Problem 1 -- Recursive filesystem exploration

The unix `find (1)` command is a powerful tool, at the heart of which is an engine which can perform a recursive walk of the filesystem. Similar functionality is provided by the `ls` command with the `-R` option. This problem set will require recursively walking a filesystem tree, obtaining `stat` information at each node, and presenting it in a human-friendly format.

Your command will be invoked like this:

```
programname options starting_path
```

Your program will begin the walk at the given starting path name (which could be relative or absolute) and will recursively explore all nodes of the filesystem at and below that point. At each node, it will print out some basic information.

Don't cheat by using `fts` or `system("find")` or something like that. Your solution should involve things such as `readdir` and `stat`.

The options are, in fact optional (for the invoker of the program, not for you) and can be any one or more of the following (if you don't know how to parse command-line arguments in C, take a look at `getopt(3)`:

`-u user`: Only list nodes which are owned by the specified user, which can be given either by name or by uid number --look up the library function `getpwnam(3)`. Assume that if a number is provided, that is the uid, otherwise look up the user name.

`-m mtime`: If mtime is a positive integer, only list nodes which have NOT been modified in at least that many seconds, i.e. they are at least that many seconds old. If mtime is negative, only list nodes which have been modified no more than -mtime seconds ago. Note that `find` syntax expresses this in terms of days, not seconds.

`-x`: When this option is specified, stay within the same mounted filesystem (we called this a "volume" in class) as the one in which your traversal began. Print a message to `stderr`,e.g.:

```
note: not crossing mount point at /dev/shm
```

`-l target`: When this option is used, only print out information about nodes which are symlinks whose targets *successfully* resolve to another node (which is assumed to exist and not itself be a symlink) called `target`. The target need not be on the same mounted volume. E.g.

```
$ ./walker -l /tmp/testing/d1/f1 /tmp/testing
0817/196271 lrwxr-xr-x  1  hak      users            8 Sep 15 2016 00:31 /tmp/testing/d2/l3
  -> ../d1/f1
```

Caution: ruminate on the following question: For a given target T, is there a unique symbolic name S which resolves to that target?

At each node which is to be listed, your program will output one line in a format similar to the output of the `find` command with the `-ls` option. An example output is:

```
$ ./walker /mnt4/D
0700/12289  drwxr-xr-x  2  root     root       1024 2016-09-08 18:47 /mnt4/D
0700/12290  crw-r--r--  1  root     root     0x1f20 2016-09-08 01:33 /mnt4/D/cdev
0700/12291  brw-r--r--  1  root     root     0x140a 2016-09-08 01:33 /mnt4/D/bdev
0700/12292  prw-r--r--  1  root     root          0 2016-09-08 01:34 /mnt4/D/npipe
0700/12294  lrwxrwxrwx  1  root     root         12 2016-09-08 01:50 /mnt4/D/symlink ->
  .././B/./101
```

The following items are required:

- •The device (filesystem identifier) and inode number of the node. Note that the former is of type `dev_t`, which is a 64-bit quantity on most Linux systems. In the example above, it is printed in hex and 0-padded to at least 4 hex digits.
- •The type of node and its permissions mask, in the format which `ls` uses. Read the `ls` man pages carefully and refer to lecture notes. Consider all of the cases (e.g. the set-gid bit is on but the group execute bit is off).
- •The number of links to the node
- •The owner (user) of the node. Print this out as a name, but if there is no integer->name translation available, print it out as an integer. Look at the man page `getpwuid(3)`.
- •The group owner of the node. Similar deal as above, `getgrgid(3)`.
- •The size of the node, in bytes. For BLOCK SPECIAL or CHAR SPECIAL device nodes, the size field doesn't make any sense, so print the raw device number in hexadecimal instead. If you don't quite understand what this means right now, don't worry about it.
- •The modification time of the node. Use any reasonable format for printing out the date/time of this timestamp, as long as it is correct! The example above differs from `ls`. Look at `localtime(3)`.
- •The path name of the node.
- •If the node is a symbolic link, the contents of the link (via `readlink(2)`)

Robustness & errors: If you encounter a filesystem which is "deeper" than the maximum number of open file descriptors (typically 1024) then you'll have a problem. The real find/ls have ways to handle this which are not germane to this assignment. However, you should properly detect and report any system call or library errors. It is acceptable to exit after an error. Your code should be free from "leaks" such as failure to `free` storage that you allocated with `malloc` (if applicable), and failure to close directories.

