Published: 06/03/66
(Supercedes: BG.4, 2/3/66)

Identification

Page Control
R.C.Daley; G.B.Krekeler

Purpose

Page control is the module which manipulates page tables and pages of segments which have been loaded by segment control. Page control has exclusive control over the operations associated with the reading and writing of pages. Page control also moves files (as determined by the multilevel storage algorithm).

Introduction

This section describes the primitives of page control and indicates the flow of control through page control. The descriptions are detailed enough to show the flow but are not intended to be the detailed implementation flow.

Page control provides a number of calls for use by the other members of the basic file system and for the missing-page-fault catcher. The primitives and the users of the primitives are listed below.

1.	Page restore	missing-page-fault catcher
2.	Read pages into core	segment control
3.	Remove a page from core	core control
4.	Truncate a segment	segment control
5.	Release pages of a segment	segment control
6.	Unload a segment	segment control
7.	Cleanup	segment control

Each of the primitives is discussed in more detail below. When the word page is used, it means a hyperpage. (The size of the hyperpage is determined from the hyperpage multiplier in the AST).

There are many places in the discussion where it is indicated that a process is blocked pending some event. The blocking and awakening mechanism for the basic file system is described in BG.18.01.

Page control can cause a process to go blocked for the following reasons.

Reason	Meaning	Event variables
ASTNA	The AST is interlocked and is not available to this process	none
ASTENA	An AST entry is interlocked and is not available to this process.	AST index
PSTNA	The PST is interlocked and is not available to this process	none
PSTENA	A PST entry is interlocked and is not available to this process.	PST index
DSTENA	A DST entry is interlocked and is not available to this process.	DST index
PTWNS	The page table word for a desired page has the page-out-of-service switch on.	AST index, page number
ASTEC	A process is waiting for a DIM cleanup to be completed so that it can deactivate a segment.	AST index

Page control attempts to awaken any concerned, blocked processes whenever it removes an interlock, sets the page-out-of-service switch off, places an AST entry in the inactive list, or deletes an entry from the AST or DST.

The primitives of page control are described first. The page control routines called getpage, removepagetable, and iodone are described following the discussion of the primitives.

Primitives

1. To restore a missing page in response to a missing-page fault, the following call is provided for the exclusive use of the missing-page fault catcher.

```
    call pagefault  (scuptr,dbr);
```

In this call, scuptr is a pointer to the information stored by an scu instruction when the fault occurs. dbr contains the descriptor base register value when the fault occurred. The following occurs.

a. Interrupts which would take the processor for a long period of time are masked and page control loops until it can set the page-table lock (PT) in the AST header. The scu information is interpreted to determine if the missing page is a descriptor segment page. If it is not a descriptor segment page, go to i.

b. If the DST entry pointed to by the page table word can be interlocked to this process, go to d.

c. Unlock PT; enable interrupts; block on DSTNA; when awakened, go to a.

d. Unlock PT; enable interrupts; increment the count of the number of pages in core in the DST entry; remove the DST entry interlock.

e. Call core control to assign a group of latched core (64 words).

f. Fill the assigned group with directed faults (segment faults).

g. Call core control to unlatch the group.

h. Return to the calling program.

i. Use the scu information to step through the descriptor segment to obtain the page table word corresponding to the missing page. If unable to obtain the page table word (descriptor segment page or page table not in core), unlock PT, enable interrupts, return to calling program.

j. If the page table word does not contain a page fault, unlock PT, enable interrupts, return to calling program.

k. If the AST entry pointed to by the page table word can be interlocked to this process, go to m.

l. Unlock PT; enable interrupts; block on ASTENA. When awakened; inhibit interrupts, loop until PT can be locked, go to i.

m. Unlock PT; enable interrupts.

n. Call getpage with blocksw ON.

o. Return to calling program.

2. To restore pages to core memory in response to declared anticipated usage, the following call is provided for the exclusive use of segment control.

call pcreadseg (index, pageno, count, blocksw);

In this call _index_ is the index of the AST entry for the segment, _pageno_ is the number of the first desired page, _count_ is the number of consecutive pages to be restored, and _blocksw_ is a switch which indicates whether the process should be blocked pending the arrival of the specified pages. The pcreadseg routine issues calls to getpage, where count determines the number of times getpage is called.

