Midterm (and some other stuff I choose to add in depending on how I structure this):

Key Definitions:
Kernel: Manages task-switching, process execution, and process management. It is always loaded in RAM. Communicates with hardware. Heart of OS
Shell: User Interface to interact with the OS. It consists of shell memory, command-line prompt, interpreter.
File System/Manager: Manages and defines file structure. FAT (file allocation table), API (application programming interface). Translates between disk and OS. Tree directory/stores commands.(Part in RAM, part is disk)
Utilities: Libraries, minimize the kernel size. Allow more commands can be added by programmer or 3rd party developers(extensible). Not all utilities are always needed so user can call them up when needed.

Utilities are on Hard Disk: because it minimizes "Kernel size", frees up RAM, third party commands and drivers don't clog it up. Also, utilities can and should be modified. The kernel is usually left alone.

Operating System: Manages computer hardware and software resources, providing common services across several programs. It is the middle-man between the user, the program, and `low-level hardware.

- **Security:** Manages passwords, encryption and file access.
- **User Interface:** Manages the Graphical User Interface (GUI), the command line interface, and the shell.
- **Memory manager:** Finds, formats and allocates memory.
- **Disk/Storage Manager:** Secondary memory manager.
- **Process Manager:** Runs and terminates programs, as well as dealing with multi-tasking.
- **Network Manager:** Manages LANs and all communications.
- **Hardware Manager:** Manages drivers, assembler, and polling.

System: A collection of sub-systems in order to create one functioning and complete application.
Sub-System: A Program that depends on another program.
Internet Protocol (IP) address: A numerical label assigned to each device within a computer network.
Network: Wire that allows communication between computers.
Local Area Network (LAN): A collection of computers or systems in the same room or building.
Wide Area Network (WAN): A collection of computers or systems connected wirelessly through one or more major network nodes. Different buildings or cities.
Internet Service Provider (ISP): Knows the location of all other Internet servers. Utilizes this to correctly deliver information packets.

The Web: A subset of the Internet dealing solely with the ISP servers. It interacts only with the browsers.

Email:

- Provider does not know how to send emails
- From Address, To Address, Message Size in Bytes, Email Message, Data Validation Code
- The mail is “sealed in a packet” and sent into the network.

Client/Server:

- Client sends a request packet to server for something
- Server attempts to find requested program or data
- Server sends a reply packet with data or program
- Client copies info into memory and executes it
- Handshaking: agree on packet format & passwords
- Comm-error: a packet was lost, resend request or time-out

Packet: carry the data in the protocols that the internet uses

Device Driver: Allow external devices to communicate generically and effectively with the computer.

Unix: Simple and optimized OS. It is driven via a command-line interface and supports a client-server architecture.

Unix Session: a session is a period of time monitored by the OS starting at login ending at logout based on the user’s name. It defines the user’s current interaction with the OS.

Root: A special administration account on a Unix system which has all permissions. (denoted with \)

Run-time Environment: when you run it on a computer. Output of a programming environment

Software Environment: coding, debugging, vim…

Switch: A parameter that modifies how a program will function.

Argument: Values which are input into a program.

Directory: Use the same permission system as files, in order to list and access them.

- **/etc:** Configuration files and passwords.
- **/bin:** Executable file related to the OS.
- **/usr:** Installation directory.
- **/opt:** Other installation directory.
- **/dev:** Contains device files for connected hardware.
- **/var:** Contains variable system files.

Relative Path: cd (in current directory), is variable depending on current directory.

Absolute Path: cd /usr/... , directory structure starting from the root directory.

Permissions: A set of rights (read, write, execute) for a file or directory which exist at 3 levels (user, group, all/other).

rLogin: “remote login”, connecting to a Unix machine, using a login name and password, from a distant machine.

sLogin: “secure login”, can be accessed from anywhere.

SSH: “secure shell”, providing a secure channel to an unsecured network, connecting an SSH client to an SSH server.

