UNIVERSITY OF EDINBURGH
Edinburgh Regional Computing Centre

User Note 76

(April 1985)

Title:

EMAS 2900 File Analysis

Author: Susan Harrower

Contact: Advisory service

Software Support Category: See Note 15

Synopsis

This note summarizes routines which are available to users of EMAS 2900. Routines CHECKFILE, FILECHECK, FILESETUP and FILEUPDATE are concerned with identifying any of the user's files which have been corrupted since last used. Routine FILESUMMARY is a means of getting an individual analysis of a large number of files without having to call ANALYSE on each file separately. Routine FIT is a means of carrying out a specific instruction repetitively on a number of files.

Keywords

CHECKFILE, file analysis, FILECHECK, FILESETUP, FILESUMMARY, FILEUPDATE, FIT

Edinburgh Regional Computing Centre

James Clerk Maxwell Building, The King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ. Telephone 031-667 1081

Access

The routines FILESUMMARY, CHECKFILE, FILESETUP, FILECHECK, FILEUPDATE and FIT are accessed by the following command

Command: OPTION SEARCHDIR=CONLIB.GENERAL

FILESUMMARY

The format for this command is

Command: FILESUMMARY files,types,outputfile,maxlines

FILESUMMARY provides a summary of some or all of a user's files. To get information on somebody else's files he or she must have given you specific permission to do so, i.e. PERMIT .ALL,yourusername. The summary is similar to that given by the ANALYSE command - giving filename, cherished status, file type, length and date last altered. In addition, the first few lines of character files are printed (the number of lines is specified by the MAXLINES parameter), and a history is given for object files. For the impatient, issuing an *Int:* T will cause FILESUMMARY to also tell you which file it is summarizing at the time.

The files summarized are selected either by matching filenames against the FILES parameter, or by filetype specified by the TYPES parameter. The summary output destination is selected using the OUTPUTFILE parameter.

FILESUMMARY's parameters can be specified either positionally (as in the Command: line above) or by giving <parameter keyword>=<parameter value> in any order. Parameter keywords are those given in UPPER case in the following list. A full description of each parameter follows.

The parameter FILES allows you to specify the user on whose files the summary will be carried out, and/or a pattern against which to select a group of files. Either, or both, parts of the parameter may be omitted, but if a username is specified it must always be followed by a dot. The pattern part can be either a specific file name, or a pattern containing * symbol(s) indicating arbitrary numbers of characters. Examples of valid FILES strings are:

```
FRED          - your file FRED
DOC*          - all of your filenames starting with DOC
*NAL*         - all of your filenames containing NAL
*ST           - all of your filenames ending with ST
EMFI45.       - all files owned (and PERMITted) by user EMFI45
ERCT92.*PAS   - all filenames owned (and PERMITted) by ERCT92
                which end with PAS
*             - all of your files
```

If the FILES parameter is omitted FILESUMMARY will prompt for a FILES value. A null value (just hit RETURN) will be taken as *.

The parameter TYPES consists of a selection of one or more characters from a list of single character options. Valid option selection characters are:

```
C   - character files
D   - data files
N   - other file types, not covered by other option characters
O   - object files
P   - partitioned files
*   - all file types
```

Thus, the selection PCO would analyse all character, object and partitioned files. The analysis of any partitioned file would include an analysis of each of its members.

If the TYPES parameter is omitted FILESUMMARY will prompt for a TYPES value. A null value (just hit RETURN) will be taken as *.

The parameter OUTPUTFILE specifies the file or device to which all output from FILESUMMARY is directed.

If a filename is specified FILESUMMARY constructs it as a VIEWable file. You can then use the F command within VIEW to obtain formatted text output.

If a device is specified all VIEW directives are suppressed in the output.

Examples of valid values for OUTPUTFILE are:

```
QUUX     - output directed to file QUUX
.OUT     - output directed to your terminal (in VIEWable form)
.LP15    - output sent to line printer 15 (JCMB)
```

If the OUTPUTFILE parameter is omitted FILESUMMARY writes the output to the temporary file T£SUMMARY, which is VIEWed automatically.

