECSIZE IS 80 BYTES, TOFILE RECSIZE IS 128 BYTES.
CONTINUE OPERATION (Y OR N) ?Y
RECORD 1 -- UNDEFINED FILE
RECORD 2 -- UNDEFINED FILE
END-OF-FILE UNDEFINED
 < CONTROL Y >
4 RECORDS PROCESSED *** 0 ERRORS
```

4)   Now list the contents of each file on your terminal. Expect
warning '*200*' and merely press <RETURN> to proceed. You should
see that the last word of undefined records is indeed propogated
through the remainder of the record.

```
>FROM=FIXED;TO=$STDLIST
*200*
RECORD 1 -- FIXED FILE
RECORD 2 -- FIXED FILE
END-OF-FILE FIXED
EOF FOUND IN FROMFILE AFTER RECORD 2
```

FILES LAB #2 (cont'd)


```
3 RECORDS PROCESSED *** 0 ERRORS
>FROM=VARIABLE;TO=$STDLIST
*200*
RECORD 1 -- VARIABLE FILE
RECORD 2 -- VARIABLE FILE
END-OF-FILE VARIABLE
EOF FOUND IN FROMFILE AFTER RECORD 2
3 RECORDS PROCESSED *** 0 ERRORS
>FROM=UNDEFINE;TO=$STDLIST
*200*
RECORD 1 - UNDEFINED FILELELELELELELELELELELELELELELELELELELELELELELELELELE
RECORD 2 - UNDEFINED FILELELELELELELELELELELELELELELELELELELELELELELELELELE
END-OF-FILE UNDEFINED D D D D D D D D D D D D D D D D D D D D D D D D D D D
EOF FOUND IN FROMFILE AFTER RECORD 2
3 RECORDS PROCESSED *** 0 ERRORS
```
|
     5)   Exit FCOPY and do a 'LISTF ,2' for each file individually.
     Observe what record length has been set for file VARIABLE in its
     disc file label.
|

```
>EXIT
 END OF PROGRAM
:LISTF FIXED,2
ACCOUNT= INTRO        GROUP= GSTUDENT
FILENAME  CODE   ------------LOGICAL RECORD-----------    ----SPACE----
                 SIZE   TYP        EOF      LIMIT R/B  SECTORS #X MX
FIXED            128B   FA          3          4    2        3  1  1
:LISTF VARIABLE,2
ACCOUNT= INTRO        GROUP= GSTUDENT
FILENAME  CODE   ------------LOGICAL RECORD-----------    ----SPACE----
                 SIZE   TYP        EOF      LIMIT R/B  SECTORS #X MX
VARIABLE         256B   VA          3          4    1       10  1  1
:LISTF UNDEFINE,2
ACCOUNT= INTRO        GROUP= GSTUDENT
FILENAME  CODE   ------------LOGICAL RECORD-----------    ----SPACE----
                 SIZE   TYP        EOF      LIMIT R/B  SECTORS #X MX
```

```
UNDEFINE            128B   UA              3           4   1         5   1   1
:BYE
CPU (SEC) = 4
CONNECT (MIN) = 13
MON, MAR  6, 1978,  4:04 PM
END OF SESSION
```

6)   Fill in the following table from information obtained from the LISTF displays:

For file:	Record Length (in bytes)	Blocking Factor	Block Length (& Buffer len) (in bytes)
FIXED	128	2	256
VARIABLE	256	1	260
UNDEFINE	128	1	128

<< End >>

F I L E S   L A B   #3 [0.5 hour]
----

1)   Use the EDITOR to create a file. Enter three records to your liking and keep it as a file called 'TEMP' with default of numbered. Exit the Editor.

```
:HELLO STUDENT.INTRO/PASSWORD
 SESSION NUMBER = #S79
 MON, MAR  6, 1978,  4:45 PM
 HP32002A.01.MR
WELCOME TO YOUR FRIENDLY HP-3000.
:EDITOR
HP32201A.7.00 EDIT/3000  MON, MAR  6, 1978,  4:46 PM
(C) HEWLETT-PACKARD CO. 1976
/A
```

```
     1      RECORD 1 -- PERMANENT FILE 'TEMP'.
     2      RECORD 2 -- I LIKE THIS ONE.
     3      RECORD 3 -- END-OF-FILE.
     4      ...
/K TEMP
/E
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
 END OF SUBSYSTEM
    2)    Now :BUILD a temporary file called 'TEMP' with 80 byte
    records blocked 16, fixed and ASCII and DISC=100.

:BUILD TEMP;REC=-80,16,F,ASCII;DISC=100;TEMP
    3)    Enter the following :FILE command:

         :FILE TEMP,NEW;REC=-80,16,F,ASCII;DISC=100;SAVE

:FILE TEMP,NEW;REC=-80,16,F,ASCII;DISC=100;SAVE
    4)    Run FCOPY and copy from $STDINX to '*TEMP'.  Do NOT specify
    this as a ;NEW file, as that has already been done in the :FILE
    command we are using.  Enter several records that meet your high
    standards and signal the end of your file by pressing <CNTL-Y>.
    You will get an error number.  Enter a printing character to get
    a tombstone and use your pocket guide to interpret the error.
    Exit FCOPY.

:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>FROM=$STDINX;TO=*TEMP
RECORD 1 -- TO EXCEEDINGLY HIGH STANDARDS
RECORD 2 -- TEMPORARY FILE 'TEMP'.
RECORD 3 -- END
 < CONTROL Y >
*103*Q
CAN'T CLOSE TOFILE
DISPLAY FILE INFORMATION (Y OR N) ?Y
```

```
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
! FILE NAME IS TEMP.GSTUDENT.INTRO               !
! FOPTIONS: NEW,A,*FORMAL*,F,N,FEQ               !
! AOPTIONS:  OUTPUT,SREC,NOLOCK,DEF,BUFFER       !
! DEVICE TYPE: 0      DEVICE SUBTYPE: 8          !
! LDEV: 1         DRT: 4            UNIT: 0       !
! RECORD SIZE: 80    BLOCK SIZE: 1280  (BYTES) !
! EXTENT SIZE: 5     MAX EXTENTS: 8              !
! RECPTR: 3          RECLIMIT: 100               !
! LOGCOUNT: 3          PHYSCOUNT: 1              !
! EOF AT: 3            LABEL ADDR: %00100052024  !
! FILE CODE: 0       ID IS STUDENT    ULABELS: 0 !
! PHYSICAL STATUS: 1000000000000000              !
! ERROR NUMBER: 100   RESIDUE: 640        (WORDS) !
! BLOCK NUMBER: 1              NUMREC: 16         !
+----------------------------------------------+
4 RECORDS PROCESSED *** 1 ERROR
>EXIT
 END OF PROGRAM
    5)   Undeterred, enter the following :FILE command and use
    LISTEQ2 to make sure you do it right:
        ':FILE TEMP,NEW;REC=-80,16,F,ASCII;DISC=100;TEMP'

:FILE TEMP,NEW;REC=-80,16,F,ASCII;DISC=100;TEMP
:RUN LISTEQ2.PUB.SYS
LISTEQ2 A01.01 (C) HEWLETT-PACKARD CO., 1976
***TEMP FILES
TEMP.GSTUDENT.INTRO
***FILE EQUATIONS
:FILE TEMP,NEW;REC=-80,16,F,ASCII;DISC=100;TEMP
 END OF PROGRAM
    6)   Do exactly the same process as in step 4, even down to the
    interpretation of the error.  Exit FCOPY.

:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>FROM=$STDINX;TO=*TEMP
RECORD 1 -- NEW FILE 'TEMP' DISPOSTION TO BE TEMPORARY.
END
 < CONTROL Y >
```

```
*103*P
CAN'T CLOSE TOFILE
DISPLAY FILE INFORMATION (Y OR N) ?Y
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
! FILE NAME IS TEMP.GSTUDENT.INTRO               !
! FOPTIONS: NEW,A,*FORMAL*,F,N,FEQ               !
! AOPTIONS:  OUTPUT,SREC,NOLOCK,DEF,BUFFER       !
! DEVICE TYPE: 0        DEVICE SUBTYPE: 3        !
! LDEV: 3          DRT: 5          UNIT: 0       !
! RECORD SIZE: 80      BLOCK SIZE: 1280  (BYTES) !
! EXTENT SIZE: 5       MAX EXTENTS: 8            !
! RECPTR: 2            RECLIMIT: 100             !
! LOGCOUNT: 2            PHYSCOUNT: 1            !
! EOF AT: 2            LABEL ADDR: %00300016277  !
! FILE CODE: 0      ID IS STUDENT    ULABELS: 0  !
! PHYSICAL STATUS: 1111000000000000              !
! ERROR NUMBER: 101   RESIDUE: 640       (WORDS) !
! BLOCK NUMBER: 1              NUMREC: 16        !
+-------------------------------------------------+
3 RECORDS PROCESSED *** 1 ERROR
>EXIT
 END OF PROGRAM
```

7)   Now attempt to :SAVE 'TEMP'. Interpret the error.

```
:SAVE TEMP
ERR 116 L
DUPLICATE NAME
```

8)   List file 'TEMP' on your terminal with FCOPY. Exit FCOPY.
From which file domain did it come by default?   Temporary   ?

```
:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>FROM=TEMP;TO=$STDLIST
*143*J
WARNING: FROMFILE IS EMPTY
0 RECORDS PROCESSED *** 0 ERRORS
>EXIT
 END OF PROGRAM
```

9)   Log-off then log-on the system. What happens to Temporary
files at the end of a Job or Session? Verify your hypothesis
with LISTEQ2.

```
:HELLO STUDENT.INTRO/PASSWORD
CPU (SEC) = 6
CONNECT (MIN) = 13
MON, MAR  6, 1978,  4:58 PM
END OF SESSION
 SESSION NUMBER = #S82
 MON, MAR  6, 1978,  4:58 PM
 HP32002A.01.MR
WELCOME TO YOUR FRIENDLY HP-3000.
:RUN LISTEQ2.PUB.SYS
LISTEQ2 A01.01 (C) HEWLETT-PACKARD CO., 1976
***NO TEMP FILES
***NO FILE EQUATIONS
 END OF PROGRAM
```

10) Now list file 'TEMP' on your terminal with FCOPY. Which file domain did this one come from? _____Permanent_____?
Exit FCOPY and log-off the system. So... we have just seen that by default files are searched for in the Temporary domain first, if not there then the Permanent domain is searched.

So... we have seen that files of the same name can exist simultaneously in each of the three domains. This works just fine until we try to move one into another domain. Then we lose the latest file we have been working on! This is a lesson to keep in mind. It means, if you are creating a NEW output file from a 3 hour job and only make it permanent upon close, make sure a duplicate permanent file does not exist at the beginning of those 3 hours.

```
:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>FROM=TEMP;TO=$STDLIST
RECORD 1 - PERMANENT FILE 'TEMP'.                        00001000
RECORD 2 - I LIKE THIS ONE.                              00002000
RECORD 3 - END-OF-FILE.                                  00003000
EOF FOUND IN FROMFILE AFTER RECORD 2
```

```
3 RECORDS PROCESSED *** 0 ERRORS
>EXIT
 END OF PROGRAM
:BYE
CPU (SEC) = 2
CONNECT (MIN) = 2
MON, MAR  6, 1978,  5:00 PM
END OF SESSION
```

<< End >>

J O B   S T R E A M   L A B   #1 [0.5 hour]

```
        ***  Please read the entire lab before proceeding !  ***
        ------------------------------------------------

A).  Write out below, then create a Job Stream file with the
Editor to compile, prep and execute the COBOL source file
'LABJOB1.PUB'.  Use three separate MPE commands; do not use
COBOLGO.  Use $NEWPASS and $OLDPASS for your USL and program
files where applicable.  Keep the Job Stream file then Stream it.
Remember, STREAM'ing will initiate a job independent of the
current Job or Session.

