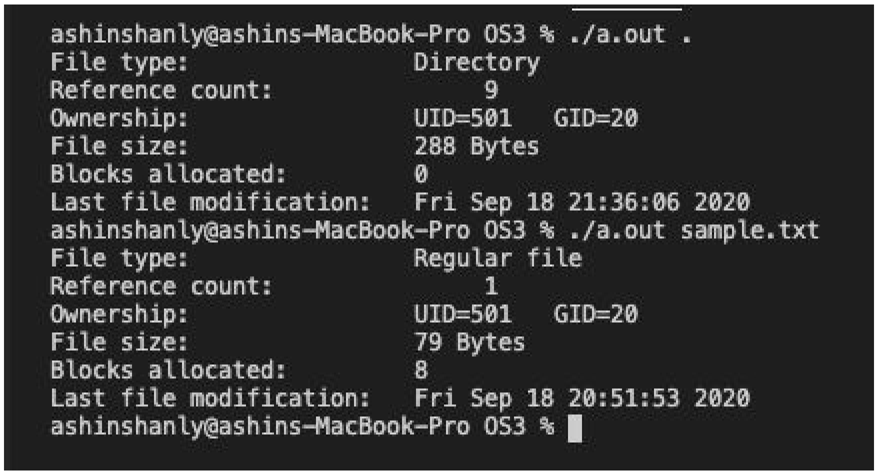
OS LAB ASSIGNMENT-3

Submitted by: KS Ashin Shanly - 20250019

Question 01: 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, and so forth. What is the link count of a directory, as the number of entries in the directory changes? Useful interfaces: `stat()`, naturally.

OUTPUT



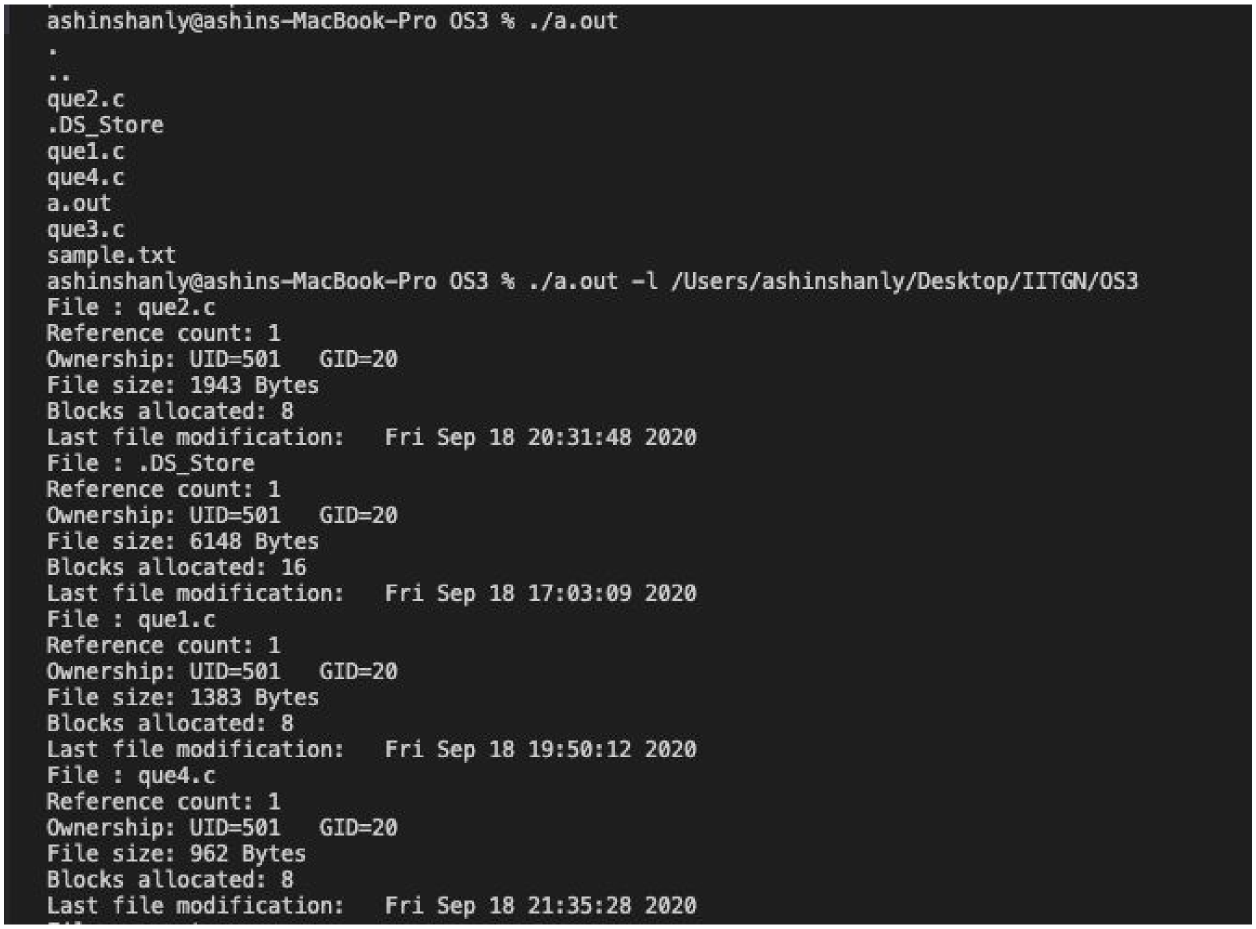
```
ashinshanly@ashins-MacBook-Pro OS3 % ./a.out .
File type:                Directory
Reference count:               9
Ownership:                UID=501   GID=20
File size:                288 Bytes
Blocks allocated:         0
Last file modification:   Fri Sep 18 21:36:06 2020
ashinshanly@ashins-MacBook-Pro OS3 % ./a.out sample.txt
File type:                Regular file
Reference count:               1
Ownership:                UID=501   GID=20
File size:                79 Bytes
Blocks allocated:         8
Last file modification:   Fri Sep 18 20:51:53 2020
ashinshanly@ashins-MacBook-Pro OS3 % 
```

