

UNSUPPORTED UTILITIES

THE UNSUPPORTED UTILITIES

This is a collection of programs that is ever growing and ever changing. We do not purport to support these, only offer words of experience and perhaps a little advice.

These programs were developed by various people and submitted for use by us all. I am breaking them into two sections; one called harmless and useful, and one called dangerous but useful. The latter group contains those utilities which either allow modification of code directly in the system or which perform their function outside of the normal channels of MPE.

Use these to the fullest; be cautious and if unsure, ask.

THE HARMLESS ONES

CLOSEDEV

This is a handy utility for the area of device hang-ups. CLOSEDEV will check the logical device number passed in the PARM part of the command. If the device is being acquired as a virtual device, but is not owned by a process, then the device is cleared and may be used again.

:RUN CLOSEDEV;PARM=

Supply the number of the device to be cleared. If you're way off base, you'll get:

"AIN'T NO SUCH DEVICE FELLA"

or

"DEVICE IS NOT A VIRTUAL FILE"

Hopefully, you'll finally get success,

"VIRTUAL FILE DELETED"

CROSSREF

This is a handy utility which takes a sequenced SPL readable source tape and produces a cross-reference. All identifiers in the program, global and local, are listed alphabetically along with the sequence numbers of every reference to that identifier. The first sequence number always refers to the actual declaration, not including FORWARD, ENTRY, or LABEL. These exceptions are included in the sequence numbers that follow. ENTRY and LABEL identifiers are assumed to be declared at the point where the label actually occurs. When DEFINE appears, a reference for all identifiers encountered within the define, and a reference also appears for the DEFINE identifier itself. The format of the print-out is as follows:

ID OF SUB OF PROC (VALUE PARAMETER REAL)

Where ID is the actual identifier name; SUB is the subroutine in which the ID was declared; and PROC is the procedure in which the SUB was declared. If ID was not declared in a subroutine then its "ID OF PROC." If ID was global, just "ID" will appear. The following resulted from mounting my source tape and typing

:RUN CROSSREF

S.P.L. CROSS REFERENCE TABLE--- APR 10, 1973 VERSION

NUMBER OF CARD IMAGES=3364. NUMBER OF SYMBOLS=250. NUMBER OF REFERENCES=2650.

ADDAUXBUF OF GETREADY (SUBROUTINE)

01440000 01690000 01708000

ANSWERED OF ERROR (LOGICAL)

00191000 00302000 00303000 00319000 00333000 00336000

AREGLEN (INTEGER)

00084000 02904000 02983000 03063000 03089000 03092000 03101000

ASCII (INTRINSIC)

00159000 00402000 00407000 00412000 00417000 00458000 00462000

ASCII TORC DIC (PROCEDURE)

02203000 02688000 03058000

ASCII TORC DIC (PROCEDURE)

02153000 02678000 03055000

ATR OF ASCII TORC DIC (BYTE ARRAY)

02208000 02278000

Obviously, this is only a sample, the listing went on and on.

The output goes to PRF which defaults to line-printer.

DECOMP

Debugging high-level language code can be very tedious without a tool to break the code into machine executable symbols. The option, CODE, on your \$CONTROL card will give an octal representation of the compiler generated code, but DECOMP gives a SPL mnemonic listing of your program with such information as all reference to external procedure names, the segment transfer table; and all entry points have their labels inserted in the proper location. The following is a DECOMP of the file FREE. Dialogue runs like this:

```
:RUN DECOMP
FILE NAME?
FREE
NO. OF SEGMENTS: 1
STARTING SEGMENT ? 0
THIS SEG HAS LENGTH OF %534
ENTER STARTING P-VALUE (PRECEDED BY %) %500

END OF PROGRAM
```

Output for this program goes to the designator OUT which defaults to line printer.

