DUMP Routine, Version 26C

17/4/80

Version 26C of the dump routine is now in service on the ERCC 2970. The object file is ENGINR.ENGPRGY_DUMP26CY; it is inserted in ENGINR.SS#DIR. The entry points are READDUMP and PRINTDUMP.

The changes in 26C with respect to the previous version (26) are as follows:

* Command: READDUMP(dumptape,file,out)

  file default is DUMPFILE
  out default is .LP

  Command: PRINTDUMP(file,out)

  out default is .LP

* The initial option-selection prompts have changed. If the response to "DUMP ANALYSIS:" is "Y" or "YES" then the following two extra options are given:

  PHOTO: "Y" or "YES" causes a dump of the photograph to be output, if available.

  AMT: "F" or "FULL" causes the AMT to be printed, including zero entries.
  "Y" or "YES" causes the AMT to be printed, excluding zero entries.
  Any other response suppresses the printing of the AMT.

  If the response to "DUMP ANALYSIS:" is "F" or "FULL" then this is taken to imply responses to "PHOTO:" and "AMT:" of "YES" and "FULL" respectively.

* The Oper output has changed, to reflect the new Oper software.

* Executing processes are printed out before the Kernel queue. They are still referred to ("EXECUTING") in the Kernel Q, Run Q1 and Run Q2 outputs.

* In the output of the AMT, the letters B, D and S are used to specify the position of an epage:

  B - both in store and on the drum. Store table index given.

  D - on the drum. Drum table index given.

  S - in store. Store table index given.

* READDUMP now takes account of segment boundaries when extending the dump file.

John M. Murison

Formatting and Labelling Routines in ENGINR

The Engineer process ENGINR has 2 utilities concerned with initialising discs. A new disc pack must first be formatted and then labelled before a file system can be established on it.

The procedure for formatting is as follows:

1) Log into ENGINR (password is ENGC).

2) Mount the new disc on a spare drive and wait for Operator Station to report:

```
        O/EDnn  LOADED  NO  LABEL
    [or O/EDnn  LOADED  xxxxxx  FRGN if from VME/B]
```

3) Type

```
        FORMAT (EDnn)
```

taking great care that the drive mnemonic is copied correctly.

4) The program prompts for lower and upper cylinder and track limits.

A lower cylinder of -1 indicates all cylinders.
A lower track of -1 indicates all tracks.

Formatting proceeds at about 25 cyls/min on a lightly loaded system.

Labelling must be done after formatting. The procedure is:

1) Log into ENGINR

2) Type DLABEL(EDnn)

3) Reply to prompt

```
    IPL OR NORMAL:
```

as appropriate. IPL labels are only needed for system (as opposed to user) discs.

4) Reply to prompt

```
    6 CHAR VOLID:
```

with the volume name.

5) After "Labelled OK" is output on the terminal check with Master Oper that the new label has been read successfully.

P.D. Stephens

EMAS 2900 UTILITY NOTE

No: 4
Date: 9/2/78

Transferring files from and to the 4-75

1. The utilities VTAPE and READVTAPE described below must be run in a privileged process on System 4 EMAS - usually 'UTILTY'. This restriction will be removed eventually and it will be possible to run them from your own process. The tapes written by VTAPE can be read on the 2970 using the VOLUMS TRANSFER command. The READVTAPE routine can be used to READ files from 2970 back-up tapes into the 4-75 file system.

2. VTAPE(tape[,start chapter])

   If 'start chapter' is omitted the file will be written starting at chapter 1. The 'start chapter' parameter can be used if it is necessary to add files to a tape which already contains some files. The command prompts for FILE: until the reply '.END' is given. The format of the response should be either

full filename	e.g. ERCC06.BASE11S
or full filename(new full filename)	e.g. ERCC06.BASE11S(ERCC27.BASES)

   The second form can be used to give the file a different name on tape, and hence on the 2970. Note that the filename can be a single member of a partitioned file, but in this case a 'new full filename' must be provided because an individual member cannot be created on the 2970; e.g.

   FILE:ERCC06.PD8_VTAPE04S(ERCC06.VTAPE)

3. READVTAPE(tape)

   The command prompts for chapters and can be terminated by replying '0'. Each chapter specified is read into the process being used and is given the same filename as it has on the tape, truncated to eight characters if necessary, and without the user name of course. After using this command the file should be OFFERed to its owner. This last requirement will be unnecessary when the utility can be run in the owner's process. The 2970 back-up tape must be mounted without a ring and should be returned promptly to the 2970 machine room. There is a line printer listing giving chapter numbers and file names, kept near the back-up tapes.

Roderick McLeod

EMAS 2900 UTILITY NOTE

No: 5
Date: 9/2/78

Input of Batch Archive Tapes to EMAS

1) In process 'UTILTY' on System 4 EMAS give the following command:

COMMAND: RUN(ERCS04.BATPIN2Y)

The program responds by giving the prompt

TAPE:

When the tape has been specified, the program outputs the name of the first file on the tape, then prompts

EMASFILENAME:

You should reply either:

filename

or REJECT (the program then goes on to the next file, if any)

or NO (the program exits to command level).

The output file is a sequential file containing EBCDIC OMF records. It is suitable as input to the OMF loader as it is.

---

2) If the file contains a microprogram, then to get a source file suitable for edit-insertion (as data in a constintegerarray) do the following:

INSERTFILE(ERCS04.OMFTOSY)

OMFTOSRC(infile, outfile)

The size of the array, in bytes and in decimal words, will be given. It is left to the user to put in the appropriate declaration.

---

3) If the file contains source, then to convert it into an EMAS source file do the following:

INSERTFILE(ERCS04.BAPPY)

BAPP(infile, outfile)

The output file is a normal EMAS source file.

A. Anderson

No: 5 (Re-issue)
Date: 21/11/78

Importation of Software from VME/B and VME/K

We can accept software from VME/B which has been written to a magnetic tape using procedure "BARCHIVE", and from VME/K via procedure "COPY OUT". There is no limit on the number of files per tape, but libraries must be written out as separate subfiles.

To obtain the facilities described below, the following should be specified:

```
OPTION (SEARCHDIR=ERCS04.OMFDIR)
```

---

1) To input files from a VME/B "BATCH ARCHIVE" tape, log into UTILTY and give the following command:

   Command: READBTAPE(tapename, filename, pdfilename)

   The parameters which the user specifies indicate which of three methods of removing files from the tape is to be used:

   a) To search the tape for a particular file.

      Command: READBTAPE(tapename, filename)

      If the filename specified is found on the tape then the prompt

      EMASFILENAME:

      will appear. Reply with the name you wish the file to have on EMAS 2900. The file will then be read in and control returned to command level.

   b) To inspect the tape file by file, transferring or skipping files as required.

      Command: READBTAPE(tapename)

      The program responds by printing the name of the first file on the tape, and prompting:

      EMASFILENAME:

      You should reply either:

```
    filename
or  R         (the program goes on to the next file, if any)
or  STOP      (the program returns to command level)
```

c) To input all the files from the tape into one or more partitioned files. This is to enable large numbers of files to be input simply.

Command: READBTAPE(tapename,,pdfilename)

The program will ask if the files are to be converted into EMAS 2900 source files as they are read in (answer Y or N). Then the program will ask for a maximum number of files to be read into the partitioned file. If this number is reached before the end of tape, then a new partitioned file can be set as destination. Thus a tape containing more files than could conveniently be kept in a single partitioned file can have its contents spread over any number of partitioned files.

On detection of a double tape mark, the message:

TAPE ENDS

is output and control returned to command level.

---

2) To input from "COPY OUT" VME/K tapes, log into process UTILTY and give the following command:

Command: READKTAPE(tapename)

and respond as to READBTAPE (method (b) above).

---

3) If the file contains a microprogram, then to get a source file suitable for edit-insertion (to be data for a constintegerarray) do the following:

Command: CMP(infile, outfile)

The size of the array, in bytes and in decimal words, will be given. It is left to the user to put in the appropriate declaration.

---

4) If the file contains source, then to convert it into an EMAS 2900 source file do the following:

Command: CS(infile, outfile)

If the file contains ISO use a third paramater, ISO, to avoid translation.

---

5) If the file contains an OMF program which has no unusual structure or run-time requirements, then it may be possible to convert it into an EMAS 2900 object file. Do the following:

    Command: COMF(infile, outfile)

Note: References beginning "ICL9CEZ" or "ICL9CM" are changed to "S#" and "M#" respectively.

If you wish to see an analysis of the OMF structure then first do the following:

    Command: OMFPARM(MAP)

---

6) If the file contains an OMF program which is unsuitable for conversion, it may still be possible to run it on EMAS 2900 by using the OMF loader. Note that conversion is preferable since the OMF loader is more limited and less efficient than the standard loader. See me for details.

---

7) If an OMF file contains unsuitable references, then they can be renamed by doing the following:

    Command: AMENDOMF(input, output)
    ACTION:

    Reply STOP in order to exit to command level
or
    RENAME/ref.name=new ref.name (with no spaces)

All occurrences of the reference will be traced and amended.

Please report any difficulties.

A. Anderson

EMAS 2900 UTILITY NOTE

No: 6
Date: 17/4/78

Magnetic Tape Utility Interface Routines

The routines described in this note provide a common interface with a series of routines available on 4-75 EMAS. They are designed primarily for "non-standard" tape utility programs, e.g. programs for writing or reading tapes for other operating systems. Several such programs already exist on the 4-75 and it should be possible to move them to EMAS 2900 with the minimum of alteration. The routines make no assumptions about the structure of information recorded on the tape.

externalroutine OPENMT (string (7) VOL)

Claims the tape with volume label VOL. If an asterisk is appended to the volume label then the tape is to be be mounted with a write permit ring; otherwise it is to be be mounted without a ring. Currently the tape must have a standard EMAS 2900 volume label (identical to 4-75 EMAS or IBM O.S. volume label). In due course VOLUMS will include a facility to allow the operator to give a temporary name to a tape with some other format of label. When the tape is claimed it is left at BT.

externalroutine UNLOADMT

Releases the tape and unloads it.

externalroutine READMT (integer AD, integername LEN, FLAG)

Reads the next block from the tape. On entry:

AD = address of an area into which to read
LEN= maximum length of read area

On exit:

FLAG = 0 successful
= 1 tape mark read
> 1 failure (details later)

If FLAG=0 then LEN = length of block read

externalroutine WRITEMT (integer AD, LEN, integername FLAG)

Writes to tape a block of length LEN from address AD. A non-zero flag on return indicates failure (details later).

externalroutine WRITETMMT (integername FLAG)

Writes one tape mark. FLAG non-zero on return indicates failure.

externalroutine SKIPMT (integer N)

Skips N blocks - forwards if N>0, otherwise backwards. For the purpose of this routine a tape mark is treated as if it were a block.

externalroutine SKIPTMMT (integer N)

Skips N tape marks forwards or backwards according to the sign of N.

externalroutine REWINDMT

Rewinds tape to BT.

Notes

These routines have been deliberately written with a simple functional interface. There is an obvious limitation (by the lack of a logical channel number) to their use for single tape jobs. The lack of failure flags for the majority of them is intentional in order to simplify their use.