        Key Points:                                  Fill In:
        -----------                                  -------

What commands delimit a job?                         :JOB & :EOJ
                                                     ----------------

What character is the STREAM command expecting?          !
                                                     ----------------

Where will your compilation listing and any          LP ($STDLIST)
program output appear and why?                       ----------------

Are there any special considerations when Keeping    ',UNN'
STREAM files with the Editor (Pre-MPE III)?          ----------------

What command can you use to find out the status      :SHOWJOB
of your job created by STREAM?                        ----------------
```

```
:HELLO STUDENT.INTRO
ACCT  PASSWORD?
PASSWORD
 SESSION NUMBER = #S91
 MON, MAR  6, 1978,  5:07 PM
 HP32002A.01.MR
WELCOME TO YOUR FRIENDLY HP-3000.
:EDITOR
HP32201A.7.00 EDIT/3000  MON, MAR  6, 1978,  5:07 PM
(C) HEWLETT-PACKARD CO. 1976
/A
    1       !JOB STUDENT.INTRO/PASSWORD
    2       !COBOL LABJOB1.PUB,$NEWPASS
    3       !PREP $OLDPASS,$NEWPASS
    4       !RUN $OLDPASS
    5       !EOJ
    6       ...
/K JOBLAB,UNN
/E
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
 END OF SUBSYSTEM
```

```
:STREAM JOBLAB
 #J6
:SHOWJOB #J6
JOBNUM   STATE IPRI JIN   JLIST      INTRODUCED   JOB NAME
#J6      EXEC         10S LP         MON  5:18P   STUDENT.INTRO
JOBFENCE= 2; JLIMIT= 2; SLIMIT= 16
:SHOWJOB JOB=STUDENT.INTRO
JOBNUM   STATE IPRI JIN   JLIST      INTRODUCED   JOB NAME
#J6      EXEC         10S LP         MON  5:18P   STUDENT.INTRO
#S91     EXEC QUIET  28  28          MON  5:07P   STUDENT.INTRO
2 JOBS (DISPLAYED):
     0 INTRO
     0 WAIT; INCL 0 DEFERRED
     2 EXEC; INCL 1 SESSIONS
     0 SUSP
JOBFENCE= 2; JLIMIT= 2; SLIMIT= 16
```
   B).   Create the same Job Stream from your Session without using
   the Editor.
```
:STREAM
>!JOB STUDENT.INTRO/PASSWORD
>!COBOL LABJOB1.PUB,$NEWPASS
>!PREP $OLDPASS,$NEWPASS
>!RUN $OLDPASS
>!EOJ
 #J7
>:EOD
:SHOWJOB #J7
JOBNUM   STATE IPRI JIN   JLIST      INTRODUCED   JOB NAME
#J7      EXEC         10S LP         MON  5:19P   STUDENT.INTRO
JOBFENCE= 2; JLIMIT= 2; SLIMIT= 16
:BYE
CPU (SEC) = 4
CONNECT (MIN) = 15
MON, MAR  6, 1978,  5:21 PM
```

END OF SESSION

J O B   S T R E A M   L A B   #2 [0.1 hour]

      As a class exercise we will contruct one Job Stream to
compile, prep, :STORE and :RESTORE files in the INTRO Account.
List any ideas you would like included below, then if time
permits, continue with JOB STREAM LAB #3.

J O B   S T R E A M   L A B   #3 [0.5 hour]

****
*   Proceed with this lab only if you have extra time.   *
****

         *** Please read the entire lab before proceeding ! ***

                 Part I -- STREAM'ing DATA
                 ----

1). We are going to submit a source disc file to the COBOL
compiler as if it were a deck of cards read through the card
reader. If it were a deck of cards, we would have to preface it
with a :DATA card containing our User and Account names plus any
associated passwords to enable MPE to identify this deck as ours
and pass it to our Job or Session. That card deck would be read
through the card reader by the input spooler into an input
spoolfile where it would remain in the 'Ready' state until
referenced by our session.

      Instead, we are going to STREAM a disc file containing an
image of this card deck with a !DATA command on the front and a
!EOD command on the back. This !DATA command must contain the
User and Account names of the session that will reference the
data exactly like a :DATA card. This !DATA command must contain
all associated passwords to be accepted by the system.

      The COBOL source deck we are going to submit is in the file
'LABJOB3.PUB'; it is unnumbered. Modify the !DATA command to
match your session's parameters. Refer to the syntax for the
:DATA command in your pocket guide (do NOT use a 'session-name'
nor a 'file-name' in your !DATA statement; they would
unneccessarily complicate things). Keep the file unnumbered as
'LABJOB3' in your group.

Exit the Editor and STREAM LABJOB3. '#Innn' should be displayed on your terminal; if not, seek aid from your instructor. Issue a :SHOWIN command and you will see an input spool-file in the 'READY' state for your User and Account that contains your COBOL source deck. Notice which device it appears to have been read from (either 5 or 10).

```
:HELLO STUDENT.INTRO/PASSWORD
 SESSION NUMBER = #S95
 MON, MAR  6, 1978,  5:21 PM
 HP32002A.01.MR
WELCOME TO YOUR FRIENDLY HP-3000.
:EDITOR
HP32201A.7.00 EDIT/3000  MON, MAR  6, 1978,  5:22 PM
(C) HEWLETT-PACKARD CO. 1976
/T LABJOB3.PUB,UNN
/L LAST
    34     !EOD
/M 1
MODIFY     1
!DATA USER.ACCOUNT
     RSTUDENT.INTRO/PASSWORD
!DATA STUDENT.INTRO/PASSWORD

/K LABJOB3,UNN
/E
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
 END OF SUBSYSTEM
:STREAM LABJOB3
 #I111
:SHOWIN
DEV/CL    DFID    JOBNUM  FNAME      STATE FRM SPACE RANK PRI #C
10        #I111                      READY         12
                  STUDENT.INTRO
28        #I109   #S95    $STDIN     OPENED
2 FILES:
    0 ACTIVE
    1 READY; INCL 1 SPOOFLES, 0 DEFERRED
    1 OPENED; INCL 0 SPOOFLES
    0 LOCKED; INCL 0 SPOOFLES
    1 SPOOFLES: 12 SECTORS
```

2). We must now reference this input spool-file with a :FILE command. If we had read a card deck through an unspooled card

reader, we would have referenced it with the command:
      ':FILE xyz;DEV=CARD'
This would have given us exclusive access to the card reader to
read in our deck.

      If we had read our card deck through a spooled card reader,
as soon as we placed it in the card reader, the input Spooler
would have read it into an input spool-file where it would remain
until referenced via this same :FILE command above.

      We have submitted a !DATA disc file to the input Spooler
with the :STREAM command, but the net effect has been exactly the
same as reading a card deck through a spooled card reader. We
can reference it with a similar :FILE command. The device a
STREAM'd file thinks it was read from depends on how the
'=STREAMS' command has been issued from the Operator's Console.
Now issue a :FILE command referencing the device number
associated with your input spool-file from the :SHOWIN display.

:FILE DATAFILE;DEV=10
      3). Use :COBOLPREP to both compile and prepare your program with
      one command. Back-reference the file name you used in your :FILE
      command in step 2 as the 'textfile' and put the resulting program
      in file 'PGM3'.

:COBOLPREP *DATAFILE,PGM3
PAGE 0001   HP32213C.02.00   (C) HEWLETT-PACKARD CO. 1977
      DATA AREA IS %000336 WORDS.
      CPU TIME = 0:00:01. WALL TIME = 0:00:08.
END COBOL/3000 COMPILATION. NO ERRORS. NO WARNINGS.
 END OF COMPILE
 END OF PREPARE
      4). Change the name of 'PGM3' to 'PROG'. If you get an error,
      chances are you have forgotten in which domain :PREP places its
      program files. Now :SAVE 'PROG' as a permanent file.

:RENAME PGM3,PROG
ERR 108 J
NON-EXISTENT FILE
:RENAME PGM3,PROG,TEMP
:SAVE PROG
      5). Issue the command to reset all active :FILE commands for
      your session. Run 'LISTEQ2.PUB.SYS' to make sure no :FILE
      commands remain active.

:RESET @
:RUN LISTEQ2.PUB.SYS

LISTEQ2 A01.01 (C) HEWLETT-PACKARD CO., 1976
***NO TEMP FILES
***NO FILE EQUATIONS
 END OF PROGRAM
  6). Using the Editor, create a Job Stream file to:
       a)   :STORE all files in your group to mag-tape.
       b)   :RESTORE files 'PROG' and 'LABJOB3' from mag-tape with
            the KEEP option specified.
       c)   Obtain a list of all files within your group on the
            line printer with a detail option of ',1'.
:EDITOR
HP32201A.7.00 EDIT/3000 MON, MAR 6, 1978, 5:30 PM
(C) HEWLETT-PACKARD CO. 1976
/A
     1   !JOB STUDENT.INTRO/PASSWORD
     2   !FILE STUDENT;DEV=TAPE
     3   !STORE ;*STUDENT;SHOW
     4   !RESTORE *STUDENT;PROG,LABJOB3;KEEP
     5   !LISTF,1
     6   !EOJ
     7   ...
/K LAB3JOB,UNN
/E
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
 END OF SUBSYSTEM
        R E M E M B E R :
        o   You must supply your Username, Acctname, and Account
             Password on the JOB command.
        o   You must KEEP the Stream file unnumbered if operating
             under pre-MPE III versions.
        o   :STORE and :RESTORE must back-reference a :FILE command
             for the tape drive. Use your User name for the file
             name so you can easily recognize which request is
             your's on the SYSTEM CONSOLE.
        o $STDLIST within a JOB will automatically be assigned to
             the line printer.
  7).   :STREAM your Job Stream file. If you have constructed it
correctly, a '#Jnnn' number will be displayed on your terminal.
You now have both a JOB and a SESSION running concurrently
logged-on under your USER.ACCT. Issue a ':SHOWJOB
JOB=@,user.INTRO' to display them both. If your JOB is in the
EXEC state and a mag-tape is available, do 'PART II -- Using
the SYSTEM CONSOLE' now. Otherwise continue with the next step
and do PART II later.

```
:STREAM LAB3JOB
 #J8
:SHOWJOB JOB=@,STUDENT.INTRO
JOBNUM   STATE IPRI JIN   JLIST      INTRODUCED   JOB NAME
#S95     EXEC QUIET  28   28         MON  5:21P   STUDENT.INTRO
#J8      EXEC        10S  LP         MON  5:34P   STUDENT.INTRO
2 JOBS (DISPLAYED):
    0 INTRO
    0 WAIT; INCL 0 DEFERRED
    2 EXEC; INCL 1 SESSIONS
    0 SUSP
JOBFENCE= 2; JLIMIT= 2; SLIMIT= 16
```

8). From within your session, obtain a list of all files within
your group on the line printer with a detail option of ',2'.

```
:FILE LP;DEV=LP
:LISTF,2;*LP
:FILE LIST;DEV=LP
:RUN LISTEQ2.PUB.SYS;PARM=1
 END OF PROGRAM
:BYE
CPU (SEC) = 12
CONNECT (MIN) = 16
MON, MAR  6, 1978,  5:37 PM
END OF SESSION
```

9). Now make a 'LISTEQ2' listing on the line printer by issuing
the following commands:
```
        :FILE LIST;DEV=LP
        :RUN LISTEQ2.PUB.SYS;PARM=1
```
Specifying ';PARM=1' directs LISTEQ2 to use file 'LIST' in a
:FILE command instead of file '$STDLIST'. You could also get a
LISTEQ2 listing to the line printer by running it from a JOB, but
it would be a listing of the file equations and temporary files
active within that JOB not within your SESSION!

End of PART I -- Do PART II if you haven't done it yet.

                 PART II -- Using the SYSTEM CONSOLE.
----
1). The SYSTEM CONSOLE looks like your Session's Terminal, but
it operates differently. Sit at the CONSOLE and attempt to key
in a command. The CONSOLE will not respond until you enter the
code: Press CNTL and upper-case 'A' simultaneously. When the
CONSOLE is ready to accept your input, it will prompt you with a

'='.   You must enter 'CNTL-A' before every command. Refer to the CONSOLE OPERATOR section of your Software Pocket Guide.

2).   Enter a '=RECALL' command to display all pending I/O requests. If a request for your tape file is there, continue with the next step and mount your mag-tape. If your request is not there, either your Job is not yet into execution or there was some error in your Job Stream file. In either case, give your tape to anyone ready to use it and take corrective action.

3).   Mount your mag-tape on the drive. It must have a write-ring to be written on. The drive's hubs do turn, if somewhat reluctantly, so put the tape reel on the top hub, thread it according to the diagram on the drive, then press the LOAD button, followed by the ON-LINE button. The tape should advance to the LOAD point and signal ON-LINE. If you get to this point by yourself, congratulations. If not, seek consolation from your instructor.

4).   The fourth item in your I/O request on the CONSOLE is your PIN (Process Identification Number). You must enter a '=REPLY' on the CONSOLE referencing your PIN and the logical device number (ldev) of the tape drive your tape is mounted on. On the drive, if 0 (zero) is lighted, the 'ldev' is 7. If '1' is lighted, the 'ldev' is 8; '2' lighted is ldev 9. Don't use 3 (ldev=10) as this is usually configured to be :JOB and :DATA accepting. If you just want to skip the whole operation, entering an ldev of '0' (zero) or 'N' will abort the I/O request.

5).   While the tape is being written, enter a =SHOWIN command on the CONSOLE. Observe that here it lists $STDIN for all users on the system by default.

6).   When the tape has been written, it will be rewound and the RESET and LOAD lights will be illuminated. At this point another I/O request should appear on the CONSOLE for your :RESTORE operation. Put the tape drive ON-LINE and reply to this new I/O request. At the completion of this operation the tape drive will again be RESET and at LOAD point. Press REWIND, then dismount the tape, give it to the next team, and get the listing from your JOB off the line printer to double check the results.

<< End >>

L A B   M P E   III

1)   Construct a JOB stream file to do a LISTF of the following
sets of files in PUB.INTRO.
      a)   All files beginning with 'K' (Use default detail for
            all LISTF's in this lab).
      b)   All files beginning with 'LAB'.
      c)   All files with at least one number in their name.
      d)   All files with the number '1' as the 4-th character in
            their name.
      e)   All files with any number as the 4-th character in
            their name.
Now STREAM the file. If a #Jnnn number is not returned, there
is a problem with your Stream file. Upon completion, go pick up
your output from the line printer.
      << EXAMPLES DONE FROM A SESSION; YOU SHOULD CREATE A JOB >>
:LISTF K@.PUB
FILENAME
KDATA   KEEPB   KEEPIT   KEEPNEW   KEEPQ   KKEY
KSAMBILD
:LISTF LAB@.PUB
FILENAME
LAB1DATA   LAB3COPY   LAB3EDIT   LAB3SOL   LAB7   LAB7DATA
LABEDIT1   LABJOB1   LABJOB3
:LISTF @#@.PUB
FILENAME
COBL1   COBPROG1   COBSUB1   COBTEST1   FILE1   FILE6
LAB1DATA   LAB3COPY   LAB3EDIT   LAB3SOL   LAB7   LAB7DATA
LABEDIT1   LABJOB1   LABJOB3   NOON206A   NEW1   PARA1
PCOB1   T1   TABDEF01   TABDEF02   TABDEF03   TRY1
TRY1UNN   WORK1000
:LISTF ????1@.PUB
FILENAME
COBL1   FILE1   PARA1   PCOB1   WORK1000
:LISTF ????#@.PUB
FILENAME
COBL1   FILE1   FILE6   NOON206A   PARA1   PCOB1
WORK1000

2)   Build a file with the following UDC's in it:
      a)   A UDC to issue a :FILE command for the line printer and
            ':SETMSG OFF' for you automatically at log-on. Also
            have it list messages on your terminal. This can be

accomplished with the :COMMENT command if the LIST
OPTION is specified. Make the message something
meaningful like "UDC has assumed control--say 'UNCL'".
b)   Place another UDC called 'Q' in the same file that will
run LISTEQ2.PUB.SYS. Use an OPTION so the UDC will not
be listed as it is executed.
Keep the file then issue a :SETCATALOG to invoke it.
3)   Use the UDC just created by entering 'Q'. Your Temp files
and File Commands should be listed. Notice -- Your line printer
File Command is not among them. Now issue a :HELLO command for
the same User.Account you are currently logged-on under. Your
log-on messages should appear. Use 'Q' again and see that your
line printer File command now exists... Huzzah.