This asked for a dump of Segment 0 beginning at location %500. Since the segment is %534, the results is as follows:

```
SEGMENT 0      PRIV=1      LENGTH =%000534

000500 004500 0 ----- DUP, NOP
000501 051607 S ----- STOR Q= 007
000502 020063 3 ----- MVB ,4 ,3
000503 031403 3 ----- EXIT %0003

000504 041605 C ----- LOAD Q= 005      <-- PROCEDURE ENTRY POINT
000505 037777 ? ----- ANDI %0377
000506 004500 0 ----- DUP, NOP
000507 022001 S ----- CMPI ,1
000510 141203 B ----- BE P+ 003
000511 031006 2 ----- PCAL DEBUG
000512 031007 2 ----- PCAL TERMINATE
000513 031402 3 ----- EXIT %0002
000514 106063 ----- SEGMENT TRANSFER TABLE (PL-%017)
000515 101420 ----- SEGMENT TRANSFER TABLE (PL-%016)
000516 101453 ----- SEGMENT TRANSFER TABLE (PL-%015)
000517 106463 ----- SEGMENT TRANSFER TABLE (PL-%014)
000520 107463 ----- SEGMENT TRANSFER TABLE (PL-%013)
000521 104463 ----- SEGMENT TRANSFER TABLE (PL-%012)
000522 101056 ----- SEGMENT TRANSFER TABLE (PL-%011)
000523 103063 ----- SEGMENT TRANSFER TABLE (PL-%010)
000524 100456 ----- SEGMENT TRANSFER TABLE (PL-%007)
000525 100470 ----- SEGMENT TRANSFER TABLE (PL-%006)
000526 100426 ----- SEGMENT TRANSFER TABLE (PL-%005)
000527 000504 ----- SEGMENT TRANSFER TABLE (PL-%004)
000530 000444 ----- SEGMENT TRANSFER TABLE (PL-%003)
000531 000110 ----- SEGMENT TRANSFER TABLE (PL-%002)
000532 000000 ----- SEGMENT TRANSFER TABLE (PL-%001)
000533 040017 -----
```

DISKDUMP

It is often necessary to see how a program is stored on the disk. This is especially true when investigating such items as USL headers or the like. Entries on the disk can be dumped by this utility in an octal format. All the user must provide is the file name and the range of records to be dumped to the output designator OUT, which defaults to the line-printer.

:RUN DISKDUMP

FILE NAME? FREE
RANGE? 0,5

END OF PROGRAM

The above example of dialogue caused five records to be dumped. The following is record 3 of the file FREE. This third record begins the segment 0 of the file as can be compared with the output from the utility PATCH. The first records of this dump were omitted.

```
FILE FREE.HPUNSUP.SUPPORT RECORD 2000003
 0 024410 013302 027410 041000 021013 000600 031012 000600 ).../.B."...2...
10 041001 040050 021001 035013 040046 031005 051010 141203 H.0(".:.062.R...
20 031006 031007 041005 000606 031012 041002 021011 021320 2.2.B..2.R."..
30 031012 000600 041005 025024 031010 004300 021015 166006 2...R.*.2..."...
40 000600 173006 040021 020121 011502 031011 004546 020220 ....0. Q.02..F .
50 040014 020122 011413 004002 041003 021007 000600 031012 0..R...B."...2.
60 140436 000414 016000 006440 006440 003221 141204 004500 ..... . ....0
70 022007 141303 000200 140417 031013 051007 141562 021001 $......2.R..R".
100 021377 131007 012602 140427 041007 031002 140464 031011 ".....R.2..42.
110 034026 035004 000700 041604 021001 000606 041004 000600 8.1...C."...R...
120 021200 041401 041402 031015 031004 000600 043004 004500 ".C.C.2.2...F..0
130 051403 021012 173006 170005 010201 140013 000000 000024 S.".....
140 021440 047506 020105 047124 051111 042523 020075 020040 # OF ENTRIES =
150 021017 020042 031014 051404 000600 021401 047004 021012 ". "2.S...#.N.".
160 041404 022417 004300 177006 170003 010201 140011 020040 C.%.....
170 020124 040502 046105 020123 044532 042440 036440 021020 TABLE SIZE = "
```

DPAN

This utility produces a formatted analysis of HP3000 system dump tape produced via the front panel of the machine. The dump itself must be acquired properly or the desired snap-shot of memory will be trash. But assuming the dump is correct, DPAN will analyze and list the following information:

1. ID page
2. Register pages
3. Low fixed core
4. System global area
5. DRT
6. CST
7. DST
8. PCB
9. Scheduling queue
10. Linked memory
11. Octal or hex main memory dump

Operation proceeds in the following steps:

1. Mount the SYSTEM DUMP TAPE.
2. Ready the MT driver and select the desired unit number.
3. Enter any /necessary FILE commands.
 - a) :FILE DPANMAST; DEV = TAPn
to assign the SYSTEM DUMP TAPE to tape unit #n.
 - b) :FILE DPANLIST; DEV = TAPn
to assign tape unit #n for all list output instead of line printer.
4. Set SWITCH REGISTER for desired options (see below).
5. Enter :RUN DPAN and wait.

The SWITCH REGISTER options are given below:

<u>BIT #</u>	<u>UP SIGNIFICANCE</u>
0	Suppress formatting of tables; octal only will be output.
1	Suppress all table analysis and printing; outputs: <ol style="list-style-type: none">a) ID PAGEb) REGISTER PAGEc) LOW FIXED COREd) Octal dump of all memory
2	Causes restart at end of current execution
3	Call DEBUG at all points present
4	Gets main memory dump in hex
5-9	Leave down
10	Terminate main memory dump immediately
11-15	Leave down

FINFO

There are times when during the course of debugging it is helpful to get file information whether it's about system files such as \$SDTIN or about files which perhaps FORTRAN, has opened for you. You may think it was opened variable; in reality it was opened fixed. To get this information is quite simple; just run FINFO and supply a qualified file name. An example of a qualified and unqualified file follows:

```
2 RUN FINFO
```

```
FILE NAME: FREE
```

```
+--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 FREE.HPUNSUP.SUPPORT ]
[ FOPTIONS: SYS,B,*FORMAL*,F,N,SL,DEQ ]
[ AOPTIONS: INPUT,SREC,NOLOCK,DEF,BUFFER ]
[ DEVICE TYPE: 0 LU: 2 DRT: 7 UNIT: 0 ]
[ RECORD SIZE: 128 BLOCK SIZE: 128 (WORDS) ]
[ EXTENT SIZE: 12 MAX EXTENTS: 1 ]
[ RECPTR: 0 RECLIMIT: 11 ]
[ LOGCOUNT: 0 PHYSCOUNT: 0 ]
[ EOF AT: 8 LABEL ADDR: %00200007350 ]
[ FILE CODE: 1029 ID IS FIELD ULABELS: 0 ]
[ PHYSICAL STATUS: 1111000000000000 ]
[ ERROR NUMBER: 0 TLOG: 0 ]
[ BLOCK NUMBER: 0 FACTOR: 1 ]
+-----+
```

```
FILE NAME:FORTRAN
```

```
+--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 TLOG: 0 ]
[ BLOCK NUMBER: 0 FACTOR: 0 ]
+-----+
```

```
FILE NAME:
```

```
END OF PROGRAM
```

Output defaults to \$STDLIST.

FREE

This program supplies information about the free space available on the system's disk. This could be valuable in assessing disk utilization. The program will prompt the user for the logical number of the disk. The following example shows two disks logical unit 1 and 2.

RUN FREE

DISC FREE SPACE LISTER

LOGICAL DEVICE #? 1
OF ENTRIES = 3 TABLE SIZE = 8 SECTORS
MIN = %34 MAX = %37777
SECTOR (%) LENGTH (%)
15532 3
16412 7
23014 14764

LOGICAL DEVICE #? 2
OF ENTRIES = 14 TABLE SIZE = 16 SECTORS
MIN = %44 MAX = %547277
SECTOR (%) LENGTH (%)
17335 10
32142 11
32164 5
37230 10
41521 16
43302 7
44022 25
55221 27
111737 5
112467 2
114552 7
115132 43
116354 52
122172 425106

LOGICAL DEVICE #?

END OF PROGRAM

The output file is LIST which defaults to \$STDLIST. A word of caution: Be sure to furnish the correct logical unit number. The results otherwise could be catastrophic.

LISTCRET

For all the millions of times, you've received a tape that could not be restored because the creator was unknown, there is LISTCRET. This utility works for STORE and SYSDUMP tapes. Output goes to \$STDLIST unless you set up a file equation setting FTN06 to the proper device. There is also a provision for multi-reel tapes. The example came from system console in session mode.