Their specification has remained virtually unchanged over 8 years and has provided a convenient interface for a large number of programs for analysing, reading and writing non-standard magnetic tapes. Only the following changes are needed for programs being moved from 4-75 EMAS:

* it is essential to set LEN before calling READMT
* WREOFMT has been renamed WRITETMMT since this is consistent with the names of the other routines

Roderick McLeod

No: 8
Date: 21/7/78

Tape Copying Utility: COPYTAPE

This command is used to make a copy of a complete magnetic tape. The routine makes no assumptions about the contents of the tape except that two consecutive tape marks are taken as an indication of the end of useful information. The command takes no parameters.

It prompts

INTAPE:	reply with VOL label of input tape
OUT TAPE:	reply with VOL label of output tape.
MAX BLK LENGTH:	reply with length of longest block on tape.
COPY LABEL?	reply: Y. if you want the output tape to be given the same name as the input tape N. if you want the output tape to retain its current label.

The program prints the number of blocks (including tape marks) copied. Failure messages should be self explanatory.

Rodgrick McLeod

Utilities for Creating Supervisor, Director and Subsystem Files

Components of the operating system are object files which have been post-processed to reduce loading them to a trivial operation. This post-processing is carried out by one of the following:

SUPFIX for Supervisors

DIRFIX for Directors

SPFIX for System Process Basefiles, e.g. for VOLUMS and SPOOLR

SSFIX for the Edinburgh Subsystem Basefile and any others based on it, e.g. currently JOBBER

In some cases further processing is required - see the notes on individual commands, below.

Parameters

All four commands take two obligatory parameters and a third optional one. The first parameter should be the name of the object file containing the compiled version of the component. This will normally have been produced by linking a number of separate components using LINK. The second parameter is the name to be given to the output file. The third parameter can be used to specify an output file or device for diagnostics and a map of the "fixed" file.

Action of routines

The action common to all the routines is as follows:

* The input file is copied into the output file.

* The GLAP in the output file is modified according to information in the load data. Any external references are satisfied from within the file by filling in the PLT descriptor in the GLAP with the value it would need to have when in use.

* Relocation of values in the GLAP is carried out, again with respect to values of the start address of the relevant areas when the file is being used.

* Any unsatisfied references are filled with the ISO characters 'NORT' followed by the first four characters of the unsatisfied entry name. The effect is that if an attempt is made to call this routine the resulting dump will show the characters in DR and thus facilitate locating the fault.

* A map of the entry points is printed.

Individual Routines

The notes below refer to each of the variants of the FIX routine.

SUPFIX

The input should be a Supervisor object file with an entry point called "ENTER" at which it is to be entered after being loaded by CHOPSUPE. After being fixed, the header has the following format:

0 total length	1 start of code	2 physical size	3 type=10
4 reserved for sum check	5 date and time	6 start of GLAP	7 offset of PLT descriptor for "ENTER"

Note that the LOAD DATA is removed from the file and that the GLAP is aligned on a page boundary.

DIRFIX

The input should be an object file containing the Director. At the beginning there must be the current version of the primitive local routine DIRLDR. This must be the first routine in the link list; it is an assembler routine which is entered by the Supervisor when the Director is started and which copies the GLAP into the GLA file and then enters Director at routine "DIRECTOR". After the file has been fixed, as described above, the load data is removed and a call is made to routine BUILDSCT to add the System Call Table to the end of the fixed file. This routine is described elsewhere. The header of the final Director file is:

0 total length	1 start of code	2 physical size	3 type=11 (fixed Director)
4 reserved for sum check	5 date and time	6 start of GLAP	7 start of SC Table

Note that as a temporary arrangement word 3 contains a copy of word 7 (start of SC Table) and word 4 contains a PC relative jump of 8 halfwords.

SPFIX

This routine is used for fixing basefiles for System processes such as SPOOLR and VOLUMS. The input should be a linked object file with routine SSLDR as its first component. The initial entry to the basefile should be:

```
systemroutine SSINIT(integer MARK, ADIRINF)
```

The use of the two parameters will be described in a Director Note. After fixing, the output file has the following format:

0 total length	1 start of code	2 physical size	3 type=12
4 reserved for sum check	5 date and time	6 start of load data	7 start of object file map

As a temporary measure word 4 contains a PC relative jump of 8 halfwords to cause a jump to byte 32 of the file.

SSFIX

This routine,which is used for linking the Edinburgh Subsystem, is the same as SPFIX except that references to items in the System Call Table are satisfied at fix time. Every time a user logs in, a check is made to see that the version of the System Call Table currently operative is the same as that at the time the fix was done. If not the references are re-filled. After calling SSFIX it is necessary to process the Subsystem file further by calling the routine MAKEBASEFILE, described elsewhere.

R.R. McLeod

EMAS 2900 UTILITY NOTE

No: 10
Date: 3/10/78

Utility for Writing Character Files to IBM format Tapes

A first version of WRITEIBMTAPE is available in UTILTY.

Command:WRITEIBMTAPE	
EMASFILE:	must be a character file
TAPE:	
DSN:	for file on tape
LABEL:	
RECFM:	FB only at present
BLOCKSIZE:	
RECORDSIZE:	

Failure or output messages should be self-explanatory.

Modifications coming:

* More than one file at a time; note that more than one file can be written on a tape using this version repeatedly and incrementing LABEL.

* Possibly VB as another format.

Note that long lines are truncated, and an appropriate warning is printed; for IMP source this is probably more useful than wrapping round.

R.R. McLeod

No: 11
Date: 9/10/78

Backup/Archive Tape Maintenance Programs

CHECKTAPE(parms)

where parms = null (in which case the tape is prompted for)

or = tape (the name of the tape to be checked)

or = tape,list (the tape name and an output file or device)

CHECKTAPE checks that the specified tape can be read, and optionally lists the contents of the chapter header of each file on the tape to the file or output device specified. It also checks the validity of each chapter header.

The checking can be abandoned by pressing ESC ("INT:") STOP.

COPYARCH(parms)

where parms = read tape, write tape (tape to be read, tape to be written)

or = null (in which case the read tape and write tape are prompted for)

COPYARCH copies the specified Backup or Archive tape to a new Backup or Archive tape, only changing the tapename in each chapter header. Full error recovery is tried when a page fails to copy. If a chapter header fails to copy, a dummy chapter is created on the write tape to replace it. This keeps the chapter numbers in correspondence. If a page in a file fails to copy it is written with what ever was actually read. If there are any failures in writing the copy is abandoned.

The copying can be abandoned by pressing ESC ("INT:") STOP.

These procedures can be accessed by inserting the two files

ERCC20.TAPECOP2Y and ERCC20.MAG3Y.

W. Laing

EMAS 2900 UTILITY NOTE

No: 12 (revised)
Date: 9/2/81

File System Maintenance Utilities

This document describes the utilities for the maintenance and repair of EMAS 2900 file systems available in release 7 of MANAGR's file system maintenance programs. The source is held in MANAGR.MANPRGS_MAINT07 and the object in MANAGR.MANPRGY_MAINTY. A description of each utility is held in the source file, which has been made VIEWable.

Note that for readability some of the command names have embedded spaces in the descriptions below. These must not be typed when the command is called if OPTION NOBRACKETS is selected.

The following is a complete list of the utilities, with brief descriptions and page number references to full descriptions, which form the bulk of this document.

*** ARCHOFF ***

Operation: Prompts for <user>, <fsys>, <filename>.

Effect: Marks a file to prevent archiving.

*** ARCHON ***

Operation: Prompts for <user>, <fsys>, <filename>.

Effect: Marks a file for resumption of archiving.

*** BAD PAGES ***

Operation: Prompts for <Fsys (or -1)>

Effect: Generates a table giving details of the bad pages on <fsys>.

*** CCKOUT ***

Operation: Prompts for <fsys>, <output file or device>.

Effect: Copies the data recording progress of file system consistency checks, from the circular file maintained by the DIRECT process into a standard text file. The maximum filesize generated is 256K.

*** CHECK DAS ***

Operation: Prompts for <low page no> and <Fsys (or -1)>.

Effect: Calls 'TO DO ALL FILES' and for each file of each user on each fsys calls DGETDA and, if the disc address of a section is less than <low page no>, prints user.file .

*** COPY INDEX ***

Operation: Prompts for <user>, <fsys>, <new user>, <new fsys>, <new index size (Kbytes)>.

Effect: Creates a new index for <new user> on <new fsys>, unless one already exists. Index attributes (delivery, basefile, file limits etc.) are copied from the old to the new index. For each file in the original index, the corresponding file in the new index is destroyed (if it exists), a file of the same name and of the same size is created in the new index, and data are copied in from the original file. The file is given the same attributes as the original file.

Purpose: To run some important file system utilities (e.g. MOVE INDEX) following corruption on a file system, it is generally necessary to complete the file system consistency check first. In the case where corruption has occurred on the disc containing MANAGR (and hence the utilities also), this can be difficult. If COPY INDEX is used (say once per week, or after updates) to copy the contents of MANAGR's index to a second disc, then the utilities should be available to deal with the original disc, even if the original MANAGR index is inaccessible. In addition, the copied index forms an on-line backup of the original.

*** CREATE LOG FILE ***

Operation: Prompts for <user>, <fsys> and <logfile>.

Effect: Creates <logfile>, permits it to MANAGR, initialises <logfile> as a circular file and calls DSFI to set the LOGFILE field in <user>s index. The contents of the file are made available by LOGOUT (q.v.).

*** DELUSER ***

Operation: Prompts for <user> and <fsys>.

Effect: Deletes the file index for <user> on <fsys>, hence removing <user> from the System. This routine requires the user to confirm the username to be deleted before actually doing so. If <fsys> is given as -1 then all occurrences of the given username will be deleted automatically. If not then all other fsys's will be searched for other occurrences of the username. Any found will be reported and the user asked if they should also be deleted.

*** DEREGISTER CLASS ***

Operation: Prompts for a "base" username in which the last two characters are decimal digits (e.g. ERCC01), <fsys> and a number, N, of usernames to be deleted.

Effect: The "base" username and N-1 consecutive usernames (last two digits being incremented) are deleted from the System. (Equivalent to repeated calls of DELUSER, q.v.).

*** DIRLOG ***

Operation: No parameters

Effect: Copies the information from Director's monitoring file VOLUMS.#DIRLOG into a character file T#OUT which can then be further edited if desired.

*** DO ALL INDEXES ***

Skeleton routine, a copy of which can be used to surround code which is meant to be executed for all indexes on an fsys.

*** ENV ***

Operation: No parameters.

Effect: Prints the current Supervisor and Director versions, the process number of the calling user and the discs currently on-line. N.B. The first disc specified is the SLOAD disc.

*** FSYS START ***

Operation: Prompts for <fsys>.

Effect: Prints the range of pages available for files on <fsys>.

*** GET ***

Operation: Prompts for <what>, <user> and <fsys>

Effect: <what> is one of

ACR	ADDRTELE	AFILES
ARCHINDUSE	BASEFILE	BATCHSS
CODES	CONNECTT	CONTROLFILE
CUMINSTRS	CUMMSECS	CUMPTRNS
CURRPROCS	DELIVERY	DINSTRS
DIRMON	DIRVSN	FILES
FUNDS	GPFSYS	GPHOLDR
INDEXUSE	ISESSM	LASTLOGON
LOGFILE	MAXFILE	MAXKB
MAXPROCS	NKBIN	NKBOUT
PRIVILEGES	SESSINSTRS	SESSMSECS
SESSPTRNS	SIGMONLEVEL	SPECIALSS
STKKB	SURNAME	

Reports the value or values of the index attribute specified (some have multiple values associated).

