DATE: August 27, 1981

TO: RD & E Personnel

FROM: Dan Swartzendruber, Barbara Kroczak

SUBJECT: New File Attributes

REFERENCE: None

KEYWORDS: None

ABSTRACT

Requirements have existed for some time for new file attributes. This document is a suggestion of how additional file attributes may be added to existing file entries.

Rev 1

The previous design has been replaced by one which provides more flexibility for the future and which will result in less movement of file entries.

Owner fields have also been ommitted from this new design.

1 Introduction

Requirements have existed for some time for new file attributes. The following are a few reasons which come to mind:

a) ROAM files should be marked as being part of a larger structure so that utilities such as FUTIL can prevent a user from deleting or copying part of the structure.

b) date/time last accessed for read or write is required for an archiving system to be implemented.

c) a flag is required for online/offline files to be implemented.

d) record/stream information, access method information will be required for intelligent handling of stream I/O.

e) a text/binary flag would help to prevent characters being deleted by accidentally running a text editor over a file.

2 Existing structure

2.1 Existing file entry

word		Notes
0	ECW type/length	(1)
1,2	BRA of file	
3-5	Reserved	
6	protection keys, delete bit	(2)
7	ACL position (offset)	
8,9	date/time last modified	
10	file type	(3)
11	SCW type/length	(4)
12 - n	file name (max 16 words)	

Notes

(1) word 0 bits 1-8 type =
- 0 old ufd header
- 1 new ufd header
- 2 vacant entry
- 3 new ufd file entry
- 5 ACL entry (see section 2.2)

bits 9-16 length=length of subentry (including ECW)

(2) 6 bits
- 1-5 unused
- 6 delete owner rights
- 7 write owner rights
- 8 read owner rights
- 9 delete flag
- 10-13 unused
- 14 delete non-owner rights
- 15 write non-owner rights
- 16 read non-owner rights

(3) 10 file type bit
- 1 set: long RAT
- 2 dumped bit
- 3 file modified under DOS
- 4 special file
- 5,6 file read/write lock
- 7,8 spare
- 9-16
  - 0 SAM file
  - 1 DAM file
  - 2 SAM seg dir
  - 3 DAM seg dir
  - 4 ufd

(4) 11 type = 0 file name subentry

length = length of subentry (including SCW)

If the name subentry were to occupy its maximum allowed length, then

the maximum length of a non-ACL type file entry would be 28 words.

2.2 Existing ACL entry

word		Notes
0	ECW type/length	(1)
1	version #	
2,3	date/time last modified	
4	type of object protected	(2)
5	spare	
6	SCW	
7 - n	ACL name	
n + 1	SCW	(3)
n+2 - m	ACL data	(4)

Notes

	word		
(1)	0	ECW	type = 5 for ACL
(2)	4	ACL	type type=6 file 7 ufd
(3)	n+1	SCW	type = 2 ACL data subentry = -1 vacant subentry (may result if part of ACL data is deleted).
(4)	n+2-m	ACL	data subentry

word		
0	SCW	type=2 , length
1	data	bits 1 non-FS flag
		2-8 MBZ
		9-11 unused
		12 P access for dirs ring1 unused for files ring1
		13 D access for dirs ring1 X access for files ring1
		14 A access for dirs ring1 unused for files ring1
		15 U access for dirs ring1 W access for files ring1
		16 L access for dirs ring1 R access for files ring1
2		bits 1-8 as bits 9-16 of word 1 but for ring 2
		9-16 as bits 9-16 of word 1 but for ring 3
3		length in characters of following access name or group
4 - n		name or access group

If the non-FS flag (bit 1 of word 1 of subentry) is off, the subentry ends here.

If the non-FS flag is on, there follows information for access control in extra modes (not yet defined)

word	
n+1	bits 1-4 no of words per mode (eml) in following entry bits 5-8 no of mode entries for ring1 9-12 no of mode entries for ring2 13-16 no of mode entries for ring3
n+2 - m	mode entries (each a constant length as stated by eml) no of mode entries per ring defined in bits 5-16 of previous word.