```
:EDITOR
HP32201A.7.0H EDIT/3000  MON, APR 24, 1978,  5:47 PM
(C) HEWLETT-PACKARD CO. 1976
/A
     1       SETUP
     2       OPTION LIST,LOGON
     3       FILE LP;DEV=LP
     4       SETMSG OFF
     5       COMMENT  UDC HAS ASSUMED CONTROL -- SAY 'UNCL'.
     6       ***
     7       Q
     8       RUN LISTEQ2.PUB.SYS
     9       ...
/K UDCLAB
/E
END OF SUBSYSTEM
:SETCATALOG UDCLAB
:Q
LISTEQ2 B00.00 (C) HEWLETT-PACKARD CO., 1976
***NO TEMP FILES
***NO FILE EQUATIONS
END OF PROGRAM
:HELLO STUDENT.INTRO/PASSWORD
CPU=48. CONNECT=18. MON, APR 24, 1978,  5:51 PM
HP3000 III.  MON, APR 24, 1978,  5:51 PM
******** WELCOME **************
FILE LP;DEV=LP
SETMSG OFF
COMMENT  UDC HAS ASSUMED CONTROL -- SAY 'UNCL'.
:Q
```

```
LISTEQ2 B00.00 (C) HEWLETT-PACKARD CO., 1976
***NO TEMP FILES
***FILE EQUATIONS
:FILE LP;DEV=LP
END OF PROGRAM
```

4)   Now add several more UDC's to the same file:
       a)   Add a UDC 'F' that runs FCOPY.PUB.SYS. Have it list
             the UDC as it is executed.
       b)   Add another UDC called 'L' that will call :LISTF with a
             detail parameter of ',2'. Construct it so a fileset or
             a different detail may be specified if the User
             chooses. Use the OPTION to inhibit BREAK.
       Now keep this file under its previous name. You must have
removed this file from the Catalog or it will still be open with
ACC=EAR and you will not be able to purge it. If you are in this
situation, keep the new file under a different name, remove your
UDC from the catalog, purge the old one and rename the new one.
       Invoke your latest copy of your UDC file. Test the new
commands by entering an 'L'. You should get a level '2' detail
listing of all files in your group. Try to interrupt the listing
with the <BREAK> Key. Specify several different filesets and
different detail levels. Try passing positional parameters and
keyword parameters. Enter 'F' just to make sure it works. Exit
FCOPY immediately.

```
:EDITOR
HP32201A.7.0H EDIT/3000  MON, APR 24, 1978,  6:16 PM
(C) HEWLETT-PACKARD CO. 1976
/T UDCLAB
/A
    9       ****
   10       F
   11       OPTION LIST
   12       RUN FCOPY.PUB.SYS
   13       *****************
   14       L FILES=@,DETAIL=2
   15       OPTION LIST,NOBREAK
   16       LISTF !FILES,!DETAIL
   17       ...
/K UDCLAB
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
!   FILE NAME IS UDCLAB.GSTUDENT.INTRO            !
!   FOPTIONS: NEW,A,*FORMAL*,F,N,DEQ              !
!   AOPTIONS:  OUTPUT,SREC,NOLOCK,DEF,BUFFER      !
!   DEVICE TYPE: 0      DEVICE SUBTYPE: 8         !
!   LDEV: 5       DRT: 4          UNIT: 1         !
!   RECORD SIZE: 80     BLOCK SIZE: 1280 (BYTES) !
```

```
! EXTENT SIZE: 10     MAX EXTENTS: 1               !
! RECPTR: 16          RECLIMIT: 16                 !
! LOGCOUNT: 16          PHYSCOUNT: 1               !
! EOF AT: 16         LABEL ADDR: %00500007466      !
! FILE CODE: 0     ID IS STUDENT    ULABELS: 0     !
! PHYSICAL STATUS: 1000000000000001                !
! ERROR NUMBER: 100   RESIDUE: 640       (WORDS)   !
! BLOCK NUMBER: 1               NUMREC: 16         !
+----------------------------------------------------+
*60*J
FCLOSE FAILURE (100)
/
:SETCATALOG
:RESUME
READ PENDING
K
 UDCLAB
UDCLAB ALREADY EXISTS - RESPOND YES TO PURGE OLD AND THEN KEEP
PURGE OLD?Y
/E
END OF SUBSYSTEM
:SETCATALOG UDCLAB
:L @.PUB
LISTF @.PUB,2
ACCOUNT=  INTRO          GROUP=  PUB
FILENAME  CODE   ------------LOGICAL RECORD-----------   ----SPACE----
                 SIZE  TYP      EOF       LIMIT R/B  SECTORS #X MX
AAUSL     USL   128W  FB       130       1023   1      256   2  8
ABUSL     USL   128W  FB         5         20   1       21   1  1
ACUSL     USL   128W  FB       130       1023   1      256   2  8
ASCII     123    80B  FA         4          4  16       10   1  1
BACKUP    EDTCT  80B  FA       530        530  16      175  12 12
BINARY    987
       . . .
TRY1             80B  FA        10         10  16       10   1  1
TRY1UNN          72B  FA        10         10  16       10   1  1
UDCMINE          80B  FA         7          7  16       10   1  1
USEFILE   EDTCT  74B  FA         4          4  16       10   1  1
USL       USL   128W  FB        64        400   1      101   1  4
VALIDNO   EDTCT  80B  FA        57         57  16       25   1  1
:L TRY1.PUB
LISTF TRY1.PUB,2
ACCOUNT=  INTRO          GROUP=  PUB
```

```
FILENAME  CODE   -----------LOGICAL RECORD----------   ----SPACE----
                 SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX
TRY1             80B   FA         10         10   16       10  1  1
:L @1@.PUB,1
LISTF @1@.PUB,1
ACCOUNT=  INTRO        GROUP=  PUB
FILENAME  CODE   -----------LOGICAL RECORD--------
                 SIZE  TYP        EOF      LIMIT
COBL1     EDTCT  86B   FA         33         33
COBPROG1  PROG   128W  FB          7          7
COBSUB1   EDTCT  78B   FA         32         32
COBTEST1  EDTCT  80B   FA         28         28
FILE1            80B   FA          0          5
LAB1DATA         71B   FA         26         84
LABEDIT1         80B   FA         61         61
LABJOB1   EDTCT  80B   FA         10         10
NEW1             50B   FA          1         20
PARA1            80B   FA          5          5
PCOB1     PROG   128W  FB         10         10
T1        EDTCT  80B   FA        276        276
TABDEF01         108B  FA          6          6
TRY1             80B   FA         10         10
TRY1UNN          72B   FA         10         10
WORK1000  BASD   308W  FB          8          8
:L DETAIL=1,FILES=W@.PUB
LISTF W@.PUB,1
ACCOUNT=  INTRO        GROUP=  PUB
FILENAME  CODE   -----------LOGICAL RECORD--------
                 SIZE  TYP        EOF      LIMIT
WORK1000  BASD   308W  FB          8          8
:F
RUN FCOPY.PUB.SYS
HP32212A.3.03 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>EXIT
END OF PROGRAM
```

5)   Write out the :STORE command to store all files beginning
with 'LAB' in all PUB groups of all Accounts (this could actually
be done only by the System Manager).

:STORE LAB@.PUB.@;*TAPEFILE;SHOW

```
    ------------------------------------------------------------------
6)    Write out a :RESTORE command to restore all files beginning
with 'LAB' into all groups of all accounts in the system from a
SYSDUMP tape.  Only restore files that did not previously exist
in any group.
            :RESTORE *TAPEFILE;LAB@.@.@;SHOW;KEEP
    ------------------------------------------------------------------
7)    Write out below a Job Stream to Run 'LABJCW1', show the JCW
setting, abort the Job if JCW is WARN or greater.  Then if JCW is
greater than 'OK' plus 100, run 'LABJCW2' then 'LABJCW3'.  Else,
just run 'LABJCW2'.
!JOB STUDENT.INTRO/PASSWORD
!RUN LABJCW1
!SHOWJCW
!IF JCW < WARN THEN
!    IF JCW > OK100 THEN
!         RUN LABJCW2
!         RUN LABJCW3
!         EOJ
!    ELSE
!         RUN LABJCW2
!         EOJ
!    ENDIF
!ELSE
!    SETJCW JCW,FATAL
!ENDIF
!EOJ
    ------------------------------------------------------------------
    ****************************************************************
*     OPTIONAL -- Proceed with the remainder of the lab only if
*     time permits.
    ****************************************************************

8)    Enhance your Log-on messages with some imaginative Terminal
      Display Enhancements.  Some of interest are:

      ESC & d J              Half-bright Inverse Display Enhancement
      ESC & d @              Reset Display Enhancements
      ESC H                  Home Cursor
      ESC J                  Clear Display
      ESC M                  Delete Line
      ESC S                  Roll Up
      ESC T                  Roll Down
      ESC z                  Terminal Self-Test

NOTE: 'ESC' represents the ESCAPE key.
                                    <<< End >>>
```

F C O P Y   L A B   #1

From a Session, concatenate into one listing on the line printer all
of the following:

1)   All records in LAB1DATA.PUB with '951' beginning in column 67.

2)   All records in LAB1DATA.PUB that DO NOT have '951' beginning in
      column 67.

3)   A 'CHAR' dump of DEFTABS.PUB.

4)   A 'CHAR' & 'HEX' dump of DEFTABS.PUB.

5)   An 'OCTAL' & 'CHAR' dump of DEFTABS.PUB.

```
:HELLO STUDENT.INTRO/PASSWORD
 SESSION NUMBER = #S58
 TUE, APR 25, 1978, 11:54 AM
 HP32002A.01.MR
:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>
:FILE LP;DEV=LP
:RESUME
READ PENDING
FROM=LAB1DATA.PUB;TO=*LP;SUBSET="951",67
EOF FOUND IN FROMFILE AFTER RECORD 25
14 RECORDS PROCESSED *** 0 ERRORS
>FROM=LAB1DATA.PUB;TO=*;SUBSET="951",67,EXCLUDE
EOF FOUND IN FROMFILE AFTER RECORD 25
12 RECORDS PROCESSED *** 0 ERRORS
>FROM=DEFTABS.PUB;TO=*;CHAR
EOF FOUND IN FROMFILE AFTER RECORD 13
14 RECORDS PROCESSED *** 0 ERRORS
>FROM=DEFTABS.PUB;TO=*;HEX;CHAR
EOF FOUND IN FROMFILE AFTER RECORD 13
14 RECORDS PROCESSED *** 0 ERRORS
>FROM=DEFTABS.PUB;TO=*;CHAR;OCTAL
EOF FOUND IN FROMFILE AFTER RECORD 13
14 RECORDS PROCESSED *** 0 ERRORS
```