Tar Files: Combine multiple files into a single file

- **-c:** Create a new TAR archive.
- **-r:** Update a TAR archive.
- **-x:** Extract contents from the TAR archive.
- **-f:** Specifies the TAR filename. Will be a generic name if this switch is left out.
- **-v:** Allows the TAR command to output information, updating the user.
- **-z:** Compresses the TAR archive into a .zip file using gzip.

Tar file declaration examples:

- tar -cvf log.tar *.log → create an archive file, specify the name, verbose.
- tar -zcvf log.tgz *.log → create a zip archive file from the created TAR file, verbose.
- tar -xvf log.tar /tmp/log → extract an archive file
- tar -zxvf log.tgz /tmp/log→ extract a zip archive file

Command-Line Commands:

- **passwd**
- **mail**
- **news**
- **logout**
- **exit**
- **ctrl+d**
- **who**
- **man TOPIC**
- **write USER MESSAGE**
- **ls PATH -l -a**
- **mv FROM TO**
- **mv OLDNAME NEWNAME**
- **cp FROM TO**
- **rm FILENAME**
    - **-i:** wait for confirmation (cp, mv, rm)
    - **-r:** recursively visit a directory (cp,rm)
    - **-f:** do not prompt (mv, rm)
- **cd PATH, cd/, cd ..**
- **pwd**
- **mkdir PATH/DIRNAME**
- **rmdir PATH/FILENAME**
- **cal YEAR**
- **cat FILENAME**
- **more FILENAME**
- **>** (redirect output) (cannot redirect output into a different program, that’s what | is for)
- **>>** (concatenate to a file)
- **<** (use file as input)

- <<
- | (pipe output as a new input)
- **chmod SWITCHES FILENAME**
    - **ex:** chmod g+rw FILEPATH
    - **ex:** chmod -r-x--x--- FILEPATH, chmod 101001000 FILEPATH, chmod 510 FILEPATH are all equivalent
- **echo**
- **head FILENAME**
- **more FILENAME**
- **less FILENAME**
- **tail FILENAME**

Editor: Allow creation and editing of files. Can be text-based or graphically-based.
Vim (Vi improved):

- **Edit Mode:** i (insert after), a (insert before), o (insert new line after), O (insert new line before)
- **Escape Mode:** ESC: dd (delete line), x (delete character), / (find)
- **Command Mode:** : while in Escape Mode. w (write), wq(write, quit)

Grep: Search for occurrences of an expression in a file.

- **-i:** Ignore case.
- **-c:** only the number of lines that match.
- **-v:** Display the lines that do not match.
- **-n:** Display line numbers.
- **-l:** list filenames where matches were found.

Script: A collection of commands grouped into a single file. Not compiled by the CPU, but interpreted by the OS.

- **Boot script:** Created by super user for all.
- **Login script:** created by owner for personal use.
- **Command-line scripts:** Automate command-line activities.
- **Backup scripts:** Archive and save files.
- **Startup script:** Run when a session begins.
- **Scheduled scripts:** Archive logs in a preset scheduled manner.
- **Maintenance scripts:** Creates a user or changes a password.
- **Programmer scripts:** Compile, control, and backup in one command.

#! (sha-bang): Indicates that the script is directly executable, followed by the name of the shell.
Escape Characters (Bash):

- **\:** turn off special meaning. ie \”
- **\n:** new line

Positional variables: Store the parameter used to start the script.

- **$#:** Number of arguments on the command line.
- **$-:** Options supplied to the shell.
- **$?:** Exit value of the last command.
- **$$:** Process number of the current process.

- **$!:** Process number of the last background command.
- **$x:** (x=0-9); arguments on the command line, in order.
    - **$0:** Name of current shell.
    - **shift:** will shift the arguments by one. ie. $3 > $2, $2 > $1
- **$*:** All arguments.
- **$@:** Some arguments, separately quoted.