3. To remove pages from core memory, the following call is provided for the exclusive use of core control.

call removepage (index, pageno, dssw, returnarg);

In this call _index_ is the index of the AST entry or DST entry for the segment, _pageno_ is the number of the page to be removed, and _dssw_ indicates whether the page is a descriptor segment page or a non-descriptor segment page. _returnarg_ is used to tell core control whether immediate unassignment can be done.

When this call is made, the page to be removed has already been latched by core control. If the AST entry or DST entry is interlocked to some other process, a return is made to the calling program with an indication that the removepage call is being ignored. If the proper entry (AST or DST) is interlocked to this process, the following occurs.

a. If dssw if OFF, go to n.

b. Decrement the count of the number of pages in core; place the DST index, a missing page fault, and the page-out-of-service switch (ON) in the appropriate page table word.

c. Clear associative memories.

d. If the DST entry cannot be interlocked to this process, return to the calling program with an indication that the page can be unassigned. Otherwise, continue to e.

e. If the number of pages in core is not zero, remove the DST entry interlock and return to the calling program with an indication that the page can be unassigned. Otherwise, continue to f.

f. If the PST entry for this process can be interlocked go to h.

g. Block on PSTENA; when awakened, go to f.

h. Mask interrupts; loop until PT can be locked.

i. Call core control to unassign the descriptor segment page table.

j. Unlock PT; unmask interrupts; clear associative memories.

k. Adjust the pointers in the DST entry chain (after interlocking the appropriate DST entries); remove the DST entry.

l. Remove PST entry interlock.

m. Return to calling program with an indication that the page can be unassigned.

n. Note - the page is not a descriptor segment page. Place the AST index, a missing-page fault, and the page-out-of-service switch (ON) into the page table word; clear associative memories. If the page-has-been-modified switch is ON, go to p.

o. If the move switch is ON and the page-has-been-moved switch is ON, or if the move switch is OFF, go to x.

p. Decrement the count of the number of pages in core; increment the count of the number of outstanding I/O requests; set the page-must-be-read switch ON. If the count of the number of pages in core is not zero, go to s.

q. Test the page table words to determine if there is a page which has not yet been moved. If all pages have been moved or moving is not required, go to s.

r. Call getpage with blocksw OFF to read a page which has not yet been moved.

s. If the move switch is ON,the page is to be written into the move file. Go to v.

t. Remove the AST entry interlock. Call I/O queue control to write the page into the execution file.

u. Return to the calling program

v. Set the page-has-been-moved switch ON; remove the AST entry interlock. Call I/O queue control to write the page into the move file.

w. Return to the calling program.

x. If the segment-kill switch is ON, return to the calling program. Otherwise, decrement the count of the number of pages in core. If this results in a count of zero go to a'.

y. Remove the AST entry interlock.

z. Return to the calling program with an indication that the page can be unassigned.

a' If the number of outstanding I/O requests is not zero, or if the page-table-hold switch is ON, remove the AST entry interlock and go to c'.

b' Call removepagetable.

c' Return to calling program with an indication that the page can be unassigned.

Note that when a write request is given, page control returns to core control before the request is completed. When the request is completed, page control is called by the DIM at the "iodone" entry. The core allocated for the page is then unassigned by calling core control, and the page-out-of-service switch is turned OFF.

4. To truncate an active segment to a shorter length, the following call is provided for the exclusive use of segment control.

```
call pctruncate (index, pageno);
```

If this call index is the index of the AST entry for the segment, and pageno is the page number of the first page to be discarded.

When this call is issued, the AST entry has alreadv been interlocked to the process by segment control. The page table words for the pages between the new length and the current length of the segment are interrogated one at a time. If the page-out-of-service switch is ON, this process is blocked on PTWNS. When the page-out-of-service switch is OFF, a test is made to see if the page is in core. If it is not in core, the page-out-of-service switch is set ON and the next page table word is interrogated.

If a page is in core, the AST index, a missing-page-fault, and the page-out-of-service switch (ON) are placed into the page table word. All associative memories are then cleared. The page is unassigned by calling core control. The count of the number of pages in core is decremented and the next page table word is interrogated.

After the affected area of the page table has been scanned, the count of the number of outstanding I/O requests is incremented. The following occurs.

a. If the count of the number of pages in core is not zero go to d.

b. If the move switch is OFF or if all pages have already been moved, go to d.

c. Call getpage with blocksw OFF to read a page which has not yet been moved.

d. Call I/O queue control to truncate the file, indicating the setting of the move switch. If the move switch is ON, the move file is truncated. Otherwise, the execution file is truncated.

e. Correct the current segment length in the AST; remove the AST entry interlock.

f. Place segment faults in all descriptor words which reference this segment and are in core. This forces the boundary field to be recalculated (when the fault occurs) for those users who do not have the append permit.

g. Return to calling program.

