(4) initalize and do OSS accounting
   open and create
   [operation check]

(5) prepare for failure from MKPTR & hash
   MKPTR -> common B
   [records]

(6) clear OSS records in MKPTR -> re-allocate & fail
   in MKPTR

(6A) Hash reserves record of memory (possible failure); release release record

(7) clear deferred reservation space to pool
destroy & destroy generator progress

(8) non-FHR

(10) extract FHR in DOS files FHR record
DPS to open = FNR + Pose and + 16 = 18

Actual DPS to open = FNR + Pose and + \( \frac{50}{8} + \left( \frac{80}{8} \times 8 \right) \)

\( \frac{m+1}{p+1} \)

\( 2 \times \left( \frac{m+1}{p+1} \right) + 1 \)

Two final files to code.

1. Fix references to FR. caps and:
   - service, smart, quick, create
   - remote, helper

2. Fix references to open case to:
   - abort, open by create

3. Create DPS allocation also in 'Disk 2' (as normal load)
   - Increase size of POS
   - manually change in 'Disk 3'
   - Don't need to delete.
   - File buffer full load everyday after change made.

Manual

Disk 1

File
on create → reserve & set standard limit in FHR after
the first call

just global open

on open → require overlay & set max in FHR →
after lib returns adjust reserve

if globally open — nothing

\[
\text{standard limit} = \left\lceil \frac{\sum \text{point block size}}{8} \right\rceil + \text{numblocks} \times 1
\]
in MKPTK (must be added after counter initialization)

increment "DC in use" in MKPTK

if reads "DC line" in MKPTK return to 86-1

in DCPTK called from oracle, helper

decrement "DC's in use" - never goes zero or negative
Each file header record will carry a counter of at least two called 'DOS limit' and 'DOS in use'. The file header block will carry the 'DOS limit' count.

On file creation, 'DOS limit' is initialized to a value sufficient to sustain one attached block. Also sufficient DOS to cover the list is withdrawn from DOS bank. The local DOS total is also incremented by the list. The "DOS in use" field in RUN is set to the amount actually used to create & hold RUN of file level.

On open file, the DOS领导者 on HBR will do its usual job only before allocating any extra space more DOS will withdraw enough to cover the 'DOS to open' counter.

The file leader in OS responsibilities is X6.

Local limit is decremented by largest amount held.
is done from bank. After the standard amount is allocated local and global logical block. On frozen file no DB allocation.

Local limit

Local limit \leq total DDS

In users are automatically allocated on disk system initialization.

Calls to increment counter in the fail or success - record back
<table>
<thead>
<tr>
<th>FILE UNIQUE NAME</th>
<th>H B S</th>
<th>DISK ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNDING DAR POINTER</td>
<td>FILE HASH-TABLE LINK</td>
<td>SUSP LIST HEAD</td>
</tr>
<tr>
<td>FROZEN OPEN COUNT</td>
<td>TOTAL OPEN COUNT</td>
<td>ECS INCMN C-LIST INDEX</td>
</tr>
<tr>
<td>FLAGS</td>
<td>NO. OF EXISTING DATA BLOCK</td>
<td>FIXED ECS NECESSARY TO OPEN</td>
</tr>
<tr>
<td>CLAIM COUNT</td>
<td>CLAIM QUEUE HEAD</td>
<td>CLAIM QUEUE TAIL</td>
</tr>
</tbody>
</table>

**MEMBERSHIP UNIQUE NAME**

**ROOT PTR**

**SHARE WORD**

**FLAGS**

- ONE-LEVEL DATA BAR FILE
- OPEN-WRITE PROBE
- XOPEN

**FRAMES**

**NAMES**

- FR.NAME
- FR.SUSP
- FR.LOCK
- FR.DAR
- FR.OPEN
- FR.CAPX
- FR.FLAG
- FR.DB5Z
- FR.PROB
- FR.WRT
- FR.SIZE
- FR.CLAIM
- FR.MEMB
- FR.FUND
- FR.ROOT
- FR.SHAP
FILE HEADER RECORD ("FHR")

ECS (Disk-Data-Structure) version of file header

Details of SHAPE WORD: see FILE HEADER BLOCK

Details of ROOT POINTER: see UNPACKED POINTER BLOCK (pointer formats)

(NOTE: in root pointer, both disk addresses are always zero)
FILE HEADER BLOCK ("FHB")
Disk version of file header

Details of FLAGS:

1. Zero-level flag
2. Data-block size

Details of SHAPE WORD:

HBS=Header Block Size
I. On Disk

II. In Disk Data Structure

Pointer Formats:

To Pointer-Block:

```
<table>
<thead>
<tr>
<th>Page 1 Addr</th>
<th>Page 0 Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page 3 Addr</td>
<td>Page 2 Addr</td>
</tr>
<tr>
<td>Page 15 Addr</td>
<td>Page 14 Addr</td>
</tr>
</tbody>
</table>
```

