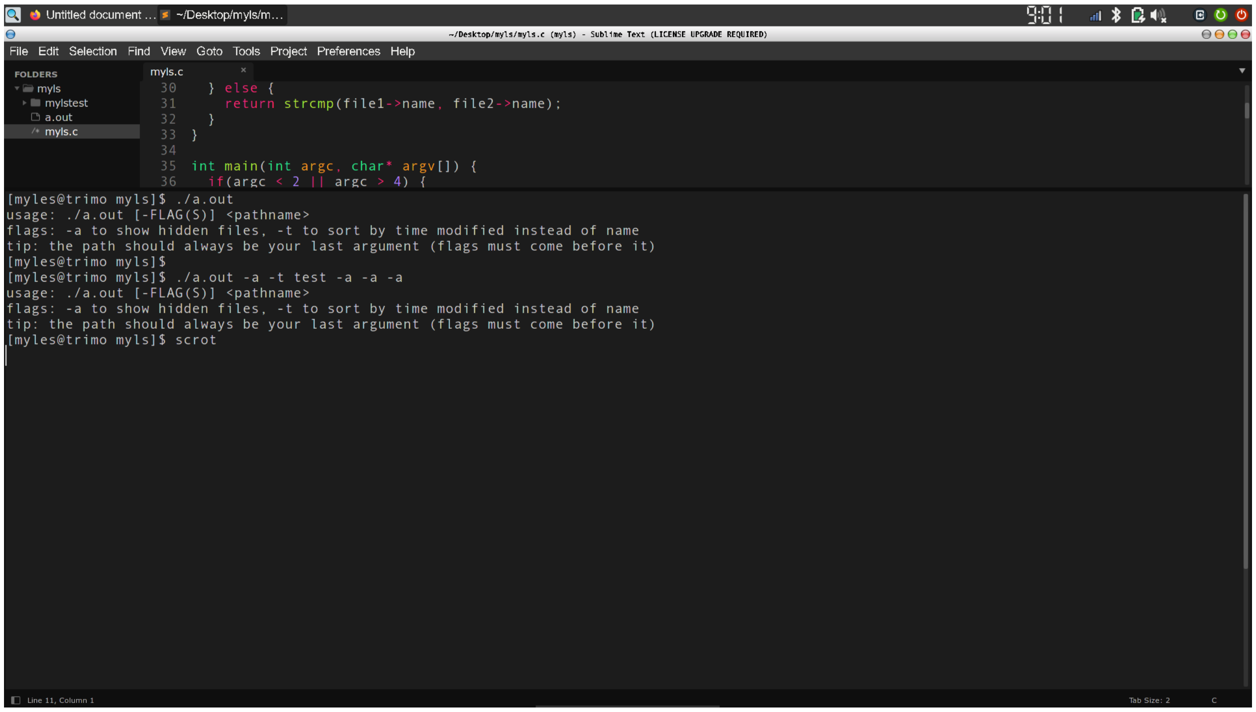
Fall 2021 Myles Greene - Beaupre

HW#2 (myls.c)

ECE322

General Design

myls.c first checks to see if the number of arguments passed to it (argc) is less than 2 (in which case the only item in argv would be the executable name) or greater than 4 (in which case the user specified more than 3 arguments, which doesn't make sense), and if either of these are true then it prints a usage message and returns -1.


Untitled document ...
~/Desktop/myls/m...
~/Desktop/myls/myls.c (myls) - Sublime Text (LICENSE UPGRADE REQUIRED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
myls
mylstest
a.out
myls.c
Line 11, Column 1
Tab Size: 2
C

```c
  } else {
    return strcmp(file1->name, file2->name);
  }
}

int main(int argc, char* argv[]) {
  if(argc < 2 || argc > 4) {
```

```
[myles@trimo myls]$ ./a.out
usage: ./a.out [-FLAG(S)] <pathname>
flags: -a to show hidden files, -t to sort by time modified instead of name
tip: the path should always be your last argument (flags must come before it)
[myles@trimo myls]$
[myles@trimo myls]$ ./a.out -a -t test -a -a -a
usage: ./a.out [-FLAG(S)] <pathname>
flags: -a to show hidden files, -t to sort by time modified instead of name
tip: the path should always be your last argument (flags must come before it)
[myles@trimo myls]$ scrot
```

If the user has passed in a valid number of arguments (aka 2, 3, or 4) then myls.c will then loop through all of the arguments between argv[0] and argv[argc-1]. This is because argv[0] is the executable name and argv[argc-1] should be the path. The user must not pass any arguments after the path according to the usage ./a.out [-FLAG(S)] <pathname>. When it is looping over all the flag arguments, if it finds -a it sets the int variable 'hidden' to 1, and if it finds the argument -t it sets the int variable 'timesort' to 1, otherwise both of these will remain 0.

