Published: 05/11/67
(Supersedes: BX.8.11, 12/13/66
BX.8.05, 01/14/65)

Identification

Get an outline of the tree structure
map_dir
E. Q. Bjorkman

Purpose

Map_dir gives the user an outline of the directory entries in a section of the file system hierarchy.

Usage

```
     map_dir -path- -ln-
```

map_dir prints the names of all directories specified in the section of the file system hierarchy (see comment), starting with path or with the working directory if path is null.

If ln is specified, printing stops with the directories ln levels inferior to the starting directory.

If path (a path name) is specified, the starting directory is the directory defined by path.

Comments

The user issuing the map_dir command must be permitted to read the directory defined by path and all directories ln levels inferior to it. If there exists directories less than ln levels inferior to the starting directory which the user cannot read, an error message is printed if the brief option is off. Printing is ended at that point for those directories and the directories inferior to them. Printing is continued for directories at superior levels and directories which are not reached via those unreadable directories.

Implementation

```
map_dir -path- -ln-
```

map_dir performs its function of outlining a section of the file system hierarchy by making ln calls to the routine, maplevel (see BY.2.06); that is, one call for each level to be listed. If ln is null, map_dir calls maplevel until all levels inferior to path have been listed, which map_dir finally discovers when it reaches directories containing no directory branches. Figure 1 illustrates levels in the file system hierarchy.

If path is null the path name of the current working directory is stored in path by calling the procedure, wdir (BY.2.05). The path name of the starting directory (path or the working directory) is written in the output stream.

The method followed for each level to be outlined is as follows: The number of the current level to be outlined is stored in nl and written in the output stream. Depth_flag is set to zero and the call

```
call maplevel (path, nl, depth_flag);
```

is made. The procedure maplevel is described in BY.2.06. Declarations in maplevel for the arguments are

```
dcl path char(*)/varying, nl fixed bin(17),
    depth_flag bit(1);
```

Maplevel formats and prints the directory entries in all directories which are nl levels inferior to the starting directory specified by path. Maplevel reaches levels inferior to the starting directory by scanning the starting directory for directory branches and then calling itself recursively. When a directory is reached which is nl levels inferior to the starting directory, maplevel sets the depth_flag to one to indicate that the specified level has been reached.

If, when maplevel returns to map_dir, the depth_flag is still zero, map_dir writes the message "end of tree structure reached" in the output stream and returns control to its caller. Otherwise the value of nl is compared with ln. If ln is null, or nl is less than ln, nl is incremented by one and the above procedure repeated. If nl equals ln (i.e., all levels specified have been mapped) control is returned to the caller of map_dir.

The possible errors for map_dir are the same as those for maplevel (see BY.2.06); e.g., the read attribute is not on for some directory in the line of execution.

If "map_dir >A>B" is typed at command level, the following could appear on the user's console:

```
                 >A>B

LEVEL            0

   DIRECTORY     B

   ENTRIES

   utility_progs

   ge, ge645_progs

LEVEL            1

   DIRECTORY     utility_progs

   ENTRIES

   DIRECTORY     ge*

   ENTRIES

LEVEL            2

   end of tree structure reached
```

*If a directory is known by more than one name, only the first name is printed following the heading, DIRECTORY.

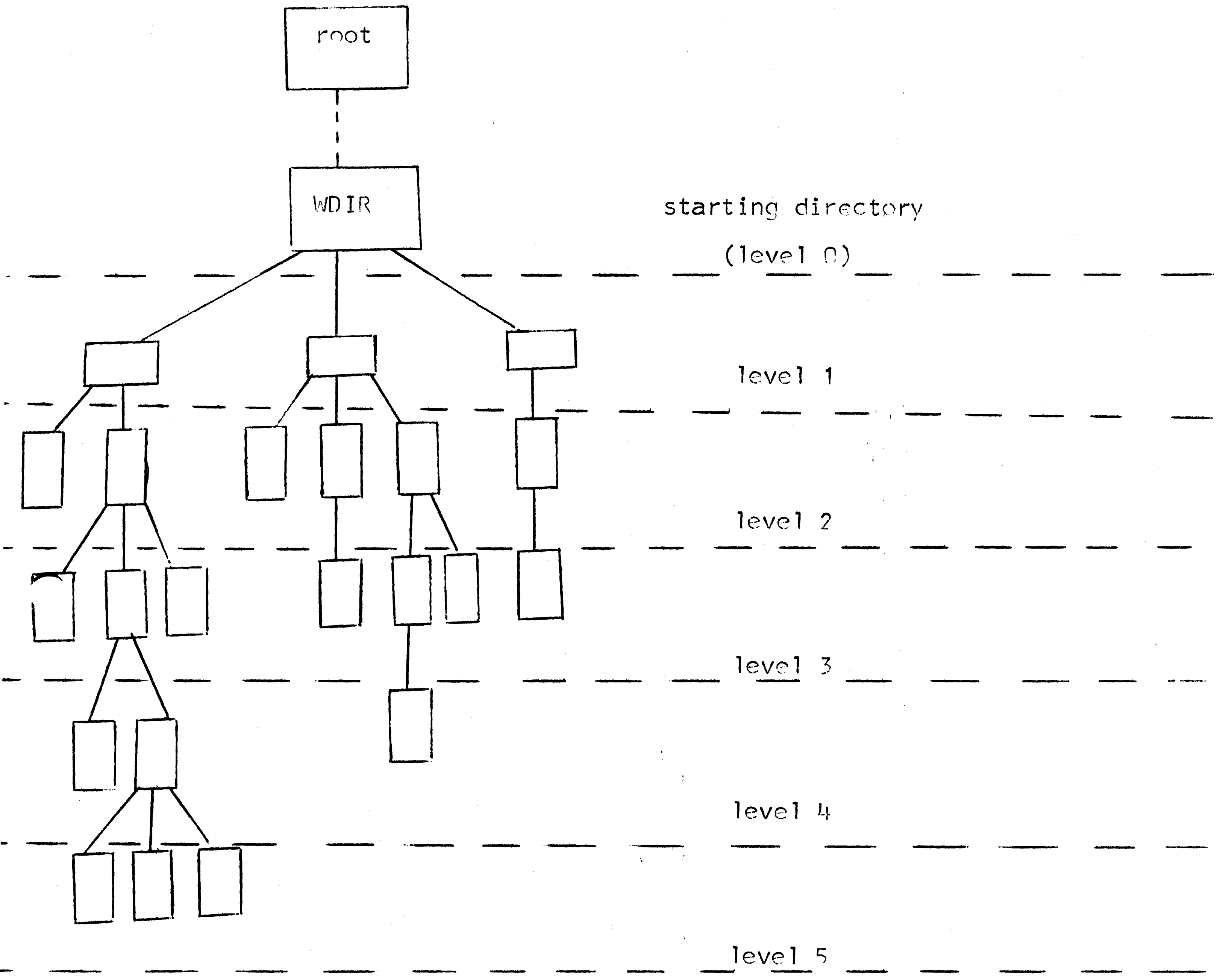

Figure 1: A hierarchy of directories showing levels.
Non-directory branches are not shown.