To Data-Block:

```
<table>
<thead>
<tr>
<th>Attach Count</th>
<th>NEW Disk Addr</th>
<th>OLD Disk Addr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

"Flags":

- Type: 0 = Data-Block, 1 = PTR-Block
- Status: 0 = OUT, 1 = IN
- ERO: 0 = FILE O.K., 1 = FILE Out of Date

At this point in the tree.
### Disk Data Structure

**Header Block Size**

- Unique Name
- DISK ADDR
- DISK INDEX
- HASH LINK
- INDEX COUNT
- OPEN CNT
- FREEZE CNT
- C-LIST INV
- FCAR
- FC-LIST
- FC-BLOCK
- FC-DAR
- FC-DSIZE
- FC-ZERO
- FC-SPACE

**Page Structure**

- `<SHARP>`
- `<POINTER>`

**Page Switch**

- Page 1
- Page 2
- Page 3
- Page 4

**Pointer Page**

- 10 pointers

**Wait**

- Wait for activity
- Freeze object
- Sequence page-disk control

**Shape Number**

- Level size = 2
- Shape #

**Attach Count**

- STATUS 0 = OUT
- 0 = IN
- 1 = COMMAND

**Page Switch**

- Page 1
- Page 2
- Page 3
- Page 4

**Page 15**

- Page 14

**Pointer Page**

- 10 pointers
## Disk Header Table

### File Header

- Unique Name
- Disk Index
- Disk Addr
- Open Count
- Suspense Queue

### Pointer

- Responsible Data Block Size
- Open Write Lock

### Shape

<table>
<thead>
<tr>
<th>Shape</th>
<th>10</th>
<th>7</th>
<th>4</th>
<th>4</th>
<th>4</th>
<th>4</th>
<th>4</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

### Shape Numbers

- Distance (bits to 0) in the AR
- Shape Numbers
- Level max = 2^5

### Pointer

- Data Block
- OLD Disk Addr
- NEW Disk Addr

### Old Disk Addr

- Attack Count

### Old Pointer Block

- Going Out
- Coming In

### Page Switch

- PAGE 1
- PAGE 2
- PAGE 3
- PAGE 4
- PAGE 5

### Pointer Page

- Ht Pointers
local file header (LFH)

<table>
<thead>
<tr>
<th>UNIQUE NAME OF ECN FILE</th>
<th>FILE ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attached</td>
<td>COUNT QUEUE</td>
</tr>
</tbody>
</table>

Frozen flag:

Attached block record (ABR)

<table>
<thead>
<tr>
<th>MAP</th>
<th>REG</th>
<th>ABR</th>
<th>LINK</th>
<th>POS1</th>
<th>PTR</th>
<th>ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Garbage

Map record (MR)

<table>
<thead>
<tr>
<th>MAP</th>
<th>CH1</th>
<th>CH2</th>
<th>CH3</th>
<th>COUNT</th>
</tr>
</thead>
</table>

GARBAGE

Multimap record (MMR)

<table>
<thead>
<tr>
<th>MAP</th>
<th>MAP1</th>
<th>MAP2</th>
<th>MAP3</th>
</tr>
</thead>
</table>

POINTER MAP/MAP
Local File Table

1. END OF FILE NO LESS = zero word
2. Free list entry = negative

MAP INDEX
MAP INDEX
MAP INDEX
100%
Low I: processor data

Local file table one entry for each file.

Local data allocated as needed in local
plus increments changing map entry to
save needed position.

<table>
<thead>
<tr>
<th>PATCH FILES</th>
<th>OPEN COUNT</th>
<th>ATTRIBUTE</th>
<th>MODE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<DIR INDEX> identifier for
<OPEN COUNT>
<ATTR QUEUE> kind of block in list.

REG | MTRN | QUEUE | BLOCK | PTR ==> FILE | N


Take back 700 ft
call $00
C.O. FAPEG
C.O. FAPEG B
DISK ALLOCATION Routines:

1. **GDSKHDR** (Get fixed disk space for file-header-block)
   - **Input:** X4=number of sectors
     - B7=return
   - **Output:** X6=Disk Addr
     - or
     - returns to B7-1 if no space available
       (due to accounting error or fragmentation; SYSERR)

2. **FDSKHDR** (Free fixed disk space for file-header-block)
   - **Input:** X4=number of sectors
     - X5=Disk Addr
     - B7=return
   - **Output:** (None)
     - or
     - returns to B7-1 if intersected free block
       (SYSERR)

3. **GDSKPTR** (Get swapped disk space for pointer-block)
   - **Input:** B7=return
   - **Output:** (None)
     - or
     - returns to B7-1 if swapped-space cutoff

4. **FDSKPTR** (Free swapped disk space for pointer-block)
   - **Input:** B7=return
   - **Output:** (None)

5. **GDSKDAT** (Get swapped disk space for data-block)
   - **Input:** X4=number of sectors
     - B7=return
   - **Output:** X6=Disk Addr
     - or
     - returns to B7-1 if swapped-space cutoff

6. **FDSKDAT** (Free swapped disk space for data-block)
   - **Input:** X4=number of sectors
     - X5=Disk Addr
     - B7=return
   - **Output:** (None)
     - or
     - returns to B7-1 if intersected free block
       (SYSERR)
What happen on DBW for 10th w.
for fit?

1885
545 EKR 71682

SH72AM
look boot file
add file
pass

black for allocation.
return on create ops

red
ECS

user AB

swaped ECS AB

holding block files cannot be on the same AB as ECS inearns of disk files

initialization -> directory alloc lock
SHAZAM for your shell file

lock both file

Xchange COS version (use holding lock file for buffers)

Pclose primary file
Attack block (detach block) (map/regular)

Parameters:
- C: EES incarnation of disc file
- D: file address 1
- D: file address 2

Action:
1. convert file capability to DHT index and find unique name - scan LFT to find file
2. scan attached blocks list for file in DHT. If file in file block - increase attachment count & exit
3. lock DHT
   - update local allocation data
   - decode address in DHT to get data block pointer (may cause pointer block reads & waits)
4. If data block in EES - up attachment count & exit

If block is in transit - up attachment count & return. If disc central well check attachment count before releasing EES space.