Quotes:

- **None:** Depends on the shell
- **‘’:** Execute as is.
- **“”:** Pre-process first.
- **``:** Execute command, output is interpreted as a string.

File tests:

- **-r file:** True if it exists and can be read.
- **-w file:** True if it exists and can be written.
- **-x file:** True if it exists and can be executed.
- **-f file:** True if it exists and is regular.
- **-d name:** True if it exists and is a directory.
- **-h file:** True if it exists and is a symbolic link.

String tests:

- **-z string:** True if the string length is zero.
- **-n string:** True if the string length is non-zero.
- **string = string:** True if they are identical.
- **string != string:** True if they are not identical.
- **string : :** True if it is not NULL

Integer tests:

- **n1 -eq n2 :** True if n1 and n2 are equal.
- **n1 -ne n2 :** True if n1 and n2 are not equal.
- **n1 -gt n2 :** True if n1 is greater than n2.
- **n1 -ge n2 :** True if n1 is greater than or equal to n2.
- **n1 -lt n2 :** True if n1 is less than n2.
- **n1 -le n2 :** True if n1 is less than or equal to n2.

C

#include <stdio.h>: Imports printf, scanf, and other I/O functions.
#include <stdlib.h>: Imports constants, simple functions, and a random number generator.
GCC Options:

- **-o filename:** Specify the name of the executable file instead of a.out
- **-v:** Verbose mode.
- **-w:** Suppress warning messages.
- **-W:** Extra warning messages.
- **-Wall:** all warning messages.
- **-O1:** Optimize for size and speed.

- **-O2:** Optimize further.

Data Types:

- int (16-bit)
- double (64-bit)
- float (32-bit)
- char (8-bit)
- char * (string in Java) (32-bit)
- boolean (binary int: 0=false; everything else is true)

Printf control codes:

- **%d:** signed integer
- **%c:** character
- **%s:** string
- **%f:** float
    - %5d - Prints an integer with 5 blank spaces on the left
    - %-5d - On the right

printf(Control code, variable name): Prints the specified strings and variables. variables must be declared in order of appearance in the string.

scanf(Control code, &variable name): Reads input from the command line during execution, copies input value into a memory address. Don't include the &variable if it is an array

- scanf("%d", &z);
- scanf("%s", stringname);
- **NO OBJECTS = NO CLASSES**

typedef: Creation of custom types.

ex: *typedef int scalefactor;* a new variable of type *scalefactor* can be declared and will be treated like an *int* in the printf and scanf commands (that is, %d)

ex: can also be used to create *booleans* of type *int.*

Scope: In C, the scope of a variable is either limited to the function in which it is declared, or can be called upon everywhere below its declaration if it was not declared in a function.

Coercion: Force a variable of one type to become another type. Sometimes implicit, but specification should be added just in case.

- **Char:** Char values are stored as ints corresponding to their placement in the ASCII table. 'A'=65; 'Z'=90; 'a'=97; 'z'=122.

Escape codes:

- **\n:** New line.
- **\r:** Return.
- **\t:** Tab.
- **\a:** Bell.
- **\b:** Backspace.
- **\\:** Backslash.
- **\":** Double quote.
- **\0:** Null.

getchar(void): retrieve one character from STDIN

putchar(int): display one character to STDOUT

gets(char array): read a string into an array
fgets(char *str, int size, FILE *stream): read a specific line from a stream and store it into str
Recursion is valid in C
//This is a comment
Libraries:

- **Stdio.h:** printf, scanf, gets, fgets, etc.
- **Math.h:** sin, cos, tan, pow, etc.
- **String.h:** strlen, strcmp, strtok, etc.
- **Ctype.h:** toupper, tolower, isnum, etc.
- **Stdlib.h:** atof, atoi, null, exit functions, etc.

Libraries are made from 2 files: .o and .h
#define: create a macro associated with a token string. This substitutes the token string in all occurrences.