```
>EXIT
 END OF PROGRAM
:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>FROM=DISCFILE;TO=
*200*
AMOS       QUITO      243-8171 1467  ANOPHELES AV      NEW ALMADEN    95143
EILEEN     SIDEWAYS   377-7545 2577  TILDEN BLVD       SAN JOSE       95111
TYRONE     SHOELACES  266-1721 17265 BLUCHER BLVD      LOS GATOS      95131
XAVIER     GREENSTAMP 247-5423 1551  PREMIUM ST        SAN JOSE       95134
AL         PINE       578-2868 1738  DRY CREEK ROAD    SAN JOSE       95116
ARMAND     HAMMER     298-4988 1350  ALKALI AV         CAMBRIAN PARK  95131
TRUDY      TEKTIFF    255-1005 17155 POIROT PL         CAMPBELL       95121
BRICK      WALL       288-7761 2950  STORY ROAD        SAN JOSE       95131
JOSE       CANUSI     214-5566 2485  ANTHEM WY         CAMPBELL       95129
ALI        FATIQ      292-0100 480   DU PONT CIR       SAN JOSE       95131
ALI        KATZ       296-7650 262   MEHITABEL AVE     SANTA CLARA    95133
BEAUFORT   SCALE      328-7540 3843  WINDY WY          SAN JOSE       95117
PHIL       ARBUSTER   997-1040 672   CONSTITUTION DR   SANTA CLARA    95110
CLARA      NETTE      243-4493 2667  GOODMAN DR        ALVISO         95143
NEIL       DU PREE    246-1112 4097  PRIE DIEUX DR     SAN JOSE       95013
OLIVER     TEETHOUT   867-0138 20085 UPPER PLATE PL    CUPERTINO      95053
AMANDA     RECKONWITH 247-9142 2474  MACHO ST          SANTA CLARA    95020
ARTHUR     MOMITER    443-5346 1554  MERCURY ST        MILPITAS       94173
RACHAEL    PREJUDICE  262-8940 1730  WARREN CT         SANTA CLARA    95035
ROSE       GARLAND    269-7132 5219  PARK MEADOW CT    SAN JOSE       95054
HY         HILL       593-8421 48709 BLUE RIDGE DR     MILPITAS       95035
KNUT       HEERJIT    923-3485 1740  VIA ABSENTIA      SAN JOSE       95053
VY         KNOTT      262-8940 1883  QUERY PL          SAN CARLOS     95014
ANNA       LOGUE      224-8934 1707  INVERSE WY        MOUNTAIN VIEW  95051
ANDY       LUCIAN     264-4169 1119  IBERIAN CT        CUPERTINO      95070
BUZZ       SAWYER     259-3434 1850  FOREST DR         CUPERTINO      95023
DEFTABS.PUB.INTRO RECORD 0 (%0)
000000:  Q".m.X.H.J..2..2..2..2..2..2..2..2..2..2..2..2..2..2..2.
000034:  .2..2..2..2..2"
DEFTABS.PUB.INTRO RECORD 1 (%1)
000000:  Q"..2..2..2..2..2..2..2..2..2..2..2..2..2..2..2..2..2
000034:  ..2..2"
DEFTABS.PUB.INTRO RECORD 2 (%2)
000000:  Q".&a1r50C<<tabs now being set>>"
000034:
DEFTABS.PUB.INTRO RECORD 3 (%3)
000000:  Q".H.J7        8 DEFAULT 1  TABS    2"
```

```
000034:
DEFTABS.PUB.INTRO RECORD 4 (%4)
000000:   Q".&a0r30C           3           4        "
000034:
DEFTABS.PUB.INTRO RECORD 5 (%5)
000000:   Q".&a0r55C      5           6 22SEP77 7"
000034:
DEFTABS.PUB.INTRO RECORD 6 (%6)
000000:   Q".&a1r00C.&dF.11.&d@23456789/123.&dF.14.&d@56.&dF.17.&d
000034:   @89.&dF.10.&d@"
DEFTABS.PUB.INTRO RECORD 8 (%10)
000000:   Q".&a1r30C1.&dF.12.&d@34.&dF.15.&d@67.&dF.18.&d@90"
000034:
DEFTABS.PUB.INTRO RECORD 9 (%11)
000000:   Q".&a1r40C.&dF.11.&d@2345678901234567890"
000034:
DEFTABS.PUB.INTRO RECORD 10 (%12)
000000:   Q".&a1r60C12345678901234.&dF.15.&d@67890"
000034:
DEFTABS.PUB.INTRO RECORD 11 (%13)
000000:   Q".&a-1r0C.G.1"
000034:
DEFTABS.PUB.INTRO RECORD 12 (%14)
000000:   SET NOTABS,FORMAT=DEFAULT,RIGHT=1,LENGTH=72,RIGHT=72,TAB
000034:   S
DEFTABS.PUB.INTRO RECORD 13 (%15)
000000:   VERIFY TABS,LEFT,RIGHT,LENGTH
000034:
DEFTABS.PUB.INTRO RECORD 0 (%0)
000000:   5122 1B6D 1B58 1B48 1B4A 091B 3209 1B32 Q".m.X.H.J..2..2
000010:   091B 3209 1B32 091B 3209 1B32 091B 3209 ..2..2..2..2..2.
000020:   1B32 091B 3209 1B32 091B 3209 1B32 091B .2..2..2..2..2..
000030:   3209 1B32 091B 3209 1B32 091B 3209 1B32 2..2..2..2..2..2
000040:   091B 3209 1B32 2220                     ..2..2"
DEFTABS.PUB.INTRO RECORD 1 (%1)
000000:   5122 091B 3209 1B32 091B 3209 1B32 091B Q"..2..2..2..2..
000010:   3209 1B32 091B 3209 1B32 091B 3209 1B32 2..2..2..2..2..2
000020:   091B 3209 1B32 091B 3209 1B32 091B 3209 ..2..2..2..2..2.
000030:   1B32 091B 3209 1B32 091B 3209 1B32 2220 .2..2..2..2..2"
000040:   2020 2020 2020 2020
```

```
DEFTABS.PUB.INTRO RECORD 2 (%2)
000000:   5122 1B26 6131 7235 3043 3C3C 7461 6273  Q".&a1r50C<<tabs
000010:   206E 6F77 2062 6569 6E67 2073 6574 3E3E   now being set>>
000020:   2220 2020 2020 2020 2020 2020 2020 2020  "
000030:   2020 2020 2020 2020 2020 2020 2020 2020
000040:   2020 2020 2020 2020
DEFTABS.PUB.INTRO RECORD 3 (%3)
000000:   5122 1B48 1B4A 3720 2020 2020 2020 2038  Q".H.J7        8
000010:   2044 4546 4155 4C54 2031 2020 5441 4253   DEFAULT 1  TABS
000020:   2020 2032 2220 2020 2020 2020 2020 2020    2"
000030:   2020 2020 2020 2020 2020 2020 2020 2020
000040:   2020 2020 2020 2020
DEFTABS.PUB.INTRO RECORD 4 (%4)
000000:   5122 1B26 6130 7233 3043 2020 2020 2020  Q".&a0r30C
000010:   2020 2033 2020 2020 2020 2020 2034 2020     3        4
000020:   2020 2022 2020 2020 2020 2020 2020 2020     "
000030:   2020 2020 2020 2020 2020 2020 2020 2020
000040:   2020 2020 2020 2020
DEFTABS.PUB.INTRO RECORD 5 (%5)
000000:   5122 1B26 6130 7235 3543 2020 2020 3520  Q".&a0r55C     5
000010:   2020 2020 2020 2020 3620 3232 5345 5037          6 22SEP7
000020:   3720 3722 2020 2020 2020 2020 2020 2020  7 7"
000030:   2020 2020 2020 2020 2020 2020 2020 2020
000040:   2020 2020 2020 2020
DEFTABS.PUB.INTRO RECORD 6 (%6)
000000:   5122 1B26 6131 7230 3043 1B26 6446 1B31  Q".&a1r00C.&dF.1
000010:   311B 2664 4032 3334 3536 3738 392F 3132  1.&d@23456789/12
000020:   331B 2664 461B 3134 1B26 6440 3536 1B26  3.&dF.14.&d@56.&
000030:   6446 1B31 371B 2664 4038 391B 2664 461B  dF.17.&d@89.&dF.
000040:   3130 1B26 6440 2220              10.&d@"
DEFTABS.PUB.INTRO RECORD 7 (%7)
000000:   5122 1B26 6131 7232 3043 3132 1B26 6446  Q".&a1r20C12.&dF
000010:   1B31 331B 2664 4034 351B 2664 461B 3136  .13.&d@45.&dF.16
000020:   1B26 6440 3738 1B26 6446 1B31 391B 2664  .&d@78.&dF.19.&d
000030:   4030 2220 2020 2020 2020 2020 2020 2020  @0"
000040:   2020 2020 2020 2020
DEFTABS.PUB.INTRO RECORD 8 (%10)
000000:   5122 1B26 6131 7233 3043 311B 2664 461B  Q".&a1r30C1.&dF.
000010:   3132 1B26 6440 3334 1B26 6446 1B31 351B  12.&d@34.&dF.15.
000020:   2664 4036 371B 2664 461B 3138 1B26 6440  &d@67.&dF.18.&d@
000030:   3930 2220 2020 2020 2020 2020 2020 2020  90"
000040:   2020 2020 2020 2020
DEFTABS.PUB.INTRO RECORD 9 (%11)
```

```
000000:  5122 1B26 6131 7234 3043 1B26 6446 1B31  Q".&a1r40C.&dF.1
000010:  311B 2664 4032 3334 3536 3738 3930 3132  1.&d@234567890 12
000020:  3334 3536 3738 3930 2220 2020 2020 2020  34567890"
000030:  2020 2020 2020 2020 2020 2020 2020 2020
000040:  2020 2020 2020 2020
DEFTABS.PUB.INTRO RECORD 10 (%12)
000000:  5122 1B26 6131 7236 3043 3132 3334 3536  Q".&a1r60C123456
000010:  3738 3930 3132 3334 1B26 6446 1B31 351B  78901234.&dF.15.
000020:  2664 4036 3738 3930 2220 2020 2020 2020  &d@67890"
000030:  2020 2020 2020 2020 2020 2020 2020 2020
000040:  2020 2020 2020 2020
DEFTABS.PUB.INTRO RECORD 11 (%13)
000000:  5122 1B26 612D 3172 3043 1B47 1B6C 2220  Q".&a-1r0C.G.l"
000010:  2020 2020 2020 2020 2020 2020 2020 2020
000020:  2020 2020 2020 2020 2020 2020 2020 2020
000030:  2020 2020 2020 2020 2020 2020 2020 2020
000040:  2020 2020 2020 2020
DEFTABS.PUB.INTRO RECORD 12 (%14)
000000:  5345 5420 4E4F 5441 4253 2C46 4F52 4D41  SET NOTABS,FORMA
000010:  543D 4445 4641 554C 542C 5249 4748 543D  T=DEFAULT,RIGHT=
000020:  312C 4C45 4E47 5448 3D37 322C 5249 4748  1,LENGTH=72,RIGH
000030:  543D 3732 2C54 4142 5320 2020 2020 2020  T=72,TABS
000040:  2020 2020 2020 2020
DEFTABS.PUB.INTRO RECORD 13 (%15)
000000:  5645 5249 4659 2054 4142 532C 4C45 4654  VERIFY TABS,LEFT
000010:  2C52 4947 4854 2C4C 454E 4754 4820 2020  ,RIGHT,LENGTH
000020:  2020 2020 2020 2020 2020 2020 2020 2020
000030:  2020 2020 2020 2020 2020 2020 2020 2020
000040:  2020 2020 2020 2020
DEFTABS.PUB.INTRO RECORD 0 (%0)
000000:  050442 015555 015530 015510  Q".m.X.H
000004:  015512 004433 031011 015462  .J..2..2
000010:  004433 031011 015462 004433  ..2..2..
000014:  031011 015462 004433 031011  2..2..2.
000020:  015462 004433 031011 015462  .2..2..2
000024:  004433 031011 015462 004433  ..2..2..
000030:  031011 015462 004433 031011  2..2..2.
000034:  015462 004433 031011 015462  .2..2..2
000040:  004433 031011 015462 021040  ..2..2"
DEFTABS.PUB.INTRO RECORD 1 (%1)
000000:  050442 004433 031011 015462  Q"..2..2
000004:  004433 031011 015462 004433  ..2..2..
000010:  031011 015462 004433 031011  2..2..2.
000014:  015462 004433 031011 015462  .2..2..2
000020:  004433 031011 015462 004433  ..2..2..
```

```
000024:    031011 015462 004433 031011 2..2..2.
000030:    015462 004433 031011 015462 .2..2..2
000034:    004433 031011 015462 021040 ..2..2"
000040:    020040 020040 020040 020040
DEFTABS.PUB.INTRO RECORD 2 (%2)
000000:    050442 015446 060461 071065 Q".&a1r5
000004:    030103 036074 072141 061163 0C<<tabs
000010:    020156 067567 020142 062551  now bei
000014:    067147 020163 062564 037076 ng set>>
000020:    021040 020040 020040 020040 "
000024:    020040 020040 020040 020040
000030:    020040 020040 020040 020040
000034:    020040 020040 020040 020040
000040:    020040 020040 020040 020040
DEFTABS.PUB.INTRO RECORD 3 (%3)
000000:    050442 015510 015512 033440 Q".H.J7
000004:    020040 020040 020040 020070         8
000010:    020104 042506 040525 046124  DEFAULT
000014:    020061 020040 052101 041123 1  TABS
000020:    020040 020062 021040 020040    2"
000024:    020040 020040 020040 020040
000030:    020040 020040 020040 020040
000034:    020040 020040 020040 020040
000040:    020040 020040 020040 020040
DEFTABS.PUB.INTRO RECORD 4 (%4)
000000:    050442 015446 060460 071063 Q".&a0r3
000004:    030103 020040 020040 020040 0C
000010:    020040 020063 020040 020040    3
000014:    020040 020040 020064 020040      4
000020:    020040 020042 020040 020040    "
000024:    020040 020040 020040 020040
000030:    020040 020040 020040 020040
000034:    020040 020040 020040 020040
000040:    020040 020040 020040 020040
DEFTABS.PUB.INTRO RECORD 5 (%5)
000000:    050442 015446 060460 071065 Q".&a0r5
000004:    032503 020040 020040 032440 5C      5
000010:    020040 020040 020040 020040
000014:    033040 031062 051505 050067 6 22SEP7
000020:    033440 033442 020040 020040 7 7"
000024:    020040 020040 020040 020040
000030:    020040 020040 020040 020040
000034:    020040 020040 020040 020040
000040:    020040 020040 020040 020040
DEFTABS.PUB.INTRO RECORD 6 (%6)
000000:    050442 015446 060461 071060 Q".&a1r0
```