2.3 Existing ufd header

word		
0	ECW type/length	type=0 old ufd header =1 new ufd header
1-3	owner password	
4-6	non-owner password	
7	spare	
8-9	quota max	
10-11	no of records used in directory	
12-13	no of records used in tree	
14-15	record/time product	
16-17	Julian age of file	
18-19	date/time quota last updated	
20	spare	
21	position of default directory ACL	
22	position of default file ACL	

2.4 Existing data returned by RDEN$$

Calling sequence to RDEN$$:

```
CALL RDEN$$ ( key,funit,buffer,buflen,rnw,filnam,namlen,code)
```

where key =

K$READ	read next entry
+ K$GACL	return name of ACL protecting entry
K$NAME	position to start of entry specified by filnam and namlen
K$GPOS	return the current position in the ufd as a 32-bit integer in filnam
K$SPOS	set the current position in the ufd from the 32-bit integer in filnam

The entry is returned in 'buffer' in a format which is slightly altered from the physical disk format.

Entry format returned by RDEN$$

word	
0	ECW type/length
1-16	file name (name of file, ufd or ACL)
17	owner/non-owner protection (not returned for ACL)
18	reserved
19	file type
20,21	date/time last modified
22-23	reserved
24	length of ACL name
25-40	ACL name (for file or ufd) blank padded

2.5 Existing file attributes set by SATR$$

Calling sequence :

```
CALL SATR$$ ( key,filnam,namlen,array,code)
```

The following keys set existing file attributes :

word		
6, bit 9	delete bit	K$SDL
6	protection keys	K$PROT
8-9	date/time last modified	K$DTIM
8-9	date/time last saved	K$DTSA
10 bits 5,6	read/write lock	K$RWLK
10 bits 2	dumped bit	K$DMPB

3 Proposed new file attributes

This proposal covers a new method of associating additional attributes with file system objects. To increase flexibility, new information will no longer be in the form of additional subentries within the file entry. This will also lead to less frequent movement of file entries as they will not need to be moved when the attributes are changed.

The new method will be to define a new class of ufd entry called an attribute entry. These will normally not be visible to the user. Attribute blocks will be valid file system entries with ECWs like any other.

Within the new class of 'attribute entry' there will be two types:

a) attributes containing information needed by the operating system

Initially there will be one such attribute, to be known as the 'OS attribute' but later, an LIO attribute may be added.

This OS attribute entry will have its own type in the ECW.

Word 3 of the existing file entry will point to the OS attribute.

b) user-definable attributes

These attributes will all have the same type value in the ECW. The name of each attribute will be stored as part of the attribute data structure.

The names of two user-definable attributes will be predefined:

1) the PRISM attribute

2) the system backup attribute

Word 4 of the existing file entry will point to the first of the user-definable attributes.

A file entry can have as many user-definable attributes as desired; each user-definable attribute will be a separate attribute entry and all user-definable attributes for a particular file entry will be chained together in a doubly-linked list. The first entry in the list and the OS entry will each point back to the entry using them.

All pointers are actually integers representing the offset of the attribute block from the start of the directory.

In the initial implementation, ACLs will not be allowed to have any user-defined attributes.

The maximum size of an attribute entry, including control information and data will be 255 words, just as with other file entries.

3.1 Predefined attribute types

3.1.1 Operating system entry

contains information needed by PRIMOS such as: File size, date/time created, date/time last accessed and possibly other information. This entry will automatically be allocated for all files.

3.1.2 System backup entry

contains information for use by a system backup utility, as well as for offline files. This entry will automatically be added the first time the entry is saved.

3.1.3 PR1SM entry

contains information needed by the PR1SM/ROAM software.

