UNIVERSITY OF EDINBURGH
Edinburgh Regional Computing Centre

User Note 77

(April 1985)

Title:

File Permissions on EMAS 2900

Author: Susan Harrower

Contact: Advisory service

Software Support Category: See Note 15

Synopsis

This User Note gives a description of the commands which are available to users of EMAS 2900 for setting and interrogating file permissions. It includes material from Chapter 5 of the EMAS 2900 User's Guide and from User Note 54 "Accessing Archived Files on EMAS 2900", as well as new material on the command LISTPERM.

Keywords

ANALYSE, APERMIT, file permissions, FILES, LISTPERM, PERMIT

Introduction

The commands described in this Note (with the exception of LISTPERM, which is accessible via CONLIB.GENERAL) are automatically accessible to all users. On-line information about each command may be found by using the Help command.

Setting Access Permissions on Files

There are two commands which allow the user to set permissions for his files.

PERMIT is used to set permissions for on-line files and
APERMIT is used to set permissions for archived files.

These are described individually below.

PERMIT

```
PERMIT FILE,user,mode
```

The first parameter is the name of the file to be PERMITted; the second is the name of the user who is to be allowed to use the file; the third indicates the modes of use which will be allowed to that user.

The 'mode' parameter consists of one or more letters. Each letter indicates a certain mode of use. Only certain letters are permitted. The letters may occur in any order, but no letter may occur more than once. R stands for 'read'; E stands for 'execute'; W stands for 'write'. Other acceptable letters are described below. For example:

```
PERMIT LUDU,ERCC14,ER
```

'All files' permissions

As well as giving other users access to a named file, it is also possible for the user to give others E and/or R access (but not W) to all his files. To do this, the user specifies the filename as .ALL, or omits it entirely. For example:

```
PERMIT .ALL,ZTBZ02,ER
PERMIT ,ZTBZ03,R
```

Note that access permissions given in this way apply to new files created after the PERMIT command is given, as well as to all files existing at the time of the command.

'All users' permissions

If the user wishes, he can allow all users to have access, either to an individual named file or to all his files. Instead of a user name, he must specify .ALL, or ??????, or omit it altogether. Note that this does not affect his own access permissions to the file(s). What he is doing is properly described as setting 'everyone else's permission', which is abbreviated to EEP in the output from some commands and

utilities. For example:

```
PERMIT .ALL,.ALL,ER
PERMIT ,??????,R
PERMIT HOLKAS,,WER
```

Permissions for user groups

The form ?????? meaning 'all users' is a special case of a 'group name'. The user can specify a group name to allow several users to access one (or all) of his files. A group name is a user name in which some of the characters are replaced by question marks, thus: ER??O?. This group would include ERCC08 and ERDB06, but not ECSC07 nor ERCC14. A group includes any user whose name matches those characters in the group name which are not question marks. For example:

```
PERMIT PUPTON,ER??O?,WR
PERMIT BAWLEY,GRY???,R
```

Own permission and P(reserve)

The user can also specify his own user name in a call of PERMIT (but only for a named file, not for 'all files'). This has special effects which are NOT achieved by specifying 'all users', nor even a user group which includes his own name. When the user first creates a file, he has full access permissions to it, so he would normally use PERMIT to restrict his access to it - typically, to prevent accidentally DESTROYing or overwriting the file. A typical command to prevent overwriting might be

```
PERMIT VITAL,myname,R
```

or, to protect against DESTROYing as well,

```
PERMIT AEONS,myname,PER
```

This demonstrates the use of P (meaning 'preserve' or 'protect') to inhibit DESTROY. The user can only specify P when giving himself permission to a file - it is not valid as a permission to other users (who cannot DESTROY his file anyway). P on its own is taken to mean PER.

Note that P will prevent the user from DESTROYing a file, but he will still be able to EDIT it if he has allowed himself W permission. COPY also needs W access to overwrite a 'protected' file, but NEWGEN can replace it with another file regardless of its other permissions. Some other commands, utilities and programs may refuse to handle a 'protected' file when they would otherwise destroy it or replace it with a new version. However, a file can still be lost using SEND file,device or OFFER file.