For the significance of the index attribute, current Director documentation on the function DSFI should be consulted for all except PRIVILEGES. See SET for a full description of available PRIVILEGES.

*** HOLES HIST ***

Operation: Prompts for <fsys>.

Effect: Generates a histogram showing how many free areas there are of each size (1 to 32 Epages) on <fsys> and how many sections.

*** HOW FULL ***

Operation: Prompts for <fsys>.

Effect: Reports the current fullness (%) of the specified <fsys>. (The figure reported when the System is open to users is not strictly comparable with the corresponding figures given by the FCHECK process at System start-up, in that a considerable quantity of temporary file-space will be included in the figure for a running System).

*** HOW FRAG ***

Operation: Prompts for <fsys> (or -1).

Effect: Calculates the degree of fragmentation on specified <fsys>. The degree of fragmentation is defined to be the percentage of the total disc space available for user files which is occupied by sections which have sizes less than a full section (i.e. 1 to 31 Epages).

*** INACTIVE USERS ***

Operation: Prompts for <trigger date> in the format DD/MM/YY and <op file/dev>.

Effect: Produces a sorted table (oldest to youngest) of entries of the form date process last used, user, fsys, surname and delivery info of those users who have not accessed their process since the trigger date.

*** LIST NNT ***

This is a synonym for command USERNAMES (q.v.).

*** LOG OUT ***

Operation: Prompts for <user>, <fsys> and <file>.

Effect: Connects <user>s logfile, if it exists, and copies it to <file> (see 'CREATE LOG FILE').

*** LOST FILES ***

Operation: Prompts for <fsys>.

Effect: Lists the contents of VOLUMS.LOSTFILES on <fsys>.

*** MOVE INDEX ***

Operation: Prompts for <user>, <fsys>, <new fsys>, <new index size (Kbytes)>.

Effect: The program first checks <fsys> and <new fsys>. If these are different then the routine will operate in COPY mode, when copies are made of the files on <fsys> to <new fsys>. When <fsys> and <new fsys> are the same then the program can potentially operate in one of two modes: COPY or TRANSFER. In TRANSFER mode the files are not copied, ownership is merely transferred. This route is faster but less safe in that a crash occurring during file transfer will leave two partial indexes and possibly result in the loss of files. In practice the TRANSFER route is only obligatory if there is insufficient space on the disc to hold a copy of all the <user>'s files. The routine checks whether this is possible given the current state of the disc and if so offers <C/T?> to the caller otherwise the move can only proceed via the TRANSFER route.

After mode has been decided a new index of <new index size> named NEWZZZ is first created on <new fsys> and <user>'s index is renamed to OLDZZZ. The <user>'s index attributes are then copied to the new index, and all the files are transferred or copied from the old to the new index. If the mode was COPY then at this point a final chance is offered to the caller to abandon the move with no side effects. If the response is to continue or mode is TRANSFER then finally the OLDZZZ index is deleted and the NEWZZZ index is renamed to belong to <user>. Note that if the original index was corrupted, or if some files are marked as being in use (either because the owner's process had terminated in disorder or because the owner or other users are actually using the files), then the program reports the number of files it is unable to transfer and prompts "Continue?", before commencing the index move. If the reply is "N" or "NO" the move is abandoned, with no side-effects. If the reply is "Y" or "YES" those files which cannot be transferred are lost when the original index is deleted. In general it is preferable not to proceed with the operation if it is likely that one or more files are actually in use by a currently existing process. The file pages which are actually in use will not be re-used before the next IPL, but the process(es) using the file(s) will be unable to disconnect the files.

Purposes: One or more of the following:

1) To transfer as many files as possible from a corrupted index into a new index.
2) To move an index onto a different disc.
3) To change the size of an index.

Apart from possible software errors, indexes become corrupt mainly when a hardware error or machine stop occurs during updating of an index. Corruption is often first noticed during the file system consistency check at System start-up, when the message

<user> CORRUPT? FSYS <fsys> or
<user> CELLS? FSYS <fsys>

is given at the main OPER. In the former case the System remains closed to users until explicitly opened, preferably following a move (or re-creation) of the affected index(es). The latter message indicates that some list-cells are not attached to any list; correction is not normally urgent.

If the System is inadvertantly or otherwise opened and processes are started following a "CORRUPT?" message, the affected indexes should be destroyed and the users re-accreditted. Corruption of other users' files could otherwise occur.

*** MOVE INDEXES ***

Operation: Prompts for <from fsys> and <to fsys>.

Effect: Moves each user's index on <from fsys> to <to fsys> with the exception of SPOOLR and VOLUMS.

*** NEW USER ***

Operation: The program prompts for the following data:

Username
File system
Index size (Kbytes) (reply 4 or 8, for about 70 or 140 files respectively)
Initials and surname
Delivery information
Foreground password (4 chars)
Background password (4 chars)
Maximum total filespace (Kbytes)
Maximum single filesize (Kbytes)
Maximum process concurrencies allowed for the user, for interactive, batch and total numbers of processes.

Effect: A new user with the specified attributes is created.

*** REGISTER CLASS ***

Operation: Prompts for a "base" username (in which the last two characters are decimal digits), e.g. ERCC01, <fsys> and a number N of usernames to be accreditted. The program further prompts for:

File system
Index size (Kbytes) (reply 4 or 8, for about 70 or 140 files respectively)
Initials and Surname
Delivery information
Password (sets the 4 characters input as both foreground and background passwords)
Maximum total filespace (Kbytes)
Maximum single filesize (Kbytes)
Maximum process concurrencies allowed for the user, for interactive, batch and total numbers of processes.

Effect: The "base" username and N-1 consecutive usernames (last two digits being incremented) are accreditted to the System, all with the same process details as input initially. The program reports each username successfully accreditted. If any username cannot be accreditted (e.g. because the file system index area is full, or because the username already exists) the program terminates at that point.

*** REFRESH FILE ***

Operation: Prompts for <user>, <fsys> and <file>.

Effect: Forces a resiting of <file> owned by <user> on <fsys>.

*** REFRESH FILES ***

Operation: Prompts for <low page no> and <fsys>.

Effect: Will resite all files on <fsys> which have sections whose disc addresses are less than <low page no>. Used, for example, in reformatting a disc as a System disc.

*** REFRESH INDEX ***

Operation: Prompts for <user> and <fsys>.

Effect: Equivalent to MOVE INDEX with oldfsys = newfsys, old index size = new index size and MODE = TRANSFER. Resites an index on the same <fsys> with the same attributes. See MOVE INDEX.

N.B. Should not be done on the same <fsys> as the routine is running from.

*** REFRESH INDEXES ***

Operation: Prompts for <fsys>.

Effect: Does a REFRESH INDEX for each user on the <fsys> specified.

*** RENAME INDEX ***

Operation: Prompts for <user>, <fsys> and <newname>.

Effect: Calls 'DRENAME INDEX'.

*** SET ***

Operation: Prompts for <what>, <user>, <fsys> and <new value> (or <new values> as appropriate to the fields).

Effect: <what> is one of

ACR	ADDRTELE	AFILES
ARCHINDUSE	BASEFILE	BATCHSS
CODES	CONNECTT	CONTROLFILE
CUMINSTRS	CUMMSECS	CUMPTRNS
CURRPROCS	DELIVERY	DINSTRS
DIRMON	DIRVSN	FILES
FUNDS	GPFSYS	GPHOLDR
INDEXUSE	ISESSM	LASTLOGON
LOGFILE	MAXFILE	MAXKB
MAXPROCS	NKBIN	NKBOUT
PRIVILEGES	SESSINSTRS	SESSMSECS
SESSPTRNS	SIGMONLEVEL	SPECIALSS
STKKB	SURNAME	

Sets the new value or values of the index attribute specified. Some have multiple values associated and will prompt appropriately. For the significance of the index attribute, consult the current Director documentation for the function DSFI for all except PRIVILEGES, which are described below.

Note that in some cases it is not sensible and in other cases not permitted to SET certain attributes. See current DSFI documentation.

PRIVILEGES

It is possible to 'SET' PRIVILEGES or 'GET' PRIVILEGES (see 'GET', above). If the operation is GET then the individual PRIVILEGES enjoyed by <user> will either be listed in the form PRIVxx where 0 <= xx <= 31, or if none are enjoyed, given as *NONE*.

If the operation is SET then the current PRIVILEGES are first given as for GET, then the caller is prompted "PRIV:". This should be responded to by a reply of the form PRIVxx where 0 <= xx <= 31 (spaces are not significant). The caller is then prompted "G/R:"> and the reply determines whether the PRIVILEGE is to be given or removed. This sequence continues

until the reply .END is received to "PRIV:", when the new set of PRIVILEGES is reported. PRIVILEGES currently available are as follows:

PRIV 04	DPRINTSTRING, DDUMP
PRIV 07	DSFI 7
PRIV 08	DPERMISSION, DFINFO, DFSTATUS, DFILENAMES on other users' files
PRIV 09	DCHECKBPASS
PRIV 10	DSFI for privileged calls and other users
PRIV 12	Ability to reset BASEFILE, CONTROL FILE, TESTBASEFILE, BATCHBASEFILE
PRIV 15	Interactive use of magnetic tapes (DMAGCLAIM)
PRIV 17	DSFI 38
PRIV 18	DPON, DPON3, DOUT, DOUT11, DOUT18, DTOFF, DLOCK
PRIV 20	BADPAGE, DSYSAD, FBASE, GETAVFSYS
PRIV 22	ACREATE2, DMODARCH, DNEWARCHINDEX
PRIV 24	DCONNECT, DDISCONNECT on # files
PRIV 25	DPRG, DUNPRG, DTRANSFER, DOFFER
PRIV 26	DEMPTYI, DRENAMEINDEX, DNEWUSER, DDELUSER, VALIND, DDUMPI, DXDUMPI, GETUSNAMES
PRIV 31	Allows ADESTROY, DCHSIZE, DCREATE, DDESTROY, DNEWGEN, DRENAME to be used on someone else's file without full index permission.

*** SET H NOARCH ***

Operation: Prompts for <fsys> (or -1).

Effect: For each user on <fsys> sets the NOARCH bit in file descriptor of file #ARCH.

*** SET MSG EEP ***

Operation: Prompts for <fsys> (or -1).

Effect: For each user on <fsys> sets EEP to 11 in file descriptors of files #ARCH and #MSG.

*** SET SSBYTE ***

Operation: The program prompt is for <user>, <fsys>, <filename> and <values>.

Effect: For the specified file the "ssbyte" is set to (the rightmost 8 bits of) <value>. The "ssbyte" is a byte in the file descriptor reserved for exclusive use of subsystems.

*** SSDESTROY ***

Operation: The program prompts for <user>, <fsys> and then repeatedly for <filename> until a response ".END" or ".E" is given.