> The link count of a file tells the total number of links a file has which is nothing but the number of hard-links a file has. The soft-link is not part of this count as the soft-link's inode number is different from the original file.
The link count of a directory increases whenever a sub-directory is created. The default link count of a directory when it is created is 2. The extra count is because for every directory created, a link gets created in the parent directory to point to this new directory.

Question 02: 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()`.

<u>OUTPUT</u>

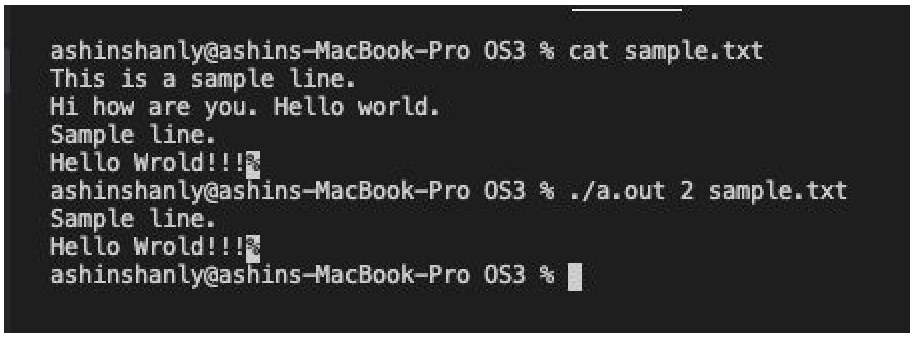


```
ashinshanly@ashins-MacBook-Pro OS3 % ./a.out
.
..
que2.c
.DS_Store
que1.c
que4.c
a.out
que3.c
sample.txt
ashinshanly@ashins-MacBook-Pro OS3 % ./a.out -l /Users/ashinshanly/Desktop/IITGN/OS3
File : que2.c
Reference count: 1
Ownership: UID=501   GID=20
File size: 1943 Bytes
Blocks allocated: 8
Last file modification:   Fri Sep 18 20:31:48 2020
File : .DS_Store
Reference count: 1
Ownership: UID=501   GID=20
File size: 6148 Bytes
Blocks allocated: 16
Last file modification:   Fri Sep 18 17:03:09 2020
File : que1.c
Reference count: 1
Ownership: UID=501   GID=20
File size: 1383 Bytes
Blocks allocated: 8
Last file modification:   Fri Sep 18 19:50:12 2020
File : que4.c
Reference count: 1
Ownership: UID=501   GID=20
File size: 962 Bytes
Blocks allocated: 8
Last file modification:   Fri Sep 18 21:35:28 2020
```

> This can be done by opening the directory using opendir() function and reading it's contents using the readdir() function. The readdir() function is given inside a while() loop which repeatedly reads all the files in the given directory. If a '-l' is specified, stat() is called on the file that is read in the while() loop.

<u>Question 03</u>: 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 backwards until it finds the requested number of lines; at this point, it should print out those lines from 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()`.

OUTPUT



```
ashinshanly@ashins-MacBook-Pro OS3 % cat sample.txt
This is a sample line.
Hi how are you. Hello world.
Sample line.
Hello Wrold!!!%
ashinshanly@ashins-MacBook-Pro OS3 % ./a.out 2 sample.txt
Sample line.
Hello Wrold!!!%
ashinshanly@ashins-MacBook-Pro OS3 %
```