For example

*Command:*filesummary
FILES:
*TYPES:*n

EKLD91.T£SUMMARY

---

*Matching with pattern **

1	*File:*	*	*SS£DIR*	*Type: Directory*	*Length: 4064 Bytes*
2	*File:*		*SS£JOURNAL*	*Type: Non-Standard*	*Length:-32 Bytes*
3	*File:*	*	*SS£OPT*	*Type: Option file*	*Length: 4064 Bytes*
4	*File:*		*T£ASTK*	*Type: Non-Standard*	*Length: 14680064 Bytes*
5	*File:*		*T£DOCINFO*	*Type: Non-Standard*	*Length: 0 Bytes*
6	*File:*		*T£LOAD*	*Type: Non-Standard*	*Length: 12256 Bytes*
7	*File:*		*T£USTK*	*Type: Non-Standard*	*Length: 204772 Bytes*
8	*File:*		*T£WRK*	*Type: Non-Standard*	*Length: 0 Bytes*
9	*File:*		*XOB*	*Type: Corrupt Object*	*Length: 0 Bytes*

End of section
View:

The parameter MAXLINES applies to files of type CHARACTER only. It specifies the number of lines to be printed at the start of the file. The MAXLINES value must be non-negative. Alternatively, if * is specified all lines in the file are printed.

If the MAXLINES parameter is omitted a value of 10 is assumed.

CHECKFILE

The format for this command is

Command: CHECKFILE filename

CHECKFILE carries out a series of checks on a given file to ascertain whether the file's internal structure is consistent. It does not check the data in the file.

For all filetypes, the file's header is checked for any corruption in filesize, data start address or data limit.

If the call of CHECKFILE is being carried out on a partitioned file, each of the file's members will be individually checked for the same things.

If the call of CHECKFILE is being carried out on a structured data file, then its records are all checked against this structure.

CHECKFILE will print warning messages if it finds any irregularities in the structure of the file. These messages are likely to be fairly obscure if you are not familiar with the details of file structure on EMAS 2900. The absence of any warnings can be taken to mean that the file is in good order; if warning messages do appear, your simplest course of action is to ask the Advisory service what is wrong! If you want more technical information, refer to User Note 35.

CHECKFILE will normally tell you

(a) the virtual address at which the file is connected
(b) the type of the file.

The appearance of these messages does NOT indicate any error condition. For partitioned files, you will also get similar messages for each member.

For example

*Command:*checkfile xob
XOB connected at 00DC0000
Corrupt object file

*Command:*checkfile scraps
SCRAPS connected at 014C0000
Partitioned file with 14 members
SCRAPS_A10C1 connected at 014C0020
Character file
SCRAPS_A10C40 connected at 014C06E0
Character file

SCRAPS_A10C80 connected at 014C0E60
Character file
SCRAPS_A1C1 connected at 014C15F0
Character file
SCRAPS_B70C1 connected at 014C1730
Character file
SCRAPS_B80C1 connected at 014C17A0
Character file
SCRAPS_B80C40 connected at 014C1F50
Character file
SCRAPS_B80C80 connected at 014C2790
Character file
SCRAPS_C140C1 connected at 014C3020
Character file
SCRAPS_C150C1 connected at 014C3090
Character file
SCRAPS_C150C40 connected at 014C3800
Character file
SCRAPS_D210C1 connected at 014C4020
Character file
SCRAPS_D220C1 connected at 014C4060
Character file
SCRAPS_C150C80 connected at 014C46A0
Character file

CHECKFILE leaves the file connected in virtual memory, so that it is accessible by the commands £SNAP, etc, described in User Note 35.

Routines FILESETUP, FILECHECK and FILEUPDATE

These three routines were designed to allow a user to check a number of files (current maximum 30) for possible corruption without having to check each file individually.

First the user must set up a "master file". This will contain records of all the files which are to be checked.

The format for this command is

Command: FILESETUP master file - if no file is given, the user will be prompted.

The user is then prompted for the names of up to 30 files. After each name is read in, the user will be asked if that file is expected to change regularly. This is because at a later stage, certain checks will only be performed for files which are not expected to change regularly. Typically, source and object code files would not be expected to change, but journals, logging files and so on would be likely to change between runs of FILECHECK.

To exit from this routine, type .END or .E in response to filename prompt - the masterfile will now contain a record for each file, consisting of its name, date last altered, a checksum of the data in the file and a flag to indicate whether or not the file will change regularly. The checksum is a number calculated by adding together all the words of data in the file. If any part of the data is altered, a

checksum calculated after the change will almost certainly be different from a checksum calculated before the change. Storing the checksum thus allows a simple check to detect any changes in the data, although the check cannot tell the user what part of the data was changed, or what the original data was.

In order to find out if any of the files whose records are held in the "master file" have been unexpectedly altered, the user should call the routine **FILECHECK**.