Full permissions

The fullest possible access permissions are WER for an individual file, and ER for 'all files'. The user may specify these permissions as A or .ALL if he prefers. For example:

```
PERMIT ARTIMON,LRHB21,A
PERMIT ,GR?C??,.ALL
```

Default permissions

If the last parameter in a PERMIT command is omitted, then ER permission is assumed, unless the user name is the user's own name, when WER (i.e., full permissions) is assumed. Thus, if a file is to be permitted to someone else, then PERMIT filename,username is nearly always sufficient. PERMIT filename,myname is a convenient way for the user to restore full access to one of his own files after he has restricted his own access - for example, when he actually wants to DESTROY the file.

```
PERMIT KRAEK,GRVA02
PERMIT ,ECSC??
```

If the user wishes to give all users ER access permissions to all his files, then the command is:

```
PERMIT .ALL
```

Multiple permissions

If the user calls the commands

```
PERMIT BONE,PTFE25,W
PERMIT BONE,PTFE25,R
```

then the file BONE is accessible to user PTFE25 with permission R, not WR as might be supposed. A PERMIT command supersedes all previous permissions granted for the same file and user (or group). The same thing is true for "all files" permissions.

However, that does not prevent a file from accumulating many different permissions for different users and groups. For example:

```
PERMIT .ALL,E?????,R
PERMIT TOKEN,GASP24,W
PERMIT TOKEN,GASP25,W
```

In this case, EJRM24 could read TOKEN (because the first PERMIT allows him to read any file), and GASP24 and GASP25 can both write into it.

Partitioned files

Access permissions can be set for the whole of a partitioned file, but not for its members individually. The members have the same permissions as the whole partitioned file.

Revoking permissions

Sometimes one needs to withdraw a permission that has already been granted. This is done by specifying the single letter C (for 'cancel') as the third parameter to PERMIT. The first two parameters must, of course, be the same as in the command which established the permission

(although the various synonyms of .ALL are not distinguished). Thus

```
PERMIT TGANSL,RAMC42,R
```

is not cancelled by

```
PERMIT .ALL,RAMC42,C
```

nor by

```
PERMIT TGANSL,RAMC??,C
```

although .ALL (all files) includes TGANSL and RAMC?? includes RAMC42. You would have to give

```
PERMIT TGANSL,RAMC42,C
```

but

```
PERMIT COGG,,ER
```

would be cancelled by

```
PERMIT COGG,??????,C
```

Incidentally, if a file is DESTROYed, there is no need to cancel the permissions for it. They are lost with the file, and will have to be set up afresh when a new file with the same name is created.

Note that the user cannot cancel his own permissions to his own files.

Restricting access to a file

One point to notice in this procedure is that finding an applicable permission prevents any further searching for other permissions. This has the apparently perverse consequence that granting an access permission can sometimes allow a user less access to a file. For example, suppose the user owns a file called SLUYS and issues the command

```
PERMIT .ALL,E?????,RW
```

Now EDBA12 has RW access to SLUYS (as well as to all the user's other files).

```
PERMIT SLUYS,.ALL,R
```

Now EDBA12 still has full access to most files, but only R access to SLUYS.

Conversely, removing a permission with PERMIT ...,...,C can have the effect of increasing a user's access to a file.

This feature can be exploited to ban certain users from using a file which is generally accessible to some group or groups. To do this, the single letter N ('no access') is used as the third parameter to PERMIT. For example

```
PERMIT CHERIMOYA,.ALL,R
PERMIT CHERIMOYA,ERCC04,N
```

This allows all users except ERCC04 to read the file. The trick depends on putting a user's name or group name into one of the 'permissions lists' with no associated permissions at all. This can be done for an

individual file or for all files, and for an individual user or for a group of users - but not for 'all users', nor for the user himself. When used for all users, it has the same effect as C.