The program then opens the directory passed as argv[argc-1] and does a while loop over it. An item in the directory is skipped if it starts with a '.' and hidden == 0. If hidden == 1 then it does not check if the item name starts with a '.' and processes it as normal. For every item in the directory, a struct file is created which has the following data:

```
struct file {
    char* name;
    char type;
    char* path;
    int size;
    char* owner;
    char* group;
    time_t modified;
};
```

Upon each iteration of the while loop, each of these pieces of data is set for the given item (the data is retrieved through lstat by testing the macros S_ISDIR, S_ISREG, S_ISLNK, and S_ISFIFO, as well as the executable permission bit S_IXUSR which must be anded with finfo.st_mode. Finally, the size of the variable ‘files’, which is a struct file*, is increased by sizeof(struct file) using realloc and files[index] is set to the current file variable.

```
files = (struct file*)realloc(files, (index+1) * structsize);
files[index] = curr;
```

Here, structsize = sizeof(struct file), index is the current iteration of the while loop, and curr is a variable of type struct file which has had all its data initialized as described above. Once the while loop finishes, we are left with an array of file structs, each with different pieces of data. These file structs represent the files in the directory we are searching.

Finally, with our array populated, we use std::qsort to sort it either by filename or by last modified time if timesort == 1:

```
qsort(files, index+1, structsize, compare);
```

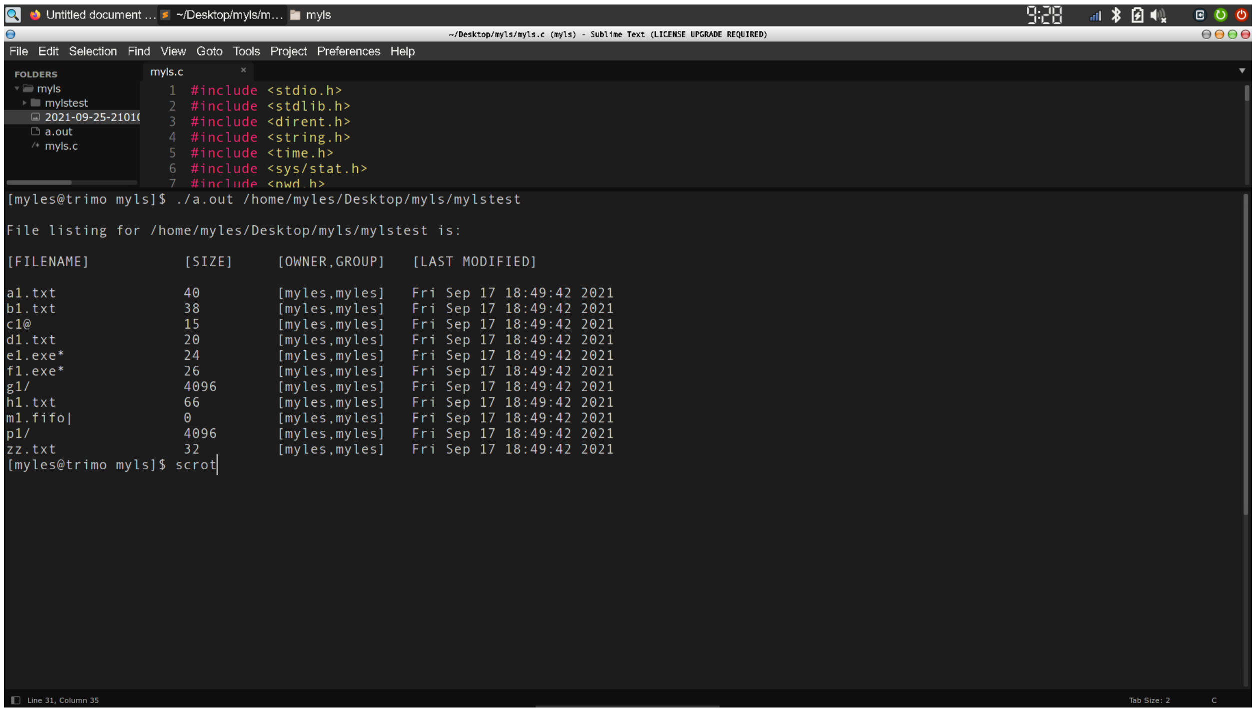
Here, compare is a pointer to the following comparison function, which will either compare by char* file.name or time_t file.modified depending on the value of timesort:

```
int compare(const void* a, const void* b) {
   const struct file* file1 = a;
   const struct file* file2 = b;

   if(timesort) {
      return difftime(file2->modified, file1->modified);
```

```
    } else {
        return strcmp(file1->name, file2->name);
    }
}
```

Once the array is sorted, all that is left to do is loop through it and print the data of each file struct:


Untitled document ...
~/Desktop/myls/m...
myls
~/Desktop/myls/myls.c (myls) - Sublime Text (LICENSE UPGRADE REQUIRED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
myls
mylstest
2021-09-25-2101(
a.out
myls.c
Line 31, Column 35
Tab Size: 2
C

```c
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <string.h>
#include <time.h>
#include <sys/stat.h>
#include <pwd.h>
```