- **#define NAME EXPRESSION**: a certain work will be replaced by the specified expression after pre-processing
- **#define MACRO EXPRESSION:** ^ditto^, but it includes a parameter separated list
- #define is simply text replacement and does not require additional memory, but it is harder to type check it, use const if it is more fitting.

#include: insert files into the source, at the same point where they are included.

- **#include <FILENAME>:** include a built in library, see above for the list
- **#include "PATH/FILENAME":** any filename and path. Used to include functions from other files, for example.

Post-Midterm (or however I decide to organize this):

Pointer: Unsigned integer variable that stores an address

- Can be of any variable type
- Less restrictions
- The address of a pointer refers to its location in memory
- **&:** give the address of a variable
- ***:** dereference operator, give the contents of an object pointed to by a pointer
- A void pointer can point to anything, other pointers are restricted

	content	Adress of
Int a	a	&a
Int *p	*p	p

Reference: point to an object
Strings are static. Arrays are dynamic.
Primitive values are passed by value

Arrays are passed by reference

- **scanf("%s", array);** put char by char into the array
- **gets(array);** ^ditto^
- **fgets(array, int, stdin);** ^ditto^ until int values are input
- **scanf("%s %s %s", array, array,array);** reads input separating by spaces into each array
- **Ex:** a swap functions must use pointers and addresses in order to be visible to the calling function

Binary Search: *see COMP 250*

The pre-processor: expands directives (such as #include and #define)
Conditional inclusion:

- Make sure a header is included only once
- Adapt code to different language
    - **#ifdef NAME**
    - **#ifndef**
    - **#else**
    - **#endif**
    - The four keys above are used to define a definition if/else conditional block. This can selectively compile the program and is used in multi-language compilations

Modular programming: allows for multiple files to operate together in a single functioning application. [RE-WORD]
Header files: contain pre-processor commands, type and variable declarations, and function prototypes
Extern: permits one and only one named-space to access the data in another space

- **extern TYPE VAR_NAME**

Make:

- Automatically determines certain parts of the code that need to be recompiled
- Can be used in any language
- Prevents the lengthy process of re-compiling hundreds of files.
- In the creation of the makefile, consider what needs to be recompiled if something changes.
- Target : Dependencies
  Commands
- **Implicit rule:** update a .o file from a correspondingly named .c file

Unit: many files compiled into a single file
Project: many units linked to one a.out file
Repository: Database that stores all versions of a file. Permits backtracking and branching.

- Keep source files in a database, all versions
- Can merge source files, if 2 edits occur simultaneously
    - Add, commit, push

RCS: a sub-folder to store all files and databases

- **Check in (regular):** delete and commit file

- **Check in (-u):** does not delete, converts to read-only
- **Check out:** get the most recent version as a read-only file
    - Or get an editable file and lock anyone out of editing until returning to the database
- **Check an old version:** -co -l -rVersion filename
- **Check in with a new version #:** -ci -rVersion filename
- **Delete a portion:** rcs -oVersionRange filename
- **See database:** rlog filename

States:

- Experimental
- Stable, Released
- Obsolete

Access Rights:

- **Add users that can access:** rcs -usernameList filename
- **Copy from another file:** rcs -oldFilename newfile
- **Delete names:** rcs -eUsernameList filename

Git:

- **Basic**
    - **Init**
    - **Add**
    - **Commit**
    - **Push**
    - **Pull**
    - **Clone**
    - **.gitignore**
    - **Status**
    - **Rm**
    - **Mv**
    - **Diff**
    - **Log**

  Branching
    - **Branch new_branch**
    - **Branch -b feature2**
    - **Commit**
    - **Checkout master**
    - **Merge**
    - **Rebase**
    - **Branch -d**

  Tagging
    - **Merge branch_name**
    - **Tag -a name**
    - **Tag -l**
    - **Push --tags**

Forking: //marked as advanced, not sure if we need to know this

GNU: Open source programming tools

- Make
- CVS, SVN
- Debug
- GProf

Bug: a fault in a program which prevents it from working as intended. Errors can range from incorrect answers to loss of data to, in extreme cases, loss of life.
Debug: Finding and fixing a bug.
Debugger: A mode which allows the programmer to examine the contents in memory, line by line.