Effect: After each input <filename> an attempt is made to destroy <filename> belonging to <user>. A result code and brief text describe the success or failure of each "destroy".

*** SSFFILES ***

Operation: The program prompts for <user>, <fsys> and <output file or device>.

Effect: The result is exactly as for SSFILES (see below), except that extra details - largely as described under SSFINFO - are additionally printed for each file. "SSFFILES" stands for "SS Full FILES".

*** SSFILES ***

Operation: The program prompts for <user>, <fsys> and <output file or device>.

Effect: A list (in alphabetical order) of all the files belonging to <user> is placed in the <output file or device>.

*** SSFINFO ***

Operation: The program prompts for <user>, <fsys> and <filename>.

Effect: Prints out a concise synopsis of the file's attributes, namely:

"*" if "cherished"
"+" if marked for archive
<filename>
connect address in caller's virtual memory, or zero if not connected in same
physical size in epages
access permission to file owner (OWNP)
general access permission to all other users (EEP, "everyone else's permission")
access permission field (APF) from segment table, if file connected in caller's virtual memory, otherwise zero
current number of users of the file
list pool number to which descriptor cells belong in <user>'s file index
the user to whom the file is "on offer", if any
whether the file is marked:
PRIVacy VIOLated
TEMPorary
VTEMPorary
NOt to be ARCHived
as having more than one generation

The current Director documentation further explains this terminology.

*** SSFSTATUS ***

Operation: Prompts for <user>, <fsys>, <file>, <act>, and <value>.

Effect: Allows attributes of <file> belonging to <user> on <fsys> to be modified by calling the Director function DFSTATUS. N.B. This routine should NOT be used unless you are absolutely sure that you know what you're doing!!

*** SSIPERMIT ***

Operation: Prompts for <user>, <fsys>, <file), <to user> and <prm 1-15>.

Effect: Gives <to user> permission to access <file> owned by <user> on <fsys> as specified by the response to <prm 1-15>. If <file> is specified as .ALL then the appropriate whole index permission is granted. See also SSREMOVEPRM.

*** SSNKB ***

Operation: The program prompts for <user> and <fsys>.

Effect: The following data are printed from <user>'s file index:

Total file space (Kbytes)
Total temporary filespace (Kbytes)
Total "cherished" filespace (Kbytes) (currently reported equal to total filespace)
Maximum permanent filespace which the user is allowed to own (on-line)
Maximum single filesize allowed for <user>.

*** SSNOF ***

Operation: The program prompts for <user> and <fsys>.

Effect: The following data are printed from <user>'s file index:

Number of files
Number of temporary files
Number of cherished files
Number of file descriptors currently extant (this may exceed number of files, as some descriptors may be marked "dead", awaiting a garbage collect)
Maximum and currently free numbers of list cells in up to 4 list-cell pools.

*** SSPERMISSIONS ***

Operation: Prompts for <user>, <fsys>, <file>

Effect: Gives OWNP and EEP for <file> and whole index permissions. If <file> is specified as .ALL then whole index permissions only are given.

*** SSPERMIT ***

Operation: The program prompts for <user>, <fsys> and <filename>.

Effect: The specified file is given general access permission, i.e. execute, write and read.

*** SS REMOVE PRM ***

Operation: Prompts for <user>, <fsys>, <file> and <to user>.

Effect: Removes individual permissions to a file or index. Reverses the effects of SSIPERMIT.

*** SSRENAME ***

Operation: The program prompts for <user>, <fsys>, <filename> and <newname>.

Effect: The specified file is renamed to <newname>.

*** SSTRANSFER ***

Operation: The program prompts for <user>, <fsys>, <filename>, <newuser>, <newfsys> and <newname>.

Effect: <filename> belonging to <user> on <fsys> is transferred to ownership of <newuser> in <newfsys> and with name <newname>.

*** TEST BAD PAGES ***

Operation: Prompts for <fsys> (or -1)

Effect: Attempts to write to each page flagged as a bad page on <fsys>, by using the Bulk Mover to write successively a page of X'FFFFFFFF', a page of X'08CEF731' (the most difficult for the hardware) and finally the empty page pattern. If all of these are successful then the user is informed, the page removed from the bad page list and returned to the system. See BADPAGES.

*** USERNAMES ***

Operation: Prompts for <FSYS>, <SORT TYPE>, <USERNAMES> (if "all FSYS" option selected), and <FILE/DEV>.

Effect: A list of usernames accredited on <fsys> (if specified - otherwise all on-line file systems), together with leading file index attributes (file limits, process concurrency limits, file space etc.) is placed in <file or device>. If

the single FSYS option is selected then the sorting can be by index number or by username. If on the other hand all on-line file systems are selected the sorting may be alphabetic by username or surname or both. Optionally in this case a file called USERNAMES may be created or updated in the calling process. This file contains a directory of usernames sorted by surname.
Command LISTNNT is a synonym for command USERNAMES.

*** WHATFILE ***

Operation: The program prompts for <disc address> and <fsys>.

Effect: File system <fsys> is searched for the file(s), if any, to which epage number <disc address> belongs. If <fsys> is specified as -1 then all file systems are searched. The filenames, if any, are reported.

*** WHATFSYS ***

Operation: The program prompts for <username>.

Effect: Each on-line file system is searched for a file index belonging to <username>. Each <fsys> found to contain <username> is reported.

C. McCallum

No: 13
Date: 25/1/79

Selective Printing of a Hardware Dump Tape

The 2900 has a hardware facility to dump the first 256K bytes of store to magnetic tape as one block. (This should not be confused with the EMAS 2900 dump to tape facility.)

A program is available to read the block and dump selected parts of it to the line printer. Since the tape is unlabelled it is necessary to give it a pseudo-label as follows:

* Mount the tape on deck Mnn without a ring.

* When VOLUMS reports that the tape is mounted with no label, type:

```
V/Mnn=vol
```

  where vol is any 6-character volume label.

* Log in to ENGINR

```
Command: PRINTHWDUMP
TAPE: vol           as given above
CODE I/E?           ISO or EBCDIC interpretation
DUMP FROM BYTE:     reply first byte required
DUMP LENGTH:        reply number of bytes required
MORE:               reply Y to return to request a further area, or N to
                    terminate the run.
```

R.R. McLeod

Exporting Object or Source Files to VME/B and VME/K

EMAS 2900 object files for transfer to VME/B or VME/K must first be converted into ICL's object module format (OMF). They can then be written to tape in magnetic engineering format (MEF). This can be read at the target system by BPRIME or KPRIME.

To obtain the facilities described below, do the following:

```
Command: OPTION (SEARCHDIR=ERCS04.OMFDIR)
```

a) To convert an object file into OMF, ordinary users should do the following:

```
Command: COBJ (EMAS object file, OMF file)
```

System staff preparing ERCC products for release should use the similar procedure:

```
Command: OPUT (EMAS object file, OMF file)
```

Notes

* In each case the OMF file appears on EMAS 2900 as a sequential data file.
* A pd file member can be specified as input or output.
* Code entries beginning "ICL9CE" and system routines will be keyed.
* The module name given is the EMAS 2900 file name.

There are three small differences in the OMF module generated by calling OPUT from that generated by calling COBJ:

1) The "library" bit in the diagnostic record is set. This causes the module to be ignored by ICL's diagnostic trace-back.

2) "ENV " is set as the diagnostic "SUBNAME".

3) The module name is prefixed by "ICL9CE".

b) A procedure is available to control some aspects of the OMF generation. It should be called before execution of COBJ or OPUT, as follows:

```
Command: OMF PARM (control)
```

Some of the "control" settings are:

MAP produces a list comprising, the module name, entries and external references, together with the "IIN" assigned to each and their key strength.

MAXKEYS sets all code and data entries to have key strength. Note that this produces a module having the same "visibility" of names as on EMAS. It is not set by default because too many keys can be a nuisance, particularly on VME/K.

NOCASCADE means that when being loaded, the module will not attempt to satisfy references from the library list.

LIBPROC sets the "library" bit.

c) Writing files to tape.

Log into process UTILTY and do the following:

Command: WRITEBTAPE or

Command: WRITEKTAPE

The program will prompt for the tape name, and then accept an unlimited number of files, by prompting:

INPUTFILE:

To close the file, reply with "NO" or "STOP".

Source files are converted to EBCDIC variable blocked records by the program. If a pd file is specified then every member is written separately to tape, each identified by its member name only.

A. Anderson

EMAS 2900 UTILITY NOTE

No: 15
Date: 28/3/79

Magnetic Tape Utilities

The utilities described below have been transferred from 4-75 EMAS. They apply only to magnetic tapes which are 9-track, 1600 bpi, phase encoded, with odd parity and an inter-block gap of approximately 0.5-0.75 inch. Block sizes must be less than 12288 bytes.

To access these utilities:

Command: OPTION(SEARCHDIR=CONLIB.TAPEUTILS)

1. Tape analysis

a) Command: QUICKANAL(tapename, outdev) [alias SHORTANAL]

gives a block-by-block structural analysis of the tape.

tapename	will be prompted for if omitted
outdev	is .OUT by default

On reaching a double tape mark

Continue?
~~CONTINUE?~~

is prompted - reply Y or N.

b) Command: IBMTAPEANAL(tapename, outdev) [alias TAPEANAL]

assumes an IBM standard labelled tape, and gives a one-line analysis for each file on it.

~~tapename will be prompted for if omitted~~ stet
outdev is .OUT by default

2. Tape reading

a) Command: UNLABLDTAPE(tapename)

is used to read files from an unlabelled or non-IBM-standard labelled tape into an EMAS 2900 process.

~~tapename will be prompted for if omitted~~ stet

For each file to be read the program prompts as follows:

NEXTFILE NO:	absolute file number, or 0 to terminate the program
RECORD FORMAT:	F, FA, V, VA

RECORD LENGTH: or MAX BLOCKSIZE:	which of these appears is dependent on the choice of RECORD FORMAT (F/FA or V/VA respectively); reply with the number of bytes
CHARS/BINARY:	C or B; if C is given the program assumes the tape to be in EBCDIC and converts to ISO
EMAS FILENAME:	name of the file to be created
[EMAS RECFM:	F or V - only appears if RECORD FORMAT reply was 'F' or 'FA']
OFFER FILE TO:	recipient process, or '.' or ME if to be retained by you
START AT BLOCK:	usually 1, but may be otherwise if part of the file is not wanted or has been read previously

The resultant file is a data file, with structure

F(A) and record length given, or V(A) and record length 1024

containing either ISO characters or binary data. If the tape file would produce more than 1 Megabyte of EMAS 2900 file, then the program produces one file for each Megabyte or part thereof, named

filename, filenameA, filenameB, etc.

N.B. When V input format is specified, the whole magnetic tape block is assumed to be a single record - you must perform any unblocking needed.

b) Command: READIBMTAPE(tapename) [alias TAPETOEMAS]

is used to read files from an IBM standard labelled tape into an EMAS 2900 process.