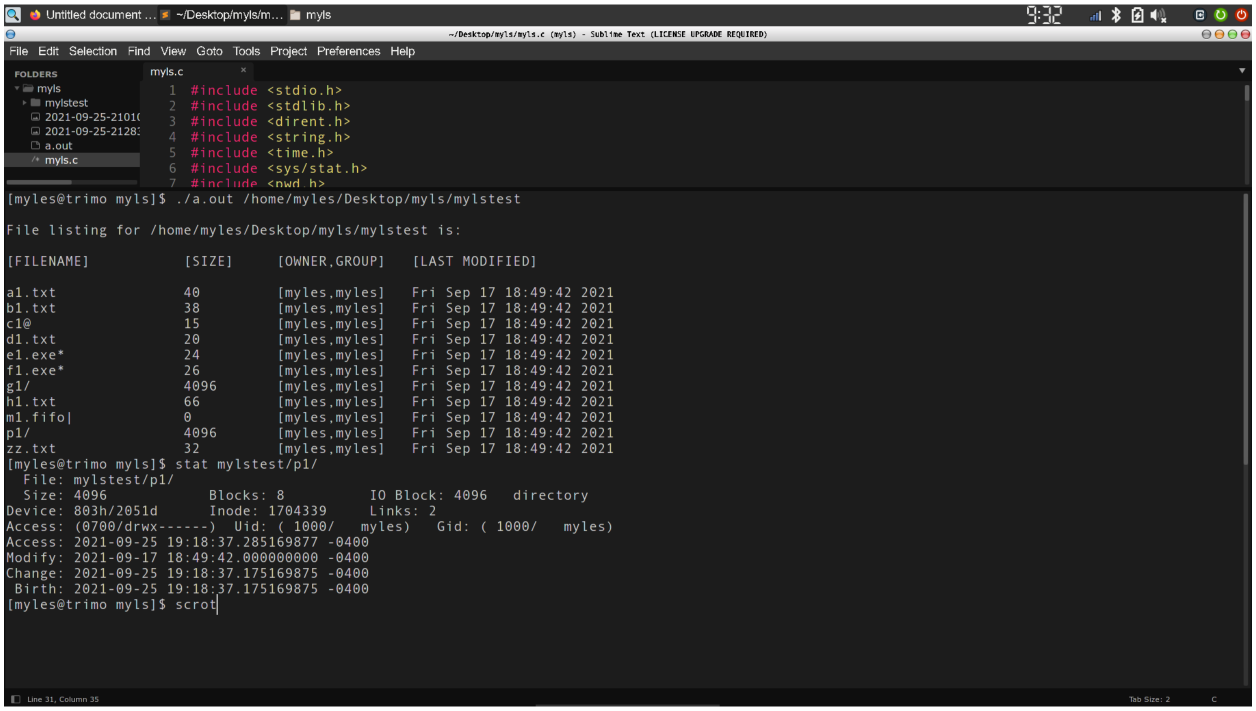
```
[myles@trimo myls]$ ./a.out /home/myles/Desktop/myls/mylstest

File listing for /home/myles/Desktop/myls/mylstest is:

[FILENAME]           [SIZE]      [OWNER,GROUP]    [LAST MODIFIED]

a1.txt               40          [myles,myles]    Fri Sep 17 18:49:42 2021
b1.txt               38          [myles,myles]    Fri Sep 17 18:49:42 2021
c1@                  15          [myles,myles]    Fri Sep 17 18:49:42 2021
d1.txt               20          [myles,myles]    Fri Sep 17 18:49:42 2021
e1.exe*              24          [myles,myles]    Fri Sep 17 18:49:42 2021
f1.exe*              26          [myles,myles]    Fri Sep 17 18:49:42 2021
g1/                  4096        [myles,myles]    Fri Sep 17 18:49:42 2021
h1.txt               66          [myles,myles]    Fri Sep 17 18:49:42 2021
m1.fifo|             0           [myles,myles]    Fri Sep 17 18:49:42 2021
p1/                  4096        [myles,myles]    Fri Sep 17 18:49:42 2021
zz.txt               32          [myles,myles]    Fri Sep 17 18:49:42 2021
[myles@trimo myls]$ scrot
```

As you can see, the files in the directory mylstest are sorted by filename by default, and each is given a symbol which represents its type. On my system the size of an empty folder is 4096 bytes instead of 64 bytes in the example out.txt, but the stat command confirms that this is the correct size:



```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <string.h>
#include <time.h>
#include <sys/stat.h>
#include <pwd.h>
```