On attack, after successful read from disk, release temp disk space occupied by the block(s) attached.
if \( EF \) then error

if \( CT \geq 0 \) then

else if \(-SF \) then \( CT = CT + 1 \);

end if

else \( SF \) true end

else start return end

else go on up
if EF then
begin if CT \neq -1 then expr.
\text{CT} < 0 \land \text{EF} < \text{false}
\text{handle gene end}

if SF then
begin SF < \text{false}
\text{CT} < \text{CT - 1}
\text{handle gene end}
\text{handle gene end}
if head events
- in FHR
  counter CT
  cir queuehead

- in LFT
  exclusive flag (EF)
  shared flag (SF)

ops exclusive

  if (EF V SF) then error
  if (CT > 0) then queue up
  if (CT = 0) then EF <- 1 nsec
  if (EF = 0) then E <- 0

ops shared

  if EF then error
  if (-EF + CT > 0) then BSF <- True
  if SF then CT < CT + 1, exit else spec set and calci
<table>
<thead>
<tr>
<th>IDENT</th>
<th>GF FILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td><strong>TEMPLATE FOR GLOBAL FILE</strong></td>
</tr>
<tr>
<td>*</td>
<td><strong>ORG 0</strong></td>
</tr>
<tr>
<td>*</td>
<td><strong>GF::SUSF BSS 0</strong> START OF SUSP WDS</td>
</tr>
<tr>
<td>*</td>
<td><strong>GF::SUSL BSS 0</strong> END OF SUSP WDS</td>
</tr>
<tr>
<td>*</td>
<td><strong>GF::IOTF BSS 0</strong> START OF I/O TABLE</td>
</tr>
<tr>
<td>*</td>
<td><strong>GF::IOTL BSS 0</strong> END OF I/O TABLE</td>
</tr>
<tr>
<td>*</td>
<td><strong>GF::DDSF BSS 0</strong> START OF DISK DATA STRUCTURE</td>
</tr>
<tr>
<td>*</td>
<td><strong>GF::DDSL BSS 0</strong> END OF DISK DATA STRUCTURE</td>
</tr>
<tr>
<td>*</td>
<td><strong>GF::HSHF BSS 0</strong> START OF HASH TABLES</td>
</tr>
<tr>
<td></td>
<td><strong>FHRHASH BSS 128</strong> FHR HASH TABLE</td>
</tr>
<tr>
<td></td>
<td><strong>DARHASH BSS 128</strong> DAR HASH TABLE</td>
</tr>
<tr>
<td>*</td>
<td><strong>GF::HSHL BSS 0</strong> END OF HASH TABLES</td>
</tr>
</tbody>
</table>

**SYMBOLS XTEXT**

**END**
attack clock

if block points in then
    begin
        if testset (busy [block]) then suspend list;
    end

    if in [block] then begin
        lock [clock];
        rent [busy];
        return end.

schedule read wait

copy to clock

set "in"

rent "busy" (do suspend list) end

end get with clock (busy)
Calls on low-level disk system

1. Open file
2. Close file
3. Pseudo-close file
4. Create file
5. Add block
6. Delete block/move block
   check block
7. Attach block (nop/nomop)
8. Detach block (nop/nomop)
9. Read
10. Write
Parameter: 

1. file (must be open) 

2. block address (file address within desired block)

A block is added to the FFS version of the file and the necessary pointers are updated in the DHT. No disc occurs is made. The new block is now attached to the process adding the block as a non-map block. Disc space is not allocated until either the file is closed or the block is detached.
Create file

Parameters:

- Unique name, header, block size
- File shape

 Allocation stuff?

- Return data path

An appropriate file header block is constructed and written onto the disk. The disk address of the new file header is returned. No data is written about is needed and the file is not open and contains no data unless it is a zero level file in which case the data exists and is all zeros.
Pseudo close file

Pseudo close is similar to "close file" except no open count are checked, no blocks are detached, and not all the space in the DHT for the file is released. Only marked pointers blocks are released in the DHT but the file pointer blocks on the disk and the file header are re-written as with the true "close file" action.
Close file.

The file is located in the DHT and the local tables. The local open count is decremented. If it remains non-zero, the action is complete. Otherwise, all locally attached blockgroups are blocked!

The global open count is decremented. If the global count remains non-zero, the action is complete. If the global count is exhausted, the EFS file may be destroyed. Finally, the file tree in the DHT must be pruned (the file has opened other data for read only be traversed, require a stack) writing out all pointer blocks to new locations on the disk and finally rewriting the header block. Then, all duplicated portion of the file on the disk (with the exception of the memory version of data blocks pointers) may be released. Also, the DHT space occupied by the file header + pointers is released. Some monky business will also be required on the allocation data for the file.

All writes can be done without wait except for the header block. Disk control will send an interrupt if trouble interrupts is checked until last write. Then, must wait for response (writes physical allocation space or error can be checked in DHT).
Open file

Parameters:
1. unique name, header block size, offset address
2. type of open: 
   - read
   - write
   - add blocks
   - edit blocks
   - move blocks
3. return code: 
   - success
   - failure
4. allocation data: 
   - disk block
   - cylinder

The unique name is hashed into the DHT. On a match, the open count is incremented and the write bit (possibly) turned on. Otherwise, space is reserved in the DHT and a disk read (open) initiated.