```
000004:    030103 015446 062106 015461  0C.&dF.1
000010:    030433 023144 040062 031464  1.&d@234
000014:    032466 033470 034457 030462  56789/12
000020:    031433 023144 043033 030464  3.&dF.14
000024:    015446 062100 032466 015446  .&d@56.&
000030:    062106 015461 033433 023144  dF.17.&d
000034:    040070 034433 023144 043033  @89.&dF.
000040:    030460 015446 062100 021040  10.&d@"
DEFTABS.PUB.INTRO RECORD 7 (%7)
000000:    050442 015446 060461 071062  Q".&a1r2
000004:    030103 030462 015446 062106  0C12.&dF
000010:    015461 031433 023144 040064  .13.&d@4
000014:    032433 023144 043033 030466  5.&dF.16
000020:    015446 062100 033470 015446  .&d@78.&
000024:    062106 015461 034433 023144  dF.19.&d
000030:    040060 021040 020040 020040  @0"
000034:    020040 020040 020040 020040
000040:    020040 020040 020040 020040
DEFTABS.PUB.INTRO RECORD 8 (%10)
000000:    050442 015446 060461 071063  Q".&a1r3
000004:    030103 030433 023144 043033  0C1.&dF.
000010:    030462 015446 062100 031464  12.&d@34
000014:    015446 062106 015461 032433  .&dF.15.
000020:    023144 040066 033433 023144  &d@67.&d
000024:    043033 030470 015446 062100  F.18.&d@
000030:    034460 021040 020040 020040  90"
000034:    020040 020040 020040 020040
000040:    020040 020040 020040 020040
DEFTABS.PUB.INTRO RECORD 9 (%11)
000000:    050442 015446 060461 071064  Q".&a1r4
000004:    030103 015446 062106 015461  0C.&dF.1
000010:    030433 023144 040062 031464  1.&d@234
000014:    032466 033470 034460 030462  56789012
000020:    031464 032466 033470 034460  34567890
000024:    021040 020040 020040 020040  "
000030:    020040 020040 020040 020040
000034:    020040 020040 020040 020040
000040:    020040 020040 020040 020040
DEFTABS.PUB.INTRO RECORD 10 (%12)
000000:    050442 015446 060461 071066  Q".&a1r6
000004:    030103 030462 031464 032466  0C123456
000010:    033470 034460 030462 031464  78901234
000014:    015446 062106 015461 032433  .&dF.15.
000020:    023144 040066 033470 034460  &d@67890
000024:    021040 020040 020040 020040  "
000030:    020040 020040 020040 020040
000034:    020040 020040 020040 020040
```

```
000040:   020040 020040 020040 020040
DEFTABS.PUB.INTRO RECORD 11 (%13)
000000:   050442 015446 060455 030562  Q".&a-1r
000004:   030103 015507 015554 021040  0C.G.1"
000010:   020040 020040 020040 020040
000014:   020040 020040 020040 020040
000020:   020040 020040 020040 020040
000024:   020040 020040 020040 020040
000030:   020040 020040 020040 020040
000034:   020040 020040 020040 020040
000040:   020040 020040 020040 020040
DEFTABS.PUB.INTRO RECORD 12 (%14)
000000:   051505 052040 047117 052101  SET NOTA
000004:   041123 026106 047522 046501  BS,FORMA
000010:   052075 042105 043101 052514  T=DEFAUL
000014:   052054 051111 043510 052075  T,RIGHT=
000020:   030454 046105 047107 052110  1,LENGTH
000024:   036467 031054 051111 043510  =72,RIGH
000030:   052075 033462 026124 040502  T=72,TAB
000034:   051440 020040 020040 020040  S
000040:   020040 020040 020040 020040
DEFTABS.PUB.INTRO RECORD 13 (%15)
000000:   053105 051111 043131 020124  VERIFY T
000004:   040502 051454 046105 043124  ABS,LEFT
000010:   026122 044507 044124 026114  ,RIGHT,L
000014:   042516 043524 044040 020040  ENGTH
000020:   020040 020040 020040 020040
000024:   020040 020040 020040 020040
000030:   020040 020040 020040 020040
000034:   020040 020040 020040 020040
000040:   020040 020040 020040 020040
```

F C O P Y   L A B   #2

Do all steps in FCOPY LAB #1 from a Job Stream. Output all listings
into the same file but don't use $STDLIST.

```
:EDITOR
HP32201A.7.00 EDIT/3000  TUE, APR 25, 1978, 12:43 PM
(C) HEWLETT-PACKARD CO. 1976
/A
    1     !JOB STUDENT.INTRO/PASSWORD
    2     !FILE LP;DEV=LP
    3     !RUN FCOPY.PUB.SYS
    4     FROM=LAB1DATA.PUB;TO=*LP;SUBSET="951",67
```

```
     5      FROM=LAB1DATA.PUB;TO=*;SUBSET="951",67,EXCLUDE
     6      FROM=DEFTABS.PUB;TO=*;CHAR
     7      FROM=DEFTABS.PUB;TO=*;CHAR;HEX
     8      FROM=DEFTABS.PUB;TO=*;OCTAL;CHAR
     9      EXIT
    10      !EOJ
    11      ...
/K FCLAB2,UNN
   |****************************************************************
   |*     OPTIONAL -- Proceed only if time permits.              *
   |****************************************************************
```

F C O P Y   L A B   #3

Modify your Job Stream from FCOPY LAB #2 to concatenate all output
in 1 disc file, then list it on the line printer honoring carriage
control characters. Execute your Job Stream.

```
/M 2
MODIFY     2
!FILE LP;DEV=LP
      RDISCFILE,NEW;REC=-80,3,F,ASCII;SAVE;CCTL
!FILE DISCFILE,NEW;REC=-80,3,F,ASCII;SAVE;CCTL
/M 4
MODIFY     4
FROM=LAB1DATA.PUB;TO=*LP;SUBSET="951",67
                       DDIDISCFILE
FROM=LAB1DATA.PUB;TO=*DISCFILE;SUBSET="951",67

/L ALL
     1      !JOB STUDENT.INTRO/PASSWORD
     2      !FILE DISCFILE,NEW;REC=-80,3,F,ASCII;SAVE;CCTL
     3      !RUN FCOPY.PUB.SYS
     4      FROM=LAB1DATA.PUB;TO=*DISCFILE;SUBSET="951",67
     5      FROM=LAB1DATA.PUB;TO=*;SUBSET="951",67,EXCLUDE
     6      FROM=DEFTABS.PUB;TO=*;CHAR
     7      FROM=DEFTABS.PUB;TO=*;CHAR;HEX
     8      FROM=DEFTABS.PUB;TO=*;OCTAL;CHAR
     9      EXIT
    10      !EOJ
/K FCLAB3,UNN
/E
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
 END OF SUBSYSTEM
```

S O R T   L A B   #1   [0.6 hr]

<< use record layout figure here (pg V-37) >>

1)   Two Employee data files exist with the above record layout,
EMPDATA.PUB which is already sorted into the desired sequence and
EMPCARD.PUB which is unsorted.

2)   Find the attributes of these files and :BUILD a permanent disc
file 'MFILE' big enough to hold both of them but otherwise with the
same attributes as the two files.

3)   Using FCOPY, make an exact copy of EMPCARD.PUB in your group called
DFILE.

4)   Sort DFILE by years of service (longest first) and put the output
back into the same file.

5)   Merge DFILE with EMPDATA.PUB and put the output in MFILE.

6)   Using FCOPY, make a listing of MFILE on the line printer, deleting
the fields from job code through the end of the record.

```
:LISTF EMPCARD.PUB,2
ACCOUNT=  INTRO       GROUP=  PUB
FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE----
                SIZE   TYP        EOF       LIMIT R/B  SECTORS #X MX
EMPCARD         80B  FA          28          28  16        15  1  1
:LISTF EMPDATA.PUB,2
ACCOUNT=  INTRO       GROUP=  PUB
FILENAME  CODE  ------------LOGICAL RECORD-----------  ----SPACE----
                SIZE   TYP        EOF       LIMIT R/B  SECTORS #X MX
EMPDATA         80B  FA          27          27  16        15  1  1
:BUILD MFILE;REC=-80,16,F,ASCII;DISC=55
:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>FROM=EMPCARD.PUB;TO=DFILE;NEW
EOF FOUND IN FROMFILE AFTER RECORD 27
28 RECORDS PROCESSED *** 0 ERRORS
```

```
>EXIT
 END OF PROGRAM
:RUN SORT.PUB.SYS
HP32214B.01.05 SORT/3000   TUE, APR 25, 1978,  1:01 PM
(C) HEWLETT-PACKARD CO. 1976
>INPUT DFILE
>OUTPUT DFILE
>KEY 29,2,DESC
>VERIFY
INPUT FILE = DFILE
OUTPUT FILE = DFILE
KEY POSITION    LENGTH      TYPE      ASC/DESC
        29          2       BYTE        DESC    (MAJOR KEY)
>END
PURGE OLD OUTPUT FILE DFILE.GSTUDENT.INTRO ? YES
                    STATISTICS
NUMBER OF RECORDS =                              28
NUMBER OF INTERMEDIATE PASSES =                   0
SPACE AVAILABLE (IN WORDS) =                 12,203
NUMBER OF COMPARES =                            144
NUMBER OF SCRATCHFILE IO'S =                     20
CPU TIME (MINUTES) =                            .01
RECORD SIZE (IN BYTES) =                         80
SCRATCH FILE SIZE (# SECTORS) =                  93
 END OF PROGRAM
:RUN MERGE.PUB.SYS
HP32214B.01.05 MERGE/3000   TUE, APR 25, 1978,  1:03 PM
(C) HEWLETT-PACKARD CO. 1976
>INPUT DFILE,EMPDATA.PUB
>OUTPUT MFILE
>KEY 29,2,DESC
>VERIFY
INPUT FILES = DFILE,EMPDATA.PUB
OUTPUT FILE = MFILE
KEY POSITION    LENGTH      TYPE      ASC/DESC
        29          2       BYTE        DESC    (MAJOR KEY)
>END
PURGE OLD OUTPUT FILE MFILE.GSTUDENT.INTRO ? YES
```

```
                        STATISTICS
NUMBER OF INPUT FILES =                         2
NUMBER OF RECORDS =                            55
SPACE AVAILABLE (IN WORDS) =               12,220
NUMBER OF COMPARES =                           53
CPU TIME (MINUTES) =                          .01
ELAPSED TIME (MINUTES) =                      .03
RECORD SIZE (IN BYTES) =                       80
 END OF PROGRAM
:FILE LP;DEV=LP;REC=-31
:RUN FCOPY.PUB.SYS
HP32212A.02.0 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>FROM=MFILE;TO=*LP
*200*J
WARNING: FROMFILE RECSIZE IS 80 BYTES, TOFILE RECSIZE IS 31 BYTES.
CONTINUE OPERATION (Y OR N) ?Y
EOF FOUND IN FROMFILE AFTER RECORD 54
55 RECORDS PROCESSED *** 0 ERRORS
>EXIT
 END OF PROGRAM
 ********************************************************************
 *      OPTIONAL -- Proceed only if time permits.                  *
 ********************************************************************

 7)   Build and execute a Job Stream to accomplish the above lab.  From
 a Job the output of Sort is not allowed to go into the input file, so
 create a temporary file to contain the output of step 4) and be the
 input for step 5).
                                                        << End >>
:EDITOR
HP32201A.7.00 EDIT/3000  TUE, APR 25, 1978,  1:08 PM
(C) HEWLETT-PACKARD CO. 1976
/A
     1      !JOB STUDENT.INTRO/PASSWORD
     2      !BUILD TEMP;REC=-80,16,F,ASCII;DISC=100;TEMP
     3      !RUN SORT.PUB.SYS
     4      INPUT DFILE
     5      OUTPUT TEMP
     6      KEY 29,2,DESC
     7      END
     8      !RUN MERGE.PUB.SYS
     9      INPUT EMPDATA.PUB,TEMP
```

```
   10        OUTPUT MFILE
   11        KEY 29,2,DESC
   12        END
   13        !FILE LP;DEV=LP;REC=-31
   14        !RUN FCOPY.PUB.SYS
   15        FROM=OUTPUT0;TO=*LP
   16        EXIT
   17        !EOJ
   18        ...
/K SORTJOB,UNN
/E
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
 END OF SUBSYSTEM
:STREAM SORTJOB
 #J4
```

S E G M E N T E R   L A B   #1 [1.0 hour]

```
       ***  Please read the entire lab before proceeding!  ***
Create an SL
------------

1)    Compile the COBOL program VALIDNO.PUB into a USL file by
      itself.  You can guarantee this by specifying a 'uslfile' of
      '$NEWPASS'.
2)    Invoke the Segmenter.
3)    Point to $OLDPASS as the 'uslfile'.
4)    Build an SL file called 'SL' in your group 20 records long
      in 1 extent.
5)    List the SL file -- it should be empty (about 10% of its
      space will be reserved for the directory, however).
6)    List the USL file.
7)    Use 2 ADDSL commands to copy both Code Segments in the USL
      into the SL (COBOL generates an additional initialization
      code segment for each code segment normally generated).
8)    List the SL to make sure both segments are there.
9)    Exit the Segmenter.
```