When, at a later time, the truncate request is completed, page control is called by the DIM at the iodone entry. The page-out-of-service switch and the page-must-be-read switch are turned off for all pages above the current segment length (formerly the new length).

5. To release a portion of a segment from core memory, the following call is provided for the exclusive use of segment control.

```
call pcfreecore (index, pageno, count);
```

In this call index is the index of the AST entry for the segment, pageno is the number of the first page to be released, and count is the number of consecutive pages to be released. The AST entry has already been interlocked to This process by segment control. The page table words corresponding to the area to be released are interrogated one at a time. If a page is not in service or is not in core, no action is taken. If a page is in core and in service, the group is latched by a call to core control. The page is removed in the manner described in the removepage call except that the AST entry interlock is not removed. This interlock is removed only after all affected page table words have been interrogated.

6. To force a segment to become unloaded (remove its page table), the following call is provided for the exclusive use of segment control.

```
call segunload (index, deactivatesw);
```

In this call index is the index of the AST entry for the segment. deactivatesw is a switch which indicates whether the segment is being made inactive. If deactivatesw is ON, segment control is deactivating the segment and the AST entry interlock will remain locked upon the return to segment control.

Segment control has already interlocked the AST entry when this call is made, and it remains locked until every page of the segment has been released and the page table has been removed.

If the AST entry indicates that the segment is already unloaded, an immediate return is made. If the segment is loaded, the segment-kill switch is set ON in the AST entry, and segment faults are stored in the descriptor segment words which refer to this segment. Associative memories are then cleared.

There are two scans of the page table words. In the initial scan if a page is in core and the page-out-of-service switch is OFF, the page is latched by calling core control. The page is then removed in the manner discussed in the removepage call except that the AST entry interlock is not removed. If the page-out-of-service switch is ON, the next page table word is interrogated.

After the entire page table has been interrogated in the manner described above, the second scan is made. In this pass through the page table, the process is blocked until the page corresponding to a page table word is removed from core (if necessary). The next page table word is not interrogated until all operations are completed and the page-out-of-service switch is off.

At the completion of the second scan, the segment-kill switch is turned off. A call to removepagetable is then made. After the page table has been removed, a return is made to the calling program.

7. To notify the DIM that it can clean up its core-resident history of a file, the following call is provided for the exclusive use of segment control.

```
     call pcclean (index);
```

In this call <u>index</u> is the index of the AST entry for the segment. Segment control has already locked the AST entry before making this call.

Page control increments the count of the number of outstanding I/O requests and calls I/O queue control at the cleanup entry and returns to the calling program.

When, at a later time, the request is completed, page control is called at by the DIM at the iodone entry. The count of the number of I/O requests is decremented, and the process blocked on ASTEC is awakened.

Page Control Utility Routines

The routines getpage, removepagetable, and iodone have been referred to in the previous section. The first two routines are called by page control. iodone is called by the DIM after it has completed an I/O request. These routines are discussed in detail below.

1. Getpage

Page control uses the getpage routine to retrieve a page.

```
call getpage (index, pageno, blocksw);
```

index is the index of the AST entry for the segment, pageno is the number of the desired page, and blocksw is a switch which indicates whether the process should be blocked pending the arrival of the specified page.

If the AST entry specified by index is interlocked to some other process, the current process is blocked on event ASTENA. Once the entry is available and is interlocked to this process, the pageno in the call and the page table pointer in the AST entry are used to obtain the page table word corresponding to the desired page. If the page is in core, and immediate return is made. If the page-out-of-service switch is ON, the AST entry interlock is removed and a return is made if blocksw is OFF. If the page-out-of-service switch is ON and blocksw is ON, (the AST entry interlock has been removed) the process is blocked on PTWNS. After being awakened, the process is treated as if getpage were just called.