Rcis central will complete the entry (leaving the take care of zero level pointers of data, entry locked), and create an appropriate OS file (possibly with data set or not).

The local process then take care of allocation process, construct the necessary local file data, and fabricate the appropriate capability and return it to the caller.
Close disc file (pseudo close)

Parameters:
1. capability for EFS incarnation of file
   - by user
   - by high level user
2. unique name, header block size, and disc address of header block

Actions:
1. If capability param convert to unique name using handler from capability into
   the DHT
   - decrement open count - if not zero then exit

2. If local count went to zero - detach all attached blocks

3. Block DHT
   - decrement global open count

4. If open to zero - then copy all data blocks to a new disc loc & copy all
   pointer blocks which have disk label

5. To new position on close - this must be done from the tree leaves toward the root
   - the fill header is re-written last - this operation must go to completion

6. Check suspense queue on fill - only possible for someone to be opening fill

7. Destroy DHT entry & ECS capability

8. Notify suspense queue
Frozen: foot files allocated from fixed disk space.
Action of reading file [free open speed until while]
Bozen
Identifier user file by LFD hold EDS unique name.
<table>
<thead>
<tr>
<th>Disk I/O Errors</th>
<th>Error Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative file address</td>
<td>E.NFAD</td>
</tr>
<tr>
<td>Negative record count</td>
<td>E.NWCT</td>
</tr>
<tr>
<td>File address too large</td>
<td>E.LFAD</td>
</tr>
<tr>
<td>File not in KPT</td>
<td>E.NOFILE</td>
</tr>
<tr>
<td>Block not attached</td>
<td>E.NAHT</td>
</tr>
<tr>
<td>Block does not exist</td>
<td>E.NOBLK</td>
</tr>
<tr>
<td>Insufficient swapped space</td>
<td>E.NOSWP</td>
</tr>
<tr>
<td>Insufficient disk space</td>
<td>E.NOSDK</td>
</tr>
<tr>
<td>I/O error</td>
<td>E.IOERR</td>
</tr>
<tr>
<td>Out of attached block records</td>
<td>E.NABR</td>
</tr>
<tr>
<td>Data block exists</td>
<td>E.NSBR</td>
</tr>
<tr>
<td>Negative number of levels</td>
<td>E.NLEV</td>
</tr>
<tr>
<td>Too many levels &gt; 10</td>
<td>E.NDBS</td>
</tr>
<tr>
<td>Negative data block size too small</td>
<td>E.NOTPOW</td>
</tr>
<tr>
<td>Shape num not power of 2</td>
<td>E.LDBS</td>
</tr>
<tr>
<td>Large data block size</td>
<td>E.NECS</td>
</tr>
<tr>
<td>Insufficient freed ECS</td>
<td>E.NSBN</td>
</tr>
<tr>
<td>Negative shape number</td>
<td>E.LSBN</td>
</tr>
<tr>
<td>Large shape num</td>
<td>E.NFL</td>
</tr>
<tr>
<td>File too large</td>
<td>E.NOTMP</td>
</tr>
<tr>
<td>No temp disk space</td>
<td></td>
</tr>
</tbody>
</table>
frozen file is charged to temp directory (i.e. process directory)
extra open file is done on temp directory
name of temp directory (possibly?) is stored in PTR when file is frozen
no local attached block records are created!
Frozen space should be accounted as fixed EOS space

CREBLK
And must do, in the case of a deleted
block which has an old address

\[
\begin{align*}
\text{new addr} & \quad \text{old addr} \\
\text{new} & \quad \text{old}
\end{align*}
\]

must refund the proper number
of sectors to the owning
directory (easily done)

In case of a pt. block which exist
in the old copy (i.e. deleted
non-zero) but not in the new-

must refund one sector to owning
directory (easily done)

close only

must do a 'close' on the
owning directory

Delete block will make a reference of a new
cdisk sector exist — also will
deallocate the actual disk
space for the data block whenever

\[
\begin{align*}
\text{new addr} & \quad \text{old addr} \\
\text{new} & \quad \text{old}
\end{align*}
\]

new space exists
Create block charges owning directory except when:
new addb = -1

(i.e. old addb = 0 and block non-exist.)

Create block charge for all pointers block created.

Destroy seize control of file.
and uses same charging algorithm as close.
on detach - if clean

on every attach (swaps excluded) (notate
increased attach count of sectors
of temp space

temporary disk
also inc. or read local temp + space

on close (or P close) reduce proc. local
temp disk space, temp + space

paying extra only results from explicit
attach/detach/attach cycles

Pay for temp disk space on basis
of total of attached sectors from
that attached to close or P close
of the file(s)

on delete new temp space, new disk
addr + 0
ehr bss student

var bss

temp1 bss 1

temp2 bss 1

temp3 bss 1

temp4 bss 1

temp5 bss 1

clear storage

split1 bss 1

split 2 bss 1

use ds bss 5

use rsp bss 5

use ebp bss 5

 realloc fill

read/write swapped fill

my alloc

my malloc

my malloc

far dop

far dop

far dop

near dop

dcl x 0

defines

left bss left len

ret1 bss 1

ret2 bss 1

ret3 bss 1

opinr equ ret1

attach equ ret1

sizes

16, 16, 16, 16
EQU
EQU
EQU

file

EVC

EVC

MEQCL

MEQCL

DARHASH
DARHASH

unique

unique names

Hash table for DNP's
" " TNP's
Block must exist in source file to target file.

Block may either be unattached or attached only to the caller—it may not refer to any map of the caller.

Files are locked in order of the magnitude of their OSS usage planes.

Old disk space scheduled for release now most prevalent.

Release:

Source print:

```
1 x12 ➞ 1 x1-2
1 x10 ➞ 1 x1-2
1 010 ➞ 1 0
1 014 ➞ 1 0
```

Target print:

```
1 000 ➞ 1 1010
1 x1-2 ➞ 1 x1 0
```
Delete Block

Parameters:
- Def  (must be open)
- file address

The requested block is detached (must be already attached). If the global attach count does not expire or if the block is locally attached as a map block, the action is return. Otherwise, the DST is updated to forget the block. Both options for the block are not released until the file is closed.
1. if deletion count in DHT is non-zero, the file block

2. find pointer to the data block

3. release space on disk
4. zero file addr fields of pointer

5. if no existing block - added in OS -
nothing done on disk [modifies pointer block
parameters]

Action:
if (not attached to local process and
global attach count is non-zero)
or (attached to local process and global
attachment > 1) the file

if file 2 at address 2 is not empty
then error
otherwise move pointers around - field
attached block counts in local file
table as needed.
File Header Block (on disk)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Name</td>
<td></td>
</tr>
<tr>
<td>Disk Addr</td>
<td></td>
</tr>
<tr>
<td>Disk Addr</td>
<td></td>
</tr>
<tr>
<td>Data Length</td>
<td></td>
</tr>
<tr>
<td>Fixed Disk</td>
<td></td>
</tr>
<tr>
<td>User Space</td>
<td></td>
</tr>
<tr>
<td>User Space</td>
<td></td>
</tr>
<tr>
<td>File Size</td>
<td></td>
</tr>
</tbody>
</table>

Notes:
- File Size = L
- Additional field: SHAPE
1) CREATE FILE
2) DESTROY FILE
3) OPEN FILE
4) CLOSE FILE
5) RETURN READ
6) RETURN WRITE
7) RETURN FILE TO FILE COPY
8) ATTACH BLOCK (S)
9) DETACH BLOCK (S)
10) PUT FILE SECTION IN MAP
11) DELETE MAP ENTRY
12) CREATE FILE DATA BLOCK
13) DESTROY FILE DATA BLOCK
14) SWAP OUT ALL ATTACHED BLOCKS
15) SWAP IN ALL ATTACHED BLOCKS
Parameters:
1) number of levels
2) pointer to array of shape numbers
3) directory allocation data

Action: The shape numbers are checked and a file header record (FHR) is constructed (except for the disk address of the header block). The FHR is inserted into the disk data structure (DDS) and an DDS file of the proper shape is created. The file header record (FHR) is used to construct a file header block for the disk which is allocated and copied to the disk. The local file table is updated to reflect the new file and appropriate account is done to account for disk and DDS space. Finally, the capability for the DDS file (open, write) and the disk address, header block size, and unique name of the file are returned.
open file

Parameters:
1) unique name, header block size, & disk address
2) directory allocation data
3) open for write?
4) force open?

Action: The designated file is logically opened
within the requesting process. If the file has previously been opened (and is still open) then the local file table (LFT) is simply updated and the file capability is returned. Otherwise, the file header block must be read from the disk, the directory allocation data installed in the disk data structure (DDS), and a file header record (FHR) created for the file in DDS.

The action aborts if the file is already open and a force open is not requested. The options for writing, reading, and deleting disk blocks are disabled if the capability returned if open write is not specified.

For zero level files, the data block is automatically attached and is accounted for as fixed ECS space.
If the file is open and frozen, no accounting takes place. Note that for a frozen file, all blocks are permanently attached.
1) 'DARKEN' or 'FAHREN' must be set to middle lock end of record.

2) When waking up the suspense list, event = zero means record has been destroyed.
create file header
Parameter: date for header block (in words) / size of block

return disk address (also inserted as last word of header)
allocate free part stuff in buffer
wait for completion

write pointer block in current set, file unlocked
allocate disk space
action: construct block in buffer
(release space in PAS
(wait for completion)
write

central line: if successful
fix up address in PAS
release old disk space
set "out" flag really out
notify completion

Note: don't pick up
no more requests
reply then send
channel queue -1
Read:

1. file header (open)
   disk addr, size
   FHR addr

2. pointer block
   disk addr (size = 64)
   FHR addr
   DOS addr of pointer
   disk addr of data block
   wait/nowait

Write:

1. create file
   disk addr, size
   ptr to head block (rest of block should be zeroed)

2. file header
   disk addr, size
   FHR addr

3. pointer block
   disk addr (size = 64)
   FHR addr; DOS addr of pointer; wait/nowait

4. data block

pseudocode or not
disc access interface

1) write
   a) disk add
   b) stuffer add
   c) size

   disc control -- no action except reply

   write full header at file creation time
   no DDS data on the file
   should wait for completion

2) open file
   a) disk add
   b) header block size
   c) FHR add

   note: file header locked already

disk control
1) complete FHR allocate c-list
2) create CDS
3) unlock FHR check c-list
   done
   wait
should wait for completion
## Open File

**Parameters**

1. Unique name, mode, block size, and disc order of file bloc
2. Type of open
   - Read
   - Write
   - Read/Write
3. Disc file or normal file

**Action**

1. Local file table (LFT) scanned for unique name. If locally open, then count increased and file capability corresponding to type of open is returned.

2. Unique name hashed into DHT.
   - If entry in DHT:
     1. DHT open status updated
     2. LFT entry made (open count = 1)
     3. Proper file capability returned
   - Else (DHT entry not found):
     1. Lock DHT
     2. DHT entry started
     3. DHT entry finished by disc central
     4. Disc central also creates EFS file.
     5. After response from disc central:
        1. Above or below the relax file “lock”
        2. Construct user capability w/proper file "lock"

3. If suspense list, then start over.

4. All allocation charges for the file is to point person to open the file.

5. User will fund all EFS allocation.

User is charged for time space for all allocated blocks.
Request disposition

Keep an entry indexed by "request aleo index"

<table>
<thead>
<tr>
<th>DIS</th>
<th>APPR</th>
<th>DATE</th>
<th>APPR INDEX</th>
<th>FILE ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>16</td>
<td>18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

User IP in request frame (20 B/TS)