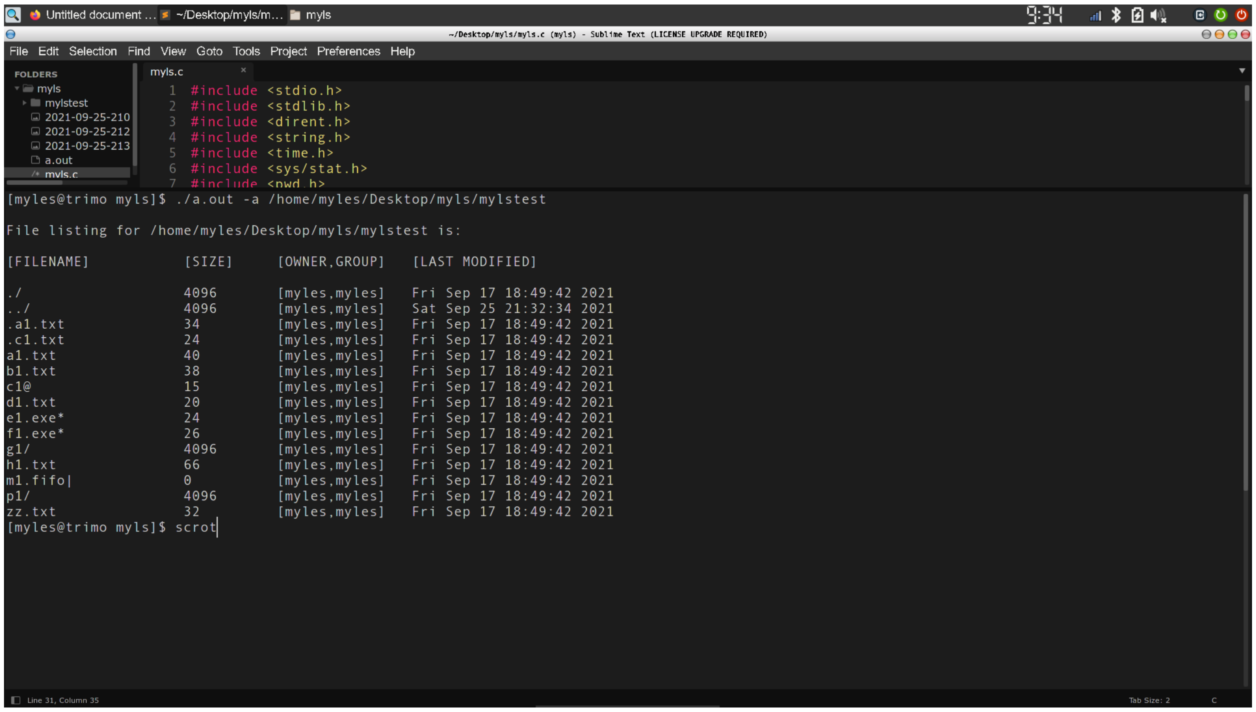
```
[myles@trimo myls]$ ./a.out /home/myles/Desktop/myls/mylstest

File listing for /home/myles/Desktop/myls/mylstest is:

[FILENAME]          [SIZE]     [OWNER,GROUP]   [LAST MODIFIED]

a1.txt              40         [myles,myles]   Fri Sep 17 18:49:42 2021
b1.txt              38         [myles,myles]   Fri Sep 17 18:49:42 2021
c1@                 15         [myles,myles]   Fri Sep 17 18:49:42 2021
d1.txt              20         [myles,myles]   Fri Sep 17 18:49:42 2021
e1.exe*             24         [myles,myles]   Fri Sep 17 18:49:42 2021
f1.exe*             26         [myles,myles]   Fri Sep 17 18:49:42 2021
g1/                 4096       [myles,myles]   Fri Sep 17 18:49:42 2021
h1.txt              66         [myles,myles]   Fri Sep 17 18:49:42 2021
m1.fifo|            0          [myles,myles]   Fri Sep 17 18:49:42 2021
p1/                 4096       [myles,myles]   Fri Sep 17 18:49:42 2021
zz.txt              32         [myles,myles]   Fri Sep 17 18:49:42 2021
[myles@trimo myls]$ stat mylstest/p1/
  File: mylstest/p1/
  Size: 4096            Blocks: 8          IO Block: 4096   directory
Device: 803h/2051d      Inode: 1704339     Links: 2
Access: (0700/drwx------)  Uid: ( 1000/   myles)   Gid: ( 1000/   myles)
Access: 2021-09-25 19:18:37.285169877 -0400
Modify: 2021-09-17 18:49:42.000000000 -0400
Change: 2021-09-25 19:18:37.175169875 -0400
 Birth: 2021-09-25 19:18:37.175169875 -0400
[myles@trimo myls]$ scrot
```

Here is an example with the -a flag to show hidden files:



```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <string.h>
#include <time.h>
#include <sys/stat.h>
#include <pwd.h>
```