:RUN LISTCRET

IS THE TAPE A CONTINUATION REEL (Y/N) ?
N

?IO/16:49/#S1/24/LDEV# FOR FTN01 ON TAPE (NUM)=REPLY 24,7

STORE TAPE LIST PROGRAM*****

FILE NAME

CREATOR

INSTALL .HPUTIL	.SUPPORT	FIELD
INSTRUCT.HPUTIL	.SUPPORT	FIELD
J00J212A.HPUTIL	.SUPPORT	FIELD
M00M212A.HPUTIL	.SUPPORT	MGR
P00P212A.HPUTIL	.SUPPORT	FIELD
U00U212A.HPUTIL	.SUPPORT	MGR
END OF PROGRAM		

LISTDIRC

This little utility prints out directory information in either an expanded or security form. The user specifies what level he wants from file to user. The expanded print-out has the precaution of allowing system manager unlimited access to all entries, account manager unlimited access to all entries in his account and anyone else can access only his own entries, the results is as follows:

```
:RUN LISTDIRC

ABREVIATED SECURITY PRINT? (Y/N) => Y
FILE NAME? =>FREE
GROUP NAME? =>HPUNSUP
ACCT NAME? =>SUPPORT
FILE NAME: FREE.HPUNSUP.SUPPORT.

SECURITY:      READ: AC
(ACCOUNT)      APPEND: AC
                WRITE: AC
                LOCK: AC
                EXECUTE: AC
                SAV FILE: AC

SECURITY:      READ: GU
(GROUP)        APPEND: GU
                WRITE: GU
                LOCK: GU
                EXECUTE: GU
                SAV FILE: GU

SECURITY:      READ: ANY
(FILE)         APPEND: ANY
                WRITE: ANY
                LOCK: ANY
                EXECUTE: ANY
FILE SECURITY OFF
FILE ACCESS FOR FIELD.SUPPORT: READ,WRITE,APPEND,LOCK,EXECUTE

FILE NAME? =>
TERMINATE? (Y/N) => Y

END OF PROGRAM
```

Output goes to the file OP which defaults to \$STDLIST.

LISTEQS

This program is used to list all temporary files and file equations that are present in the current job session. This information is needed most urgently at around 2:30 a.m. when your code begins to blur and you can't remember if you set your output file to LP or CARD.

To use the program, type:

```
:RUN LISTEQ
```

The following example of output shows that this session had no temporary files and one file equation; i.e.,
:FILE OUT;DEV=LP.

END OF PROGRAM

Output goes to LIST which defaults to the line printer.

LISTLOG

With the logging features of MPE this utility provides a dump of the log file. The formal designator of the input or log file is LOGFILE. The formal designator for the output file is LOGLIST. A parameter is provided to suppress certain record types. By setting the bit for the appropriate type to 1, its output is suppressed. Record types and bit positions are:

Logging Error	0
System Up	1
Job Initiation	2
Job Termination	3
Process Termination	4
File Close	5
System Shutdown	6

An example is:

To list Job Initiation and Job Termination records on Tape

```
:FILE LOGFILE = LOG0034
:FILE LOGLIST; DEV = TATE
:RUN LISTLOG; PARM = %163
```

The following example produced much output. I enclose a sample for your enjoyment.

```
:FILE LOGLIST;DEV=LP
:FILE LOGFILE=LOG0077
:RUN LISTLOG.HPUNSUP.SUPPORT
```