For each file to be read the program prompts as follows:

DSN:	file name, or '.' if you do not know nor care; .END terminates the program
CHARS/BINARY:	C or B; C causes EBCDIC to ISO translation
EMAS FILENAME:	file to be created
OFFER TO?	recipient process, or '.' or ME if to be retained by you
SKIP TO BLOCK?	usually 1, but may be otherwise if part of the file is not wanted or has been read previously

The resultant file is a character or data file, according to the CHARS/BINARY reply. If it is a data file, its structure will be as acquired from the DCB information in the tape file header block. If the tape file contains more than 1 Megabyte of data the program produces one file for each Megabyte or part thereof, named

filename, filenameA, filenameB, etc.

3. Tape writing

a) Command: WRITEIBMTAPE(tapename) [alias STTOTAPE]

is used to write character files to an IBM standard labelled tape.

tapename will be prompted for if omitted

The program initially prompts as follows:

START AT LABEL:	specify first free IBM label number on the tape
DSNS=FILENAMES?	Y or N, depending on whether, for each file, the tape data set name is to be constructed from the EMAS 2900 name
CONSTANT RECLS?	Y or N, depending on whether LRECL is the same for every data set to be written
CONSTANT BLKSZ?	Y or N, depending on whether BLKSIZE is the same for every data set to be written

Then, for each file to be written to the tape, further prompts are as follows:

EMASFILE:	character file name; .END to terminate the program
[DSN:	only if DSN=FILENAMES? was N]
LRECL:	number of bytes - only ONCE if CONSTANT LRECLS?=Y
BLKSIZE:	number of bytes - only ONCE if CONSTANT BLKSZ?=Y

4. Tape dumping

Command:DUMPMT (tapename, outdev)

gives a character and hexadecimal printout of the contents of part of a magnetic tape.

outdev is .LP by default

The program prompts:

CODE I/E:	to select ISO or EBCDIC interpretation of the bytes on output.
SKIP:	to specify the number of blocks to be skipped before dumping.
BLOCKS:	to specify the number of blocks to be dumped.

Note that for the last two replies each tape mark counts as one block.

M.D. Brown
R.R. McLeod

Procedures in Files CONLIB.SERV1Y and CONLIB.SERV2Y

Notes

* Routines intended primarily to be console commands (having string parameters) are marked "C" in the list below, and are described as commands in the text. Other routines and functions are given formal IMP specifications in the text.

* The column headed File indicates whether the procedure is in file SERV1Y or SERV2Y.

* Routines marked '*' in the following list should be regarded as unsupported or subject to change.

* Queries should be directed to the originator, J.K. Yarwood, ERCC Room 2013, ext 2647.

CONTENTS (in alphabetical order)

QINFO	1	C	Brief "fileinfo"	1
RDFILEAD	1		Gives address of read-mode file	30
RDINT	1		Interactive integer read routine	21
RDINTS	1		Interactive integer real function	67
RECODELINES	2	C	De-assembles into 2900 machine code	24
REPLACE	2	C	Replaces pdfile member	54
RSTRG	1		Reads a line into a string	19
SEARCHF		*	Searches for text in pdfile	69
SEPARATE	1	*	Separates command parameters	22
TIM	1	C	Time-of-day to console	3
TPFILEAD	1		Gives address of temporary file	32
TSEARCH	2	C	Searches for text in pdfile	70
TSEARCHALL	2	C	All occurrences of text in pdfile	71
UDERRS	1		Prints a Director error message	23
UPDATE	2	C	Prepares job to update pdfile	61
VAL	1		Validates specified VM for access	72
WRFILEAD	1		Gives address of write-mode file	31
YCOMP	1	C	Binary comparison of two files	38
YSEARCH	1	C	Searches a file for integer or string pattern.	37

---

1) QINFO(file1, file2, ...)

A quick FILEINFO on one or more files - gives a one-liner describing each file.

2) AINFO(file1, file2, ...)

Gives own, general and individual access permissions for each file.

3) **TIM**

Gives time of day.

4) COMPARE(file1, file2)

Compares the two text files file1 and file2 on a line-by-line basis, stopping and printing out differing lines (if any). When a difference has been found, a prompt ':' is given, and the following commands may then be typed.

A	Advance one line in each file and restart the comparison.
Mfn	Move, in file f, n lines (f=1 or 2).
Q	Quit
E	End (same effect as Quit).
Pf	Print the current line of file f (f=1 or 2).
PB	Print the current lines of both files.
GO	Restarts the comparison of successive lines.
Fftext	Find in file f the text "text" (f=1 or 2).
FBtext	Find in both files the text "text".

The commands are required to be typed in a strict format and are carefully checked before being effected. An error leads to rejection, with the message NO.

The program terminates with the message EOF1, EOF2 (end of file1 or file2 reached) or COMPARISON COMPLETE (end of both files reached simultaneously).

5) MINIT
6) METR

Gives CPU time, page turns and number of SVCS since the previous call of METR. MINIT should be called once to start with for initialisation.

7) PIM(file, parm, parm, ...)

Calls the IMP compiler to compile the source file which is the first (or only) parameter. If the filename is FILE, the object and listing files are taken as FILEY, FILEL. (If the filename ends with 'S', the 'S' is dropped before the 'Y', 'L' are appended. The filename must not have 8 characters unless the last is 'S'. If the filename contains a 6-character username or an 8-character pdfile name, the object and listing filenames are generated from the 8-character file name or member name).

In addition, the following keywords may be appended:

.NY	object file to be .NULL
.N	listing file to be .NULL (also adds PARM (NOLIST))
.OUT	error lines to interactive terminal
.LP	listing file copied to .LP
.LPD	(".LP and Destroy") listing file sent to .LP
X	object filename to have X rather than Y (or Z) as last character
NEWGEN	causes object file to have X as a last character and provided compilation is successful, the object file is NEWGENned onto the existing 'Y' (or 'Z') object file.

NOLIST
OPT
NOCHECK
NOTRACE
NOARRAY
NODIAG
MAP
STACK
PARMX
PARMY
DEBUG
MAXDICT

} cause corresponding PARM to be set.

When the compilation is completed, the PARM for the process is reset to the parms which previously obtained.

9) **DUMPVM**

Produces a hexadecimal dump of an area of virtual memory. Prompts for start address or segment number; if a segment number is given, prompts additionally for a relative start address within the segment. Then prompts for a relative finish address and for output file or device.

11) externalstringfn FROMSTR(string (255) S, integer I, J)

Operates analogously to the IMP intrinsic string function FROMSTRING, except that conditions leading to the run-time fault STRING INSIDE OUT instead simply return a null result.

11) externalstringfn ITOS(integer I)

Returns a decimal digit string representation of the integer I (string commences "-" if I negative).

12) **NKB**

Prints the following file index data for the process:

maximum file size allowed
total file space limit
total Kbytes of file space
number of files
total Kbytes of temporary file space
number of temporary files.

13) **DETA**

No parameters. Prompts ':' and accepts input lines for a detach file until an input line comprises:

%C when it detaches the file and returns to command level.

integer when it detaches the file for execution with that time limit (minutes) and returns to command level.

Q when it just returns to command level.

14) **DUMPFILE(file, relstart, relfinish, filedev)**

Places a hexadecimal dump from address "relstart" to address "relfinish" in file "file" into file "filedev". "Filedev" may be a filename or .OUT or .LP. The dump has 32 bytes per line (16 for .OUT) with graphic representations for alphabetic and numeric characters on the right. Unless "filedev" is .OUT, the output file contains the name of the input file as a heading. "relstart" and "relfinish" may be given as decimal or hexadecimal (preceded be X) addresses relative to the start of the file. Only the first N of the four parameters need be given (0<= N<=4), as the program prompts for successive parameters which are not given (or which are given incorrectly).

15) DUMPCODE(file, relstart, relfinish, filedev)

Operates analogously to DUMPFILE (q.v.), placing de-compiled assembler from "file" into "filedev". Identical to NRCODE (q.v.).

16) EBCDICDUMP(file, relstart, relfinish, filedev)

Operates analogously to DUMPFILE above, except that graphic representations are for EBCDIC byte values rather than ISO.

17) EXFILE(file, text1, text2, filedev)

Extracts lines from "file" starting with line containing first occurrence of "text1" up to and including line containing first subsequent occurrence of "text2". Extracted file is sent to .LP. Fast , with minimum page turns. If "text1" is null, extract commences at start of file. If "text2" is null, extract ends at end of file. (Both "text1" and "text2" may not be null). Multiple consecutive underline characters in "text1" or "text2" are replaced by single space. If fewer than three parameters are supplied in the command file, the program prompts for its further requirements.

18) externalintegerfn BIN(string(255) S)

Result is the value represented by the string of up to eight decimal or hexadecimal (preceded by X) digits. Error result is X'80308030'.

19) externalroutine RSTRG(stringname S)

Reads characters from the currently selected input stream and sets S to be the string of characters obtained (excluding the newline). (Will not deliver a null string, and keeps reading until a non-null line is obtained).

20) externalintegerfn HXSTOBIN(string(255) S)

Result is the value represented by the string of up to eight hexadecimal digits (preceded by X). Error result is X'80308030'.

21) externalroutine RDINT(integername i)

Reads the next decimal or hexadecimal (preceded by X) number from the user's terminal. This routine is eqivalent to the standard IMP READ routine except that: (a) it will not read reals; (b) it accepts hexadecimal numbers; (c) it does not fail "symbol in data" for invalid input, but gives an error message for faulty input, abandoning remaining input on the same line and indicating the last valid number read (if any) which was supplied to the program, and then prompts for further input.

22) externalstringfn SEPARATE(stringname S)

Separates the string S into substrings comprising things between commas. At successive calls of the function the result and the string S are set to the 'next' substring. Result is null when there are no substrings left. (A null substring (i.e. ', , ') in the original also terminates the set of substrings).

23) externalstringfn UDERRS(integer I)

If I is an error number returned by a Director procedure, the function yields a text interpretation of the number.

24) RECODELINES(file, start line, end line, filedev)

De-assembles (2900 machine code) to file or device "filedev" from an IMP object program in "file" compiled with line-number updating between source line-numbers "start line" and "end line" (or line numbers reasonably close to same if code for stated line numbers is not found). Prompts if no parameters supplied, or parameters supplied incorrectly.

25) externalroutine DUMP(start, finish, printstartaddress, columns)

Prints a hexadecimal dump from virtual address "start" to virtual address "finish" on the currently selected output stream. The dump has either 16 (if "columns" <=72) or 32 bytes per line with graphic representations for alphbetic and numeric characters on the right. Multiple successive zero lines are printed once only followed by "ZEROES".

26) COPF(file1, file2)

Operates analogously with the Edinburgh Subsystem command COPY except that the file header words are not inspected and the physical size of the output file (file2) is the same as the physical size of the input file (file1). (Intended for non-standard files.)

30) externalintegerfn RDFILEAD(string(15) file)

Connects "file" in the "most suitable/possible" mode for reading. If the connect request cannot be satisfied, output stream zero is selected, the Subsystem routine CONNECT error flag is printed with an English interpretation, and a result of zero is returned. Otherwise the result is the virtual address of the start of the file (segment-aligned).

31) externalintegerfn WRFILEAD(string(15) file)

Connects "file" in write-mode, creating the file (1 page, but leaving a 4-segment VM gap) if it does not exist. If the connect request cannot be satisfied, output stream zero is selected, the Subsystem routine CONNECT error flag is printed with an English interpretation , and a result of zero is returned. Otherwise the result is the virtual address of the start of the file (segment-aligned).