The format for this command

Command: FILECHECK masterfile - if no file is given, the user will be prompted

FILECHECK examines every record in the master file to ascertain the following facts:

(a) Does the file still exist?
(b) If the file doesn't exist, has it been archived since the date last altered that is stored in the master file record?

Additionally, if the file was not expected to alter regularly, the date last altered which is stored in the master file record is compared with the date last altered in the file's header. If these two dates do not match, a warning is printed, a call of CHECKFILE is made on the file (see above), a new checksum is calculated and put in the master file's record and the 'date last altered' is updated in the master file's record. If the dates do match, a checksum is calculated for the file and compared with the checksum held in the master file. If these checksums differ, a warning is printed and a call is made on CHECKFILE (see above).

For example

*Command:*filesetup
Master file : manfred
Filename : scraps
Alter often ? n
Filename : xob
Alter often ? n
Filename : diary
Alter often ? y
Filename : .end
There are 3 entries in your master file MANFRED

*Command:*filecheck
Master file : manfred
EKLD91.SCRAPS checksum unaltered
EKLD91.XOB checksum unaltered
EKLD91.DIARY

If the user wishes to alter the "master file" in any way, this is done by a call on **FILEUPDATE**

The format for this command

is *Command:* FILEUPDATE master file - if no file is given, the user is prompted

Valid internal commands for FILEUPDATE are

I (to insert record(s)),
R (to remove record(s)),
L (to list all records)
and
Q (to quit from
fileupdate)

If the user is inserting records, he will be asked if the file is likely to change often - this is for the same reason outlined above in FILESETUP. To exit from Insert or Remove, type .END or .E

For example

*Command:*FILEUPDATE
Master file : manfred
Valid operations : I = insert, R = remove, L = list, Q = quit
Operation : i
Filename : nlfor
Alter often : n
EKLD91.NLFOR has been inserted into the list
Filename : .end
Valid operations : I = insert, R = remove, L = list, Q = quit
Operation : q

*Command:*filecheck
Master file : manfred
EKLD91.SCRAPS checksum unaltered
EKLD91.XOB checksum unaltered
EKLD91.DIARY
EKLD91.NLFOR checksum unaltered

FIT

FIT is a routine designed to enable the user to carry out a command on a number of files quickly, without having to call the command on each file individually.

The idea of FIT is that the user should be able to say "for each file whose name has such-and-such a form, do command X". In order to get the full benefit of the command FIT, the user is required to supply any parameters of the command that he wishes to be carried out. The user puts a % symbol in the parameters wherever a file name is to be substituted. FIT then lists the commands, then asks whether you wish to obey them or not.

Thus, the command FIT %=*S;ANALYSE(%,E)

would perform ANALYSE filename,e on all the user's files ending with s.

Another example would be

Command: FIT %=*S;FORT77(%,*Y)

which means "for each file % whose name has the form *S (where * is any string) do FORT77 %,*Y, replacing % by the whole filename and * by the appropriate part of the name". For another example, consider

Command: NEWPDFILE SOURCES

Command: FIT %=MAP*;COPY(%,SOURCES_*)

to copy a set of files into a partitioned file. In this example file mapf22 would be copied into sources as member f22.

The forms of filenames which can be used are

*	any file
*ABC	any file whose name ends with ABC
DEF*	any file whose name begins with DEF
*PQR?	any file whose name includes PQR
ST*UV	any file whose name begins with ST and ends with UV

The user does not have to use the * symbol; he can choose any of *, @, &, ? provided he uses it consistently in any one call of FIT. In the case of *PQR?, he needs to use two different marks, because he may need * to mean "the part of the name before the PQR", and ? to mean "the part of the name after the PQR".

N.B. Do not use a * as a 'wild card' and include it within the operation parameters if * means something specific to the particular operation, e.g. FIT =*S;ANALYSE(,*) is ambiguous!

The user can also specify C%=*ABC and so on, to mean "all cherished files with names of that form", or H%=*ABC meaning "all non-cherished files ...", or A%=*ABC to mean "all archived files ...". Another alternative is to retain a file which would contain the names of all the files the user would wish to carry the command out on (one to a line). If the file was called (for example) II, then FIT [II]%=*ABC would mean "all files whose names were contained in file II and whose names were of that form".

Alternatively, the user can put %=MYPD_*ABC to mean "all members of MYPD whose names have that form".

In any of these cases, if the user wants to indicate "any filename" or "any membername", then he can leave the * out altogether, like this -

Command: FIT %=;ANALYSE(%)