TO: MSPM Distribution
FROM: C.A. Cushing
SUBJ: BG.8.00 and BG.8.01
DATE: January 27, 1967

Section BG.8.00 has been rewritten to give an overview of the directory control module by explaining its interaction with the general user and the other modules (namely Segment Control) of the Basic File System.

Many minor revisions were made to BG.8.01, particularly in the calls, in order to make them agree with the actual implementation of the module.

Published: 01/27/67
(Supersedes: BG.8.00, 5/10/66)

Identification

Directory Control
C. A. Cushing

Purpose

Directory Control (DC) is a module made up of two sub-sections. One is the directory supervisor (DS), the other is the directory maintainer (DM). The function of DC is to maintain the special segments in the tree hierarchy called directories and make selected contents of these segments available to the general user. DS provides this user with an interface to the file system and DM provides DC with certain utility routines. The only way for a user to manipulate directories is through directory control.

Interaction with General Users

Directory data bases are totally inaccessible to a user. However, directory control as part of the hard-core supervisor is able to access these protected segments for the user. For example, if a user procedure wishes to see the contents of a directory entry, directory control copies selected items from the entry into a structure provided by this procedure.

A list of all such items returned from the contents of a directory entry follows (see BG.7.00, Directory Data Base, for a complete description of each item).

for branches:

directory switch	(dirsw)
unique identifier	(uid)
date/times segment last used,	(dtd)
last modified,	(dtm)
last dumped	(dtd)
date/time branch last modified	(dtbm)
retention date	(rd)
options switches	(optsw)
usage status of segment	(usage)
count of users	(usagect)
no more users switch	(nomore)
maximum length of segment	(ml)
current length of segment	(cl)
number of bits in segment	(bc)
account number	(acct)

high and low multilevel limits(hlim, llim)
apparent mode of uer (amode)
protection list
gate list
names of branch

for links:

unique identifier (uid)
date/times link last used, (dtu)
last modified, (dtm)
last dumped (dtd)
path name of entry to which this is a link
names of link

Interaction with Segment Control

The primitives to the supervisory section of directory control expect a symbolic name (path name or tree name) of the directory whose contents are to be manipulated. This name is then passed to the maintainer whose function is to find a specified entry in this directory.

The directory maintainer calls segment control to get the segment pointer associated with the given directory path name. If the segment pointer is returned, i.e., the directory with the given path name is known to this process, then the directory maintainer uses this segment pointer (ITS) pair, pointing to the base of the directory segment to read this directory in search of the given entry.

When directory maintainer calls segment control to get the segment pointer of a directory given its path name, this name may not exist in the known segment table (KST). If this is the case, the segment is not known to the process and a segment pointer cannot be returned immediately. A circular recursive pattern of calls is then established between segment control, directory supervisor and directory maintainer in an attempt to find the given directory in the hierarchy and to make it known to the current process. After each complete turn, the original path name is broken down to get the name of the next superior directory (e.g., "(root)>a>b" becomes "(root)>a" then "(root)" until one of these directories is found in the KST or the root of the hierarchy (which is known to the directory supervisor) is reached. In either case, the circular process can then begin to unwind.

During the unwinding process each directory beginning with the first one found in the KST or the root, is searched for the branch which points to that inferior directory listed next in the original path name (e.g. if "(root)>a" is known, it is searched for "b"). When this branch is found, the information in it is used to make the directory to which it points a known segment. Now that this directory is known, it can be searched for the branch pointing to the next inferior directory and so on until the original directory is made known, and its segment pointer returned to directory maintainer.

The root of the hierarchy is known by directory supervisor and need never specifically be named by the general user when he writes a path name. Generally, a path name of a directory given to directory supervisor is relative to the root, i.e., the path name begins with ">" (see BX.8 for path-naming conventions). In this case, directory supervisor prefixes the name of the root to the given path name. In actual implementation, two directory names which are set in the process data segment are prefixed to the given path name (see BG.8.04, "set_base_dir").

In the circular process mentioned above the last directory name in the path name, i.e., the root, has a branch which is known to directory supervisor. Circling as far as the root is usually done once, the first time a directory supervisor primitive is invoked by the process. It then becomes a known segment and stays known until all segments inferior to it (the root) become unknown.

Hashing

The method of searching a directory for an entry, given its symbolic entry name, is by hash coding (for more detail, see hash primitive in BG.8.01). There is a hash formula which is applied to the given name to produce an integer which points to a location in a hash table contained in the directory. Each location in this table contains a slot number which is an index into another table, called a slot table, within the directory. Each slot in this slot table contains a pointer to an entry in the directory. There is a location in the hash table for each name of every entry in the directory. Since a single entry may have more than one name, there may exist more than one occurrence of the same slot number in the hash table which points (through the slot table) to the same entry. Once a slot number is taken from the hash table, it is used to obtain the names of the entry to which it points in order to see if one of the names matches the given name. If there is a match, then the entry is found. If not,

then a second slot number from the next location in the hash table is used to test another entry. Successive slot numbers are extracted from the hash table until the desired entry is found or the search is terminated either by an empty hash table location or in the pathological case, by scanning every entry in the hash table, i.e., table completely full. Depending upon the hash formula and the number of locations in the table, the number of tries should be small.