```
1 3 4 12
```

- Type: 1
  - Action: Unset (000)
  - Open: 001
  - Close: 010

Pointer block (no release): 011
Print the above
must be made with plates
detail dump of file
attack block
PO D: attack (non-map),
P1 C: file
P2 D: file adodb.1
P3 D: file celob.2
P3 > P2

A) convert P1 to DHT index
   read DHT
   scan local file data for file
   test P2, P3
   etc.
   PA = P2 (mod 5n)

B) find file for appropriate data block (using pointers)
bloks as needed

C) scan local attached block. If already attach
   if so increase non-map attch count
   and go to 4.

lock section DHT

D) make local attachment (enter block in DHT)

E) update accounting for swapped space (local)

F) increment reference count on global data block pointer

G) if data block "in" unlock DHT
   goto 4.

H) if data block "out" set "comming in"
   initiate transfer (no wait), goto 4.

I) if data block "comming in" unlock DHT
   goto 4.

J) FA ← FA + Sn
   if FA > P3 exit
   else goto B
Control process

on data block write

check attack count before releasing file data block

on open

if zero level - [Handwritten note]

on attack

must copy to disc map buffer which is then moved to file and file
calls: read (always wait) — attach & detach
write
open
close
attach (clean/dirty)
detach (clean/dirty)

**READ**

increment local attach count
increment global attach count if local was zero

block in?
- block coming in?
  - add to suspension list
- block going out?
  - remove from suspension list
  - mark block "coming in"
  - make disc request
  - wait
  - deliver
  - detach

receive disc response — look up user request
disc read?
copy block to file
  - set block in
  - get head of suspension list

send normal event to local proc if wanted
send events to suspension queue

exit
atomic action

if dest busy: if busy then busy else suspend (2)

Allocate DHT

DHT lock

Suspend lock

Allocate lock

Lock 1: write bit, attach paint

Lock 2: open count, suspend (and lock DHT), suspend file chain

Lock N entries in DHT

File lock

Note: must compare user capabilities with disc version of same for match on MNT and incore files

must inhibit interrupt counter

locks
directory entry

FILE NAME
  ACCESS LIST
  DISC ADDR and UNIQUE NAME

LINK NAME
  ACCESS LIST
  DIRECTORY (DISC ADDR and UNIQUE NAME)
  NAME
  ACCESS KEY

DIRECTORY NAME
  ACCESS LIST
  DISC ADDR and UNIQUE NAME

REALLOCATION BLK NAME
  ACCESS LIST
  (contains (or o.k.) list of o.k. obj)

ACCESS KEY NAME
  ACCESS LIST
  VALUE

EVENT QUEUE NAME
  ACCESS LIST
  SHAPE (size of event queue and initial events)
  unique name

SUBSYSTEM NAME
  ACCESS LIST

  1) access key name (for subp name)
  2) file for map (LINK)
  3) entry for subp (ENTRY-FL-EMS-MAP)
  4) C-LIST formula (for LINK)
local file table
file header record

DHT index 12 bits
attached block queue 12 bits
open count
<table>
<thead>
<tr>
<th>Unit Member Installation</th>
<th>Inst. Code</th>
<th>V / A</th>
<th>Representative</th>
<th>Individual(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lawrence Radiation Laboratory, Berkeley</td>
<td>BKY</td>
<td>V</td>
<td>Mr. James G. Baker</td>
<td>Miss J. V. Lawrence</td>
</tr>
<tr>
<td></td>
<td>6600 (9)</td>
<td>131K</td>
<td>12/65</td>
<td>Bldg. 50B, Room 1237</td>
</tr>
<tr>
<td></td>
<td>6600 (27)</td>
<td>131K</td>
<td>7/67</td>
<td>415-843-2740, X-5529</td>
</tr>
<tr>
<td></td>
<td>Lawrence Radiation Laboratory</td>
<td>Berkeley, California 94720</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lawrence Radiation Laboratory, Berkeley</td>
<td>CAL</td>
<td>V</td>
<td>Mr. Kendrick J. Hebert</td>
<td>Dr. Loren P. Meissner</td>
</tr>
<tr>
<td>Berkeley, Computation Center</td>
<td></td>
<td></td>
<td>Computation Center</td>
<td>Bldg. 50B, Room 3245</td>
</tr>
<tr>
<td></td>
<td>6400 (12)</td>
<td>65K</td>
<td>11/66</td>
<td>415-843-2740, X-5739</td>