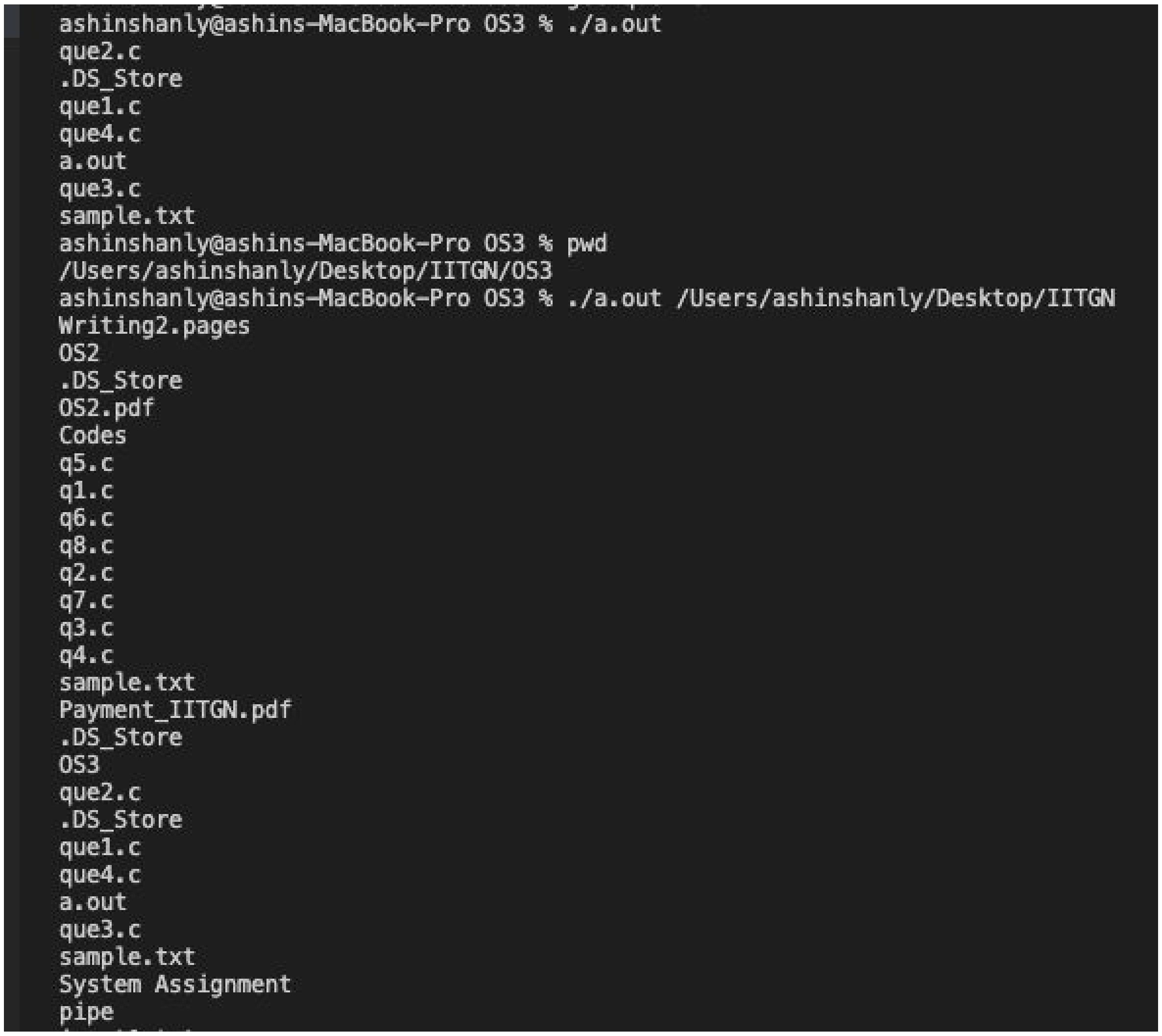
> One way to do this is by starting from the end of the file and counting backwards how many lines we need to show and place the index to start printing at that position. The start printing the lines from that index till the end of the file has been reached. If the given number of lines is greater than the total number of lines in the entire file, then the whole file will be printed line by line.

Note: The input format to the code given is "mytail n filename".

The time complexity of the above program will be O(n), where n is the number of characters in the file. It would take utmost n number of iterations to figure out the position in the file to start printing from and n number of iterations to finally print it. Therefore, O(n+n) which is the same as O(n).

Question 04: 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. Refine your recursive search with more fun options, similar to the powerful `find` command line tool. Useful interfaces: figure it out.

OUTPUT



```
ashinshanly@ashins-MacBook-Pro OS3 % ./a.out
que2.c
.DS_Store
que1.c
que4.c
a.out
que3.c
sample.txt
ashinshanly@ashins-MacBook-Pro OS3 % pwd
/Users/ashinshanly/Desktop/IITGN/OS3
ashinshanly@ashins-MacBook-Pro OS3 % ./a.out /Users/ashinshanly/Desktop/IITGN
Writing2.pages
OS2
.DS_Store
OS2.pdf
Codes
q5.c
q1.c
q6.c
q8.c
q2.c
q7.c
q3.c
q4.c
sample.txt
Payment_IITGN.pdf
.DS_Store
OS3
que2.c
.DS_Store
que1.c
que4.c
a.out
que3.c
sample.txt
System Assignment
pipe
```

> This program recursively prints out the contents in a directory including the contents of its sub directories and their subdirectories and so on. Inside the loop, while reading any content of the directory, it is initially checked whether it is a file or a directory. If it is a file, it's file name is simply printed and if it's a directory, the function recursively calls itself with the path being updated to the new path by concatenating the previous path with the directory name currently being read. This will print the entire contents of it;s sub directories and their sub subdirectories until the end of the tree.