CSCI 3323 (Principles of Operating Systems), Fall 2018
Homework 6

Credit: 30 points.

1 Reading

Be sure you have read, or at least skimmed, 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 one of my mailboxes (outside my office or in the ASO).

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? Note 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 any experience you have with actual cameras.)

2. (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". This happens even when the user trying the command is `root` (a.k.a. the superuser). What's wrong?

3. (5 points) Section 4.5.1 describes several MS-DOS FAT filesystems. These systems require keeping a FAT in memory, which given that FATs have an entry for each disk block seems like it might require a lot of memory. But FAT-12, FAT-16, and FAT-32 impose limits on the number of disk blocks (based on the number of bits used to represent the block — e.g., 12 for FAT-12), which might mean the amount of memory needed is less. How much memory is required for a FAT for each of these filesystems? You can express your answers in terms of powers of two, or in terms of kilobytes (2^{10} bytes), megabytes (2^{20} bytes), or gigabytes (2^{30} bytes). To be consistent with what's in the textbook, also assume that each entry in the FAT is a whole number of bytes (e.g., 12 bits is rounded up to 2 bytes), though some online sources indicate that this is not the case, at least for FAT-12. Also note that FAT-32 is somewhat badly named in that it uses 28 bits for block number and not 32.

4. (10 points) Section 4.5.2 describes a UNIX filesystem 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 address 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.)

5. (5 points) Supposedly one of the advantages of a filesystem using i-nodes over one using a FAT is that for the former we only have to keep in memory the i-nodes for any open files. However, if an i-node can reference indirect blocks, as described in problem 4, it seems plausible that we might also need to keep in memory all the indirect blocks. If that's the case, and if blocksize and size of disk addresses are as in problem 4 and the "attributes" part of an i-node requires 144 bytes (a guess based on the size of the `struct` built by library function `fstat`), what's the minimum and maximum amount of memory we would need for an i-node and any associated indirect blocks? (*Note:* All i-nodes are the same size.)

3 Programming Problems

Do as many of the following programming problems as you like. 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 and the assignment (e.g., "csci 3323 hw 6" or "O/S hw 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 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 each.) Each of these problems asks you to do something with all files in a directory and its subdirectories. To get maximum points, your program(s) 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.)

   Hint: I found it helpful to structure the programs using recursion.

   (a) 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 = `/lib64`, $B = 1024$, and $M = 20$, on Dias01:

```
Unable to open /usr/lib64/Pegasus:  Permission denied
Results for directory /usr/lib64 with blocksize 1024:
```

```
6821 files of size          1 blocks
2999 files of size          2 blocks
2067 files of size          3 blocks
1344 files of size          4 blocks
1061 files of size          5 blocks
1059 files of size          6 blocks
 949 files of size          7 blocks
 807 files of size          8 blocks
 611 files of size          9 blocks
 500 files of size         10 blocks
 682 files of size         11 blocks
 753 files of size         12 blocks
 595 files of size         13 blocks
 354 files of size         14 blocks
 362 files of size         15 blocks
 695 files of size         16 blocks
 286 files of size         17 blocks
 260 files of size         18 blocks
 233 files of size         19 blocks
 556 files of size         20 blocks
7690 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.)

(b) 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 (i.e., symbolic links that point to files that don't exist). Here is sample output for running the program with D = `/users/bmassing/Local/HTML-Documents/Classes/CS3323_2018fall/Homeworks/HW06/Problems/T`

```
Broken symbolic links in /users/bmassing/Local/HTML-Documents/Classes/CS3323_2018fall/Homeworks/HW06/Problems/TestData:

/users/bmassing/Local/HTML-Documents/Classes/CS3323_2018fall/Homeworks/HW06/Problems/TestData/foobar
/users/bmassing/Local/HTML-Documents/Classes/CS3323_2018fall/Homeworks/HW06/Problems/TestData/barfoo
```

(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`.)

(c) 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/Prob`
`/users/bmassing/Local/HTML-Documents/Classes/CS3323_2018fall/Homeworks/HW06/Problems/T`

```
Files with multiple hard links in /users/bmassing/Local/HTML-Documents/Classes/CS3323_2018fall/Homeworks/HW06/Problems/TestData:

/users/bmassing/Local/HTML-Documents/Classes/CS3323_2018fall/Homeworks/HW06/Problems/TestData/bbb
/users/bmassing/Local/HTML-Documents/Classes/CS3323_2018fall/Homeworks/HW06/Problems/TestData/b
/users/bmassing/Local/HTML-Documents/Classes/CS3323_2018fall/Homeworks/HW06/Problems/TestData/bbbb
/users/bmassing/Local/HTML-Documents/Classes/CS3323_2018fall/Homeworks/HW06/Problems/TestData/bb

/users/bmassing/Local/HTML-Documents/Classes/CS3323_2018fall/Homeworks/HW06/Problems/TestData/dd
/users/bmassing/Local/HTML-Documents/Classes/CS3323_2018fall/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 `foo2` pointing to `foo`, you type `ln foo foo2`.)

4 Honor Code Statement

Include the Honor Code pledge or just the word "pledged", plus *at least one of the following* about collaboration and help (as many as apply).[1] Text *in italics* is explanatory or something for you to fill in. For programming assignments, this should go in the body of the e-mail or in a plain-text file `honor-code.txt` (no word-processor files please).

- This assignment is entirely my own work. *(Here, "entirely my own work" means that it's your own work except for anything you got from the assignment itself — some programming assignments include "starter code", for example — or from the course Web site. In particular, for programming assignments you can copy freely from anything on the "sample programs page".)*
- I worked with *names of other students* on this assignment.
- I got help with this assignment from *source of help — ACM tutoring, another student in the course, the instructor, etc. (Here, "help" means significant help, beyond a little assistance with tools or compiler errors.)*
- I got help from *outside source — a book other than the textbook (give title and author), a Web site (give its URL), etc.. (Here too, you only need to mention significant help — you don't need to tell me that you looked up an error message on the Web, but if you found an algorithm or a code sketch, tell me about that.)*
- I provided help to *names of students* on this assignment. *(And here too, you only need to tell me about significant help.)*

5 Essay

Include a brief essay (a sentence or two is fine, though you can write as much as you like) telling me what about the assignment you found interesting, difficult, or otherwise noteworthy. For programming assignments, it should go in the body of the e-mail or in a plain-text file `essay.txt` (no word-processor files please).

[1] Credit where credit is due: I based the wording of this list on a posting to a SIGCSE mailing list. SIGCSE is the ACM's Special Interest Group on CS Education.