END OF PROGRAM

TIME TYPE JOB#

DATE: FRI, JAN, 18, 1974

LOGFILE: LOG0077

1115 11 18	FILE SYS	LOG0077 .PUB	.SYS	DISP	DOM	SECTORS	DEV T/M	RECORDS	BLOCKS
				1	0	128	0 /2	0	0
1115 12 13	UP SYS	UP00 04	CORE	CST	PCB	100	TRL	ICS	JOBSSIMAX / RUNNING
			64	157	137	57	64	48	511 20 21
1115 13 19	JOB S1	USER FIELD	ACCOUNT	JOB	LOGON G	LDEV IN OUT	LINE LIM	CPU LIMIT	INP OUTP
			SUPPORT	MPUNSUP	23	23	0	0	8 8
1115 14 13	FILE S1	STDLIST	FILE NAME	DISP	DOM	SECTORS	DEV T/M	RECORDS	BLOCKS
				0	1	0	16 /23	5	5
1115 14 25	FILE S1	VAMP	MPUNSUP	SUPPORT	DISP	DOM	SECTORS	DEV T/M	RECORDS
				4	1	330	0 /2	0	0
1115 15 11	FILE S1	T	FILE NAME	DISP	DOM	SECTORS	DEV T/M	RECORDS	BLOCKS
				0	1	0	24 /8	45	45
1115 15 17	FILE S1	CANDIDAT	MPUNSUP	SUPPORT	DISP	DOM	SECTORS	DEV T/M	RECORDS
				0	0	18	0 /2	2	3
1115 15 19	FILE S1	ERROR	MPUNSUP	SUPPORT	DISP	DOM	SECTORS	DEV T/M	RECORDS
				0	0	8	0 /2	0	0
1115 15 31	FILE S1	GOOD	MPUNSUP	SUPPORT	DISP	DOM	SECTORS	DEV T/M	RECORDS
				0	0	14	0 /2	1	2
1115 15 32	FILE S1	SYSLIST	FILE NAME	DISP	DOM	SECTORS	DEV T/M	RECORDS	BLOCKS
				0	1	0	16 /23	2	2
1115 15 47	FILE S1	LOAD	FILE NAME	DISP	DOM	SECTORS	DEV T/M	RECORDS	BLOCKS
				0	1	201	0 /1	2	1
1115 15 46	PROC S1	PROG SEG	SL SEG	MAX STACK	MAX DS	VIRT ST			
		1	0	3840	16	62			
1115 15 46	FILE S1	SL	FILE NAME	DISP	DOM	SECTORS	DEV T/M	RECORDS	BLOCKS
				0	1	1792	0 /1	61	60
1115 15 46	FILE S1	EDITOR	FILE NAME	DISP	DOM	SECTORS	DEV T/M	RECORDS	BLOCKS
				0	1	233	0 /2	33	1
1115 15 47	FILE S1	EDITON	FILE NAME	DISP	DOM	SECTORS	DEV T/M	RECORDS	BLOCKS
				0	1	233	0 /2	2	1
1115 16 24	FILE S1	VAMP	MPUNSUP	SUPPORT	DISP	DOM	SECTORS	DEV T/M	RECORDS
				0	1	322	0 /2	960	320
1115 17 10	FILE S1	EDTLIST	FILE NAME	DISP	DOM	SECTORS	DEV T/M	RECORDS	BLOCKS
				0	0	0	32 /5	13	13
1115 17 30	FILE S1	EDTLIST	FILE NAME	DISP	DOM	SECTORS	DEV T/M	RECORDS	BLOCKS
				0	0	0	32 /5	18	18
1115 18 47	FILE S1	EDTLIST	FILE NAME	DISP	DOM	SECTORS	DEV T/M	RECORDS	BLOCKS
				0	0	0	32 /5	10	18

35

LISTLST

This utility gives the analyst loader segment table information. Depending upon which bit or combination of bit is set, the resulting printout is

- 1 - Loaded programs
- 2 - Directory
- 3 - Actual CST map

or any combination. The first part is important for finding out how many users are sharing the code for the loaded programs. The second part furnishes such information as the number of CSTs required for the loaded program. The third part is a list of all CSTs available to the system. The "A" means allocated; "C" means core resident; and "X" means system code. The code types are

SSL - System Segment Library
PSL - Public Segment Library
GSL - Group Segment Library
PROG - Program file

Output is on file FTNOG which defaults to \$STDLIST. The command is :RUN LISTLST;PARM= where PARMs bit patterns are as follows:

- (15:1) = 1 - Prints loaded programs
- (14:1) = 1 - Prints Directory
- (13-1) = 1 - Prints CST Table

The following example shows PARM = 7 which turns all three bits on.

