Practical 1 – System Util Report

Overview

The task was to implement the Unix command *ls -n* in C without using any library functions apart from *localtime()*. No extensions were attempted.

Design

In my design, system calls were contained within a separate wrapper functions so that their implementation can be easily modified without affecting the code significantly (e.g. the function header stays the same). These were initially developed with the *syscall()* wrapper function but were later modified to use Inline Assembly instead.

There are many helper functions for other necessary functionality, such as converting ints to strings (*my_int_to_str*), measuring the length of strings (*my_strlen*), combining two strings (*my_strcat*) and 3 functions for printing out the data (*write_permissions*, *write_date* and *write_ls*).

The write system call has been split into 2 distinct functions: one that prints to *stdout*, and one that prints to *stderr* to make the method header less cluttered. I believe this decreases the user's chance of making mistakes and reduces code repetition (as *stderr* is used in rare cases, such as error handling or occasionally debugging).

Where appropriate, error handling was implemented. If failure is detected in any of the *syscalls*, the program prints a helpful error message and then exits (see Figure 4). There is also a more subtle error handling in *write_date()*, where if the month is not recognised, it simply prints out *"--Err--"*. The reasoning behind this is that it is not a fatal error, but the user should still be notified if it happens (although it is very unlikely to happen in practice). System call numbers were retrieved from the s*yscall.h* library during development stages and then stored as constants (Figures 1 & 2).

The output of the program was regularly compared to the output of Unix call *ls -n* (see Figure 3). Although it was not a requirement in the specifications, hidden files and directories (including current and parent directory) are filtered out and not displayed to mimic *ls -n* better. The Unix command had some additional functionality, such as formatting and ordering of data, which I did not attempt to implement.

A lot of the code used is modified from other sources, such as the man page for the *getdents* and *stat* system calls. These code segments are noted on each occasion, as well as what part has been modified from the original. All references used are noted in the References section of this report, and at the relevant sections in the code.

Testing

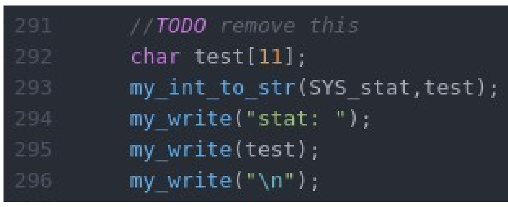
The code was tested by modifying the contents of the project directory and comparing the output of *myls* to Unix command *ls -n* (Figure 3).

Initially, a solution with the *syscall()* wrapper function was created. When switching over to Inline Assembly, the outputs of the two implementations were compared as well, to ensure identical behaviour.

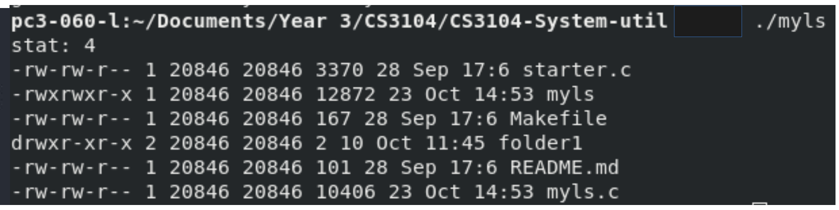
Program behaviour was tested without arguments (Figure 3), with a sub-directory given as argument (Figure 5), with a parent directory given as argument (Figure 6) and with an invalid path (Figure 7) to ensure that it behaves as expected. When provided with invalid path, *ls -n* prints out an error message, but I chose not to implement this into my design.

Examples

Figure 1 & 2: An example of correct *syscall* number retrieval from the *sys/syscall.h* library:

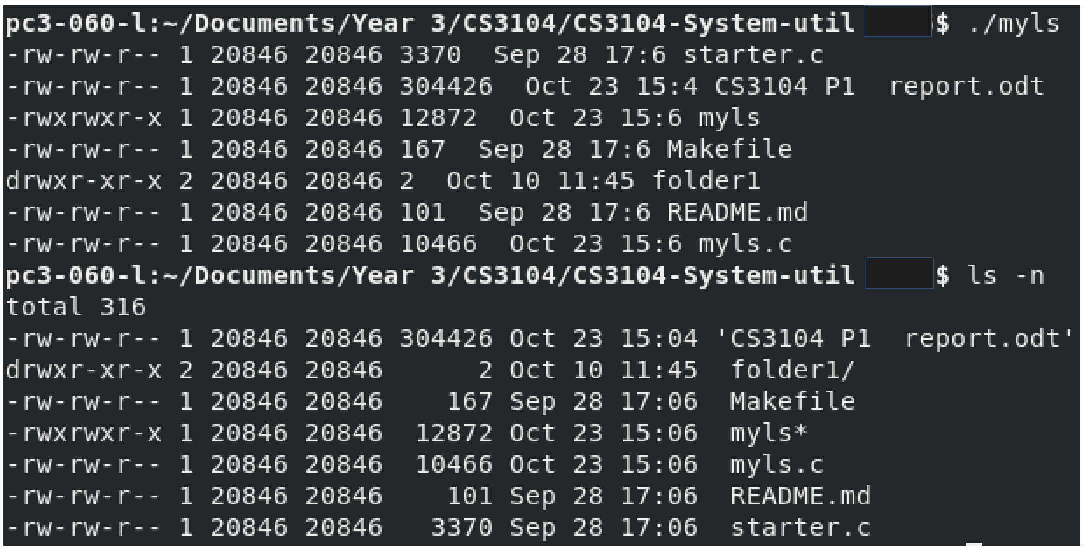


```
    //TODO remove this
    char test[11];
    my_int_to_str(SYS_stat,test);
    my_write("stat: ");
    my_write(test);
    my_write("\n");
```



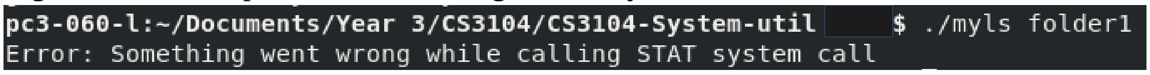
```
pc3-060-l:~/Documents/Year 3/CS3104/CS3104-System-util [redacted] ./myls
stat: 4
-rw-rw-r-- 1 20846 20846 3370 28 Sep 17:6 starter.c
-rwxrwxr-x 1 20846 20846 12872 23 Oct 14:53 myls
-rw-rw-r-- 1 20846 20846 167 28 Sep 17:6 Makefile
drwxr-xr-x 2 20846 20846 2 10 Oct 11:45 folder1
-rw-rw-r-- 1 20846 20846 101 28 Sep 17:6 README.md
-rw-rw-r-- 1 20846 20846 10406 23 Oct 14:53 myls.c
```

Figure 3: A comparison of the output of *mylist* and the output of *ls -n* within the same directory.



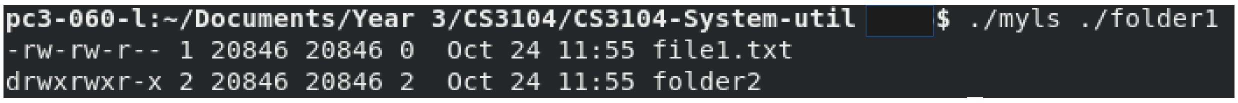
```
pc3-060-l:~/Documents/Year 3/CS3104/CS3104-System-util [redacted]$ ./myls
-rw-rw-r-- 1 20846 20846 3370  Sep 28 17:6 starter.c
-rw-rw-r-- 1 20846 20846 304426  Oct 23 15:4 CS3104 P1  report.odt
-rwxrwxr-x 1 20846 20846 12872  Oct 23 15:6 myls
-rw-rw-r-- 1 20846 20846 167  Sep 28 17:6 Makefile
drwxr-xr-x 2 20846 20846 2  Oct 10 11:45 folder1
-rw-rw-r-- 1 20846 20846 101  Sep 28 17:6 README.md
-rw-rw-r-- 1 20846 20846 10466  Oct 23 15:6 myls.c
pc3-060-l:~/Documents/Year 3/CS3104/CS3104-System-util [redacted]$ ls -n
total 316
-rw-rw-r-- 1 20846 20846 304426 Oct 23 15:04 'CS3104 P1  report.odt'
drwxr-xr-x 2 20846 20846      2 Oct 10 11:45  folder1/
-rw-rw-r-- 1 20846 20846    167 Sep 28 17:06  Makefile
-rwxrwxr-x 1 20846 20846  12872 Oct 23 15:06  myls*
-rw-rw-r-- 1 20846 20846  10466 Oct 23 15:06  myls.c
-rw-rw-r-- 1 20846 20846    101 Sep 28 17:06  README.md
-rw-rw-r-- 1 20846 20846   3370 Sep 28 17:06  starter.c
```

Figure 4: An example of error handling when a system call fails.