Summary of parameters for PERMIT

	Set	Revoke	Debar
Specific file			
self	file,myname,perms#1	-	-
user	file,user,perms#2	file,user,C	file,user,N
group	file,group,perms#2	file,group,C	file,group,N
all users	file,,perms#2	file,,C	-
All files			
self	-	-	-
user	,user,perms#3	,user,C	,user,N
group	,group,perms#3	,group,C	,group,N
all users	,,perms#3	,,C	-
	Acceptable:	Default:	A means:
perms#1	P, W, E, R	WER	WER (P means PER)
perms#2	W, E, R	ER	WER
perms#3	E, R	ER	ER

APERMIT

APERMIT FILE,date,user,mode

When a file is archived, it retains the individual file permissions that it had when it was on-line. Its permissions can be changed by APERMIT whilst it is still on archive. When a file is restored from archive, it becomes available on-line with those permissions. A user can use the RESTORE command to get another user's file restored, provided that he has access permission to the file.

The APERMIT command may be used to extend or restrict access to one of the user's archived files for other users. It may also be used to find out the access permissions of one of his archived files.

The first parameter is the name of the file;
the second is the date the file was archived;
the third is the name of the user who is to be allowed to use the file;
the fourth indicates the modes of use which will be allowed to that user.

The date parameter can be found by a call of FILES filename,a. Alternatively if the date parameter is omitted, then the date the latest version of filename was archived is assumed.

'All files' permissions

Unlike on-line files, it is not possible for the user to give another user permission to all of his archived files. However, if the user has set a permission for 'all on-line files' (for example PERMIT .ALL,ERCC01,R) then that permission also applies to all his archived files.

'All users' permissions

The user can allow all users to have access to an individual file. However, he cannot permit all of his archived files to all users. Instead of a user name, the user must specify .ALL or ?????? or omit the user name altogether. For example:

```
APERMIT FRED,17/01/85,.ALL,R
APERMIT FRED,,??????,RE
APERMIT FRED,,,R
```

Permssions for user groups

The form ?????? meaning 'all users' is a special case of a 'group name'. The user can specify a group name to allow several users permission to a file. Again, the user cannot give permission to all of his archived files to a 'group name'. A group name is a user name in which some of the characters are replaced by question marks.
For example

```
APERMIT JOB1,06/06/84,ER??0?
APERMIT JOB1,,GRY???,RE
```

Own permission

The user cannot alter his own permission for his own archived files. He will always have sufficient permission to restore his own archived files, and he can change his own permission to a file when it is restored. There is no permission setting which will prevent the user from discarding any of his archived files.

Full permissions

The fullest possible access permission for an archived file is WER. The user may specify this permission as A or .ALL if he prefers.
For example

```
APERMIT MYJOB,01/01/85,ERCC01,A
APERMIT FILEAB,,ERC???,.ALL
```

Default permissions

If the last parameter of an APERMIT command is omitted, then ER permission is assumed. Thus if the user wishes to permit a file to someone else, then APERMIT filename,date,user is nearly always sufficient. For example

```
APERMIT run1,02/09/81,ercc01
APERMIT run1,,er????
```

Multiple permissions

If the user calls APERMIT FRED,,ERCC01,W followed by
APERMIT FRED,,ERCC01,R

then the file FRED is only accessible to ercc01 with permission R and not WR as might be supposed. This is because an APERMIT command supersedes all previous permissions granted for the same file and user. However, this does not prevent a file from accumulating many different permissions for many users and groups. For example

```
APERMIT FILE,,ERCC01,R
APERMIT FILE,,ERCC02,ER
```

In this case, ercc01 has read permission only, and ercc02 has read and write permission.

Revoking permissions

To withdraw the permission on a file, the user must call APERMIT with the letter 'C' (for cancel) as the fourth parameter. When making this call on APERMIT, the user parameter must match exactly the user parameter that was given when the file was permitted. Thus

```
APERMIT FILE1,,ERCC01,R
```

can only be cancelled by

```
APERMIT FILE1,,ERCC01,C    and not by APERMIT FILE1,,ERCC??,C
```

If a file is discarded, there is no need to cancel the permissions for it, as they will have been lost with the file.