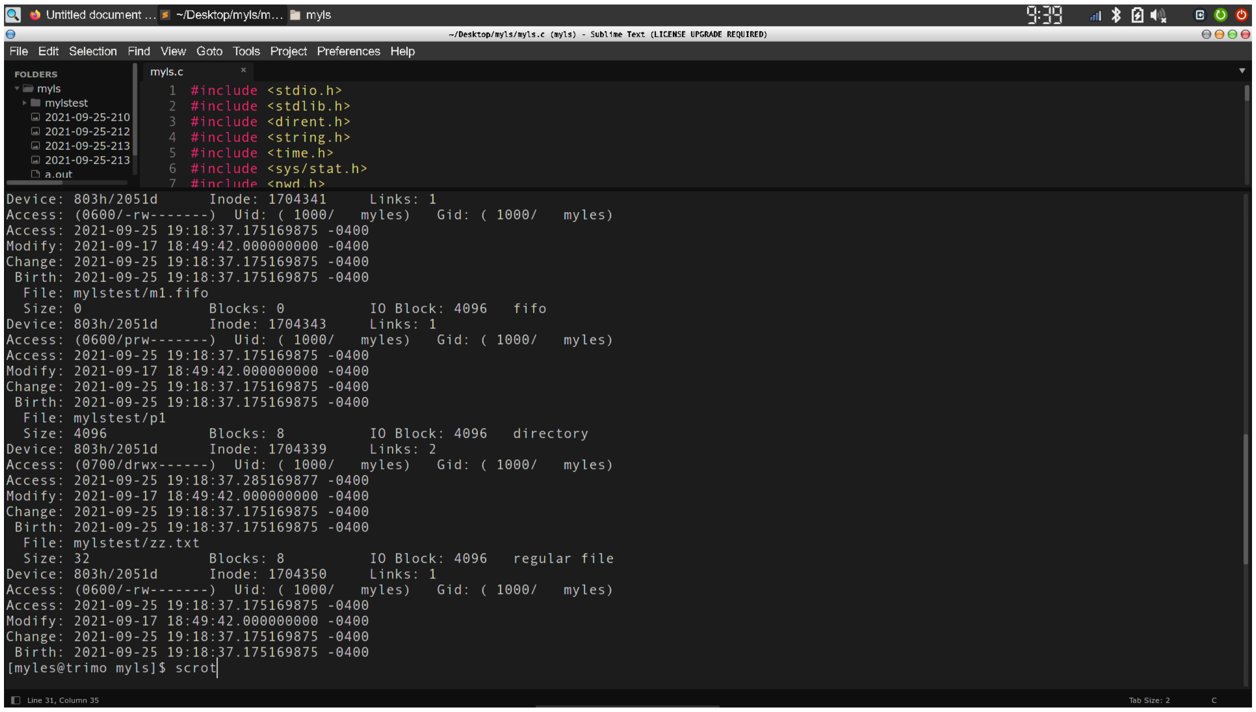
```
[myles@trimo myls]$ ./a.out -a /home/myles/Desktop/myls/mylstest

File listing for /home/myles/Desktop/myls/mylstest is:

[FILENAME]          [SIZE]     [OWNER,GROUP]   [LAST MODIFIED]

./                  4096       [myles,myles]   Fri Sep 17 18:49:42 2021
../                 4096       [myles,myles]   Sat Sep 25 21:32:34 2021
.a1.txt             34         [myles,myles]   Fri Sep 17 18:49:42 2021
.c1.txt             24         [myles,myles]   Fri Sep 17 18:49:42 2021
a1.txt              40         [myles,myles]   Fri Sep 17 18:49:42 2021
b1.txt              38         [myles,myles]   Fri Sep 17 18:49:42 2021
c1@                 15         [myles,myles]   Fri Sep 17 18:49:42 2021
d1.txt              20         [myles,myles]   Fri Sep 17 18:49:42 2021
e1.exe*             24         [myles,myles]   Fri Sep 17 18:49:42 2021
f1.exe*             26         [myles,myles]   Fri Sep 17 18:49:42 2021
g1/                 4096       [myles,myles]   Fri Sep 17 18:49:42 2021
h1.txt              66         [myles,myles]   Fri Sep 17 18:49:42 2021
m1.fifo|            0          [myles,myles]   Fri Sep 17 18:49:42 2021
p1/                 4096       [myles,myles]   Fri Sep 17 18:49:42 2021
zz.txt              32         [myles,myles]   Fri Sep 17 18:49:42 2021
[myles@trimo myls]$ scrot
```

