File System Utilities

Note: This is an individual project.
Description: In this project, you will write four different programs based on various UNIX utilities.

Stat: Write your own version of the command line program stat, which simply calls the stat() system call on a given file or directory. Print out file size, number of blocks allocated, reference (link) count, file permissions, and file inode.

Useful interfaces: stat()

List Files: Write a program that lists files in the given directory. When called without any arguments, the program should just print the file names. When invoked with the -l flag, the program should print out information about each file, such as the owner, group, permissions, and other information obtained from the stat() system call. The program should take one additional argument, which is the directory to read, e.g., myls -l directory. If no directory is given, the program should just use the current working directory.

Useful interfaces: stat(), opendir(), readdir(), getcwd().

Tail: Write a program that prints out the last few lines of a file. The program should be efficient, in that it seeks to near the end of the file, reads in a block of data, and then goes back until it finds the requested number of lines; at this point, it should print out those lines
from the beginning to the end of the file. To invoke the program, one should type: mytail -n file, where n is the number of lines at the end of the file to print.

Useful interfaces: stat(), lseek(), open(), read(), close().

Recursive Search: Write a program that prints out the names of each file and directory in the file system tree, starting at a given point in the tree. For example, when run without arguments, the program should start with the current working directory and print its contents, as well as the contents of any sub-directories, etc., until the entire tree, root at the CWD, is printed. If given a single argument (of a directory name), use that as the root of the tree instead.

Useful interfaces: you figure it out.

Running Example:

```
________________________________________________ $
./myls
makefile
myls
mysearch
mystat
mytail
myls-Wanwan.c
mysearch-Wanwan.c
mystat-Wanwan.c
mytail-Wanwan.c
________________________________________________ $
./mysearch
Current Path /home/w/wanwan/proj4/
Filename: makefile
Filename: myls
Filename: mysearch
Filename: mystat
Filename: mytail
Filename: myls-Wanwan.c
Filename: mysearch-Wanwan.c
Filename: mystat-Wanwan.c
Filename: mytail-Wanwan.c
Search has finished!
________________________________________________ $
./mystat ./makefile
File Information of ./makefile
File Size: 259 Bytes
Number of Blocks Allocated: 8
Number of Links: 1
File Permissions: -rw-r--r--
File Inode: -198486821
```

$

```
./mytail 2 ./makefile
These are the last 2 lines of ./makefile:
mytail: mytail-Wanwan.c
 gcc -o mytail mytail-Wanwan.c
```