32) externalintegerfn TPFILEAD(string(15) file, integer PAGES)

Operates analogously with WRFILEAD (q.v.) except that the file is created with the attribute TEMPORARY if it does not already exist (that is, the file will not survive a logout or a System crash).

33) externalintegerfn NWFILEAD(string(15) FILE, integer PAGES)

Operates analogously with WRFILEAD (q.v.) except that if file already exists it is connected in "newcopy" mode (previous file contents are not to be read).

35) PRHEX(integer I)

Prints "integer" in hexadecimal representation on the currently selected output stream.

36) PROCT(integer I)

Prints the octal representation of the RH half of "integer" on the currently selected output stream.

37) YSEARCH

Prompts for the name of a file in which a "binary" search for a byte pattern is to be made. Prompts RELSTART and RELFINISH (which may be given in decimal or hexadecimal (preceded by X) representation) for search limits relative to the start of the file. Then the program prompts STR/SHORT/INT: to determine from reply (STR, SHORT or INT) whether (respectively) a multi-character pattern, a halfword pattern (halfword aligned) or a fullword pattern (halfword aligned) is to be searched for. Then it prompts respectively for a non-null string of characters, a 16-bit integer or a 32-bit integer as appropriate (the integer may be typed in decimal or hexadecimal (preceded by X) representation). The search then commences. NOT FOUND is printed if the pattern is not found; otherwise an area around the first occurrence of the pattern is printed at the teletype.

38) YCOMP

Used to compare two files ("binary" compare). Prompts for the two filenames and a relative start position (which may be given in decimal or hexadecimal (preceded by X) representation). Compares the two files up to the end of the shortest file or up to the first difference in content. Prints the relative address at which the comparison terminates, with the contents of the words which are not equal if termination is as a result of difference between the files.

40) NOF

Prints the number of files in the process owner's index.

41) **AWAIT**

Suspends process without a prompt string on the terminal, thus allowing operator messages to be printed while the process is asleep.

42) **BEL**

Sends about 7 BEL characters to the interactive terminal.

49) **CONC(file1, file2, .../outfile[, .NP])**

Concatenates the text files file1, file2, ..., placing the resultant file in outfile (which may be the same name as one of the input files). If the keyword .NP is appended, a new page in the output file is placed at the end of each input file.

50) **DELI**

Prompts for up to 19 characters of delivery information (which may contain spaces), centralises it in a 19-character string and calls DELIVER.

52) **PDINSERT(pdfile1_member1, file2_member2, ...)**

Calls COPY(member1, pdfile1_member1),
COPY(member2, pdfile2_member2).

(The members must not already exist in the pdfiles.)

53) **EXTRACT(pdfile1_member1, pdfile2_member2, ...)**

Calls COPY(pdfile1_member1, member1),
COPY(pdfile2_member2, member2), ...

(Files called member1, member2, ... must not already exist.)

54) **REPLACE(pdfile1_member1, pdfile2_member2, ...)**

Calls COPY(member1, pdfile1_member1),
COPY(member2, pdfile2_member2), ...

(The members must already exist in the pdfiles.)

55) externalstringfn HTOS(integer I, PLACES)

The result is a string length of PLACES giving the (least significant digits of the) hexadecimal representation of the integer I.

57) **LI (file1, file2, ...)**

Calls LIST(file1, .LP),
LIST(file2, .LP), ...

59) **NRCODE(file, relstart, relfinish, filedev)**

Operates analogously to DUMPFILE (q.v.), placing de-compiled 2900 assembler from "file" into "filedev".

60) **PDCHECK(pdfile1, pdfile2, ...)**

For each pdfile a comparison is performed between members and ordinary files with the same name, and a summary is printed. At the same time a file SS#DESRP is created containing data for the command UPDATE (q.v.).

61) **UPDATE**

Data in file SS#DESRP is used to create a file SS#DETAC to be OBEYed or DETACHed to update the pdfiles in the most recent call of command PDCHECK (q.v.). The program gives groups of filenames: first those which would be destroyed, then those which would replace members of the pdfiles. For each group a corresponding number of Y's and N's must be typed (if the number does not match the number in the group the input is ignored). Where Y's are given, a DESTROY/REPLACE command is put in the SS#DETAC file. When all Y's/N's have been supplied the SS#DETAC file is listed, and the prompt "DETACH/OBEY:" is given. The replies are:

Q	No further action
NOW	SS#DETAC is detached for NOW execution
NOW, integer	As NOW, integer specifying "seconds"
integer	SS#DETAC is detached, integer specifying "seconds"
OBEY	The prompt .LP/.OUT is issued; .LP or .OUT is typed, to give OBEY(SS#DETAC, .LP or .OUT).

67) externalintegerfn RDINTS(string(63) S)

The result is the next integer which would have been delivered by routine RDINT (q.v.), except that if S contains valid integers separated by spaces or newlines, the next integer from S will be delivered.

69) externalintegerfn SEARCHF(params)

Not supported.

70) TSEARCH (text, file)

Searches for "text" in "file". Multiple underline characters in text are replaced by a single space. If a null parameter is given, the program prompts for text (which may include spaces) and for files. Lines surrounding the first occurrence of "text" in file are printed. If "file" is a pdfile, all character-type members are searched, except that if (for example) member FILE1 and FILE1L both exist, the FILE1 member is not searched.

71) TSEARCHALL(text, file)

Operates analogously with TSEARCH (q.v.), except that all occurrences of "text" in "file" are printed.

72) externalintegerfn VAL(integer ADR, LENGTH, RW, PSR)

Validates the virtual memory described by ADR, LENGTH for access according to RW (0=read, 1=write). If PSR is zero, the validation is at the ACR level of the caller; otherwise PSR should be the program status register value for the validation.
Result = 1 access allowed
= 2 access not allowed

J.K. Yarwood

Tape Utilities

The utilities described in this Note are for processing tapes written to recognisable standards. The tapes that the user can process with these utilities are user tapes, not tapes used by the System for its own private purposes (such as archiving). Users should note that although user (owned) tapes can be processed on EMAS 2900, they should consider the relative merits of the archive system compared with the use of private tapes for long term storage of data.

It is intended to provide the following utilities and detailed documentation for them. Only those marked with an asterisk are described in the current version of this Note.

TAPEANAL*	Summarise contents of a tape.
COPYTAPE	Copy one tape to another or copy selected files from several tapes to one output tape.
COPYTOTAPE* COPYFROMTAPE*	Copy files between user tapes and disc and vice-versa, including the handling of character files.
TAPECONVERT	Convert a tape written to an alien but reconisable standard to the System (IBM) standard.
DUMPTAPE	Dump out selected parts of a tape.

The latter two utilities will only be of help to those with a good knowledge of tape and file structures.

The utilities have been designed as a set, giving a reasonably consistent appearance, and can be used from a foreground or background (batch) process with reasonable ease. However, it should be noted that access to user magnetic tapes from a foreground process is not normally permitted; consult your local Advisory Service for details.

The parameters are specified in a parameter list to the call on the utility required, or in response to prompts, but not a mixture of the two. Positional and keyword parameter specification can be mixed, the use of a keyword parameter implying movement to the position of that keyword parameter for the purpose of identifying subsequent positional parameters.

Default values can be invoked when using a parameter list by omitting the relevant parameter, and at any point all remaining parameters can be defaulted by terminating the parameter list. If the user elects to be prompted for parameters (usually in foreground) he may specify that a default for a given parameter is to be taken by replying with the stop (".") character to the prompt for that parameter. If he wishes that all subsequent parameters (including the current one) should be defaulted, he should respond with the slash ("/") character; no more prompts will then be issued. Note, however, that some parameters do not have default values.

COPYFROMTAPE

This utility can be used for copying files from tape to the EMAS filestore. The tape must have been written to the accepted System standard, i.e. the 'IBM' standard. The utility takes the following parameters, which can be specified positionally (in the order below), by keyword or in response to prompts. Normally only the first four parameters will be needed.

Note that if the output file already exists, it will be overwritten by this utility.

Command:COPYFROMTAPE(name, label, type, rename, skip, process, maxrecl, size)

Parameter	Default	Description
NAME	none	The name of the file on the tape.
TAPE	none	The name of the tape volume.
LABEL	1	The position of the file on the tape.
TYPE	DATA	This parameter determines the format of the output file: it has four possible values. DATA file will be copied as data file. CHAR file will be copied as a character file. SEE NOTE (1) CHARX as CHAR except that newline characters will not be added at the end of each record processed. SEE NOTE (2) CHARA as CHARX except that if the input file is labelled as having ASA format control characters in its records, then those characters will be interpreted as the file is copied. (Not yet implemented.)
RENAME	tape file name	The name to be given to the disc file if its name is to be different from that of the file on tape. This must not be a member of a PD file.
SKIP	0	The number of records in the tape file to be skipped before processing commences.
PROCESS	*	The number of records to be processed. If all records from the starting point (i.e. after any SKIPped records) are required then specify '*'.
MAXRECL	10000	This specifies the maximum length of record in the input file. If this is greater than 10000 then this parameter must be used to specify the largest record in the input file.

Parameter	Default	Description
SIZE	255	This specifies (in Kbytes) the upper limit on the size of the output file when being created by COPYFROMTAPE. This is a temporary upper limit (see the Edinburgh Subsystem command DEFINE).

Examples

To copy the first file called SAMPLE1 from a tape AB4567 to a data file of the same name:

```
Command:COPYFROMTAPE(SAMPLE1,AB4567)
```

To copy the seventeenth file called STRANGE from tape KT5432 to a character file called FAMILIAR:

```
Command:COPYFROMTAPE(STRANGE,KT5432,17,CHAR,FAMILIAR)
```

The last example could be specified using a mixture of keyword and positional parameters, thus:

```
Command:COPYFROMTAPE(TAPE=KT5432,17,NAME=STRANGE,TYPE=CHAR,FAMILIAR)
```

Notes

1) When copying to a character file, all trailing spaces in each record are removed. Also, if TYPE=CHAR is specified, a newline character is added to the end of each record processed.

2) If TYPE=CHARX is specified, any records consisting entirely of space characters would be lost, as all the spaces would be removed and no newline character would be output.

COPYTOTAPE

This utility can be used to copy character or data files from the EMAS 2900 file system to a magnetic tape, which must already have a volume label. All the files will be written in accordance with the accepted System standard, i.e. the "IBM" standard. The utility takes the following parameters, which can be specified positionally (in the order below), by keyword or in response to prompts. Normally only the first four parameters will be needed; only the first two must be specified.

