CS312 Lab 1

August 7, 2024

- Part A has to be shown in the lab on the same day and has to be submitted on Moodle before 4 pm. Use the command `history` to view command history.
- Part B is take-home. This will not be evaluated. You do not have to submit on Moodle.

Part A

1. Determine the following in the computer you are sitting on. Do this on a terminal.
   (a) CPU, CPU clock speed, amount of RAM, secondary storage usage. Is it a HDD or an SSD? The following files/commands may be useful: `/proc/cpuinfo, /proc/meminfo, free, df, lsblk, lscpu`. See the man pages of the commands (eg. `man df`).
   (b) What Linux distro are you running? What is the kernel version? (`uname`)
   (c) What is your ip address? Are you connected to the internet? (`ip address, ping`)
   (d) What shell are you running?
   (e) What versions of Python are installed on the computer? Which one executes when you type `python`?
2. See the man pages and become familiar with the following commands (in case you aren't): `ls, who, less, cat, cp, mkdir, mv, rm`
3. See the man page of the `wc` command. Use `wc` on the files `input.txt, input2.txt` and understand the output.
4. Perform shell redirection on the `ls` command on the `/etc` directory to form the file `input2`. (`ls /etc > input2`). Understand the difference between `>` and `>>`.
5. Use shell pipes to pass the output of `cat` to `wc`. (`cat input2.txt|wc`) How many files are present in the `/etc` directory?
6. Experiment with the commands `head` and `tail`.
7. Extract the archive `lab1code.tar.gz`. Study the code and build it.
8. Add a new program called `printlog.c` which accepts a number x as a command line argument and prints $\log_e x$. Check the validity of the input and handle errors as needed. Update the Makefile.
9. Consider the function `swap1`, having the declaration below. Write the body of the function. Explain why it cannot be used to swap the value of two variables in the calling function. Write the code to demonstrate this. This is *call-by-value.*

```
void swap1(int x, int y); /* wont work */
```

10. Fix the above program by performing *call-by-reference*, as shown below.

```
void swap2(int *px, int *py); /* should work */
```

Part B

11. Consider the declaration `int a[10];`, which declares an integer array `a` having ten elements. `a[i]` refers to the ith element of the array. If we have a pointer `int *pa;` which is set to `pa = &a[0];`, then `pa+i` is the address of `a[i]`. Thus, this means `*(pa+i)` refers to the contents of `a[i]`.

    `pa = &a[0];` is equivalent to writing `pa = a;`. As a result, `a[i]` can also be written as `*(a+i)`.

    Experiment with the above by writing a short program until it is clear. What happens if another data type (other than `int`) is used? Will the above work? How?

12. Determine the size (in bytes) of the following data types in C by writing code: `int, char, float, double`. What about a `struct`?

13. Complete the program `recprint.c` that converts a number into a string. In other words convert 32 to "32".