</tr>
<tr>
<td></td>
<td>6400 (69)</td>
<td>32K</td>
<td>12/68</td>
<td></td>
</tr>
<tr>
<td>Lawrence Radiation Laboratory,</td>
<td>LRLL</td>
<td>V</td>
<td>Dr. Sidney Fernbach</td>
<td>Mrs. Penny Gee Library</td>
</tr>
<tr>
<td>Livermore, Computation Department</td>
<td></td>
<td></td>
<td>Computation Department</td>
<td>Bldg. 103 (L-61)</td>
</tr>
<tr>
<td></td>
<td>6600 (1)</td>
<td>131K</td>
<td>9/64</td>
<td>415-642-5265</td>
</tr>
<tr>
<td></td>
<td>6600 (13)</td>
<td>131K</td>
<td>3/66</td>
<td>Mr. David F. Hussey</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Computation Department</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bldg. 113 (L-61)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>415-447-1100</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Librarian</td>
</tr>
<tr>
<td>Lehigh University</td>
<td>LUCC</td>
<td>V</td>
<td>Mr. John E. Smith</td>
<td>Mr. Albert Siegel</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Room 117, Packard Laboratory</td>
<td>215-867-5071, X-456</td>
</tr>
<tr>
<td></td>
<td>6400 (53)</td>
<td>65K</td>
<td>8/68</td>
<td>Mr. Larry White</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>215-867-5071, X-355</td>
</tr>
<tr>
<td>Lockheed Georgia Company</td>
<td>LGC</td>
<td>V</td>
<td>Mr. C. C. Douglas, Jr.</td>
<td>Mr. A. B. Atchley</td>
</tr>
<tr>
<td></td>
<td>6400</td>
<td></td>
<td>Dept. 87-11, MZ 463</td>
<td>Dept. 87-10, MZ 288</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Lockheed Georgia Company</td>
<td>404-424-7725</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>86 S. Cobb Drive</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Marietta, Georgia 30060</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Phone: 404-424-3455</td>
<td></td>
</tr>
<tr>
<td>Lockheed Missiles &amp; Space Company</td>
<td>ML</td>
<td>V</td>
<td>Mr. Conrad K. Bedient</td>
<td>Mr. Jim Hollingworth</td>
</tr>
<tr>
<td></td>
<td>6400 (6)</td>
<td>32K</td>
<td>Dept. 19-50, Bldg. 181</td>
<td>Dept. 19-50, Bldg. 181</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Lockheed Missiles &amp; Space Company</td>
<td>408-742-8993</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P. O. Box 504</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sunnyvale, California 94088</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Phone: 408-742-7865</td>
<td></td>
</tr>
<tr>
<td>Lockheed Missiles &amp; Space Company</td>
<td>MS</td>
<td>V</td>
<td>Dr. J. N. Christensen</td>
<td>Mr. J. E. Sherman</td>
</tr>
<tr>
<td></td>
<td>6400 (33)</td>
<td>65K</td>
<td>2/68</td>
<td>Dept. 19-63, Bldg. 190</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>408-742-2692</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Mr. W. E. Plunkett</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Dept. 19-63, Bldg. 190</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>408-742-2643</td>
</tr>
<tr>
<td>Los Alamos Scientific Laboratory,</td>
<td>LASL</td>
<td>V</td>
<td>Mr. Charles A. Folkner</td>
<td></td>
</tr>
<tr>
<td>University of California</td>
<td></td>
<td></td>
<td>University of California</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6600 (14)</td>
<td>65K</td>
<td>6/66</td>
<td>Los Alamos Scientific Laboratory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C-2, P. O. Box 1863</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Los Alamos, New Mexico 87544</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Phone: 505-667-5081</td>
</tr>
<tr>
<td>Unit Member Installation</td>
<td>Inst. Code</td>
<td>V / A</td>
<td>Representative</td>
<td>Individual(s)</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------</td>
<td>------</td>
<td>----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>United Computing Systems Inc.</td>
<td>UCSI 6400 (47)</td>