Restricting permissions

In order to prevent a user, group of users or all users from accessing an archived file, the single letter 'N' (no access) is given as the fourth parameter of APERMIT For example:

```
APERMIT FRED,01/01/85,ERCC01,N
APERMIT PRIVATE,,ERC???,N
APERMIT PRIVATE,,.ALL,N
```

Summary of parameters for APERMIT

	Set	Revoke	Debar
self	-	-	-
user	file,date,user,perms	file,date,user,C	file,date,user,N
group	file,date,group,perms	file,date,group,C	file,date,group,N
all users	file,date,,perms	file,date,,C	-

Acceptable permissions are W, E, R
Default permission is ER
Permission 'A' means WER

Some guidelines for permitting files

What permissions are needed?

Most uses of a file will need R access to it. E access is necessary for files of object code.

For an archived file, the owner may always issue the RESTORE command regardless of access permissions. Any other user who has any access permission for the file may also restore it. When the file is restored, the permissions for the archived file will be copied for the on-line version. Subsequently the owner may change the permissions for the on-line version and the permissions for the archived version will NOT be changed (and vice versa).

W access allows a user to overwrite the contents of another user's file, but not to change its size, nor to DESTROY it. This means that most programs, commands and utilities (including those supplied by the system) cannot change another user's file even if W access permission has been granted. In particular, it is not possible to EDIT the file. In fact, W access is most often useful for Direct Access and Store Map files.

It is apparent by now that the matter of permissions is not simple. There are, as we have seen, several classes of permissions: for specific files and for all files; for the owner, for other named users, for groups of users and for all users.

How permissions are stored

You will observe that the various permissions are not all muddled up together. The system keeps the different kinds of permissions in separate places:

Permissions for individual files:	
Owner's permission	OWNP ('own permission')
Other named users	in a list
Groups of users	in the same list
All other users	EEP ('everyone else's permission')
Permissions for all files:	
Owner's permission	(no such thing!)
Other named users	in a list (not the same as the list for individual files)
Groups of users	in the same 'all files permissions' list
All other users	EEAFP ('everyone else's all files permission')

Each individual archived file has a set of permissions just like an on-line file. The "permissions for all files" apply equally to archived and on-line files; there are no separate "permissions for all archived files".

Can a user access a file?

The question arises of how the system decides what access permissions are allowed to a particular user. The rule goes like this:

1. If the user is the owner of the file, then OWNP applies and no other permissions are relevant (which explains why there are no 'own permissions' to 'all files').

2. If the user is not the owner, then the individual file's permission list is scanned for the user's name (groups are ignored at this stage). If his name is found, then the associated permission applies and no other permissions are relevant - even if the permission found is inadequate for what he wants to do. Note that this stops the search, so that a more generous permission in, for instance, the 'all files' permission list, will not be found.

3. If the user's name was not found at stage 2, the individual file's permission list is scanned again for a group name which includes the user's name. If any such group is found, then the associated permission is taken, and, as before, no other permissions are relevant, and all the same remarks apply. Note also that if there is more than one group name which includes the user's name (as, for instance, EPA??? and ?P?D2? both include EPAD25), then the outcome is unpredictable, since the permission for either group may be taken. It is up to you to avoid putting overlapping groups in a permissions list.

4. If a permission has still not been found, then EEP is examined. If it allows any access at all, then EEP applies, and the search stops as before; but if EEP does not allow any of read, write or execute, then the search continues with stage 5.

5. The 'all files' permissions list is searched for the user name, (ignoring group names, as at stage 2). If the user's name is found, then the associated permissions are taken and the search stops.

6. If a permission has still not been found, the 'all files' permissions list is searched again for a group which includes the user's name. If such a group is found, the associated permission is taken. All the remarks at stage 3 apply equally to this search.

7. If a permission has still not been found, then EEAFP is examined. If it allows any access at all, then EEAFP applies; otherwise the user is allowed no access.

