CSCI 4320 (Principles of Operating Systems), Fall 2010

Homework 6

Credit: 20 points.

1 Reading

Be sure you have read Chapter 4.

2 Problems

Answer the following questions. You may write out your answers by hand or using a word processor or other program, but please submit hard copy, either in class or in my mailbox in the department office.

1. (5 points) Consider a digital camera that records photographs in some non-volatile storage medium (e.g., flash memory). Photographs are recorded in sequence until the medium is full; at that point, the photographs are transferred to a hard disk and the camera's storage is cleared. If you were implementing a file system for the camera's storage, what strategy would you use for file allocation (contiguous, linked-list, etc.) and why? Notice that this camera does not have the ability to delete photographs from its storage one at a time, so your file system does not need to support that. (It's probably best to think of this as a somewhat hypothetical problem, using only the description supplied, rather than trying to extrapolate from your experience with actual cameras.)

2. (5 points) The textbook describes more than one strategy for keeping track of free blocks in a file system (free blocks, bitmaps, and FATs). All of these strategies rely on information that is kept both on disk and in memory, sometimes with the most-current information only in memory. What would happen if whatever data structure is used to keep track of free blocks was lost or damaged because of a system crash — is there a way to recover, or do you have to just reformat the disk and hope you backed up any really important files? Answer separately for MS-DOS FAT-16 (which uses a FAT) and UNIX V7 filesystems (which uses one of the other strategies).

3. (5 points) Linux includes code to access several types of Windows filesystems, including FAT-32. So on a system where one of the disk partitions holds a FAT-32 filesystem, one can configure Linux to access this filesystem through pathname `/windows/fat` for example. However, all the files in `/windows/fat` appear to be owned by user `root`, and attempts to change their ownership (with the `chown` command) fail with an error message "Operation not permitted". What's wrong?

4. (5 points) Consider a UNIX filesystem (as described in section 4.5.3) in which each i-node contains 10 direct entries, one single-indirect entry, one double-indirect entry, and one triple-indirect entry. If a block is 1KB (1024 bytes) and a disk addresses is 4 bytes, what is the maximum file size, in KB? (*Hint:* Use the blocksize and size of disk addresses to determine how many entries each indirect block contain.)

3 Programming Problems

For extra credit, do one or more of the following programming problems. You will end up with at least one code file per problem. Submit your program source (and any other needed files) by sending mail to `bmassing@cs.trinity.edu`, with each file as an attachment. Please use a subject line that mentions the course number and the assignment (e.g., "csci 4320 homework 6"). You can develop your programs on any system that provides the needed functionality, but I will test them on one of the department's Fedora Linux machines, so you should probably make sure they work in that environment before turning them in.

1. (Optional — up to 5 extra-credit points) Write a program that given a directory D, blocksize B, and maximum number of blocks M as command-line arguments prints out how many files in D and its subdirectories are of size B or less, how many are of size between B and $2B$, etc., up to size MB. (This might be useful in getting an idea of what size files are typical, so if you had a choice of blocksize you would know what choice might make the most sense.) Include directories and symbolic links (but count the size of the link and not the file/directory it links to). Also turn in output of running this program on your home directory in `/users` with B and M as below.

   Here is sample output for running the program with $D =$ `/lib`, $B = 4096$, and $M = 20$, on Xena00:

```
Results for directory /lib with blocksize 4096:

                1553 files of size          1 blocks
                1985 files of size          2 blocks
                1296 files of size          3 blocks
                 679 files of size          4 blocks
                 443 files of size          5 blocks
                 336 files of size          6 blocks
                 265 files of size          7 blocks
                 243 files of size          8 blocks
                 175 files of size          9 blocks
                 157 files of size         10 blocks
                 107 files of size         11 blocks
                  56 files of size         12 blocks
                  74 files of size         13 blocks
                  65 files of size         14 blocks
                  50 files of size         15 blocks
                  31 files of size         16 blocks
                  24 files of size         17 blocks
                  30 files of size         18 blocks
                  43 files of size         19 blocks
                  51 files of size         20 blocks
                 491 files of size         21 blocks or more
```

   (Of course, you won't be able to examine files in directories you don't have access to. Just print error messages for files/directories you can't access.)