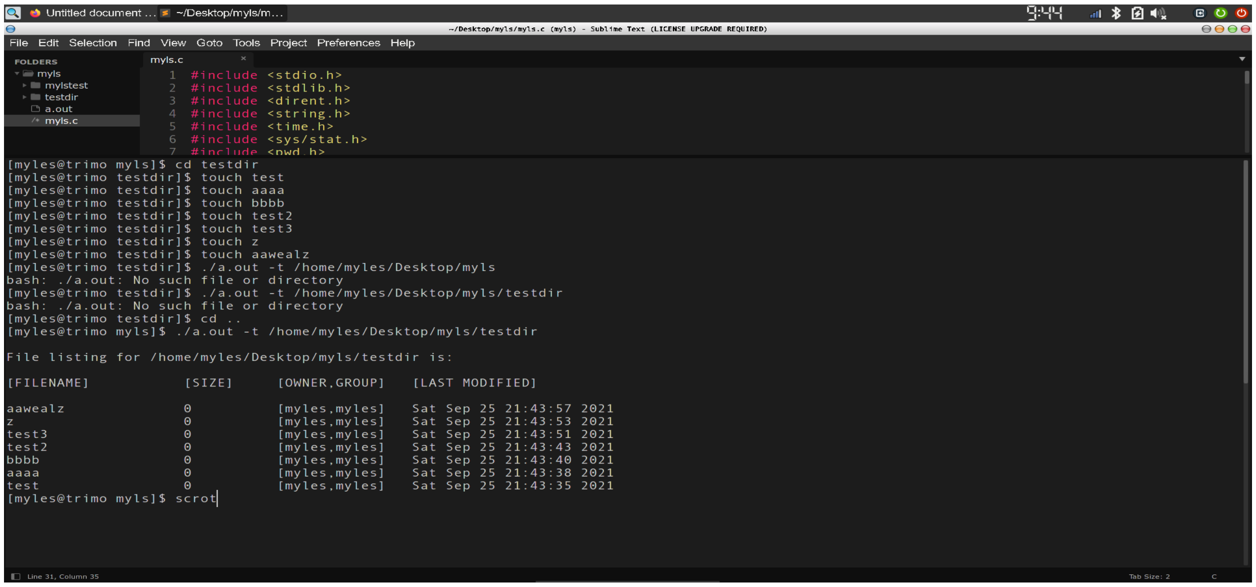
When I downloaded the mylstest tarball and untarred it, every file and directory inside had the same exact value for last modified time, even the nanosecond portion (they all ended in .000000000):



```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <string.h>
#include <time.h>
#include <sys/stat.h>
#include <pwd.h>
```

```
Device: 803h/2051d      Inode: 1704341     Links: 1
Access: (0600/-rw-------)  Uid: ( 1000/   myles)   Gid: ( 1000/   myles)
Access: 2021-09-25 19:18:37.175169875 -0400
Modify: 2021-09-17 18:49:42.000000000 -0400
Change: 2021-09-25 19:18:37.175169875 -0400
 Birth: 2021-09-25 19:18:37.175169875 -0400
  File: mylstest/m1.fifo
  Size: 0               Blocks: 0          IO Block: 4096   fifo
Device: 803h/2051d      Inode: 1704343     Links: 1
Access: (0600/prw-------)  Uid: ( 1000/   myles)   Gid: ( 1000/   myles)
Access: 2021-09-25 19:18:37.175169875 -0400
Modify: 2021-09-17 18:49:42.000000000 -0400
Change: 2021-09-25 19:18:37.175169875 -0400
 Birth: 2021-09-25 19:18:37.175169875 -0400
  File: mylstest/p1
  Size: 4096            Blocks: 8          IO Block: 4096   directory
Device: 803h/2051d      Inode: 1704339     Links: 2
Access: (0700/drwx------)  Uid: ( 1000/   myles)   Gid: ( 1000/   myles)
Access: 2021-09-25 19:18:37.285169877 -0400
Modify: 2021-09-17 18:49:42.000000000 -0400
Change: 2021-09-25 19:18:37.175169875 -0400
 Birth: 2021-09-25 19:18:37.175169875 -0400
  File: mylstest/zz.txt
  Size: 32              Blocks: 8          IO Block: 4096   regular file
Device: 803h/2051d      Inode: 1704350     Links: 1
Access: (0600/-rw-------)  Uid: ( 1000/   myles)   Gid: ( 1000/   myles)
Access: 2021-09-25 19:18:37.175169875 -0400
Modify: 2021-09-17 18:49:42.000000000 -0400
Change: 2021-09-25 19:18:37.175169875 -0400
 Birth: 2021-09-25 19:18:37.175169875 -0400
[myles@trimo myls]$ scrot
```

So I wasn't able to test the -t flag with those files, but I did test it with my own files and as you can see it lists them in order from last modified to first modified:


Untitled document ...
~/Desktop/myls/m...
~/Desktop/myls/myls.c (myls) - Sublime Text (LICENSE UPGRADE REQUIRED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
myls
mylstest
testdir
a.out
myls.c
Line 31, Column 35
Tab Size: 2

```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <string.h>
#include <time.h>
#include <sys/stat.h>
#include <pwd.h>
```