<td>65K</td>
<td>10/68</td>
<td>Mr. G. C. Baker, United Computing Systems Inc., 3130 Broadway, Kansas City, Missouri 64111, Phone: 816-753-4500, Mr. Robert W. Franklin, 816-753-4500</td>
</tr>
<tr>
<td>Universita di Bologna</td>
<td>UBOL 6600</td>
<td>65K</td>
<td>9/69</td>
<td>Prof. Giuseppe Mannino, Direttore del Centro di Calcolo, 33, Via Zamboni, Bologna, Italia, Phone: 222-956, Mr. J. A. Baudot, 514-343-6018, Mr. Bernard Dumont, 514-343-6018</td>
</tr>
<tr>
<td>Universite de Montreal</td>
<td>UMTL 6400 (65)</td>
<td>65K</td>
<td>11/68</td>
<td>Dr. J. St. Pierre, Directeur, Centre de Calcul, Universite de Montreal, Case Postale 6128, Montreal, Quebec, Canada, Phone: 514-343-6018, Dr. J. N. Weadon, Dept. of Computing Science, 23-4333</td>
</tr>
<tr>
<td>University of Adelaide</td>
<td>ADEL 6400 (5)</td>
<td>32K</td>
<td>7/66</td>
<td>Prof. J. A. Ovensstone, Computing Centre, The University of Adelaide, Adelaide, South Australia, Australia, Phone: 23-4333, Mr. William Clark, 602-884-3493</td>
</tr>
<tr>
<td>University of Minnesota</td>
<td>MINN 6600 (16)</td>
<td>65K</td>
<td></td>
<td>Dr. Marvin L. Stein, Numerical Analysis Laboratory, University of Minnesota, Minneapolis, Minnesota 55455, Phone: 612-373-4381</td>
</tr>
</tbody>
</table>
000001, 000000, 00000003450000 ERROR
BOO, BEAD GHOST HERE
@PURGE
COMMAND PROCESSOR HERE!
BAD SUB
!BASIC
BASIC HERE
: R, TVR0<01
: M/250; C . . . ; P; M/331; C. TS: TS: TS: TS: P
250 PRINT TAB(5),
331 PRINT TS: TS: TS: TS: TS: TS:
: F, TVR01
COMMAND PROCESSOR HERE! RECALL
UNEXPECTED FRETURm DURING COMMAND PROCESSOR, B7 = 006721
!BASIC
BASIC HERE
: R, TVR01
: RUN

53 39.23591
54 39.7 LINE 1030 PANIC
55

: T; USER ERROR
000001, 000000, 00000003450000 ERROR
BOO, BEAD GHOST HERE
@RETRY
BAD SYNTAX
@RETURN
USER ERROR
000001, 000000, HERE
PANIC IN REA.D GHOST
PURGE

Panic

Panic
10 DIM A(3), B(3), C(3), D(6)
20 DEF FNA(N) = PI*N/180
30 DEF FNB(N) = 180*N/PI
40 DEF FNC(N) = N/SQR(1-N*N)
50 DEF FND(N) = 3-ABS(3-N)
100 REM
110 LET 0 = FNA(23+(27+15/60)/60)
120 LET K1 = FNA(30)
200 FOR N0 = 53 TO 53
210 PRINT TAB(5), N0, N0+1, N0+2
220 FOR N1 = 0 TO 2
230 LET D(N1+1) = FNB(ATN(FNA(N0+N1)))*COS(0))
240 NEXT N1
250 PRINT TAB(5)
260 LET T = 3
265 GOSUB 1000
270 FOR N1 = 0 TO 2
280 LET A(N1) = ATN(TAN(FNA(N0+N1)))*COS(0))
290 LET B(N1) = FNC(SIN(FNA(N0+N1)))*SIN(0))
300 LET C(N1) = ASN(SIN(FNA(N0+N1)))*SIN(0))
310 NEXT N1
320 PRINT
330 LET T$ = "---------"
331 PRINT T$:T$:T$:T$:T$:T$
340 FOR T0 = 10 TO 60 STEP 3
350 LET T1 = FNA(T0)
360 FOR N1 = 0 TO 2
370 LET T = ASN(TAN(T1)*B(N1))
380 LET D(2*N1+1) = FNB(ATN(SIN(T/3)/C(N1)))))
390 LET D(2*N1+2) = FNB(ATN(SIN(T*2/3)/C(N1)))))
400 NEXT N1
410 PRINT T0;
420 LET T = 6
425 GOSUB 1000
430 NEXT T0
440 NEXT N0
450 PAUSE "DONE!"
1000 REM MAT PRINT
1010 FOR Q0 = 1 TO T
1020 PRINT D(Q0);
1030 NEXT Q0
1040 PRINT
1050 RETURN
99999 END
PANIC IN COMMAND PROCESSOR
!WHO
COMMAND PROCESSOR
!LOGOUT
TSS NOW UP
TSS NOW UP
TSS NOW UP
CAL TSS VERSION 1.0
NAME YOUR PERMANENT DIRECTORY
.PDLIST
TSS NOW UP
CAL TSS VERSION 1.0
NAME YOUR PERMANENT DIRECTORY
.GUEST
GIVE PASS WORD
.ENTER TENTATIVE NAME FOR TEMPORARY DIRECTORY
.DAVE
COMMAND PROCESSOR HERE
!BASIC
BASIC HERE
:I
XXXYYZZ
ERROR LINE NUMBER MISSING
LET J = 5
ERROR LINE NUMBER MISSING
<10 LET J = 5
:M.LET;E
M.LET↑
ERROR LINE NUMBER MISSING
:
?????
:Q
COMMAND PROCESSOR HERE
!LOGOUT
GOOD DAY
\*AL ISS VERSION 1.0
NAME YOUR PERMANENT DIRECTORY..PD\*L\*ST
GIVE PASS WORD..ROODT
PASS WORD NOT CONFIRMED
NAME YOUR PERMANENT DIRECTORY..EGL
UNEXPECTED FRETURN DURING LOGON, B7 = 006721
NAME YOUR PERMANENT DIRECTORY..PD\*L\*ST
GIVE PASS WORD..ROODT
ENTER TENTATIVE NAME FOR TEMPORARY DIRECTORY..EGL
COMMAND PROCESSOR HERE
!EDITOR
EDIT
I
1234567890

C\*rnt E in editor

?????
E
E

?????
P
E
T;P$
E
F
*GIVE FILE NAME EXPLICITLY
F ← NULL
COMMAND PROCESSOR HERE
!LOGOUT
GOOD DAY