- It can halt the program and list the source code
- Print the type of a variable
- Jump to any line
- Can be used on an already crashed process.

GDB is a debugger for C, C++, Java, Assembly, and others.

- Use the -g flag during compilation in order to use GDB, as additional information is included in the compilation
- Use *gdb executableName* in order to use the debugger
- *list* to view the source code
- *step* and *next* allow the user to run the program line by line
    - *step* enters functions
    - *next* goes line by line
- All commands:
    - Quit
    - List
    - Run
    - Backtrace
    - Whatis
    - Print
    - Break
    - Continue
    - Watch
    - Set
    - Ptype
    - Call
    - Where
- Managing breakpoints:
    - Delete
    - Clear
    - Disable
    - Enable

    - Enable once

Profilers: dynamic performance analysis tools

- Record data to determine which sections of a code need optimization

GPROF:

- Use the -pg flag in compilation
- Creates a gmon.out binary file
- Gprof (*flags*) a.out gmon.out > textfile
- Flags:
    - -b (not verbose)
    - -s (merge all)
    - -z (functions not called)
- Command line flags:
    - -a (supress printing private functions)
    - -e *function* (do not print this functions info)
    - -E *function* (do not count time in this function)
    - -f *function* (only print this function's info)
    - -F *function* (only count time in this function)
    - -v (print version number)
    - -z (mention all functions)
    - -b (not verbose)
    - -s (summarize info in gmon.sum)

Flat Profile: output of GPROF

- Table containing all called function in a table, with columns corresponding to:
    - %time
    - Cumulative seconds
    - Self seconds
    - Calls
    - Self ms/call
    - Total ms/call
- This table gives the amount of proportional time within each function in order to determine slow-running processes. Use this to your advantage.

File structure: a file is stored in a straight line in memory

Stdio.h: contains functions for calling to files

- fopen(const char* file, const char *mode)
    - Returns a FILE pointer, which contains information such as location, current position, and errors
    - Mode indicates type of access ("rt", "wb", "r+wt", etc.)
    - Returns null in case of error
- Int fscanf(FILE *fp, char *format, ...)
- Int fprintf(FILE *fp, char *format, ...)
- Char *fgets(char *line, int maxline, FILE *fp) - get line by line
- int fputs(char *line, FILE *fp) - write a new string into the file

- Int fgetc(FILE *) - return value of char at ptr
- Int fputc(int c, FILE *) - write a value at the pointer
- Int fclose(FILE *fp)
- Int feof(FILE *) - return 1 if end of file

System(): executes a command in the shell

Fork(): execution of both parent and child processes

- Must #include <sys/types.h> and <unistd.h>

Struct: Composes variables into a single record

Union: Merges variables into a single variable

Typedef: define a new type identifier

- Cuts out the need to specify the struct at every declaration

Enumerated types: list of constants that can be referred to as integers

- Enum season {summer, autumn, winter, spring};
- Summer has value 0, autumn 1, …
- Can be overridden, any value can be assigned as you wish

Dynamic memory:

- Void *malloc(int numberOfBytes);
- Void *calloc(int repeat, int numberOfBytes)
    - These return a void pointer, must be cast before being used
- Void free(void *ptr) to release allocated memory
- **->** is the dereference operator, to access members of a data structure

Data structures:

- **Linked list:**
-
    - Starts with a pointer and ends in NULL, composed of one or many nodes
    - A node is composed of a data portion and a pointer to the next node
    - A node is a struct
- **Binary tree:**
    - Pointer points to the head of the tree, every data box is a node
    - A node is composed of a data portion, a left pointer, and a right pointer
    - Again, a node is a struct

Arrays are static, in order to change their size, must use calloc(), copy the array into that space, and free() the old array.