   To get maximum points, your program should be in C or C++ and make no use of system commands such as `ls`. (You can use another language, or even write a shell script, but you will get fewer points.) Library functions `opendir`, `readdir`, and `lstat` will probably be helpful. You might also be interested in functions `chdir` and `strerror`. These functions are

described by man pages. (Remember also that `man -a foo` gives all man pages for `foo`. This can be helpful if there is both a command `foo` and a functionfoo.)

Here is some starter code[1] that parses/checks the command-line arguments.

2. (Optional — up to 5 extra-credit points) Write a program that given a directory D as a command-line argument prints all the "broken" symbolic links in D or any of its subdirectories — that is, symbolic links that point to a file that doesn't exist. Here is sample output for running the program with D = `/users/bmassing/Local/HTML-Documents/CS4320/Homeworks/HW04/Problems`:

```
Broken symbolic links in /users/bmassing/Local/HTML-Documents/CS4320/Homeworks/HW06/Problems:

/users/bmassing/Local/HTML-Documents/Classes/CS4320_2010fall/Homeworks/HW06/Problems/TestData/barfoo
/users/bmassing/Local/HTML-Documents/Classes/CS4320_2010fall/Homeworks/HW06/Problems/TestData/foobar
```

(Again, you won't be able to examine files in directories you don't have access to, so just print error messages. You should be able to access everything in the above directory, however. If you want to create some test data of your own, remember that to make a symbolic link called `sym` pointing to `foo`, you type `ln -s foo sym`.)

To get maximum points, your program should be in C or C++ and make no use of system commands such as `ls`. (You can use another language, or even write a shell script, but you will get fewer points.) The library routines mentioned for the previous problem may be helpful. The starter code may also be helpful, in reminding you how to access command-line arguments in C.

3. (Optional — up to 5 extra-credit points) Write a program that given a directory D as a command-line argument finds all the files in D or any of its subdirectories to which there are two or more hard links and prints, for each of them, all the paths within D that point to that file. Here is sample output for running the program with D = `/users/bmassing/Local/HTML-Documents/CS4320/Homeworks/HW06/Problems`:

```
Files with multiple hard links in /users/bmassing/Local/HTML-Documents/CS4320/Homeworks/HW06/Problems:

/users/bmassing/Local/HTML-Documents/Classes/CS4320_2010fall/Homeworks/HW06/Problems/TestData/bbbb
/users/bmassing/Local/HTML-Documents/Classes/CS4320_2010fall/Homeworks/HW06/Problems/TestData/b
/users/bmassing/Local/HTML-Documents/Classes/CS4320_2010fall/Homeworks/HW06/Problems/TestData/bb
/users/bmassing/Local/HTML-Documents/Classes/CS4320_2010fall/Homeworks/HW06/Problems/TestData/bbb

/users/bmassing/Local/HTML-Documents/Classes/CS4320_2010fall/Homeworks/HW06/Problems/TestData/dd
/users/bmassing/Local/HTML-Documents/Classes/CS4320_2010fall/Homeworks/HW06/Problems/TestData/d
```

This output means that the two pathnames in the first group reference the same file, the four pathnames in the second group reference the same file, etc. Output can be in any order as long as paths that reference the same file are grouped together. (Again, you won't be able to examine files in directories you don't have access to, so just print error messages. You should be able to access everything in the above directory, however. If you want to create some test data of your own, remember that to make a hard link called `sym` pointing to `foo`, you type `ln foo sym`.)

To get maximum points, your program should be in C or C++ and make no use of system commands such as `ls`. (You can use another language, or even write a shell script, but you will get fewer points.) The library routines mentioned for the previous problems may be helpful. The starter code may also be helpful, in reminding you how to access command-line arguments in C.

[1] `http://www.cs.trinity.edu/~bmassing/Classes/CS4320_2010fall/Homeworks/HW06/Problems/filesizes.c`