```
:HELLO STUDENT.INTRO/PASSWORD
 SESSION NUMBER = #S98
 MON, MAR  6, 1978,  5:38 PM
 HP32002A.01.MR
WELCOME TO YOUR FRIENDLY HP-3000.
:COBOL VALIDNO.PUB,$NEWPASS
PAGE 0001    HP32213C.02.00  (C) HEWLETT-PACKARD CO. 1977
PAGE 0002    VALID-NO          VALIDATE PART-NO AS HAVING CORRECT CHECK-DIGIT -
<< SL SUBPROGRAM >>.
    DATA AREA IS %000376 WORDS.
    CPU TIME = 0:00:02.  WALL TIME = 0:00:08.
END COBOL/3000 COMPILATION.  NO ERRORS.  NO WARNINGS.
 END OF COMPILE
:SEGMENTER
SEGMENTER SUBSYSTEM (C.0)
-USL $OLDPASS
-BUILDSL SL,20,1
-LISTSL
SL FILE SL.GSTUDENT.INTRO
USED                    600            AVAILABLE               4200
```

```
-LISTUSL
USL FILE $OLDPASS..
VALIDNO'
    VALIDNO'          254  P  A C N R
VALIDNO
    VALIDNO           623  P  A C N R
    VALIDNO'S              CP A C   R
FILE SIZE           377600
DIR. USED              125        INFO USED         1273
DIR. GARB.               0        INFO GARB.           0
DIR. AVAIL.          37253        INFO AVAIL.      336505
-ADDSL VALIDNO'
-ADDSL VALIDNO
-LISTSL
SL FILE SL.GSTUDENT.INTRO
SEGMENT   0 VALIDNO'           LENGTH   260
    ENTRY POINTS    CHECK CAL STT  ADR
    VALIDNO'          0    C   1     0
    EXTERNALS       CHECK STT SEG
    VALIDNO'S         0    2   1
11
SEGMENT   1 VALIDNO            LENGTH   634
    ENTRY POINTS    CHECK CAL STT  ADR
    VALIDNO           2    C   1     0
    VALIDNO'S         1    C   2   620
    EXTERNALS       CHECK STT SEG
    C'DISPLAY'FIN     0    10  ?
    C'DISPLAY'L       0     7  ?
    C'DISPLAY'INIT    0     6  ?
    C'DISPLAY'ID      0     5  ?
    C'TST'NUM         0     4  ?
    VALIDNO'          0     3  0
11
USED             3600        AVAILABLE        1200
-EXIT
 END OF SUBSYSTEM
 :
```

Create an RL
----

10) Compile the COBOL program DISCIO.PUB into $NEWPASS (specify
    $NEWPASS to make sure the object output of this compile goes
    into a different USL file than the previous one).
11) Invoke the Segmenter.
12) Point to $OLDPASS as the USL file.
13) Build an RL file called 'SEGRL' in your group 30 records
    long in 1 extent.
14) List the RL file -- it should be empty but have space
    reserved for the directory.
15) List the USL then use 2 ADDRL commands to copy RBM's
    "DISCIO'" and "DISCIO" into the RL file.
16) List the RL file to make sure both RBM's are there.
17) Exit the Segmenter.

```
:COBOL DISCIO.PUB,$NEWPASS
PAGE 0001    HP32213C.02.00   (C) HEWLETT-PACKARD CO. 1977
PAGE 0002    DISC-IO          DO DISC I/O FOR ALL INVENTORY PROGRAMS - < RL SUB
PROGRAM >.
    DATA AREA IS %000551 WORDS.
    CPU TIME = 0:00:06.   WALL TIME = 0:00:13.
END COBOL/3000 COMPILATION.   NO ERRORS.   NO WARNINGS.
 END OF COMPILE
:SEGMENTER
SEGMENTER SUBSYSTEM (C.0)
-USL $OLDPASS
-BUILDRL SEGRL,30,1
-LISTRL
RL FILE SEGRL.GSTUDENT.INTRO
* ENTRY POINTS *
* EXTERNALS *
USED                400            AVAILABLE            7000
-LISTUSL
USL FILE $OLDPASS..
```

```
DISCIO'
   DISCIO'          164  P  A C N R
DISCIO
   DISCIO          3343  P  A C N R
   DISCIO'S              CP A C   R
FILE SIZE         377600
DIR. USED            210            INFO USED           4661
DIR. GARB.             0            INFO GARB.             0
DIR. AVAIL.        37170            INFO AVAIL.       333117
-ADDRL DISCIO'
-ADDRL DISCIO
-LISTRL
RL FILE SEGRL.GSTUDENT.INTRO
* ENTRY POINTS *
DISCIO'S         2  3331       1200
DISCIO          2     0       1200     2  3343  5023
DISCIO'         0     0        400     1   164   212
* EXTERNALS *
DISCIO'S         0  3331       1200
DISCIO'          0     0        400
C'DISPLAY'INIT   0
C'DISPLAY'L      0
C'DISPLAY'ID     0
C'DISPLAY'FIN    0
C'READD          0
C'DISPLAY'INIT   0
C'DISPLAY'L      0
C'DISPLAY'ID     0
C'DISPLAY'FIN    0
C'WRITED         0
C'PERFORM        0
C'PERFORM        0
C'READD          0
C'DISPLAY'INIT   0
C'DISPLAY'L      0
C'DISPLAY'ID     0
C'DISPLAY'FIN    0
C'DISPLAY'INIT   0
C'DISPLAY'L      0
C'DISPLAY'FIN    0
C'DISPLAY'INIT   0
C'DISPLAY'L      0
C'DISPLAY'FIN    0
C'WRITED         0
```

```
C'PERFORM          0
C'READD            0
C'PERFORM          0
C'DISPLAY'INIT     0
C'DISPLAY'L        0
C'DISPLAY'FIN      0
C'DISPLAY'INIT     0
C'DISPLAY'L        0
C'DISPLAY'FIN      0
C'OPEN'            0
C'READD            0
C'WRITED           0
C'DISPLAY'INIT     0
C'DISPLAY'L        0
C'DISPLAY'FIN      0
C'DISPLAY'INIT     0
C'DISPLAY'L        0
C'DISPLAY'FIN      0
C'CLOSE            0
C'ENDPAR           0
C'ENDPAR           0
C'DISPLAY'INIT     0
C'DISPLAY'L        0
C'DISPLAY'ID       0
C'DISPLAY'FIN      0
C'DISPLAY'INIT     0
C'DISPLAY'L        0
C'DISPLAY'FIN      0
USED                      6500          AVAILABLE              700
-EXIT
  END OF SUBSYSTEM
    Use the RL and SL
    ------------------

    18)   Compile the COBOL program 'INVUPD.PUB' into $NEWPASS.
    19)   Invoke the Segmenter.
    20)   Point to $OLDPASS as the USL file.
    21)   Prepare this USL creating a program file 'SEGRUN'.  Obtain a
          PMAP, use MAXDATA=10000, and use the RL file you created in
          steps 10 through 17.
    22)   Exit the Segmenter.
    23)   Run 'SEGRUN'.  You should get a LOAD ERROR 201,27
          "Unresolved Prog External VALIDNO".
    24)   Run 'SEGRUN' again, this time obtaining an LMAP and
          specifying LIB=G to use your group SL you created in steps 1
          through 9.
    25)   Enter a '/' to exit program.

:COBOL INVUPD.PUB,$NEWPASS
```

```
PAGE 0001   HP32213C.02.00   (C) HEWLETT-PACKARD CO. 1977
PAGE 0002   INV-UPDATE          UPDATE INVENTORY MASTER RECORDS - MAIN PROGRAM
    DATA AREA IS %000464 WORDS.
    CPU TIME = 0:00:04.   WALL TIME = 0:00:10.
END COBOL/3000 COMPILATION.   NO ERRORS.   NO WARNINGS.
 END OF COMPILE
:SEGMENTER
SEGMENTER SUBSYSTEM (C.0)
-USL $OLDPASS
-PREPARE SEGRUN;PMAP;RL=SEGRL;MAXDATA=10000
PROGRAM FILE SEGRUN.GSTUDENT.INTRO
INVUPDATE            0
    NAME            STT   CODE  ENTRY  SEG
    INVUPDATE'       1     0      0
    DEBUG            2                  ?
    COBOLTRAP        3                  ?
    INVUPDBEGIN00'   4                  1
    C'GOTO           5                  ?
    SEGMENT LENGTH         54
INVUPDBEGIN00'       1
    NAME            STT   CODE  ENTRY  SEG
    INVUPDBEGIN00'   1     0      0
    DISCIO           2                  2
    C'DISPLAY'INIT   3                  ?
    C'DISPLAY'L      4                  ?
    C'DISPLAY'FIN    5                  ?
    C'ACCEPT         6                  ?
    C'DISPLAY'ID     7                  ?
    VALIDNO          10                 ?
    C'TST'NUM        11                 ?
    C'PERFORM        12                 ?
    C'ENDPAR         13                 ?
    TERMINATE'       14                 ?
    SEGMENT LENGTH        2014
RL SEGMENT           2
    DISCIO           1     0      0
    DISCIO'S         3     0    3331
    C'DISPLAY'INIT   4                  ?
    C'DISPLAY'L      5                  ?
    C'DISPLAY'ID     6                  ?
    C'DISPLAY'FIN    7                  ?
```

```
        C'READD          10                    ?
        C'WRITED         11                    ?
        C'PERFORM        12                    ?
        C'OPEN'          13                    ?
        C'CLOSE          14                    ?
        C'ENDPAR         15                    ?
        DISCIO'           2    3343   3343
        SEGMENT LENGTH         3550
PRIMARY DB          0    INITIAL STACK    1440    CAPABILITY        600
SECONDARY DB     1235    INITIAL DL          0    TOTAL CODE       5640
TOTAL DB         1235    MAXIMUM DATA    23420    TOTAL RECORDS      43
ELAPSED TIME   00:00:09.959                 PROCESSOR TIME   00:01.829
-EXIT
 END OF SUBSYSTEM
:RUN SEGRUN
UNRESOLVED PROG EXTERNAL VALIDNO
ERR 201,27 J
LOAD ERROR
:RUN SEGRUN;LMAP;LIB=G
PROGRAM FILE SEGRUN.GSTUDENT.INTRO
VALIDNO         PROG 2   10    1 GSL   2    1    1
C'CLOSE         PROG 0   14    2 SSL   0   26  165
C'OPEN'         PROG 0   13    2 SSL   0   10  165
C'WRITED        PROG 0   11    2 SSL   0    1  165
C'READD         PROG 0   10    2 SSL   0    6  165
TERMINATE'      PROG 0   14    1 SSL   0    2   43
C'ENDPAR        PROG 0   15    2 SSL   0   12  164
                         13    1
C'PERFORM       PROG 0   12    2 SSL   0    5  164
                         12    1
C'TST'NUM       PROG 0   11    1 SSL   0    4  163
C'TST'NUM       GSL  0    4    1 SSL   0    4  163
C'DISPLAY'ID    PROG 0    6    2 SSL   0   17  165
                          7    1
C'DISPLAY'ID    GSL  0    5    1 SSL   0   17  165
C'ACCEPT        PROG 0    6    1 SSL   0   31  165
C'DISPLAY'FIN   PROG 0    7    2 SSL   0   20  165
                          5    1
C'DISPLAY'FIN   GSL  0   10    1 SSL   0   20  165
C'DISPLAY'L     PROG 0    5    2 SSL   0   15  165
                          4    1
C'DISPLAY'L     GSL  0    7    1 SSL   0   15  165
C'DISPLAY'INIT  PROG 0    4    2 SSL   0   16  165
                          3    1
C'DISPLAY'INIT  GSL  0    6    1 SSL   0   16  165
C'GOTO          PROG 0    5    0 SSL   0   11  164
COBOLTRAP       PROG 0    3    0 SSL   0    6  166
```

```
DEBUG          PROG 0   2   0 SSL  0   1  57
301 302 303
INVUPD(20AUG77) -- UPDATE INVENTORY MASTER.
  <ENTER DISCIO> TRANS-TYPE = O
  <DISCIO> OPEN INV-FILE.
**<DISCIO> FILE INITIALLY EMPTY -- MOVING EOF TO RECORD 1000.**
  ********************************************
  * formal-file-designator EXPECTED BY THIS *
  * PROGRAM IS 'INVFILE'.  REMEMBER... IF    *
  * COBOL DOES NOT FIND A PERMANENT FILE BY  *
  * THAT NAME, IT WILL CREATE ONE IN THE     *
  * NEW DOMAIN WHICH WILL DISAPPEAR UPON     *
  * BEING CLOSED UNLESS TOLD OTHERWISE BY    *
  * A FILE EQUATION!  (@#!*?/]&%@)           *
  ********************************************
  <EXIT DISCIO>
BE SURE TO ENTER ALL CHARS FOR A FIELD.
(INCLUDING TRAILING BLANKS).
VALID TRANS-TYPES ARE:
N = NEW RECORD.
R = READ EXISTING RECORD.
A = ADD TO INVENTORY.
S = SUBTRACT FROM INVENTORY.
/ = END-OF-PROGRAM.
TRANS-TYPE?
/
  <ENTER DISCIO> TRANS-TYPE = C
  <DISCIO> CLOSE INV-FILE.
  <EXIT DISCIO>
 END OF PROGRAM
   *************************************************************************
   *       Problems you may encounter.                                    *
   *************************************************************************