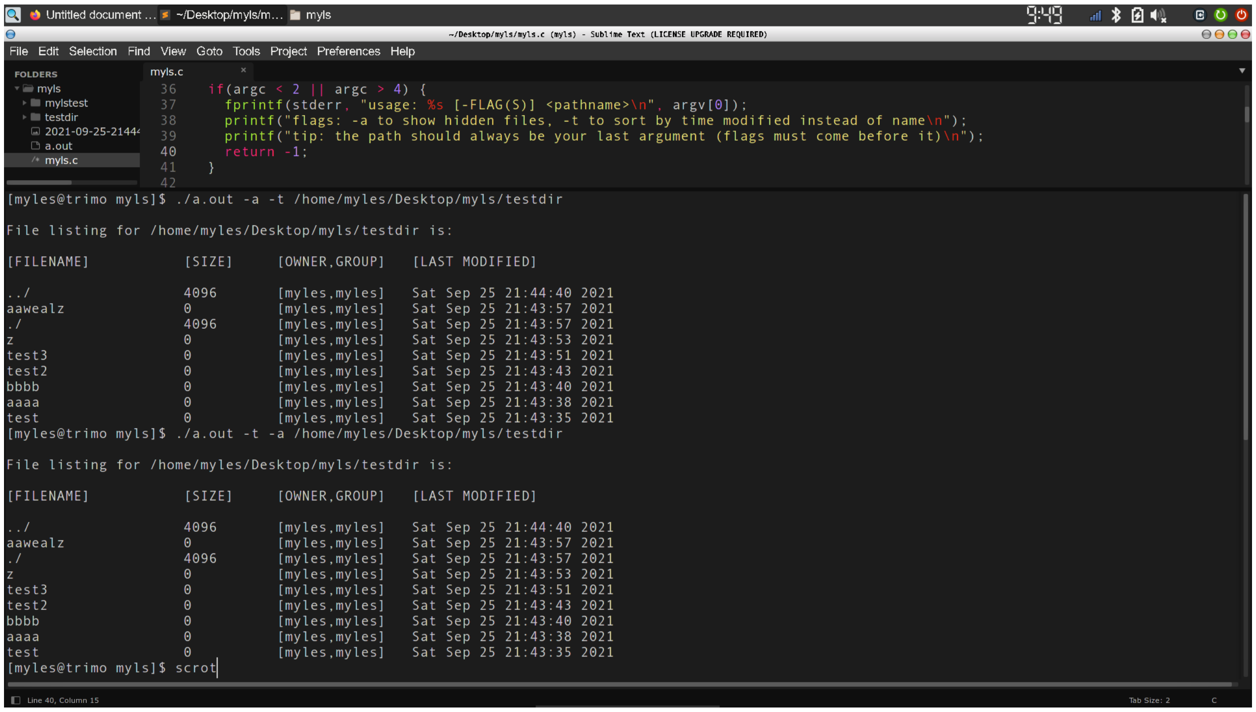
```
[myles@trimo myls]$ cd testdir
[myles@trimo testdir]$ touch test
[myles@trimo testdir]$ touch aaaa
[myles@trimo testdir]$ touch bbbb
[myles@trimo testdir]$ touch test2
[myles@trimo testdir]$ touch test3
[myles@trimo testdir]$ touch z
[myles@trimo testdir]$ touch aawealz
[myles@trimo testdir]$ ./a.out -t /home/myles/Desktop/myls
bash: ./a.out: No such file or directory
[myles@trimo testdir]$ ./a.out -t /home/myles/Desktop/myls/testdir
bash: ./a.out: No such file or directory
[myles@trimo testdir]$ cd ..
[myles@trimo myls]$ ./a.out -t /home/myles/Desktop/myls/testdir

File listing for /home/myles/Desktop/myls/testdir is:

[FILENAME]           [SIZE]     [OWNER,GROUP]   [LAST MODIFIED]

aawealz              0          [myles,myles]   Sat Sep 25 21:43:57 2021
z                    0          [myles,myles]   Sat Sep 25 21:43:53 2021
test3                0          [myles,myles]   Sat Sep 25 21:43:51 2021
test2                0          [myles,myles]   Sat Sep 25 21:43:43 2021
bbbb                 0          [myles,myles]   Sat Sep 25 21:43:40 2021
aaaa                 0          [myles,myles]   Sat Sep 25 21:43:38 2021
test                 0          [myles,myles]   Sat Sep 25 21:43:35 2021
[myles@trimo myls]$ scrot
```

And here is an example with both the -a and -t flags, in either order:


Untitled document ...
~/Desktop/myls/m...
myls
~/Desktop/myls/myls.c (myls) - Sublime Text (LICENSE UPGRADE REQUIRED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
myls
mylstest
testdir
2021-09-25-21444
a.out
myls.c
Line 40, Column 15
Tab Size: 2

```
  if(argc < 2 || argc > 4) {
    fprintf(stderr, "usage: %s [-FLAG(S)] <pathname>\n", argv[0]);
    printf("flags: -a to show hidden files, -t to sort by time modified instead of name\n");
    printf("tip: the path should always be your last argument (flags must come before it)\n");
    return -1;
  }
```

```
[myles@trimo myls]$ ./a.out -a -t /home/myles/Desktop/myls/testdir

File listing for /home/myles/Desktop/myls/testdir is:

[FILENAME]           [SIZE]     [OWNER,GROUP]   [LAST MODIFIED]

../                  4096       [myles,myles]   Sat Sep 25 21:44:40 2021
aawealz              0          [myles,myles]   Sat Sep 25 21:43:57 2021
./                   4096       [myles,myles]   Sat Sep 25 21:43:57 2021
z                    0          [myles,myles]   Sat Sep 25 21:43:53 2021
test3                0          [myles,myles]   Sat Sep 25 21:43:51 2021
test2                0          [myles,myles]   Sat Sep 25 21:43:43 2021
bbbb                 0          [myles,myles]   Sat Sep 25 21:43:40 2021
aaaa                 0          [myles,myles]   Sat Sep 25 21:43:38 2021
test                 0          [myles,myles]   Sat Sep 25 21:43:35 2021
[myles@trimo myls]$ ./a.out -t -a /home/myles/Desktop/myls/testdir

File listing for /home/myles/Desktop/myls/testdir is:

[FILENAME]           [SIZE]     [OWNER,GROUP]   [LAST MODIFIED]

../                  4096       [myles,myles]   Sat Sep 25 21:44:40 2021
aawealz              0          [myles,myles]   Sat Sep 25 21:43:57 2021
./                   4096       [myles,myles]   Sat Sep 25 21:43:57 2021
z                    0          [myles,myles]   Sat Sep 25 21:43:53 2021
test3                0          [myles,myles]   Sat Sep 25 21:43:51 2021
test2                0          [myles,myles]   Sat Sep 25 21:43:43 2021
bbbb                 0          [myles,myles]   Sat Sep 25 21:43:40 2021
aaaa                 0          [myles,myles]   Sat Sep 25 21:43:38 2021
test                 0          [myles,myles]   Sat Sep 25 21:43:35 2021
[myles@trimo myls]$ scrot
```

