COEN 146 Computer Networks

Unix/Linux Commands and Overview of C Programming

General Announcements

TA email: magupta@scu.edu, anagarkar@scu.edu

TA: Mridul Gupta, Aman Nagarkar

Demoing & Grading:

- Grading will be based on
  - Whether code works
  - Programming style
  - Embedded comments
  - Observations/answer for questions
- Code (50%)
- Demo (40 %)
- Attendance (10%)

Note: Late penalties will be 15% till first 24 hours and 25% for the next 24 hours and 0 will be awarded after. Demo needs to be completed within the submission deadline unless otherwise specified.

Lab instructions:

- If needs to attend other sessions, let TAs know ahead of time
- Can finish lab early and just come to demo

Terminal/Remote into the ECC

- In general, you need to be running on a linux OS
- Recommended to SSH into engineering linux computers
  - The *username* and *password* is the same as your scu email entries
  - ssh *username*@linux.dc.engr.scu.edu
- Get file from and to linux machine:
  - sftp *username*@linux.dc.engr.scu.edu
  - cd : move around
  - put filename : put file onto remote machine
  - get filename : get file from remote machine

Terminal Basics

To navigate around use cd *dir_name*

To go to parent folder use cd ..

To learn more about a command, use man

- man *command*
- Press q to exit when done reading

Command will work with different options: -a: all, -b: buffer, -c: command, -d: debug, -e: execute, -f: file, -l: list, -o: output, -u: user

- ls: lists all files and directories (try with options: -a, -al)
- cat: displays file content (try cat file1 file2 > file3)
- mv: moves a file to a new location (try mv file1 file2)
- rm: deletes a file
- cp: copy file
- man: gives help information on a command
- history: gives a list of past commands
- clear: clear the terminal
- mkdir: creates a new directory
- rmdir: deletes a directory
- echo: writes arguments to the standard output (try echo 'Hello World' > myfile)
- df: shows disk usage
- apt -get: install and update packages
- mail -s 'subject' -c 'cc-address' -b 'bcc-address' 'to-address' < filename: sends email with attachment
- chown/ chmod: change ownership/ permission of file or directory
- date: show the current date and time
- ps: displays active processes
- kill: kills process
- sh: bourne shell – command interpreter (good to learn about shell programming)
- grep: searches for pattern in files
- Ctrl+c: halts current command
- Ctrl+z: stops current command and resumes with foreground
- Ctrl+d (exit): logout of current session

Pipe and redirect

- Pipe
  - Send output of one program to input of another
  - Can be done by using ‘|’
  - Eg. cat /dev/random | head -c 100000
- Redirect
  - Output redirection can be done by using ‘>’
  - Can be used to output to a file
  - Eg. cat /dev/random | head -c 100000 > src1.dat

Generate random file

- cat /dev/random | head -c <bytecount> > <outputfile>
  - cat: read data from a file
  - /dev/random: a special file that serve as pseudo random generator
  - head: command that output first part of file
  - -c <bytecount>: command line option to take <bytecount> bytes

Use Vi editor(optional)

- vi *filename* : open a file
- Press i to enter edit mode
- Press esc in edit mode to get back to command mode
- In command mode
  - :w : to save
  - :q : to quit
  - :wq : to save and quit
  - :q! : to quit without saving

Remember to have following information

```
# Name: <your name>

# Date: <date> (the day you have lab)

# Title: Lab1 - task

# Description: This program computes ... <you should

# complete an appropriate description here.>
```

Copy with functions

- #include <stdio.h>
- FILE *fopen(const char *filename, const char *mode)
- size_t fwrite(void* ptr, int size, int n, FILE *fp); or fprintf() (for text files)
- size_t fread(void* ptr, int size, int n, FILE *fp); or fscanf() (for text files)
- fclose(ptr);
- More info: https://www.cplusplus.com/reference/cstdio/fopen/
- modes:

"r"	**read:** Open file for input operations. The file must exist.
"w"	**write:** Create an empty file for output operations. If a file with the same name already exists, its contents are discarded and the file is treated as a new empty file.
"a"	**append:** Open file for output at the end of a file. Output operations always write data at the end of the file, expanding it. Repositioning operations (fseek, fsetpos, rewind) are ignored. The file is created if it does not exist.
"r+"	**read/update:** Open a file for update (both for input and output). The file must exist.
"w+"	**write/update:** Create an empty file and open it for update (both for input and output). If a file with the same name already exists its contents are discarded and the file is treated as a new empty file.
"a+"	**append/update:** Open a file for update (both for input and output) with all output operations writing data at the end of the file. Repositioning operations (fseek, fsetpos, rewind) affects the next input operations, but output operations move the position back to the end of file. The file is created if it does not exist.

Copy with system calls

```
#include<sys/types.h>
#include<sys/stat.h>
#include <fcntl.h>
int open (const char* Path, int flags [, int mode ]);          //access mode flags: O_RDONLY, O_WRONLY, O_RDWR
size_t read (int fd, void* buf, size_t cnt);
size_t write (int fd, void* buf, size_t cnt);
int close(int fd)
```

More info: https://man7.org/linux/man-pages/man2/open.2.html

Time with clock()

```
#include <time.h>

    clock_t start, end;

    double cpu_time_used;


    start = clock();

    ... /* Time consuming process. */

    end = clock();

    cpu_time_used = ((double) (end - start)) / CLOCKS_PER_SEC;
```

Threads

- #include <pthread.h>
- Compile with -lpthread flag
- Functions to create threads
  - pthread_create(pthread_t *thread, pthread_attr_t *attr, void *(*start_routine) (void *arg), void *arg)
  - pthread_exit()
  - pthread_join(pthread_t thread, void **retval)

Compile with GCC

- gcc *filename*
- gcc -o *executable_name filename*
- If not specified, the output name will be called a.out
- To compile a file using pthread, -lpthread flag will be used
  - gcc -lpthread -o *executable_name filename*
- After compilation, you can execute the file
  - *./executable_name [parameters]*

Makefile

- Makefile will be useful when you have multiple files to compile.
- General format:

```
target … : prerequisites …
        recipe
        …
        …
```

eg:

```
all: 1.c 2.c
    gcc -o 1 1.c
    gcc -o 2 2.c

clean:
    rm 1 2
```

- To compile all files: make
- To clean up: make clean
- For more information: go to https://www.gnu.org/software/make/manual/make.html

Steps to follow

For steps 1 and 2, create a .txt/.docx/.pdf file and submit on Camino.

For steps 3-5, create a .c file and submit on Camino.

Demo the result to TA before submission deadline.