        If you repeat steps, be careful.  When you build a library
   file with the Segmenter, it builds it as 'NEW' with a close
   disposition of 'SAVE'.  This means it is possible to create a
   second library file with the same name and you will only learn of
   the conflict when you close the newly created file, normally as
   you exit the Segmenter.  If you already have a file of that name,
   point to it, don't create another one.
                                     << End >>
```

‹‹‹ use figure from KSAM-19 here ›››

A KSAM file with the above format resides in PUB.INTRO. Keys
have been defined for positions 1, 11, 53 and 67 of the record.
The data file is 'KDATA'; the key file is 'KKEY'.

1)   Using FCOPY, list KDATA.PUB on your terminal in chronolog-
ical sequence (the order in which the file was written).
:HELLO STUDENT.INTRO/PASSWORD
HP3000 III. MON, APR 24, 1978, 6:36 PM
******** WELCOME **************
:RUN FCOPY.PUB.SYS
HP32212A.3.03 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>FROM=KDATA.PUB;TO=
PHIL   ARBUSTER   997-1040 672   CONSTITUTION DR SANTA CLARA   95110
JOSE   CANUSI   214-5566 2485   ANTHEM WY   CAMPBELL   95129
NEIL   DU PREE   246-1112 4097   PRIE DIEUX DR   SAN JOSE   95013
ALI   FATIQ   292-0100 480   DU PONT CIR   SAN JOSE   95131
ROSE   GARLAND   269-7132 5219   PARK MEADOW CT   SAN JOSE   95054
XAVIER   GREENSTAMP 247-5423 1551   PREMIUM ST   SAN JOSE   95134
ARMAND   HAMMER   298-4988 1350   ALKALI AV   CAMBRIAN PARK 95131
KNUT   HEERJIT   923-3485 1740   VIA ABSENTIA   SAN JOSE   95053
HY   HILL   593-8421 48709 BLUE RIDGE DR   MILPITAS   95035
ALI   KATZ   296-7650 262   MEHITABEL AVE   SANTA CLARA   95133
VY   KNOTT   262-8940 1883   QUERY PL   SAN CARLOS   95014
ANNA   LOGUE   224-8934 1707   INVERSE WY   MOUNTAIN VIEW 95051
ANDY   LUCIAN   264-4169 1119   IBERIAN CT   CUPERTINO   95070
ARTHUR   MOMITER   443-5346 1554   MERCURY ST   MILPITAS   94173
CLARA   NETTE   243-4493 2667   GOODMAN DR   ALVISO   95143
AL   PINE   578-2868 1738   DRY CREEK ROAD   SAN JOSE   95116
RACHAEL   PREJUDICE   262-8940 1730   WARREN CT   SANTA CLARA   95035
AMOS   QUITO   243-8171 1467   ANOPHELES AV   NEW ALMADEN   95143
AMANDA   RECKONWITH 247-9142 2474   MACHO ST   SANTA CLARA   95020
BUZZ   SAWYER   259-3434 1850   FOREST DR   CUPERTINO   95023
BEAUFORT   SCALE   328-7540 3843   WINDY WY   SAN JOSE   95117
TYRONE   SHOELACES   266-1721 17265 BLUCHER BLVD   LOS GATOS   95131
EILEEN   SIDEWAYS   377-7545 2577   TILDEN BLVD   SAN JOSE   95111
OLIVER   TEETHOUT   867-0138 20085 UPPER PLATE PL   CUPERTINO   95053
TRUDY   TEKTIFF   255-1005 17155 POIROT PL   CAMPBELL   95121
BRICK   WALL   288-7761 2950   STORY ROAD   SAN JOSE   95131
EOF FOUND IN FROMFILE AFTER RECORD 25

26 RECORDS PROCESSED *** 0 ERRORS
2)   List the file in alphabetical order by last name on your
terminal.
>FROM=KDATA.PUB;TO=;KEY=11

PHIL	ARBUSTER	997-1040	672	CONSTITUTION DR	SANTA CLARA	95110
JOSE	CANUSI	214-5566	2485	ANTHEM WY	CAMPBELL	95129
NEIL	DU PREE	246-1112	4097	PRIE DIEUX DR	SAN JOSE	95013
ALI	FATIQ	292-0100	480	DU PONT CIR	SAN JOSE	95131
ROSE	GARLAND	269-7132	5219	PARK MEADOW CT	SAN JOSE	95054
XAVIER	GREENSTAMP	247-5423	1551	PREMIUM ST	SAN JOSE	95134
ARMAND	HAMMER	298-4988	1350	ALKALI AV	CAMBRIAN PARK	95131
KNUT	HEERJIT	923-3485	1740	VIA ABSENTIA	SAN JOSE	95053
HY	HILL	593-8421	48709	BLUE RIDGE DR	MILPITAS	95035
ALI	KATZ	296-7650	262	MEHITABEL AVE	SANTA CLARA	95133
VY	KNOTT	262-8940	1883	QUERY PL	SAN CARLOS	95014
ANNA	LOGUE	224-8934	1707	INVERSE WY	MOUNTAIN VIEW	95051
ANDY	LUCIAN	264-4169	1119	IBERIAN CT	CUPERTINO	95070
ARTHUR	MOMITER	443-5346	1554	MERCURY ST	MILPITAS	94173
CLARA	NETTE	243-4493	2667	GOODMAN DR	ALVISO	95143
AL	PINE	578-2868	1738	DRY CREEK ROAD	SAN JOSE	95116
RACHAEL	PREJUDICE	262-8940	1730	WARREN CT	SANTA CLARA	95035
AMOS	QUITO	243-8171	1467	ANOPHELES AV	NEW ALMADEN	95143
AMANDA	RECKONWITH	247-9142	2474	MACHO ST	SANTA CLARA	95020
BUZZ	SAWYER	259-3434	1850	FOREST DR	CUPERTINO	95023
BEAUFORT	SCALE	328-7540	3843	WINDY WY	SAN JOSE	95117
TYRONE	SHOELACES	266-1721	17265	BLUCHER BLVD	LOS GATOS	95131
EILEEN	SIDEWAYS	377-7545	2577	TILDEN BLVD	SAN JOSE	95111
OLIVER	TEETHOUT	867-0138	20085	UPPER PLATE PL	CUPERTINO	95053
TRUDY	TEKTIFF	255-1005	17155	POIROT PL	CAMPBELL	95121
BRICK	WALL	288-7761	2950	STORY ROAD	SAN JOSE	95131

EOF FOUND IN FROMFILE AFTER RECORD 25
26 RECORDS PROCESSED *** 0 ERRORS
3)   List the file in order by zip code on your terminal.
>FROM=KDATA.PUB;TO=;KEY=67

ARTHUR	MOMITER	443-5346	1554	MERCURY ST	MILPITAS	94173
NEIL	DU PREE	246-1112	4097	PRIE DIEUX DR	SAN JOSE	95013
VY	KNOTT	262-8940	1883	QUERY PL	SAN CARLOS	95014
AMANDA	RECKONWITH	247-9142	2474	MACHO ST	SANTA CLARA	95020
BUZZ	SAWYER	259-3434	1850	FOREST DR	CUPERTINO	95023
RACHAEL	PREJUDICE	262-8940	1730	WARREN CT	SANTA CLARA	95035
HY	HILL	593-8421	48709	BLUE RIDGE DR	MILPITAS	95035
ANNA	LOGUE	224-8934	1707	INVERSE WY	MOUNTAIN VIEW	95051
OLIVER	TEETHOUT	867-0138	20085	UPPER PLATE PL	CUPERTINO	95053
KNUT	HEERJIT	923-3485	1740	VIA ABSENTIA	SAN JOSE	95053
ROSE	GARLAND	269-7132	5219	PARK MEADOW CT	SAN JOSE	95054
ANDY	LUCIAN	264-4169	1119	IBERIAN CT	CUPERTINO	95070
PHIL	ARBUSTER	997-1040	672	CONSTITUTION DR	SANTA CLARA	95110
EILEEN	SIDEWAYS	377-7545	2577	TILDEN BLVD	SAN JOSE	95111

```
AL          PINE        578-2868 1738  DRY CREEK ROAD     SAN JOSE       95116
BEAUFORT    SCALE       328-7540 3843  WINDY WY           SAN JOSE       95117
TRUDY       TEKTIFF     255-1005 17155 POIROT PL          CAMPBELL       95121
JOSE        CANUSI      214-5566 2485  ANTHEM WY          CAMPBELL       95129
TYRONE      SHOELACES   266-1721 17265 BLUCHER BLVD       LOS GATOS      95131
ARMAND      HAMMER      298-4988 1350  ALKALI AV          CAMBRIAN PARK  95131
BRICK       WALL        288-7761 2950  STORY ROAD         SAN JOSE       95131
ALI         FATIQ       292-0100 480   DU PONT CIR        SAN JOSE       95131
ALI         KATZ        296-7650 262   MEHITABEL AVE      SANTA CLARA    95133
XAVIER      GREENSTAMP247-5423 1551    PREMIUM ST         SAN JOSE       95134
AMOS        QUITO       243-8171 1467  ANOPHELES AV       NEW ALMADEN    95143
CLARA       NETTE       243-4493 2667  GOODMAN DR         ALVISO         95143
EOF FOUND IN FROMFILE AFTER RECORD 25
26 RECORDS PROCESSED *** 0 ERRORS
```

4)   List the first five records in primary key sequence on your
terminal.

```
>FROM=KDATA.PUB;TO=;SUBSET=,5
PHIL        ARBUSTER    997-1040 672   CONSTITUTION DR    SANTA CLARA    95110
JOSE        CANUSI      214-5566 2485  ANTHEM WY          CAMPBELL       95129
NEIL        DU PREE     246-1112 4097  PRIE DIEUX DR      SAN JOSE       95013
ALI         FATIQ       292-0100 480   DU PONT CIR        SAN JOSE       95131
ROSE        GARLAND     269-7132 5219  PARK MEADOW CT     SAN JOSE       95054
5 RECORDS PROCESSED *** 0 ERRORS
```

5)   List all those people who live in San Jose in zip code
sequence on your terminal.

```
>FROM=KDATA.PUB;TO=;KEY=67;SUBSET="SAN JOSE",53
NEIL        DU PREE     246-1112 4097  PRIE DIEUX DR      SAN JOSE       95013
KNUT        HEERJIT     923-3485 1740  VIA ABSENTIA       SAN JOSE       95053
ROSE        GARLAND     269-7132 5219  PARK MEADOW CT     SAN JOSE       95054
EILEEN      SIDEWAYS    377-7545 2577  TILDEN BLVD        SAN JOSE       95111
AL          PINE        578-2868 1738  DRY CREEK ROAD     SAN JOSE       95116
BEAUFORT    SCALE       328-7540 3843  WINDY WY           SAN JOSE       95117
BRICK       WALL        288-7761 2950  STORY ROAD         SAN JOSE       95131
ALI         FATIQ       292-0100 480   DU PONT CIR        SAN JOSE       95131
XAVIER      GREENSTAMP247-5423 1551    PREMIUM ST         SAN JOSE       95134
EOF FOUND IN FROMFILE AFTER RECORD 25
9 RECORDS PROCESSED *** 0 ERRORS
```

6)   List 10 records beginning with record number 6 in
alphabetical sequence by first name on your terminal.

```
>FROM=KDATA.PUB;TO=;KEY=1;SUBSET=6,10
ANNA        LOGUE       224-8934 1707  INVERSE WY         MOUNTAIN VIEW 95051
ARMAND      HAMMER      298-4988 1350  ALKALI AV          CAMBRIAN PARK 95131
ARTHUR      MOMITER     443-5346 1554  MERCURY ST         MILPITAS      94173
BEAUFORT    SCALE       328-7540 3843  WINDY WY           SAN JOSE      95117
BRICK       WALL        288-7761 2950  STORY ROAD         SAN JOSE      95131
BUZZ        SAWYER      259-3434 1850  FOREST DR          CUPERTINO     95023
```

```
CLARA      NETTE       243-4493 2667  GOODMAN DR       ALVISO        95143
EILEEN     SIDEWAYS    377-7545 2577  TILDEN BLVD      SAN JOSE      95111
HY         HILL        593-8421 48709 BLUE RIDGE DR    MILPITAS      95035
JOSE       CANUSI      214-5566 2485  ANTHEM WY        CAMPBELL      95129
10 RECORDS PROCESSED *** 0 ERRORS
    7)  Copy the file in chronological sequence to a non-KSAM file
    in your group.  List this non-KSAM file on your terminal, then
    purge it.
>                                          << BREAK pressed >>
:LISTF KDATA.PUB,2
ACCOUNT= INTRO       GROUP=  PUB
FILENAME  CODE   ------------LOGICAL RECORD----------  ----SPACE----
                 SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX
KDATA            80B   FA         26        30   1      30   6  7
:BUILD NEWFILE;REC=-80,16,F,ASCII;DISC=100
:RESUME
READ PENDING
FROM=KDATA.PUB;TO=NEWFILE;KEY=0
EOF FOUND IN FROMFILE AFTER RECORD 25
26 RECORDS PROCESSED *** 0 ERRORS
>FROM=NEWFILE;TO=
NEIL       DU PREE     246-1112 4097  PRIE DIEUX DR    SAN JOSE      95013
OLIVER     TEETHOUT    867-0138 20085 UPPER PLATE PL   CUPERTINO     95053
AMANDA     RECKONWITH247-9142 2474    MACHO ST         SANTA CLARA   95020
AMOS       QUITO       243-8171 1467  ANOPHELES AV     NEW ALMADEN   95143
ARTHUR     MOMITER     443-5346 1554  MERCURY ST       MILPITAS      94173
EILEEN     SIDEWAYS    377-7545 2577  TILDEN BLVD      SAN JOSE      95111
TYRONE     SHOELACES 266-1721 17265   BLUCHER BLVD     LOS GATOS     95131
XAVIER     GREENSTAMP247-5423 1551    PREMIUM ST       SAN JOSE      95134
RACHAEL    PREJUDICE 262-8940 1730    WARREN CT        SANTA CLARA   95035
AL         PINE        578-2868 1738  DRY CREEK ROAD   SAN JOSE      95116
ARMAND     HAMMER      298-4988 1350  ALKALI AV        CAMBRIAN PARK 95131
TRUDY      TEKTIFF     255-1005 17155 POIROT PL        CAMPBELL      95121
BRICK      WALL        288-7761 2950  STORY ROAD       SAN JOSE      95131
JOSE       CANUSI      214-5566 2485  ANTHEM WY        CAMPBELL      95129
ROSE       GARLAND     269-7132 5219  PARK MEADOW CT   SAN JOSE      95054
HY         HILL        593-8421 48709 BLUE RIDGE DR    MILPITAS      95035
KNUT       HEERJIT     923-3485 1740  VIA ABSENTIA     SAN JOSE      95053
VY         KNOTT       262-8940 1883  QUERY PL         SAN CARLOS    95014
ALI        FATIQ       292-0100 480   DU PONT CIR      SAN JOSE      95131
ALI        KATZ        296-7650 262   MEHITABEL AVE    SANTA CLARA   95133
ANNA       LOGUE       224-8934 1707  INVERSE WY       MOUNTAIN VIEW 95051
ANDY       LUCIAN      264-4169 1119  IBERIAN CT       CUPERTINO     95070
BEAUFORT   SCALE       328-7540 3843  WINDY WY         SAN JOSE      95117
BUZZ       SAWYER      259-3434 1850  FOREST DR        CUPERTINO     95023
```