Linked lists are created and modified cell-by-cell

Software system: application made from many different languages and technologies working together in order to provide a common tool.

Introduction to web technology

Internet as a system:

Client PC -> Browser ---> CGI/HTML <--- Web Server(Apache) <- PC <- Database

Browser interprets the language stack. A reader function takes text, analyzes it, and gives it to a display function.

Server Side: STDIN (packet for server to browser) and STDOUT (packet for user to server app)
Server HTTP Status Codes:

80: normal HTTP
443: SSL HTTP
200: ok
401: unauthorized
403: forbidden
404: not found
500: internal server error

HTML Programming:

Tags: html, head, title, body
Attributes: h1-6, p, br, b, u, i, big, small, em, strong, blockquote, q, cite, ul, ol
<img src="URL" width="int" height="int">
text
Th, td, tr for tables
<!-- comment -->

Character entities: &#int;

- 160 for space
- 60 <
- 62 >
- 38 &
- 34 "
- 39 '
- 169 copyright
- 174 registered
- 247 division

Internet: All computers connected together using the technology IP
Web: A subset of the internet. It is only the public folders.

CGI: Common Gateway Interface

- <form name="input" action="script.py" method="get/post">
- **Get:** stores in URL, logged by server, shell memory
    - Allocate a pointer to it, use sscanf or parse char by char
- **Post:** transfers the data in query packet, stdin
    - Allocate an int to getenv("CONTENT_LENGTH"), place char by char into an array?
- <input type="text" key="name">
- <input type="radio" key="name" value="value">
- <input type="checkbox" key="name">
- <input type="submit/reset" value="value">
- **Output:** printf("Content-Type:text/html\n\n")

Python: scripting language. Interpreted, not compiled.

Object oriented

- def NAME(parameters) :
  STATEMENTS
- “““comments”””
- Functions are objects
- **Importing:** use the filename, or an object from a file
  - Form = cgi.FieldStorage()
  - if form.has_key(‘x’):
    theValue = form[“x”]

Interpreted line by line, much like Bash, although a main function can be declared if so desired

Dictionary: data type consisting of of key-value pairs

- dic_name={“key”:value,”key”:value}
- dic_name[“key”]=value
- clear(), copy(), get(key), get(key, value), has_key(key), items(), keys(), values()
- Mutable

List: data type that stores objects

- name=[‘string’, int]
- append(o), count(o), index(o), insert(i, o), pop(), pop(i), remove(o), reverse(), sort(), sort(f)

Tuples: like lists, but are immutable

- Tuple=”string”, int

No variable declaration

Anything 0 or empty is false

Strings:

- **‘** : text as-is
- **“** : processed, like in printf
- **“““** : comments
- %s for formatting, concatenate with +

Mapping: li = [1, 9, 8, 4;]li2 = [x*2 for x in li]

.join: link 2 lists

```
if <COND>:
        STATEMENTS
else:
        STATEMENTS

for X in Y_LIST:
        STATEMENTS

for I in range(INT,INT-1):
        STATEMENTS

while <CONDITIONS>:
        STATEMENTS
```

Try:
    REF=open(“FILE”,”TYPE”)
Except IOError:
    statements
Else:
    Statements
Finally:
    Statements

Command-line arguments: import sys, then use sys.argv[1], as argv[0] is the name of the script

Object definition:

Class NAME (PARENTLIST):
    Def __init__ (self, PARAMETERLIST): ← like a constructor in Java
        Statements

When defining attributes:
    _NAME is private
    NAME is public

Sys:

- exit()
- Argv
- Path
- ps1

OS:

- Name
- Systen
- Mkdir
- getcwd

Re:

- Search
- Match
- Findall
- Split
- Sub, subn

Time:

- Time
- Gmtime
- Localtime
- Mktime
- Sleep

Random:

- Randint

- Sample
- Seed

Math:

- Sin, cos, …
- Log
- Ceil, floor
- *Pi, e