:RUN LISTLST;PARM=7

LOADER SEGMENT TABLE DATE: THU, JAN 24, 1974, 4:02 PM

*****LOADED PROGRAMS

LOAD	.PUB	.SYS	, # USERS= 1
LISTLST	.HPUNSUP	.SUPPORT	, # USERS= 1
COBOL	.PUB	.SYS	, # USERS= 1

*****DIRECTORY

- 1) ENTRY TYPE: SL FILE
SEGMENTS COPIED TO SYSTEM DISC: NO
FILE NAME: SL .PUB .SYS
OF CST'S REFERENCED: 68
- 2) ENTRY TYPE: PROGRAM FILE
SEGMENTS COPIED TO SYSTEM DISC: NO
LOAD MODE: NORMAL
FILE NAME: LOAD .PUB .SYS
OF CST'S REFERENCED: 1

*****ACTUAL CST MAP

ACTUAL CST #	LOGICAL CST #	FLAGS	CODE TYPE	REF	SOURCE FILE	
020	001	X	SSL	3	SL	.PUB
021	002	X	SSL	3	SL	.PUB
022	017	X	SSL	3	SL	.PUB
023	023	X	SSL	3	SL	.PUB
024	025	X	SSL	3	SL	.PUB
025	031	X	SSL	3	SL	.PUB
026	037	X	SSL	3	SL	.PUB
027	041	X	SSL	3	SL	.PUB

.SYS
.SYS
.SYS
.SYS
.SYS
.SYS
.SYS
.SYS

SLLIST

In order to list out the system SL, this utility is provided. It simply lists all the segments and corresponding segment numbers. There are many applications for this utility, not the least of which is to check whether or not a segment made it into the SL after a shakey SYSDUMP. The command is all that is needed. Output goes to the designator OUT which defaults to the line-printer.

RUN SLLIST

SL FILE SL.PUB.SYS

1 FILESYS2	41 DATASEG	102 JOBTABLE
2 FILESYS1	42 S'LIB'01	103 RINS
3 TRACE1'	43 CHECKER	
4 TRACE0'	44 UTILITY	105 CLIB'01
5 COBLIB15	45 SEGUTIL	106 LOADER1
6 COBLIB14	46 MMDISK	107 PROCMAIL
7 COBLIB10	47 FILESYS7	110 SDMMCOMM
10 COBLIB09	50 MERGSEG2	
11 COBLIB05	51 DEBUG	
12 COBLIB03	52 SYSDEBUG	
	53 SYSDSPLY	
		117 CISUBS
17 MORGUE	60 FILESYS6A	120 CIORGMAN
	61 FILESYS6	121 CIINIT
	62 FILESYS5	122 CIFILEM
	63 FILESYS4	123 CIFILEB
23 ABORTRAP		124 CIERR
	65 IOPM	125 CILISTF
25 DISKSPC	66 TTYINT	126 CIMISC
26 COBLIB02	67 IOUTILITY	127 STORE
	70 CLIB'09	130 RESTORE
	71 CLIB'08	131 CXSTOREST
31 MESSAGE	72 CLIB'07	132 MMCORER
32 COBLIB01	73 CLIB'06	
	74 CLIB'05	134 CROUTINE
34 S'LIB'04	75 CLIB'04	
	76 CLIB'03	136 PCREATE
36 S'LIB'03	77 DIRC	137 PINT
37 FILESYS3	100 ALLOCATE	140 SORTSEG2
40 S'LIB'02	101 CLIB'02	141 FIRMWARESIM

END OF PROGRAM

TAPECOPY

One of the hardest working utilities is TAPECOPY. This program copies and verifies up to six copies of a STORE, Cold Load, or SPL readable tape. From a terminal a user may quickly copy a tape, verify that it is good and terminate or make multi-copies.

```
:RUN TAPECOPY
```

```
HP 3000 TAPE COPY AND VERIFY PROGRAM
```

```
VERSION 2.0
```

```
ENTER FORMAT(STORE/COLD LOAD/SINGLE FILE)
```

Entering the chosen format, the console will request a tape unit for the master and then a unit for the copy. A message then appears, if all is well (good tape, proper density, etc.), that says the copy is in progress. At the end the tape rewinds and the verification process begins. Upon completion, a message is issued whether the copy was good or bad; and if it was good, do you wish to copy another?

