CSC 252 – Computer Organization

Fall 2020

The purpose of this document is to familiarize you with C and some basic Linux commands and utilities.

1. Undergraduate majors and CS MS/PhD students should already have accounts on the CS instructional machines. If you don't, visit `accounts.csug.rochester.edu` and follow the instructions there.

2. You can remotely connect to cycle machines using SSH. To use SSH, open Terminal on MacOS/Linux or PowerShell on Windows 10, and type:

   ```
   ssh <netid>@cycle1.csug.rochester.edu
   ```

   cycle1 above can be replaced with cycle2, cycle3, or cycle4.

3. Some terminal shortcuts

   - Pressing `tab` will autocomplete file and folder names
   - `Control+C` will stop execution of your current program
   - `Control+L` will clear your screen

4. Linux file pathing

   - `~` is an alias for your home directory
   - `.` is an alias for your present working directory
   - `..` is an alias for the parent directory of your present directory
   - `-` is an alias for the last directory you were in

5. Here are some basic Linux commands that we will walk through. Arguments in angular brackets are required arguments while the ones in square brackets are optional arguments.

   - `ls [dir]`: list
     Used to list files in the current directory or, if provided, the directory name.

   - `pwd`: present working directory
     Tells you your present working directory

   - `cd <dir>`: change directory

   - `mkdir <dir name>`: make directory

   - `mv <src> <dest>`: move
     Move src file/directory to dest file/directory. Can also be used for renaming.

   - `cp [-r] <src> <dest>`: copy
     Exactly like mv, but copies instead. Use -r if copying a directory.

   - `tar <options> <filename>`: tape archive
     Compression utility, just like ZIP or RAR. All assignments will be in this format. The most commonly used option will be xvf (x – extract, v – verbose, f – file input).

- `scp [-r] <src> <dest>`: secure copy
  Uses SSH to copy files to/from a remote machine. Just like cp, except you need to specify full host names and enter their password. For example,

  ```
  scp <netid>@cycle1.csug.rochester.edu:~/hello.txt hello.txt
  ```

- `rm [-r] <filename1> [filename2] ...`: remove
  Used to remove a file or directory. Need the -r option only if removing a directory. Use with care, since there is no way of recovering a deleted file/directory.

- `cat [-n] <filename>`
  Prints the content of a file into the terminal. Good for viewing small files. Use -n option to print line numbers too.

- `less <filename>`
  Provides a scrollable interface to read a file (you can use arrow keys and page up/down keys to scroll). Good for viewing large files. To quit, press q.

6. Try out the "man" command. At the prompt, type

   ```
   man passwd
   ```

   You will see the name of the passwd command, the list of possible command line options, and a description of what the command and the options do. You will find that almost every command has more options than you realized, most of which no one ever uses (try man cat!). Try also the following commands (don't bother reading all the details about every option - you won't remember them anyway).

   ```
   man less
   man ls
   man od
   ```

   You can find out more about the man command itself (which stands for "manual") by typing

   ```
   man man
   ```

7. Write a "hello, world" program in C. You can use the following program.

   ```
   #include <stdio.h>

   int main()
   // this is a comment
   /* this is also
      a comment */
   {
       int i;
       printf("Hello, world.\n");
       i = 3;
       printf("i = %d\n", i);
       return 0;
   }
   ```

   Use an editor (vim/emacs/nano) to type this into a file named "hello.c". Here are a couple of useful tips if you are using nano:

   - `^` corresponds to `Control`
   - `M` corresponds to `Alt`

8. Compile the "hello, world" program using the command

```
gcc hello.c
```

9. You should now have an executable file "a.out" which you can run by simply typing

```
./a.out
```

10. Read the man page for the compiler and find the command-line switches that will cause it to save the various intermediate files you saw in step 8 above. Take a look at these files, using 'less' to look at text, and 'nm', 'od', or 'objdump' to look at object (machine language) files. You can find out if a file is text or data by using the 'file' command with the filename as the argument.

11. Type the following lines into a file named "Makefile":

```
hello: hello.c hello.h
	gcc -o hello hello.c
```

    Be warned that the indented lines have to start with a tab character, not a series of spaces. Create a file called "hello.h" with the following line in it:

```
const int VAL = 3;
```

    Finally, edit hello.c and change it so that it reads as follows:

```
#include <stdio.h>
#include "hello.h"

int main()
// this is a comment
/* this is also
   a comment */
{
    int i;
    printf("Hello, world.\n");
    i = VAL;
    printf("i = %d\n", i);
    return 0;
}
```

    Type "make" and notice the output. You should now have a file called "hello" which you can run. Type "make" again and notice that it does not recompile hello (because you haven't changed it).

12. Use `gdb` to run one of the above programs or one of your own. You should change your Makefile so that it compiles your program with the "-g" option, which facilitates debugging.

```
hello: hello.c hello.h
	gcc -g -o hello hello.c
```

    To use gdb type "gdb filename" where filename is the name of the executable file. If you have not used gdb before, type "help" at the prompt. Experiment with setting breakpoints and single-stepping, and use the commands "list", "display", "where", and "print".

This document draws heavily from Prof. Michael Scott's Assignment 0 for CSC 254.