CSC209F L0101 Fall '99

Assignment 2

Handed Out: October 7, 1999 Due: October 29, 1999

Question 1 (15 Marks)

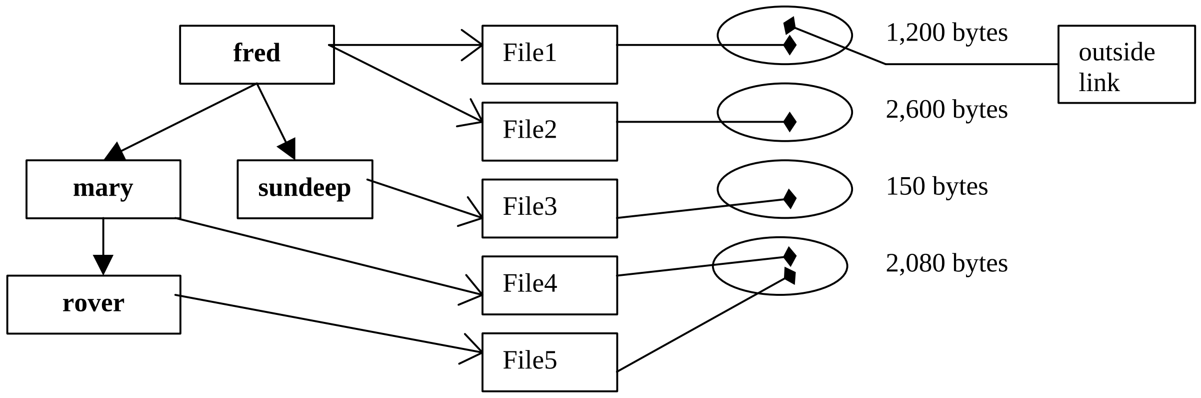
Write a C program `dirStat.c` to compute directory statistics. The program is called as

`dirStat [-r] [<`*`directory`*`>]`

The optional parameter <*directory*> is the name of the directory for which statistics are to be collected. If absent, the current working directory is assumed. If the optional parameter `-r` is specified `dirStat` will recursively compute statistics for any subdirectories encountered. The following statistics are computed:

- number of regular files in the directory/directories, and their total size
- number of directory files encountered, and theamount of space used by these files
- total space & number of files: don't count file space twice in the total space count, so ...
    - if a file has more than 1 link to it, keep track of the file in case you see it again; if you do see it again, don't count it again
    - report space used by multi-linked files whose extra links are not within this tree ...
- if you encounter *special files* (other than directory files), don't include them in your statistics

Sample output from `dirStat` called on the following directory structure might look like:



```
% dirStat -r fred
Directory: fred
  Total file links: 2
  Total file space: 3,800 bytes
  Total sub-directories: 2
  Total sub-directory file space: 1024 bytes
Directory: fred/mary
  Total file links: 1
  Total file space: 2,080 bytes
  Total sub-directories: 1
  Total sub-directory file space: 512 bytes
Directory: fred/mary/rover
  Total file links: 1
  Total file space: 2,080 bytes
  Total sub-directories: 0
  Total sub-directory file space: 0 bytes
Directory: fred/sundeep
  Total file links: 1
  Total file space: 150 bytes
  Total sub-directories: 0
  Total sub-directory file space: 0 bytes
Total directories encountered: 5
Total file links: 5
Total files: 4
Total file space: 6,030 bytes (didn't count the 2,080 byte file twice)
Files linked outside directory structure: 1
File Space linked outside directory structure: 1,200
```

In the diagram, the filled rectangles are directories, the open rectangles are file links, and the ovals represent actual disk files. We see that `File4` and `File5` are links to the same physical space, which should not be counted twice in totals … and File1 is linked from another directory structure.

Hint: you can tell how many links a file has to it by examining the data structure returned by `stat()`. The only way to tell if two different file names refer to the same file is to compare the "inode" numbers in the data structure returned by `stat()`.

Hand in a printed version of your program, as well as submitting it electronically on CDF using "submit -N a2 csc209h dirStat.c". You can overwrite a previous submission by adding the "-f " switch to the submit command.

Question 2 (10 Marks)

Repeat Question #4 from Assignment 1, but this time programmed in C. That is, write a C program named "recent.c" with usage as follows:

```
recent [-t] <progName> {<progArgs>}
```

The argument `<progName>` specifies the name of a program you are to locate in the search path and execute. Specifically, search the directories listed in the $PATH environment variable and list each directory which contains an executable file named `<progName>`. If you find a file named `<progName>` in one of the directories which is not executable, print a message after the directory name indicating that it cannot be executed by you. Execute the first instance you find (after listing the directory names). The program has an optional argument "`-t`": when it is specified you are to execute the version of `<progName>` with the most recent modification time instead. You may assume that 1) <progName> never starts with a '`-`' character, and 2) that the `-t` option, if present, is always the first command line parameter. If there are insufficient arguments your program should print an error message and return with exit status 42 (any other conditions causing your program to prematurely terminate should return exit status 1). The program will treat any parameters after `<progName>` as a list of arguments that should be passed to `<progName>` when it is executed. Your program is to wait for <progName> to terminate execution, and is to print the exit status code returned by `<progName>`. The following may help: `getenv()`, `strtok()`, `fork()`, the family of `exec()` functions, `wait()`, `stat()`.

Hand in a printed version of your csh script, as well as submitting it electronically on CDF using "`submit -N a2 csc209h recent.c`". You can overwrite a previous submission by adding the "-f " switch to the submit command.