TO: MSPM Distribution
FROM: M. A. Padlipsky
SUBJ: BG.10.00
DATE: 02/14/68

The attached revision of BG.10.00 reflects the current design of the File System DIM.

Published: 02/14/68
(Supersedes: BG.10.00, 06/27/66)

Identification

Overview of the File System DIM
L. Whitehead

Purpose

The File System device interface module (DIM) is the interface between the Multics File System and those devices used for on-line storage of files. These devices currently include the firehose drum, the disc and the RACE. This use of the term DIM should not be confused with the same term still used in some (obsolete) I/O system documentation. Note that the File System DIM is part of the wired-down hardcore supervisor; hence, it and its data bases are shared by all Multics processes.

Overview

The DIM serves as an input/output manager for the file system - it performs the functions of device storage management and supervision of device I/O. There are two basic parts of the DIM, which perform the tasks of

1. overall management
2. device control

Device supervision is handled by a device control module for each device (e.g., drum control, disk control, etc.). Overall management is handled by several modules which can process several requests for various devices (simultaneously) if several processors are available. These modules are discussed below.

The DIM has two entry points:

1. dim$file_io; this entry is used to make a request for I/O.
2. dim$run; this entry is to give the DIM and the several device control modules the opportunity to run and accomplish any outstanding work. This call is used when Multics has no other useful work that can be done. The entry is also used whenever an interrupt occurs on an associated device.

Two major calls are made by the DIM to routines exterior to itself.

1. Iodone. A call is made to "iodone" to signal the completion of each request.

2. Multilevel. A call to multilevel move control (BH.1.01) is made when a device becomes overcrowded. This is a request to move some of the files from the particular device to another device. The actual I/O is performed by the DIM, but decisions concerning what file to move are made outside the DIM.

The exact calling sequences and parameter declarations for calls to the DIM are given in BG.10.07.

Structure of a File

One job of the DIM is the management of files on external storage. To the DIM, a file is a linear array of information which is described in terms of record numbers where a record is 64 words. Figure 1 describes a file. In general a file is not stored as a continuous unit in core or on a device, so Figure 1 is actually an abstraction of a file. Requests to the DIM refer to the starting record number and a number of contiguous records, thus describing the portion

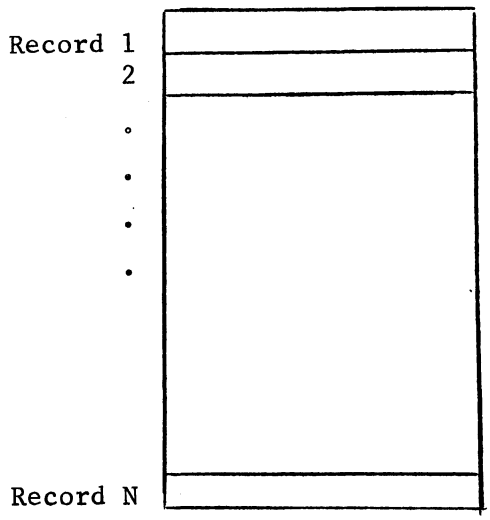

Figure 1. A File as Seen by the DIM

of the file to be processed. All requests are oriented toward 64 word records which must be stored contiguously in core.

Types of Requests

Three types of requests may be made of the DIM by the file system.

1. Read. A request to read records from some device into core.

2. Write. A request to write records from core memory onto some device.

3. Delete. A request to modify the description of a file so that certain records are no longer considered to exist for a file.

For example, a request to delete records 13 through 20 would cause a file that was 20 records long to become only 12 records long. The exact call parameters are given in BG.10.01.

Restrictions Related to Requests

Two important restrictions should be noted by users of the DIM.

1. Requests to the DIM are not necessarily processed on a first-in, first-out basis, but in an order appropriate to the optimum strategy for each device.

2. The DIM cannot handle two requests for the same record of a file. For example, a request for records 5-10 and another request for records 8-12 would not be legal - assuming the requests were for the same file. Checking to prevent request's conflicts is done in page control (see BG.4).

Obviously the two restrictions are related. They are stated separately to emphasize their meaning to the user of the DIM.

Definitions of Some Terms

Several terms require careful definition because of their importance in discussions of the DIM.

All record-keeping related to maintaining files on on-line devices is done by the DIM. A request for a file consists of a description of the part of the file to be serviced and a pointer to unique file information (the filemap). The DIM must determine where the file is on the device or where to place the file if it is being written for the first time. Files are of varying lengths and their lengths may change frequently; therefore they are stored in random locations on a device. The device addresses of the various parts of a file are maintained in a table called a filemap. There is one filemap for each file. The filemap is kept in the file directory when the file is inactive and is placed in core in the AST while the file is active. The AST is wired down, thus the filemap is in core at all times and available to the DIM while the file is active. The filemap is initialized and its length established and changed by calls to service routines provided in the DIM. These routines are discussed in BG.10.03.

