CSC209

sean.ryan

January 2023

1 Software Tools and System Programming, an introduction

Software tools:

- Efficiently use the Unix Command Line
- Understand what the shell is, write basic shell scripts
- use the Make tool by create Makefiles

System programming

- C programming
- files
- processes
- process communication (e.g. signals, sockets)

one of the reason we are using the C program is that we can have more control over the memory. In a lot of programming languages the compiler or interpreter handles most of the data allocation, but in C we can take allocating memory to store data into our own hands. It is going to be up to use to manage all of that memory by ourselves. Keyword, garbage collection.

Considering process communication, lets think of a hypothetical question; how does a program run another program? how can files interact? how can we write a script to mod the content of a game, changing which files are selected on run?

Unix Principles

- Do one basic thing well
  - with basic variations
- Simple input formats

  - plain text
  - dont require interactive input
  - stdin to stdout/stderr
- Simple output format
  - plain text
  - expected to be an input of another tool

Demo of basic commands

- **cd [directory]** - change directory
- **ls** - a command that shows all directories and files in the current path
  - adding -l to the ls command prints more information about each file in current path
- **wc [document]** - stands for word count in this format: lines, words, character in document
- sort [document] - sorts text files alphabetically and displays the contents

Shells A shell is a program that acts as an interface or intermediary between a human and your computer's operating system.

$$\$\ wc\ hello.c$$

- the $ is a shell prompt
- the text "wc hello.c" is the command
- the first word, wc, is the name of an executable file (program) to run
- the remaining hello.c is an argument to be passed into the program
-
  - finds the executable program on your computer
  - interprets the arguments
  - runs the program with the given inputs

Standard outputs, Standard inputs, and Pipes
When wc is run without giving an argument seems like it does nothing, but instead is waiting for another input from the user. Instead of counting the words of a document, it'll count the inputted text instead.

to exit, use 'crtl d'

Consider

$$\$\ wchello.c\ >\ wc_output.txt$$

nothing seems to happen? trying running ls and you'll see a new file in your current directory has been created named wc_output.txt. this file will have the output stored into it.

piping: | is the piping operand. it'll run the left hand side of the operand and feed it into the standard input of water command on the Right hand side. for example:

$$\$\, ls\ -l \,|\, wc$$

The result will take the output from 'ls -l', and then feeding it into 'wc' program (conceptually)

the 'man' command using the 'man' command it will print the manual, giving you a reference for whatever program follows the 'man' command. for example:

$$\$\, man\, ls$$

will give you a complete manual about the ls command

2 Intro to file system

the file system is commonly thought up as a tree, with extending branches as you travel down the nodes and leaves. **Some details**

- an inode is a data structure that contains info on a file
  - metadata (e.g. owner, creation time)
  - where the file contents are stored on disk
- A directory is a special type of file that contains directory entries
  - a directory entry is a mapping from file name to inode
  - directories can also contain other directories, creating a directory hierarchy
- the root directory is the top of the hierarchy (on unix, simply seen as)

file permissions since multiple users can share the same file system, we need a way to restrict file/directory access to particular users.

levels of user access:

- user who owns the file
- group (of users) that the file is associated with
- other users

types of user access:

- view contents of the file or directory
- edit/delete files from directory
- execute a file as a program. for directories, this means to go through a folder to access sub directories

how does this translate to code? well using the 'ls -l' gives us the ability to see our permissions with regard to files in the current directory. by running 'ls -1'

```
-rw---------
```

this above is what details all permissions, group the dashes after 'rw' in 3. there are 3 dashes in groups of 3, the groups represent the levels of access in order. and the 3 dashes inside the group represent the types of user access, in order.

how to change permissions:

$$\$\ chmod\ o - x\ hello$$

the o represents the other group, the - represents subtract or take away, and the x represents the executable user access. use the 3 characters before the file to detail the changes you wish to make.

$$\$\ chmod\ 750\ hello$$

this is another way we can use chmod, and it is the numeric mode for changing permissions. What does the 750 mean? each number represents owner (7), group (5), and other users (0) which, when turned into their binary representations, translate into permissions

3 Creating programs

to run a script using the command line, assuming you've got the language, all you must do is reference the compiler installed. for example:

$$\$\ python3\ hello.py$$

will run the hello.py file using python, and output the result in the command line

python, however, is very simple. C requires a much more complex command to run a C command. first, we must compile the file with this command, using hello.c as an example:

$$\$\ gcc\ -Wall\ -g\ -std=gnu99\ -o\ hello\ hello.c$$

is used to compile the file but then we must execute it with:

$$\$\ ./hello$$

4 Arrays and Pointers

An array is a contiguously allocated set of objects of a fixed type, they do not store their own length.

A pointer is an object whose value provides a reference (or memory address) to an object of a different type. it is initialized using *.

Two key operators

- * (deference): when x is a pointer to type T, *x evaluates to the object of type T referenced by x
- (address of). when x has type T, x evaluates to the object of type "pointer to T" containing the memory location