THE "BE SURE YOU KNOW WHAT YOU'RE DOING" ONES

DISKEDIT

This is another utility which allows you to modify the code on disk directly, according to absolute sector addresses. Parameters are free format.

Commands:

>LIST ldn

Specifies the device to which >DUMP listing is to go

Defaults:

no parameter - job list device

Initially: job list device.

>DISC ldn

Specifies the disc.

Initially: 1

>MODIFY sectornum, relwdaddr [,numwds]

- <sectornum> - the absolute sector address (decimal or octal)
- <relwdaddr> - the word address of the first word to be modified, relative to word 0 of <sectornum>. This may span sectors.
- <numwds> - the number of words to modify. This may span sectors. Default: 1.

The command returns with the sector number being modified (octal), and for each word to modified:

aaa: dddddd, %

<aaa> is the sector relative word displacement (octal)

<dddddd> is the contents of this word (octal)

The user then responds with

<newval> - new octal replacement, or

* - leave alone or

/ - abort MODIFY operation

The message, "WRITTEN" is printed when each sector is physically written. Before this, the sector has not been altered.

> EXIT

Just for a visual example the following provides a dialogue.
The prompt is given and the user must enter the desired
command.

RUN DISKEDIT

DISC EDIT/DUMP

<DISC 1

<MODIFY 5,24

SECTOR %000000000005

030: 041414,%*

WRITTEN

<EXIT

END OF PROGRAM

:

GIVEFILE

With MPE Version B, the user is not allowed to rename files across accounts boundaries. This utility gives the user the ability to move certain files from one account to another. Once the file has been moved, the entry disappears from the account giving up the file. The user must be running GIVEFILE from the account to which he wishesto bring the file. Otherwise, he will get:

DIRECINSERTFILE ERROR

If the file is not qualified properly or does not exist, the following appears:

FILE NOT FOUND

Certain files are protected as is FORTRAN in the following example. Note J00J222A in the SUPPORT account, HP32222 group, is being moved to group HPUNSUP and renamed to JOB.

:RUN GIVEFILE

OLD FILENAME? FORTMAN.MANAGER.SYS
FILE NOT FOUND
OLD FILENAME? FORTMAN.PUB.SYS
UNABLE TO OPEN FILE EXCLUSIVELY
OLD FILENAME? M00M102A.HP32102.SUPPORT
FILE NOT FOUND
OLD FILENAME? J00J222A.HP32222.SUPPORT
CREATOR = FIELD ? YES
NEW FILENAME? JOB
OLD FILENAME?

END OF PROGRAM

This program produces a snapshot of memory on specified second intervals. The matrix produced by the memory scan is recursive and when studied as a whole, it can show such things as

- Output defaults to \$STDLIST with the PRINT intrinsic. The following is an example of MEMMAP run with a 15 second interval which must be specified by the user.

INTERVAL IN SECS 15

[illegible]

```

LEGEND:-  S - SYSTEM CODE          #22      N - NON LINKED MEMORY
          U - USER CODE            #3       . - FREE MEMORY
          D - ANY DATA STACK       #7
          E - ANY EXTRA DATA SEG #1      64 WORD RESOLUTION

```

[illegible]

HP-3000 CORE MAPPER FRI, JUN 29, 1973, 8:46 AM

[illegible]

41

PATCH

For the analyst who thinks he has found the solution to a bug and wishes to try a patch to a lame program file, we have PATCH. This utility allows both displaying and modifying of the contents of program files on disk. The following is an example of a dump of the file FREE.

```
:RUN PATCH
  FILE=?FREE
?D,0,0,1
  024410
?D,0,0,5
  024410
  013302
  027410
  041000
  021013
?

  END OF PROGRAM
```

The prompt is "?". The user may then give one of the following directives,

D,segment number, address, number of locations.
M,segment number, address, number of locations.

inorder to dump or modify a code segment. If changes are to be made to the global area use:

M
D , D, address, number of locations.

As shown above, the contents of the specified addresses are displayed. If you are modifying, the change will be echoed to insure the correctness of the change.