Command:COPYTOTAPE(name, tape, label, format, rename, skip, process, blocksize, option)

parameter	default	description
NAME	none	The name of the file in the EMAS file system. If a PD file member is specified, the RENAME parameter must be used to specify a valid IBM filename (see note 5).
TAPE	none	The name of the tape volume.
LABEL	1	The position of the file on the tape.
FORMAT	VB200	The required tape file format.
RENAME	input file name	The name to be given to the tape file if its name is required to be different from that of the input file.
SKIP	0	The number of records or "lines" to be skipped before processing commences.
PROCESS	*	The number of records or "lines" to be processed. If all records from the starting point are required then specify '*'.
BLOCKSIZE	4096	The length of blocks written to the tape. If this parameter is defaulted the System will use a blocksize as near to 4096 as is consistent with the specification of FORMAT (see above).
OPTION	STOP	For character files: specifies what action is to be taken in the event of a "line" in the input file being longer than the maximum record length of the output file. It has three possible values: STOP Processing is abandoned SPLIT The "line" is split across two or more output records (not yet implemented) TRUNC The "line" is truncated

Examples

1) To copy a character file called COMPRESS to the fifth file on tape AB1234:

```
Command:COPYTOTAPE(COMPRESS,AB1234,5)
```

2) If the above file were required to be copied in "card image" format with a blocksize of 2000 the call would be:

```
Command:COPYTOTAPE(COMPRESS,AB1234,5,FB80,,,,2000)
```

3) A keyword could be used to specify the blocksize in example (2) above:

```
Command:COPYTOTAPE(COMPRESS,AB1234,5,FB80,BLOCKSIZE=2000)
```

Notes

1) When a file is copied to label "n" on a magnetic tape, any file recorded at that label position and any files recorded at further label positions are DESTROYED.

2) Program source files being copied to tape for transfer to other installations should probably be in "card image" format, i.e. with FORMAT set to FB80.

3) When a character file is copied to a file on tape with fixed length records, all "lines" are padded with spaces.

4) Multiple newlines from a character file copied to a tape file with variable length records are recorded as single records consisting of a space.

5) To conform to the IBM file naming convention, names of files on private tapes must:

   (a) be no longer than 17 characters in total.

   (b) comprise one or more fields separated by periods, each field consisting of 1 to 8 alphanumeric (A->Z, 0->9) characters starting with a letter.

   e.g. FILEA
   SAMPLE1.GLASGOW
   G35A.STAT.LEVEL2

This utility is used to scan a tape and print out a summary of its contents, the thoroughness of the scanning being controlled by the LEVEL parameter. It should be noted that reading every block on a tape, especially when it holds a lot of data, is a relatively expensive process. Using the LEVEL parameter set to 1 avoids this problem. The other values of LEVEL are normally used for diagnostic purposes.

Command:TAPEANAL(tape, output, level, dumpout)

parameter	default	description
TAPE	none	The name of the tape to be analysed.
OUTPUT	.LP	Destination for the summary output from the analysis. This can be an output device or a file.
LEVEL	1	The type of analysis obtained. It takes three possible values, the first two applying only to tapes conforming to the accepted System standard: 1 A scan and extraction of information from the labels on the tape. This gives an efficient summary of the characteristics of all the files on the tape. 2 As 1 but including a scan of the files as well as the labels, thus checking for I/O errors and reporting the longest block in each file. This is less efficient than 1. 3 This option assumes nothing about the structure of the tape but simply gives a block by block summary of each physical file on the tape.
DUMPOUT	null	If a dump of the first 96 bytes of the first two blocks of each file is required then this parameter must be used to to specify the file for the dump output.

Examples

1) To obtain a simple summary of the contents of tape AB1234:

```
Command:TAPEANAL(AB1234)
```

2) To obtain a complete summary of the same tape, including a dump of the first 96 bytes of the first two blocks of each (labelled) file on the tape, and then to print the summary and dump together:

```
Command:TAPEANAL(AB1234, SUMRYFILE, 2, DUMPFILE)

Command:LIST(SUMRYFILE+DUMPFILE, .LP)
```

Notes

1) Although the headings on the output for a level 1 analysis are reasonably self explanatory, the following notes may help:

   Block Length: the maximum permitted size for blocks in the file.

   Label No.: the file sequence number that appears in the file label (this need not necessarily start at one for the first file on the tape).

2) For a level 2 analysis three extra fields are printed:

   Max Block: the length of the largest block actually found in the file.

   Byte Count: the total count of all bytes in the blocks of the file, excluding the file labels but including all 'red tape' for variable format files.

   Length: an estimate of the actual length of tape occupied by the file (including its labels) rounded up to the next foot. This assumes an inter-block gap of 0.6 inch.

3) The output from a level 3 analysis relates to the physical files on the tape. The output for each file consists of the physical file number followed by a number of fields, as follows:

```
BLOCKNO( COUNT * SIZE)
```

   where SIZE is the length (in bytes) of COUNT blocks starting at block BLOCKNO, and taking the first block on the tape as block one. E.g.

```
13(   9 * 1600)
```

   would mean that there are 9 blocks of 1600 bytes starting from and including block 13 on the tape.

4) Users who wish to restrict the output from this utility to 72 characters per line should prefix the value for the LEVEL parameter by the letter 'S'. This suppresses the output of some of the less important information. For example:

```
Command:TAPEANAL(AB1234,.OUT,S1)
```

B.R.P. Murdoch

No: 19
Date: 26/05/80

PROBE INDEX

Source: MANAGR.MANPRGS_PROBEINnnS

Object: MANAGR.MANPRGY_PROBEINDEXY

Originator: C. McCallum

Parameters: None

Operation: Prompts for <user>, <fsys> and <output to>.

Effect: Reads the file index for <user> from <fsys> specified and outputs an analysis of it to a file or device as specified.

A. Gibbons

EMAS 2900 UTILITY NOTE

No: 20
Date: 30/3/81

SHOWTAPE

This is an interactive tape examination program, driven by commands similar to ECCE SHOW. The position of each byte of data on the tape is defined by its byte, block, and file displacement from the start of the tape. Commands exist to move a notional cursor to any byte, and to output one or more bytes onwards from the current cursor position in octal, hexadecimal, character (optionally translated), or in a combined form of hexadecimal and character side by side.

Primitive commands consist of an operation, generally specifying a cursor-moving operation or an output operation, followed by a repetition count. For a cursor-moving operation the repetition count specifies the number of bytes, blocks or files by which the cursor is to be moved; for an output operation the repetition count specifies the number of bytes onward from the current position to be output.

The cursor-moving primitive commands are:

Un - move the cursor to byte 1, block 1 of the nth file Up the tape from the current file.(Up is towards the end of the tape.)

Dn - move the cursor to byte 1, block 1 of the nth file Down the tape from the current file.

Fn - move the cursor to byte 1 of the nth block Forward from the current block, within the current file. (Forward is towards the end of the file.)

Bn - move the cursor to byte 1 of the nth block Backwards from the current block, within the current file.

Rn - move the cursor to the nth byte Right from the current byte, within the current block. (Right is towards the end of the block.)

Ln - move the cursor to the nth byte Left from the current byte, within the current block.

The output primitives cause the output of n bytes of the current block starting with the current byte:

On - output in Octal.

Hn - output in Hexadecimal.

Pn - output as Printing characters.

Cn - output in Combined hexadecimal and character side by side.

Clearly such primitives can fail; for example if the specified cursor destination does not exist, or if there are insufficient bytes remaining in the current block to output.

If any output command or the cursor-moving command R fails, the end of block failure message

```
**EOB** AFTER BYTE n
```

is produced. (In the case of an output command the output up to the failure is produced.) If the cursor-moving command L fails, the start of block failure message is simply

```
**SOB**
```

since it must have failed at byte 1. Similarly, the F and B commands produce the file failure messages

```
**EOF** AFTER BLOCK n
```

and

```
**SOF**
```

respectively, and the U and D commands produce the tape failure messages

```
**EOT** AFTER FILE n
```

and

```
**SOT**
```

respectively.

The repetition count for a primitive command can be of two forms:

a) explicit - a positive integer, meaning do it that many times (an explicit repetition count of 1 need not be typed)

b) indefinite - 0 or ?, meaning do it until EOT, SOT, EOF, etc., and produce the appropriate failure message.

Two types of output have already been mentioned - that produced by the O, H, P and C commands, and failure messages. A third type of output is produced when an explicit U, D, F or B command succeeds.

Whenever an explicit U or D command succeeds, the file descriptor for the newly current file is output in the form

```
FILE n
```

Whenever an explicit F or B command succeeds, the block descriptor for the newly current block is output in the form

```
BLOCK n LENGTH=m BYTES
```

Multiple primitive commands can be input on the same line, forming a compound command whose elements are executed from left to right. Additionally, parentheses may be used to define compound commands which, exactly like primitive commands, are followed by an explicit or indefinite repetition count. (An unparenthesised compound command of one or more primitive commands behaves exactly as if it had outer parentheses and a repetition count of 1, and should be so considered in the description of termination and failure which follows.)

An explicit primitive command terminates when its repetition count is exhausted. It terminates and fails at EOT, SOT, EOF, etc. An indefinite primitive command terminates only at EOT, SOT, EOF, etc. It never fails (although it produces a failure message).

An explicit compound command terminates when its repetition count is exhausted (i.e. all its elements have been executed from left to right without failure n times). It terminates and fails when any element of it fails. An indefinite compound command terminates only when an element of it fails. The compound command itself never fails.
(N.B. in view of which, care should be taken to ensure that an indefinite compound command contains a subsequence whose repetition will ultimately give a failure, otherwise the program will loop indefinitely. In this context note that the output commands do not move the cursor.)

The primitive command \ can be used to cancel any failure in the immediately preceding primitive or compound command.

Some examples follow to clarify these points. At the start of a session the cursor is positioned at byte 1, block 1, file 1 and the file descriptor and first block descriptor are output. From this position the
command.....produces the output

U?.....EOT failure message only; i.e. the number if files.

(U1)?.....file descriptors for files 2 onwards to EOT.

F?.....EOF failure message for file 1; i.e. the number of blocks.

(F?U)?.....EOF for file 1, then file descriptors and EOF for all files to EOT.

((F1)?U)?.....block descriptors for blocks 2 onward to EOF for file 1, then file and block descriptors for all blocks in all files from 2 onward to EOT.

(H50\F1)?.....hex of first 50 bytes in block 1 then block descriptors and hex 50 for all blocks from 2 onward to EOF; note the use of the \ to cancel failure in the event of a block having fewer than 50 bytes, which would otherwise terminate the sequence.

(F1B1)?.....loops indefinitely without output.

(P10)?.....the first 10 characters of block 1 indefinitely.

(F?)?.....EOF for file 1 indefinitely.

Now to the question of the cursor position after failure. With three exceptions there is always a defined current file, block and byte. As mentioned above, at the start of a session the cursor is positioned at file 1, block 1, byte 1 unless

a) the first file contains no data blocks (i.e. a tape mark at the start of tape) when the empty file condition is raised. The file descriptor and the empty file failure message is output. Thereafter, the commands described so far except U and D produce an appropriate failure message and terminate. This condition can only be raised at file 1. Elsewhere it would correspond to two adjacent tape marks which signals EOT. The EOT condition is raised when a U command is executed and the current file is the last on the tape. Thereafter, the block and byte positions are undefined, commands F,B,L,R and the output commands give an appropriate failure message and terminate and U gives the EOT failure message and terminates. Only D of the commands described so far is effective,

or b) the first block of the first file is a bad block (i.e. one which cannot be successfully read from the tape because of parity errors), when the bad block condition is raised; this condition is also raised when a U,D,F or B command is executed and the newly current block is bad. A warning message is output in the form

```
** BAD BLOCK **
```

The block descriptor takes the form

```
BLOCK n   ** BAD BLOCK **
```

and the current byte becomes undefined. Thereafter the output and L and R commands give an appropriate failure message and terminate.