3.2 Changes to existing file entry

word	
3	OS attribute position (offset)
4	Position of 1st user definable attribute (offset)

3.3 Structure of new attribute entries:

3.3.1 OS attribute entry

word		
0	ECW	Entry Control Word: type=6, length=16
1	reserved	
2	bakptr	Pointer back to object using this entry
3-4	numrec	Size of file in records/pages
5	lwords	Size of last record/page in words
6-7	dta	Date/time last accessed
8-9	dtc	Date/time created
10-11	lra	Last record in file
12-15	reserved	

3.3.2 User-definable attribute entry

word		
0	ECW	Entry Control Word: type=7, length>=9
1	flink	Position of next attribute in chain (offset)
2	blink	Position of previous attribute or main entry
if first		
3-5	reserved	
6	SCW	Subentry control word for name of attribute
7 - n	name	Name of attribute
n+1	SCW	Subentry control word for attribute data structure
n+2 - m	data	Data structure of attribute

3.4 Automatic conversion of old-style entries:

Anytime a file without an OS entry is accessed, the file system will allocate an OS entry and and set word 3 of the file entry to point to this new attribute entry. This will allow PRIMOS to automatically convert any file from an older Rev19 partition to the new format. Also, anytime a file with an old-style PR1SM type entry is accessed, the file system will allocate a new style attribute entry, copy the information into it, delete the old subentry from the file entry, and chain the new attribute block onto the file's attribute list. This will allow PRIMOS to automatically convert any file with an old-style PR1SM subentry to the new format. System backup attributes will be added to file entries the first time they are saved by the system backup program. There is a possibility that a disk partition might run out of space during one of these "automatic" attribute adding procedures.

4 New routines to set/read attributes:

A subroutine called WATTR$ will be provided to define an attribute and associate it with a file. This routine will also be used to change an attribute already associated with a file. A subroutine called RATTR$ will be provided to return any attribute of a file. A subroutine called DATTR$ will be provided to delete any desired attribute of a file. RATTR$ and DATTR$ will allow specification of the attribute by its name or its ordinal number in the list of attributes for that file. This second option is provided primarily for efficiency in fetching and deleting attributes of a file. The user should be aware that adding new attributes to a file may change the position of others in the list. Possible errors are E$ATNF - attribute not found, or E$EOF when ordinal number is out of range.

4.1 Routine to read named/numbered attribute of a file

```
CALL RATTR$(key, filnam, fillen, attrnm, attrln,
            atridx, data, dataln, code)
```

where	key = K$READ	- read an attribute by name	(Input)
	+ K$POSN	- specified by ordinal number	
	filnam	- file whose attribute is wanted	(Input)
	fillen	- length of file name	(Input)
	attrnm	- name of attribute	(Input)
	attrln	- length of attribute name	(Input)
	atridx	- ordinal number (for K$POSN)	(Input)
	data	- buffer attribute is returned in	(Output)
	dataln	- maximum length of attribute data, NOT of total structure	(Input)
	code	- error code	(Output)

4.2 Data structure returned by RATTR$

```
dcl 1 attribute based,
        2 attr_name char(32) var,          /* name of attribute */
        2 attr_data_len fixed bin,         /* length of data */
        2 attr_data(dataln) fixed bin;     /* attribute data */
```

4.3 Routine to set any user-definable attribute of a file

```
CALL WATTR$(key, filnam, fillen, attrnm, attrln, atridx
            data, dataln, code)
```

where	key = K$WRIT + K$POSN	- define an attribute by name - specified by ordinal number	(Input)
	filnam	- name of file	(Input)
	fillen	- length of file name	(Input)
	attrnm	- name of attribute	(Input)
	attrln	- length of attribute name	(Input)
	atridx	- ordinal number (for K$POSN)	(Input)
	data	- buffer containing attribute data	(Input)
	dataln	- length of data to use	(Input)
	code	- error code	(Output)

4.4 Routine to delete any user-definable attribute of a file

```
CALL DATTR$(key, filnam, fillen, attrnm, attrln, atridx, code)
```

where	key = K$DELE + K$POSN	- delete an attribute by name - specified by ordinal number	(Input)
	filnam	- name of file	(Input)
	fillen	- length of file name	(Input)
	attrnm	- name of attribute	(Input)
	attrln	- length of attribute name	(Input)
	atridx	- ordinal number (for K$POSN)	(Input)
	code	- error code	(Output)