Device addresses in the filemap describe a unit of storage called a hyper-record. A hyper-record is a group of contiguous records on a device. The hyper-record size is a constant for each device. The concept of hyper-record was developed to reduce the number of addresses needed to describe a file. As an example, the hyper-record size on the drum might be four, implying that four records can be described by the address of the first record of the group. The number of records in a hyper-record is called blocksize in the DIM. Figure 2 illustrates the concept of hyper-record with a blocksize of four.

The concept of hyper-record is not related to the concept of hyper-page. There need be no relation between them. If the hyper-page size and hyper-record size were always equal, I/O would be more efficient; however the DIM operates correctly if there is no relation between the two sizes. Requests to the DIM are in terms of records, and all necessary translation is made by the DIM between record numbers and hyper-records.

All addresses in the filemap are hyper-record addresses. A hyper-record address

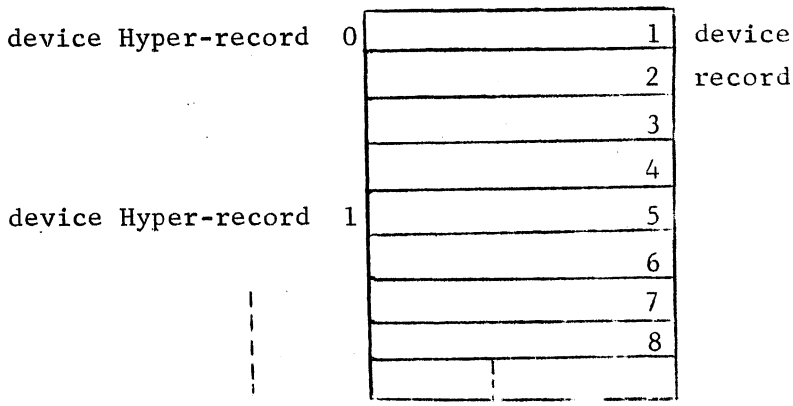

Figure 2. Hyper-records

is defined to be the absolute device address of the first record of a hyper-record, divided by blocksize. Thus the hyper-record address drops the redundant trailing zeros. For example, given a blocksize of 8 and a device address of 34040(8), the hyper-record address is 3404(8).

A hyper-record address is defined to be 18 bits in length. This requires that the blocksize for each device must be large enough that the quotient defined above can be held in 18 bits.

In order to be able to write new files on a device, records must be kept of device addresses which are not being used. The housekeeping of addresses of unused hyper-records is done by the free storage module of the DIM. Tables used by free storage are defined in BG.10.04.

In summary, the device addresses of the parts of a file are maintained by the DIM in the filemap. All device addresses are carried as hyper-record address. The DIM makes all necessary translations between record numbers of a file and the associated hyper-record. The concept of hyper-record is transparent to the user and to the file system (except the DIM, obviously).

Some Important Data Bases

The general management portion of the DIM utilizes two data bases - the filemap and the i/o queue (ioq). Each device control module maintains several data bases but their content and use is device dependent; therefore for information on these the reader is referred to the BG section for each device.

The filemap is initialized and maintained by the DIM. It must be initialized before a newly created file can be referenced. The use and content of the filemap were discussed above. The first word of each filemap is a lock of the sort manipulated by the standard file system locking routines. (BG.15.02)

The filemap is initialized with a special unused address which indicates that the hyper-records have never been written (the special address is discussed in more detail in BG.10.02). Until one of the records in a hyper-record is written, this special address indicates that the contents of these records of a file can be assumed to be zero. The unused records then take no space on the device. Figure 3 indicates the structure of the filemap.

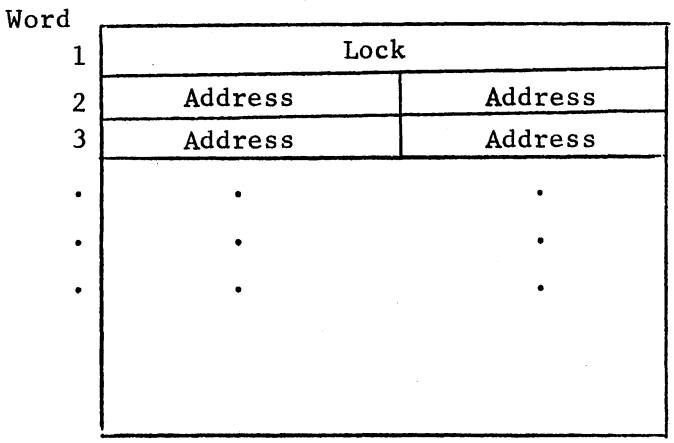

Figure 3. A Filemap

The ioq is a data base maintained internal to the DIM. When a request enters the DIM, an ioq entry is obtained and data which must be maintained until the request is completed is placed in the entry. The DIM command module (see module descriptions below) places additional information in the entry as parameters for device control. Device control maintains records in the ioq entry while I/O for the request is being accomplished. The declaration of the ioq can be found in BG.10.08. A discussion of the contents occurs in the sections of BG.10 relative to modules where the items are created and used.

