Seng265 Midterm Review

Unix

- Unix is: multi-user, multi-tasking OS, freeware, portable, easy, security issues, remote processing, stable, large user community
- Shell responsible for communication between user and kernel
- Kernel responsible for: memory allocation, **file system**, communication, load and execute
    - Core of the OS
- .. – refers to parent directory
- . – refers to the current directory
- ~ - refers to the home directory
- pwd – display current directoy
- anything can be abstracted as a file (from programs, data, to memory)
- Use forward slash "/" to separate directory and files
- "/" is the root directory
- permissions for files and directories:
    - User (owner) has full control over permissions even if you delete all permissions of your own
    - user ("u") [-rwx------] - the file owner has full permission
    - group ("g") [----rwx---] -full permission for group
    - other ("o") [-------rwx] -full permission for others
    - all ("a") – user + group + other
        - read (r) = file to be read / directory content to be read
        - write (w) = file to be modified / directory to be modified (deleted, create)
        - execute (x) = file is executable / permission to navigate into directory
        - dash (-) = no permisions
- chmod sets the permissions of the file/directories
    - chmod =r test.c - set file to read only
    - chmod 444 test.c -read only
    - chmod a-wx, a+r test.c -read only
    - chmod u=rwx, g=rx, o=x test.c
- less test.c or more test.c -display file one page at a time
- Stream redirection
    - - 0 stdin
    - - 1 stdout
    - - 2 stderr
    - - can redirect input by using '<'
    - - can redirect by using '<' for changing input
    - - sort < kylechat.txt will output whatever is in the text file to the console
    - - echo $SHELL > a.txt will change the output stream from console to the text file
- pipes route standard output of one command into standard input of another command
- execute multiple commands by cmd1; cmd2; cmd3 etc.

- group and redirect commands together by (date; who; pwd) > logfile

    - file name expansion:
    - * - any number of characters
    - ? – exactly one of any character
    - [abc] – any character in the set [abc]
    - ![abc] – any character not in set [abc]
- Bash shell history – uses read line editing interface to show most recent commands, usually remembered across login sessions

Version control

- Repository remembers every committed change to every controlled file
- Downsides of "locking" down working copies
    - Cause administration problem
    - Awkward and inconvenient
    - False sense of security
- "Copy modify merge" – allows users to work in parallel because the changes do not overlap, consistency, files can be merged
    - - Copy-modify-merge allows users to work in parallel
    - (i) User 1 and User 2 checkout a file B from the repository
    - (ii) Both users modify their files, call them B' and B'' for user 1 and user 2 respectively.
    - (iii) user 1 commits his changes first
    - (iv) user 2 tries to commit his change but fails to as you get an out of date error
    - (v) user 2 gets a copy of the newly updated one (B'), and merges the changes of B' and B'' (his own copy) to make B*. He commits his changes after.
    - (vi) User 1 now updates his B' to ensure everything is going well.
    - - git push updates your working copy to the remote repository
    - -git pull gives you a local copy from a repository
- Git – framework for version-control system workflows
    - Tracks changes to files and directories over time
    - Remembers changes to all files
    - Allow concurrent access to repository over a network
- Working copy is a normal directory on your local system
- Git ls-files will display the files that you have committed and pushed to the remote repository
- Each commit results in a new snapshot of code in the working copy
- Git log – shows the snapshots in order
- Git status – what has changed in our working copy
- Git pull = git fetch + git merge

C programming

- No array access bounds checking
- No null-pointer checks
- No checks on uninitialized variables

- Declarations must be at the beginning of a scope
- Does **not** have Boolean values
- char *s = “test \n”; - **s is an address to a char. It is a variable holding an address to a static string table.**
- Arrays – may be statically or dynamically allocated. Static can’t grow at runtime, dynamic can
    - If array dimensioned to hold **size** elements, then the array indexed from **0 to size-1.**
    - Again, **NO array bounds checking**
    - Must know the size of array when declaration
- C does not have true and false
    - Use relational operators, equality, and logical operators instead
    - An expression = 0 is false; otherwise true
- Functions
    - int main(int argc, char *argv[]) {
    - **argv – array of strings, one for each word or quotes phrase**
    - **argc – keeps track of argv[] length**
    - double fmax(double x, double y) {
    - a type void – when the function has no return value or no parameters
- Addresses and Pointers
    - All variables are data
    - All data are in memory
    - Every memory location has an address
    - Pointers hold the address memory of a location which is stores a value of a DT
    - & - obtain an address
    - * - use an address
    - Read from right to left
    - int *a; - a is an address to an int
    - char *st[10]; -st is an array that is an address to a char
    - Get the 5th element of “grades”: &grades[4]
- Strings – memory for strings is not automatically allocated
    - Can’t merge 2 strings with “+”
    - Strings are character arrays
    - **Start of a string is an address to a char**
    - **End of a string is indicated with a \0 null character, and takes 1 space in the array**
    - strncpy(words, “the quick brown fox”, 20);
    - pw = &words[0] is the same as pw = words;
    - strncat – to merge 2 or more strings together (similar to “hello” + “world” in java) **– unlike strncpy, this will properly cap the resulting string with a null \0.**
    - strncmp – compares 2 strings that returns either a zero, or positive int
    - strlen – length of string (**not counting \0 null at the end**
- File io
    - fopen(test.txt, “r”) – open file according to filename
    - fgets (char ***buff**, int **n**, FILE ***stream**) – read at most **n-1** characters from **stream** and copy to location **buff**
    - fprintf (stderr, “input the wrong file”)– like printf but the output goes to the stream

    - fclose – closes the stream
- Words are separated by whitespaces
- Tokenization – to extract text from an array
    - strtok(char* what_to_tokenize, string_w_separators); **use this first load up string**
    - strtok(NULL, string_w_separators); **use this to get more tokens from the same string**
- Program scope
    - Variable exists for the programs lifetime, can be accessed from any file
- File scope
    - Variable is visible from declaration to end of file
    - Define variable file outside a function with **static keyword**
- Function scope
    - Variable at the beginning of a function to the end
- Block scope
    - Variable from declaration to the end of the block
- Continue – starts the next loop iteration checking the while condition
- Break – exit loop immediately, resume after the while body
    - Continues exec at closing brace

```
char buff[50];
char *cursor;

cursor = &buffer[0]; and
cursor = buffer; are the same!
```