Apart from these three cases, the cursor position after failure is as follows:

command	file	block	byte
D	1	1	1
F	current	last	1
B	current	1	1
R	current	current	last
L	current	current	1
output	unchanged		

Output from the execution of each O, H, P or C command begins on a new line. Within one such command, newlines are inserted if required on the basis of the page width which by default is 132. This width can be reset at any time using the W command:

Wn - set the page width to n; n<12 or n=? produces a failure message and fails.

The output produced by the O and H commands is always an exact representation of the bit pattern on the tape. The character output produced by P and as part of C is a translated form of the bit pattern.

The translation is controlled by the current translation mode. If the mode=0 (the default) the tape bytes are assumed to be ISO characters and the bottom 7 bits of each byte are used, non-printing characters being translated to space and newline to ^. If the mode=1 the tape bytes are assumed to be EBCDIC characters and are translated to their ISO equivalents, non-printing to space and newline to ^. The translation mode can be changed at any time using the M command:

Mn - set the translation mode=n; n>1 or n=? produces a failure message and fails.

After some interactive movements within the files on a tape, it may be convenient to have the current cursor position output explicitly. This is done using the T(Tell) command:

Tn - output the current cursor position (current file descriptor, block descriptor and byte number); the repetition count is discarded.

The analysis is stopped and the tape released using the S command:

Sn - end the analysis, rewind and release the tape and stop.

Any multi-character console interrupt will cause the current command line to be abandonned, after which it would be advisable to use the T command to locate the cursor.

.....

The program currently resides in ERCC15.SHOWTAPEY on the 2980 and is run by

Command:SHOWTAPE(<tape id>, <in>, <out>)

The tape identifier is required. <out> is where the output is sent and defaults to .OUT. <in> is the source of the command input and defaults to .IN. However, since it can be a file, an arbitrary number of tailor-made analyses can be obtained simply by inserting the appropriate command string in a file (suitably pretested for indefinite looping, especially if the output is not to the console). As a final example, one likely candidate for inclusion in such a set of analyses is the command string

M1((C?F1)?U1)?S

which produces - the file descriptor, the block descriptor, a side by side hex/character dump of all the bytes in the block, for all the blocks, for all the files in an EBCDIC coded tape, the output being suitable for a page width of 132.

.....

Appendix A - Primitive commands

Un - move the cursor up n files.

Dn - move the cursor down n files.

Fn - move the cursor forward n blocks.

Bn - move the cursor backward n blocks.

Rn - move the cursor right n bytes.

Ln - move the cursor left n bytes.

On - output n bytes in octal.

Hn - ouput n bytes in hexadecimal.

Pn - output n bytes as printing characters.

Cn - ouput n bytes in combined hex/character side by side.

Wn - set the page width to n.

Mn - set the translation mode to n.

Tn - tell the current cursor position.

Sn - rewind, release the tape and stop,

\ - cancel failure in immediately preceding command.

.....

C.D. McArthur

EMAS 2900 UTILITY NOTE

No: 21
Date: 30/3/81

BECCE: Binary ECCE (Version 2)

BECCE is an editor available on EMAS at ERCC. It provides facilities for examining and modifying an Edinburgh Subsystem file of any type, presenting it to the user as a series of bytes without regard to any internal structure which it might have (e.g. object file, fixed format data file, partitioned file).

It is based on the text editor ECCE - in fact it is a version of ECCE, with some facilities removed and others added. The use of ECCE is described in detail in the EMAS 2900 User's Guide (2nd Edition), and the reader of this description of BECCE is assumed to be conversant with the facilities provided by ECCE and the way in which they are presented.

Summary of approach

ECCE operates in terms of a 'current line' of the file being edited, and of a 'current position' within the current line. Text specified (for various purposes) by the user is given as a string of characters. ECCE's output consists of the current line as a string of characters, with the current position indicated.

A "line" in the file is delimited by newline characters. This brings us to the first problem in applying the ECCE approach to "binary" files - files which are to be regarded simply as a series of bytes: there is no certainty that the newline character will appear with sufficient regularity (if at all) to enable the file to be split up into convenient 'lines'. The second problem is that the bytes in the file being edited need not correspond to printing characters, and some alternative means of representing them, for input and output, must therefore be provided.

The approach adopted in BECCE to solve the 'line' problem is as follows: on entering BECCE the user is asked to give a record length in bytes (default 20); he is then asked to give a 'separator' byte, expressed as a decimal value. If he does not give a value, BECCE repeatedly scans the file to be edited until a byte pattern, starting with 255 decimal and working downwards, is found which does not occur in the file. If one cannot be found, BECCE selects an infrequently occurring byte pattern and uses that. It also gives a warning that this has been done.

BECCE then copies the input file, inserting the separator byte at intervals corresponding to the specified record length. Any occurrence in the input file of the separator byte is represented in the copy by two such bytes. The editor proper is then entered with the separator byte doing duty as the 'newline' character. The file being edited is thus made up of a series of 'lines' of identical length (the specified record length), apart from occurrences of two separator bytes together (as explained above); these would appear as blank lines in the file being edited. As stated above, no account is taken of any internal structure which the file might have, and it is up to the user to ensure that any such structure is preserved.

At the end of the edit session, BECCE removes all single occurrences of the separator byte on copying the workfile to the output file, and (if no unique byte was found at the start of the editing) each occurrence of two separator

bytes together is replaced by a single separator byte. Thus if the user wishes to insert the separator byte at some point in the file he can do this by inserting a blank line (e.g. by use of R*B or L*B or BB) during the editing. Note that if the edited file is not required, it is quicker to exit from BECCE by means of 'INT:A' rather than '%C'.

The file header is not made available for editing. For data files, the "no of records" figure in the header is adjusted to the appropriate value at the end of the edit session, if the file format is F; if the format is V (or Unstructured) no such adjustment is made.

Byte formats

As noted above, the contents of a binary file cannot in general be represented by character strings. BECCE allows the user to indicate how his input strings are to be specified, and how the output 'lines' are to be printed. It does this by means of two new '%' commands:

1) >%I=<letter>

The %I command specifies the input format of strings in commands to BECCE. <letter> can be one of

H or h	for hexadecimal
D or d	for decimal
O or o	for octal
C or c	for character

With each letter there corresponds a 'minimum number of symbols' to represent any byte. These are as follows:

H	2	(e.g. A4)
D	3	(e.g. 194)
O	3	(e.g. 176)
C	1	(e.g. z)

Examples of input strings, with format H:

```
f/A31f24/ s/7B C 34/
```

There are no spaces in the 'f' string and so the 'minimum number of symbols' value, in this case 2, is used to split up the hexadecimal digits. Thus f/A31f24/ corresponds to f/A3 1F 24/. If space characters are present they delimit the specification of each byte. Note in the 's' string above that it is not necessary to specify two digits if one is sufficient, so long as spaces are used. Note also that f/A31F2/ would be interpreted as f/A3 1F 02/.

Further examples of input strings, this time with input format D:

```
f/140034128/ s/45 2 254/
i/P37/ v/64qe/
```

The first line above follows on from the hexadecimal examples. The second line includes characters which are not digits in the range 0-9. This is allowed: if a character is encountered which is not a digit of the specified input format then it is treated as a 'C' format byte specification in itself. Effectively one can use the 'C' input format whenever convenient. Note that f/1q/ is acceptable: it is treated as f/001 q/. These remarks apply to all the numerical input formats.

In the case of the 'O' and 'D' formats, the 'minimum number of symbols' is 3. However, if the value resulting from taking three adjacent digits is greater than 255 decimal, then the first two digits alone are assumed to define a byte. For example, /15151/ specifies bytes 151 and 51, but /51515/ specifies bytes 51, 51 and 5. To avoid confusion, it is better to use spaces or leading zeroes rather than exploit this rule.

In the case of the 'H' input format, the letters 'A' to 'F' (and 'a' to 'f') are treated as hexadecimal digits 'A' to 'F'.

It follows from the above that every possible input string has a meaning.

The default input format is H.

Finally, note that %L is the default, not %U. It is recommended that %U be not used, as the effect of case conversion can be confusing when a numerical input format is in use.

2) >%O=<letter>[<1-9>][C]

(The parts in square brackets are optional; the square brackets themselves are not typed)

This specifies the format of the output lines produced by BECCE. The letters permitted are the same as for %I. The optional digit following the letter specifies how many character positions are to be allotted. If this is not specified, the 'minimum no. of symbols' plus 1 is used; this results in each byte being output preceded by a single space. If a value less than the 'minimum number of symbols' is given then it is ignored and the default is taken.

The letter 'C' (or 'c') can be appended to the %O command. It causes each line output by BECCE to be repeated in 'C' format.

If the output format is C then bytes which do not correspond to printable characters are represented by '_'.

The default output format is H3.

Changing the size of the file

The use of some ECCE commands change the size of the file, e.g. e, k, i/text/. However, since there are applications where it is intended not to change the file size but merely to replace bytes by the same number of new bytes, a warning can be generated when the current position is moved off a line whose length is not a factor of the specified record length; the text "Line?" is printed.

To switch this warning on: >%?=1

To suppress it: >%?=0

ECCE facilities not in BECCE

The following ECCE facilities are not available in BECCE:

* Secondary input (but see the next section)
* Secondary output
* SHOW command

Marking and using text already in the file

BECCE enables the user to mark a string of bytes in the file for subsequent use with any of the text location and manipulation commands, viz. D, F, I, S, T, U, V.

The symbol '^' can be used in a command line. It causes the current position to be noted. Subject to certain conditions detailed below, a second use of '^' on the SAME file line causes a string of bytes to be defined, i.e. the bytes between the two positions noted.

If any of the commands D, F, I, S, T, U, V is subsequently used in a command line followed by '@' rather than the usual string specification, e.g. I@ instead of I/Fred/, then the string most recently defined via '^' is implied.

```
Example:   2A 73 2B^ 64 01 FF     (current line)
           >L ^ R* ^ B I@ M- P2   (input to ECCE - spaced out for clarity)
           2A 73 2B 64 01 FF
           2B 64 01 FF           (this line was created by B and I@)
```

Notes

* The file being edited must not be altered between the two uses of '^'. If it is, the earlier '^' is ignored.

* A pair of '^'s can define a null string - a subsequent use of @ would then always succeed. Initially the @ string is null.

* A string can be defined 'the wrong way round'; i.e. the first '^' can point to a position beyond the second '^' position.

* The only failure with '^' occurs when the defined string is longer than 100 bytes.

* The @ string can be redefined as often as desired.

Access

Access to BECCE is via directory CONLIB.GENERAL:

```
Command:OPTION(SEARCHDIR=CONLIB.GENERAL)
```

Thereafter BECCE is invoked with file specification identical to that of ECCE:

```
Command:BECCE(input,output)
   or Command:BECCE(input)
   or Command:BECCE(,output)
```

(Lack of) Support

BECCE is not supported by the ERCC Advisory Service. While I shall try to fix errors, I cannot undertake to do so as a matter of urgency. I am not willing to change BECCE, but am prepared to give a copy of the source to anyone who would like to.

John M. Murison