Entries in the ioq are linked in a "free list" when they are unused. They are not linked, from the time request data is inserted until all I/O for the request is completed. When all I/O for a request is completed, the associated entry is linked in the "done list" until program control in the DIM reaches the point that page control can be notified of the completion.

Strategy in the DIM

One of the major philosophies in the design of the DIM was to minimize the number of device interrupts required. Details of the workings of the parts of the DIM are given below and in other BG sections. The intent of these paragraphs is to give the reader insight into the global strategy.

When I/O on a device is completed, it is conventional to generate an interrupt for interpretation by the associated software. Since Multics utilizes a distributed system concept, a process swap must occur to permit this associated software to run (see BJ sections on process swaps, wake ups, etc.). It is desirable to reduce the overhead of process swaps by reducing the number of interrupts generated. This is particularly true on devices with a high transfer rate which need frequent "feeding".

The DIM and its associated device control modules attempt to maintain a high level of throughput with a small number of interrupts. This is accomplished by maintaining relatively long hardware queues (of commands) for each device and servicing these queues each time the DIM runs. If the DIM is called frequently, few if any interrupts will occur.

More specifically, calls to dim$file_io generate at least one call to one device control. Before the return to the file system is made, a call is made to each device control at a $run entry to permit them to do any outstanding work. On calls to dim$run a call is made to each device control at the $run entry. The $run entry permits work such as posting all completed commands and issuing any commands which might have been waiting because of lack of room in the hardware queue.

The details in the various device controls are different and the reader should consult the appropriate BG sections for each device.

Major Modules of the DIM

A list of the major parts of the DIM and their associated BG section follows.

1. DIM Driver (BG.10.01)
2. DIM Command Processor (BG.10.02)
3. Device Free Storage Management (BG.10.04)
4. Drum Control Module (BG.11)
5. Disk Control Module (BG.12)
6. Race Control Module (BG.13)
7. Functions Provided for File System Use (BG.10.06)
8. Service Routines for Use by the DIM (BG.10.03)

A brief discussion of the first six of the categories is given.

DIM Control Module

This module receives calls to the DIM, analyzes requests and directs control to various pats of the DIM. On calls to dim $ file_io a check is made to see if the associated device is operating and a call is made to the appropriate entry point in the command module (i.e., READ, WRITE, DELETE). On return from the command module, a call is made to the various device control modules at the entry device_control$run to permit housekeeping before return to page control.

On call to dim $ run, calls are made to the various device control modules to permit any backlog of I/O to be serviced.

At both entries, before returning, DIM control calls a routine (service_done_list) which calls iodone once for each request that has been completed.

DIM Command Module

This module has one entry point for each type of DIM request - read, write, delete. The various requests are analyzed and calls to device control are made to effect the necessary I/O. One call to device control is generated for each

hyper-record that must be referenced. If records of a file are written for the first time, calls are made to free storage to obtain device addresses. If records are being deleted, calls are made to free storage to return device addresses. If a read request occurs for a record that has never been written, a call is made to a routine write zeros in the associated core area.

Free Storage Module

This module is responsible for keeping a record of the unused (free) storage for each device under the file system. The routine operates on tables referenced by an array of pointers which are indexed on the device identification number, so that free storage for all devices is maintained by one routine.

The table of free addresses is maintained on the associated device and is called the free storage map. A small portion of this map for each device is maintained in core for fast reference.

The free storage module is also responsible for monitoring the usage level of each device and alerting the multilevel management system when a device is overloaded.

Device Control Switch

Calls to device control are made with device id (did) as one of the parameters. The device control switch receives all calls to device control and calls drum control, disk control or race control as required. No DIM module or file system module differentiates between devices until this routine is reached.

Device Control Modules

Each device has a device control module which is responsible for effecting device I/O and handling device errors. The major responsibility of this module is efficient I/O to the device. The drum is connected directly to memory (i.e., is an active unit). For this reason, drum control talks directly to the drum. All other devices are connected to the GIOC and their control modules interface with the GIM for I/O.

Posting of completed commands to the ioq is done for device control by a DIM service routine. This technique permits each device control to be essentially separate from the rest of the DIM and to work with the DIM through a simple interface.