If the page-out-of-service switch is OFF, the following occurs.

a. Increment the count of the number of pages in core; set the page-out-of-service switch ON; remove the AST entry interlock.

b. Call core control to assign a latched page.

c. If the page-must-be-read switch is ON, go to e. Otherwise, fill the assigned core with zeros and prepare a new page table word, remove the fault, store the page address, turn the page-out-of-service switch OFF. The page table word is stored with one instruction and the AST entry interlock is not disturbed or interrogated.

d. Return to calling program.

e. Increment the count of the number of I/O requests.

f. Call I/O queue control to read from the appropriate file. If the move switch is ON and the page-has-been-moved switch is ON, the move file is read. Otherwise, the execution file is read.

g. If blocksw is OFF, return to the calling program. Otherwise, block the process until the I/O is completed; when awakened, return to calling program.

2. Removepagetable

Page control uses the following call to remove a page table.

call removepagetable (index, deactivatesw);

index is the index of the AST entry for the segment to be unloaded, and deactivatesw is a switch which indicates whether the AST entry interlock is to be removed. The AST entry is already locked to this process when the call is received.

All descriptor words which point to the page table to be removed must be set with a segment fault. From the process numbers in the AST entry trailers, page control obtains the information necessary to track down the DST entries. Each process number is used to gain access to the PST, and the PST entries point to the DST entry chains. The segment number in the AST entry trailers indicates to which descriptor segment words the segment fault is to be stored. After the faults are stored, all associative memories are cleared.

The segment-loaded switch is turned off, and core control is called to unassign the page table. The entire AST is interlocked to this process (or this process is blocked on event ASTNA), and all of the AST entry trailers are released. If the AST entry-hold switch is on, all interlocks are removed and a return is made. If the AST entry-hold switch is ON, the deactivatesw is interrogated. If deactivatesw is OFF, all interlocks are removed and a return is made to the calling program.

If the entry-hold switch is OFF, and the number of inferior AST entries is zero, the AST entry is placed in the list of candidates for removal after the link-forward and link-backward pointers are established. If deactivatesw is OFF, all interlocks are removed. If deactivatesw is ON, all interlocks except the AST entry lock are removed. A return is then made to the calling program.

3. Iodone

Whenever an I/O request initiated by page control is completed, the DIM calls page control with the following call.

```
     call iodone (stateword);
```

This call may be issued at any time and will be handled by the process which is running at the time it is issued. In this call stateword provides the information which was given when the I/O request was initiated. It defines the operation (read, write, truncate, cleanup), the AST entry index, the core address of the group, the initial page number, and an indication of any errors found while processing the request.

If the operation is read, a new page table word is prepared and stored - the address of the page is indicated, the page fault is removed, and the page-out-of-service switch is turned OFF. A call to notify is made to awaken any process which may be blocked waiting for the page to be in service. If the segment-hold switch is OFF, core control is called to unlatch the page. A return is then made to the DIM.

If the operation is write, the core allocated for the page is unassigned by a call to core control. The page-out-of-service switch is set OFF. If the segment-kill switch is ON, a return to the calling program (DIM) is made. If the segment-kill switch is OFF, the process attempts to interlock the AST entry. It is blocked on ASTENA until the AST entry is available. Once the AST entry is interlocked to this process, the count of the number of outstanding I/O requests is decremented. IF this new count is not zero, the AST entry interlock is removed and a return is made to the calling program.

If the number of outstanding I/O requests is zero, the count of the number of pages in core is checked. If this count is non-zero, the AST entry interlock is removed and a return is made. If both counts are zero, and the page-table-hold switch is OFF, a call to removepagetable is made. A return to the calling program (DIM) follows the return from removepagetable. If the page-table-hold switch is ON, the AST entry interlock is removed and return is made to the DIM.

If the operation is truncate, the AST entry is interlocked or the process is blocked on ASTENA until the AST entry is available. The page-out-of-service switch and the page-must-be-read switch are turned OFF for all pages above the current segment length. The number of outstanding I/O requests is decremented and tested, in the manner described in the write operation above, to see if the segment should be unloaded.

If the operation is cleanup, the AST entry is already locked (to some other process). The count of outstanding I/O requests is decremented and the process blocked on ASTEC is awakened. A return is made to the DIM.

PL/I Declarations

The PL/I declarations for the parameters to page control primitives are given below.

```
     declare

     blocksw bit (1),            /* block switch */
     count bit (8),              /* number of pages */
     deactivatesw bit (1),       /* deactivate switch */
     dssw bit (1),               /* descriptor segment switch */
     index bit (18),             /* AST or DST index */
     pageno bit (8),             /* page number */
     scuptr ptr,                 /* pointer to */
     stateword bit (72);         /* iodone information */
```