```
PHIL      ARBUSTER 997-1040 672   CONSTITUTION DR  SANTA CLARA    95110
CLARA     NETTE    243-4493 2667  GOODMAN DR       ALVISO         95143
EOF FOUND IN FROMFILE AFTER RECORD 25
26 RECORDS PROCESSED *** 0 ERRORS
>EXIT
END OF PROGRAM
:PURGE NEWFILE
   *****************************************************************
   *     OPTIONAL -- Proceed only if time permits.                *
   *****************************************************************

   8)  'Build a KSAM file in your group by modifying and running a
   job stream.  From the editor, text in KSAMBILD.PUB unnumbered.
   Modify the ':JOB' command in record 1 to reference your Username.
   Keep this job stream unnumbered in a file 'LABKSAM1' in your
   group.  Now ':STREAM' this file.  Upon completion of that JOB,
   inspect the line printer listing and find KSAMDATA and KSAMKEY
   have been built in your group.
:EDITOR
HP32201A.7.0H EDIT/3000  MON, APR 24, 1978,  6:48 PM
(C) HEWLETT-PACKARD CO. 1976
/T KSAMBILD.PUB,UNN
/M 1
MODIFY       1
!JOB MYNAME.INTRO/TRAIN
      RSTUDENT.INTRO/PASSWORD
!JOB STUDENT.INTRO/PASSWORD
/K LABKSAM1,UNN
/EXIT
END OF SUBSYSTEM
:STREAM LABKSAM1
 #J54
:SHOWJOB #J54
 NO SUCH JOB(S)
JOBFENCE= 2; JLIMIT= 2; SLIMIT= 16
:LISTF
FILENAME
KSAMDATA     KSAMKEY      LABKSAM1
   9)   Load KSAMDATA in your group from LAB1DATA in PUB using
   FCOPY.
:RUN FCOPY.PUB.SYS
HP32212A.3.03 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
```

```
>FROM=LAB1DATA.PUB;TO=KSAMDATA
EOF FOUND IN FROMFILE AFTER RECORD 25
26 RECORDS PROCESSED *** 0 ERRORS
>EXIT
END OF PROGRAM
    10)  Run KSAMUTIL and using 'HELP' list all KSAMUTIL commands on
    your terminal.
:RUN KSAMUTIL.PUB.SYS
HP32208Z.1.6  MON, APR 24, 1978,  6:52 PM
>HELP
VALID COMMANDS ARE:
BUILD, TO CREATE A FILE
ERASE, TO RESET A FILE TO INITIAL CONDITIONS
EXIT, TO LEAVE THIS ROUTINE
PURGE, TO DELETE A FILE
RENAME
SAVE, TO RETAIN A TEMPORARY FILE
VERIFY, TO DESCRIBE FILE CHARACTERISTICS
MORE (Y/N)?Y
ENTER COMMAND NAME: BUILD
BUILD <DATAFILEREF>
[;DEV=<DEVICE>]
[;DISC=[<NUMREC>][,<NUMEXTENTS>][,<INITALLOC>]]]
[REC=[<RECSIZE>][,[<BLOCKFACTOR>][,[F ][,BINARYASCII]]]]
[;TEMP]
[;CODE=<FILECODE>]
;KEY=<TYPE>,<POSITION>[,[<LENGTH>][,[<BLOCKING>][,DUPLICATE]]]
[;KEY=<TYPE>,<POSITION>[,[<LENGTH>][,[<BLOCKING>][,DUPLICATE]]] ...]
[;LABELS=<NUMBERLABELS>]
[;FIRSTREC=0]
[;KEYDEV=<DEVICE>]
;KEYFILE=FILEREFERENCE2
[;KEYENTRIES=<NUMBER>]

<TYPE>::=B\D\I\R\L\N\P\*

MORE (Y/N)?Y

ENTER COMMAND NAME: EXIT

EXIT
```

```
MORE (Y/N)?N
     11)  Use the VERIFY command to display the attributes of           |
     KSAMDATA.                                                          |
>VERIFY KSAMDATA
WHICH (1=FILE INFO, 2=KSAM PARAMETERS,  3=KSAM CONTROL, 4=ALL, 5=NONE)?4
KSAMDATA.GSTUDENT.INTRO     CREATOR=STUDENT
FOPTIONS(004005)=KSAM, :FILE, NOCCTL, F, FILENAME, ASCII, PERM
AOPTIONS(000400)=DEFAULT, NOBUF, DEFAULT, NO FLOCK, NO MR, IN
RECSIZE:SUB:TYP:LDNUM:DRT:UN.:  CODE:LOGICAL PTR: END OF FILE:FILE LIMIT
    -71: 3: 0:   3: 5: 1:     0:           0:           26:        30
  LOG. COUNT:PHYS. COUNT:BLK SZ:EXT SZ:NR EXT: LABELS:LDN:   DISCADDR:
          1:         1:  -72:    5:    7:     1: 3:00000003316:
KEY FILE=KSAMKEY  KEY FILE DEVICE=4          SIZE=          66 KEYS=      4
FLAGWORD(000000)=RANDOM PRIMARY, FIRST RECORD=0, PERMANENT
KEY TY LENGTH   LOC. D KEY BF   LEVEL
   1  B     10     11 Y    112       1
   2  B     10      1 Y    112       1
   3  B     14     53 Y     92       1
   4  B      5     67 Y    144       1
DATA FILE = KSAMDATA     VERSION= Z.1.6
KEY CREATED=114/'78 18:50:34.6    KEY ACCESS= 114/'78 18:53:39.8
KEY CHANGED=114/'78 18:51:39.7    COUNT START=114/'78 18:51:40.3
DATA RECS  =           26 DATA BLOCKS=         25 END BLK WDS=        36
DATA BLK SZ=           36 DATA REC SZ=         71 ACCESSORS=           0
FOPEN                   2 FREAD                 0 FCLOSE               1
FREADDIR                0 FREADC                0 FREADBYKEY           0
FREMOVE                 0 FSPACE                0 FFINDBYKEY           0
FGETINFO                3 FGETKEYINFO           0 FREADLABEL           0
FWRITELABEL             0 FCHECK                0 FFINDN               0
FWRITE                 26 FUPDATE               0 FPOINT               0
FLOCK                   0 FUNLOCK               0 FCONTROL             0
FSETMODE                0
KEYBLK READ           107 KEYBLK WROTE         51 KEYBLK SPLIT         0
NEXT KB REC            34 FREE KEY HD           0
MIN PRIME             24 MAX PRIME            12
DATA FIXED           TRUE DATA B/F              1 TOTAL KEYS           4
FIRST RECNUM            0 MIN RECSIZE          71
WHICH (1=FILE INFO, 2=KSAM PARAMETERS,  3=KSAM CONTROL, 4=ALL, 5=NONE)?5
     12)  Use one ERASE command to delete all entries from both the     |
     data and key files but leave the structure intact.                 |
     13)  Use the VERIFY command to make sure there are no remaining     |
     entries in either file.                                            |
>ERASE KSAMDATA
>VERIFY KSAMDATA
WHICH (1=FILE INFO, 2=KSAM PARAMETERS,  3=KSAM CONTROL, 4=ALL, 5=NONE)?1
```

```
KSAMDATA.GSTUDENT.INTRO      CREATOR=STUDENT
FOPTIONS(004005)=KSAM, :FILE, NOCCTL, F, FILENAME, ASCII, PERM
AOPTIONS(000400)=DEFAULT, NOBUF, DEFAULT, NO FLOCK, NO MR, IN
RECSIZE:SUB:TYP:LDNUM:DRT:UN.:  CODE:LOGICAL PTR: END OF FILE:FILE LIMIT
    -71:  3:  0:    3:  5:  1:      0:          0:          0:        30
  LOG. COUNT:PHYS. COUNT:BLK SZ:EXT SZ:NR EXT: LABELS:LDN:   DISCADDR:
          0:          0:   -72:    5:    7:      1:  3:00000003316:
WHICH (1=FILE INFO, 2=KSAM PARAMETERS,  3=KSAM CONTROL, 4=ALL, 5=NONE)?5
>EXIT
END OF PROGRAM
    14)  Use FCOPY to copy KDATA.PUB into KSAMDATA using no 'KEY='
    parameter.  Now list KSAMDATA on your terminal in chronological
    sequence to see that the file is now stored in order by primary
    key.
:RUN FCOPY.PUB.SYS
HP32212A.3.03 FILE COPIER (C) HEWLETT-PACKARD CO. 1976
>FROM=KDATA.PUB;TO=KSAMDATA
*200*
EOF FOUND IN FROMFILE AFTER RECORD 25
26 RECORDS PROCESSED *** 0 ERRORS
>FROM=KSAMDATA;TO=;KEY=0
*200*
PHIL        ARBUSTER   997-1040 672    CONSTITUTION DR  SANTA CLARA   95110
JOSE        CANUSI     214-5566 2485   ANTHEM WY        CAMPBELL      95129
NEIL        DU PREE    246-1112 4097   PRIE DIEUX DR    SAN JOSE      95013
ALI         FATIQ      292-0100 480    DU PONT CIR      SAN JOSE      95131
ROSE        GARLAND    269-7132 5219   PARK MEADOW CT   SAN JOSE      95054
XAVIER      GREENSTAMP247-5423 1551    PREMIUM ST       SAN JOSE      95134
ARMAND      HAMMER     298-4988 1350   ALKALI AV        CAMBRIAN PARK 95131
KNUT        HEERJIT    923-3485 1740   VIA ABSENTIA     SAN JOSE      95053
HY          HILL       593-8421 48709  BLUE RIDGE DR    MILPITAS      95035
ALI         KATZ       296-7650 262    MEHITABEL AVE    SANTA CLARA   95133
VY          KNOTT      262-8940 1883   QUERY PL         SAN CARLOS    95014
ANNA        LOGUE      224-8934 1707   INVERSE WY       MOUNTAIN VIEW 95051
ANDY        LUCIAN     264-4169 1119   IBERIAN CT       CUPERTINO     95070
ARTHUR      MOMITER    443-5346 1554   MERCURY ST       MILPITAS      94173
CLARA       NETTE      243-4493 2667   GOODMAN DR       ALVISO        95143
AL          PINE       578-2868 1738   DRY CREEK ROAD   SAN JOSE      95116
RACHAEL     PREJUDICE  262-8940 1730   WARREN CT        SANTA CLARA   95035
AMOS        QUITO      243-8171 1467   ANOPHELES AV     NEW ALMADEN   95143
AMANDA      RECKONWITH247-9142 2474    MACHO ST         SANTA CLARA   95020
BUZZ        SAWYER     259-3434 1850   FOREST DR        CUPERTINO     95023
BEAUFORT    SCALE      328-7540 3843   WINDY WY         SAN JOSE      95117
```

```
TYRONE     SHOELACES 266-1721 17265 BLUCHER BLVD      LOS GATOS     95131
EILEEN     SIDEWAYS  377-7545 2577  TILDEN BLVD       SAN JOSE      95111
OLIVER     TEETHOUT  867-0138 20085 UPPER PLATE PL    CUPERTINO     95053
TRUDY      TEKTIFF   255-1005 17155 POIROT PL         CAMPBELL      95121
BRICK      WALL      288-7761 2950  STORY ROAD        SAN JOSE      95131
EOF FOUND IN FROMFILE AFTER RECORD 25
26 RECORDS PROCESSED *** 0 ERRORS

>EXIT

END OF PROGRAM
```

15) Use the PURGE command in KSAMUTIL to purge both KSAMDATA
and KSAMKEY at the same time. Using LISTF verify that they have
both been purged.

```
:RUN KSAMUTIL.PUB.SYS
HP32208Z.1.6  MON, APR 24, 1978,  6:57 PM
>PURGE KSAMDATA
KSAMDATA.GSTUDENT.INTRO & KSAMKEY PURGED.
>EXIT
END OF PROGRAM
:LISTF
FILENAME
LABKSAM1
:PURGE LABKSAM1
:BYE
CPU=27. CONNECT=22. MON, APR 24, 1978,  6:58 PM
```