You'll find that directory entries come back to you unsorted and therefore doesn't match the output of ls. Don't worry about it. Also don't worry too much about a pre-order vs a post-order vs an unordered walk. All of those things would require you to sort each directory you encounter, which would only add to the unpleasantness of this assignment.

Problem 2 -- Extra Credit -- 1pt

Attach screenshots (or terminal session text pastes) as you perform the following system maintenance tasks. Note: you must do these as **root** (uid 0) on a real UNIX system, not Cygwin. The examples below are specific to Linux. Be very careful since an error in one of these commands could destroy data on your hard disk! There are numerous online resources to help you with this process.

1) Obtain a USB memory stick or USB external hard disk that you don't care about.

2) Attach this device to your system. Using the `dmesg` command you should see some system log messages which reveal which device node the kernel has assigned to the drive. A typical series of log messages is below:

```
[1309116.520803] usb 1-4: new high-speed USB device number 3 using ehci-pci
[1309116.648472] usb 1-4: New USB device found, idVendor=154b, idProduct=005b
[1309116.648475] usb 1-4: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[1309116.648477] usb 1-4: Product: USB 2.0 FD
[1309116.648479] usb 1-4: Manufacturer: PNY Technologies
[1309116.648481] usb 1-4: SerialNumber: AA00000000000033
[1309116.648836] usb-storage 1-4:1.0: USB Mass Storage device detected
[1309116.648947] scsi10 : usb-storage 1-4:1.0
[1309118.001013] scsi 10:0:0:0: Direct-Access     PNY      USB 2.0 FD       1100 PQ: 0 ANSI: 4
[1309118.001181] sd 10:0:0:0: Attached scsi generic sg8 type 0
[1309118.002042] sd 10:0:0:0: [sdh] 63901440 512-byte logical blocks: (32.7 GB/30.4 GiB)
[1309118.002862] sd 10:0:0:0: [sdh] Write Protect is off
[1309118.002864] sd 10:0:0:0: [sdh] Mode Sense: 43 00 00 00
[1309118.003578] sd 10:0:0:0: [sdh] No Caching mode page found
[1309118.003580] sd 10:0:0:0: [sdh] Assuming drive cache: write through
[1309118.007737]  sdh: sdh1
[1309118.010834] sd 10:0:0:0: [sdh] Attached SCSI removable disk
```

Verify that the description of the device matches what you just inserted! Note that the kernel has assigned `/dev/sdh` to the overall disk, and that there is one partition `/dev/sdh1`.

3) Make a Linux EXT2 filesystem on the disk (OK to use either the partition or the entire disk). Make sure journalling is NOT enabled in this case. Show the commands and responses.

4) Mount the disk. Show the output of the `mount` commands (without any arguments) which shows the mounted volume. Run a test program to generate endless disk metadata activity, e.g. repeatedly create, rename and delete files.

5) Disconnect the disk while it is active. CAUTION: this may hang your system for a while, and may require a reboot to recover.

6) Reconnect the disk. Run fsck to repair the filesystem (note, you may have to use `fsck.ext2` specifically, depending on your distribution, as it may attempt to fsck the disk as a FAT filesystem first). You should see at the very least that the filesystem was "not cleanly unmounted." If you've generated enough corruption from your interrupted activity, you may see other errors as well. Answer "y" to allow repairs to take place. Now mount it and see if any files have appeared in `lost+found`. Show all these commands and outputs. Umount the filesystem. We are done with this first test.

7) Make a new Linux EXT3 or EXT4 filesystem on the disk (overwriting the previous filesystem). Make sure journalling IS enabled this time.

8) Mount the filesystem and perform your steps 4 and 5 again.

9) Reconnect the disk and attempt to mount it. Verify from the logs that the journal was recovered. Show the relevant lines of the dmesg logs.