8. For an archived file, if any permission (other than no access) has been found, the user may RESTORE the file.

Interrogating Access Permissions for Files

There are several commands for determining what permissions have been set for the user's files. Which command is used is determined by whether the user is interested in only one file, a group of files, all files, archived files, or even all on-line and archived files.

The commands to meet all these possibilities are described below.

Interrogating an individual on-line file

In order to discover the access permissions of a single on-line file, a call of ANALYSE can be made with the letter 'P' as the second parameter. For example:

*Command:*ANALYSE NEWFSUITES,P
*File: *NEWFSUITES Type: CHARACTER Length: 15796 Bytes*
Last altered: 16/04/85 at 11.47.28
Access Permissions: Self:All Others:None
Permissions for all files:
EKLD91 RE EKLF37 RE ???U?? None ERCQ19 RE ERCQ16 RE

Current users: 1

Note that in this example, whole index permissions are also returned.

Interrogating an individual archived file

If the single character "?" is given as the second, third or fourth parameter to APERMIT, then the command will produce a report of the existing general and individual permissions (if any) for the file and whole index permissions (if any), and it will NOT change any permissions (regardless of the values of any of the other parameters).

*Command:*APERMIT MYFILE,?
Access permissions: None

*Command:*APERMIT MYFILE,16/11/83,?
Access permissions:
EKLD91 R

Interrogation of process for whole index permissions

In order to ascertain which user(s) have whole index permission to his process the user can call routine FILES with the single letter 'P' as the second parameter. For example:

*Command:*FILES ,P

Disc files : 74 Temp files : 6 Archived files : 94
Total size : 1776K Temp total : 524K
Total limit : 4096K Maxfilesize : 10000K
Index size : 9K

Index space unused:
File Descs : 113 Sect Descs : 177 Perm Descs : 38
Archived File Descs : 87 Archived Perm Descs : 61

.ALL Access Permissions

EKLD91 RE EKLF37 RE ???U?? None ERCQ19 RE ERCQ16 RE

Multiple interrogation of on-line and/or archived files

LISTPERM is a routine which enables the user to check the permissions set on some or all of his files.

LISTPERM takes three parameters

```
LISTPERM type,group,file/device
```

where

Type is the parameter for setting the name(s) of the files. This can be in the form of a single filename, a mask (for example *S would examine all files ending with S), or null (or *) meaning all files.

Group is the parameter for setting the 'type' of file to be examined. Type 'A' refers to archived files only, type 'I' to online files only, and 'IA' refers to both. If group is '*' or null, then 'IA' is assumed.

file/device is the parameter for specifying the destination of the output. The default for this is .OUT (i.e. output is sent to the console). If the user nominates a file which already exist, he will be asked whether he wishes to overwrite the contents of that file.
For example:

Command: LISTPERM *S*,I

ONLINE FILES

File	*All Users*	*Specific Users*	*Self*
SNOOP	*RWE*	*none*	*RWE*
FILEUSERS	*no access*	*ERCQ?? RE*	*RWE*
NEWSETMODES	*no access*	*ERCQ08 RE*	*RWE*
USERNOTE	*no access*	*ERC??? RE*	*RWE*
NFSUMMARYO	*no access*	*ERCX19 RE*	*RWE*

5 on-line files are explicitly permitted to other users

These 2 permissions apply to all your on-line and archived files

???U?? no access
ERCQ?? RE

Command: LISTPERM ,A

ARCHIVED FILES

File	*Date*	*All Users*	*Specific Users*
KERN27S	*17/06/84*	*RE*	*none*
MYFILE	*01/02/84*	*no access*	*EKLD91 R*
CALIBAN	*16/11/83*	*no access*	*EKLD91 R*
CALIBAN	*19/06/83*	*no access*	*EKLD91 no access*
PROSPERO	*23/03/83*	*no access*	*EKLD?? RE*
KERN27S	*23/12/82*	*R*	*none*

6 archived files are explicitly permitted to other users

There are no "all files" permissions