Conclusion

myls.c will list the contents of any directory you give it, so long as you follow the proper usage which it will print out to you if you do something wrong. The only valid flags are -a and -t, and they must come before the directory argument. In general it works by quicksorting an array of file structs representing the files in the directory, either by their name or by their last time modified, and it's only 145 lines of code. I also uploaded source.txt alongside this document so it's easier to read, otherwise the source code is below (sorry I turned this in late I wanted to make it perfect).

Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <string.h>
#include <time.h>
#include <sys/stat.h>
#include <pwd.h>
#include <grp.h>
#include <unistd.h>
#include <sys/types.h>

struct file {
    char* name;
    char type;
    char* path;
    int size;
    char* owner;
    char* group;
    time_t modified;
};

int timesort = 0;

int compare(const void* a, const void* b) {
    const struct file* file1 = a;
    const struct file* file2 = b;

    if(timesort) {
        return difftime(file2->modified, file1->modified);
    } else {
        return strcmp(file1->name, file2->name);
    }
}

int main(int argc, char* argv[]) {
    if(argc < 2 || argc > 4) {
        fprintf(stderr, "usage: %s [-FLAG(S)] <pathname>\n",
argv[0]);
        printf("flags: -a to show hidden files, -t to sort by time
modified instead of name\n");
        printf("tip: the path should always be your last
argument (flags must come before it)\n");
        return -1;
    }

    int patharg = argc - 1;
    int hidden = 0;
    char* currflag;

    if(argc > 2) {
        for(int i = 1; i < patharg; i++) {
                currflag = argv[i];
                if(strcmp(currflag, "-a") == 0) {
                        hidden = 1;
                }
                else if(strcmp(currflag, "-t") == 0) {
                        timesort = 1;
                }
                else {
                        fprintf(stderr, "Unrecognized flag '%s': options
are -a or -t and must come before your path\n", currflag);
                        return -1;
                }
        }
    }

    char* path = argv[patharg];
    DIR* dirp = opendir(path);

    if(dirp == NULL) {
        fprintf(stderr,"Bad path: %s\n", path);
        return -1;
    }

    struct dirent* d;
    struct file* files = NULL;
    int index = -1;
    int structsize = sizeof(struct file);
    struct file curr;
    struct stat finfo;
    char* fullpath;
    char* dname;

    while((d = readdir(dirp)) != NULL) {
        curr.name = d->d_name;

        if(hidden || curr.name[0] != '.') {
            index++;

            fullpath =
(char*)malloc(sizeof(char)*strlen(path)+strlen(curr.name)+2);
            sprintf(fullpath,"%s/%s", path, curr.name);

            if(lstat(fullpath, &finfo) == 0) {
                dname =
(char*)malloc(sizeof(char)*strlen(curr.name)+2);

                curr.type = ' ';

                if(S_ISDIR(finfo.st_mode)) {
                    curr.type = '/';
                }
            else if(S_ISREG(finfo.st_mode)) {
                if((finfo.st_mode & S_IXUSR) == S_IXUSR) {
                        curr.type = '*';
                    }
                }
                else if(S_ISLNK(finfo.st_mode)) {
                                curr.type = '@';
                }
                else if(S_ISFIFO(finfo.st_mode)) {
                                curr.type = '|';
                }

                        sprintf(dname,"%s%c", curr.name, curr.type);

                        curr.path = dname;
                        curr.size = finfo.st_size;
                curr.owner = getpwuid(finfo.st_uid)->pw_name;
                        curr.group = getgrgid(finfo.st_gid)->gr_name;
                        curr.modified = finfo.st_mtim.tv_sec;

                        files = (struct file*)realloc(files, (index+1) *
structsize);
                        files[index] = curr;

                        free(fullpath);
                } else {
                        fprintf(stderr,"Error! lstat did not return 0 on
file or directory '%s'\n", fullpath);
                        return -1;
                }
        }
   }

  qsort(files, index+1, structsize, compare);
```

```
    printf("\nFile listing for %s is:\n\n", path);

    printf("[FILENAME] \t        [SIZE] \t[OWNER,GROUP]
\t[LAST MODIFIED]\n\n");

    for(int i = 0; i <= index; i++) {
        printf("%-20s ", files[i].path);
        printf("%-5d ", files[i].size);
        printf("\t[%s,%s] ", files[i].owner, files[i].group);
        printf("\t%s", ctime(&files[i].modified));
    }

    free(dname);
    free(files);

    return 0;
}
```