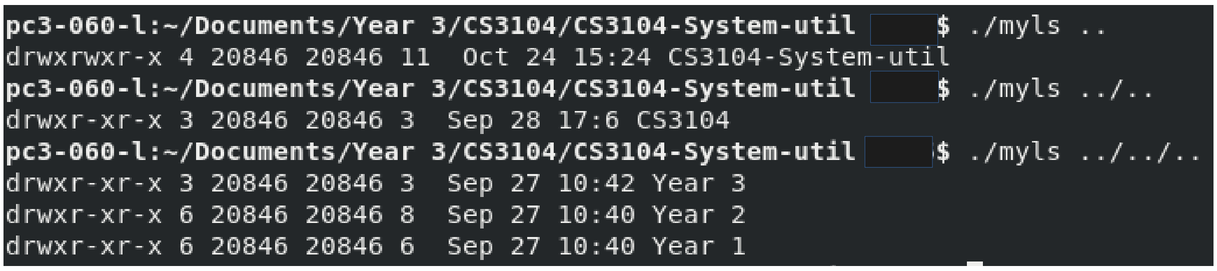
```
pc3-060-l:~/Documents/Year 3/CS3104/CS3104-System-util [redacted]$ ./myls folder1
Error: Something went wrong while calling STAT system call
```

Figure 5: An example of program functionality when accessing sub-directories.



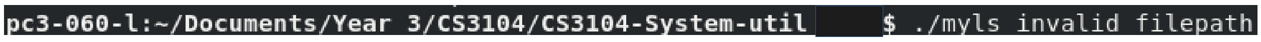
```
pc3-060-l:~/Documents/Year 3/CS3104/CS3104-System-util [redacted]$ ./myls ./folder1
-rw-rw-r-- 1 20846 20846 0  Oct 24 11:55 file1.txt
drwxrwxr-x 2 20846 20846 2  Oct 24 11:55 folder2
```

Figure 6: An example of program functionality when accessing parent directories.



```
pc3-060-l:~/Documents/Year 3/CS3104/CS3104-System-util [redacted]$ ./myls ..
drwxrwxr-x 4 20846 20846 11  Oct 24 15:24 CS3104-System-util
pc3-060-l:~/Documents/Year 3/CS3104/CS3104-System-util [redacted]$ ./myls ../..
drwxr-xr-x 3 20846 20846 3  Sep 28 17:6 CS3104
pc3-060-l:~/Documents/Year 3/CS3104/CS3104-System-util [redacted]$ ./myls ../../..
drwxr-xr-x 3 20846 20846 3  Sep 27 10:42 Year 3
drwxr-xr-x 6 20846 20846 8  Sep 27 10:40 Year 2
drwxr-xr-x 6 20846 20846 6  Sep 27 10:40 Year 1
```

Figure 7: An example of program functionality when accessing non-existent files.



```
pc3-060-l:~/Documents/Year 3/CS3104/CS3104-System-util [redacted]$ ./myls invalid filepath
```

Difficulties & Conclusion

The design of this practical was rather difficult, as I realised how much I've been relying on library functions and memory allocation so far (e.g. I initially wanted to put all file details in a linked list and implement pretty printing, which would have been easy with basic library functions, but having limited tools lead to a completely different design). I spent far more time contemplating on design decisions than writing code, which was a good choice in retrospect, as I could identify the source of errors very easily.

I also had to rely on code taken from the internet for a significant part of the practical, interpret that code to make sure it does what is needed of it, and modify it as needed. I was very concerned with accidentally plagiarising someone or not taking credit for sections I've written and changed, so keeping track of my references was an important focus during this practical.

I believe I've written a stable and complete solution to the basic requirements, although I did not have the time to attempt extensions. Given more time, I would have liked to implement more of the string library and re-implement *my_write()* as a variadic function: so that it concatenates any number of strings internally and prints them all out as one string. I also would have liked to implement other Unix commands with the use of Inline Assembly.

References

[1] Manual getdents: http://man7.org/linux/man-pages/man2/getdents.2.html

[2] Manual stat: http://man7.org/linux/man-pages/man2/stat.2.html

[3] Printing file permissions: https://stackoverflow.com/questions/10323060/printing-file-permissions-like-ls-l-using-stat2-in-c

[4] How to convert *int* to *char* in C? : https://stackoverflow.com/questions/2279379/how-to-convert-integer-to-char-in-c

[5] What is *mode_t* in C?: https://jameshfisher.com/2017/02/24/what-is-mode_t.html

[6] *strcat* implementation: https://stackoverflow.com/questions/2488563/strcat-implementation

[7] The starter code provided in *starter.c*