Systems Programming Final Exam Review

Topics Before Midterm:

Strings:

- Strings are created using a char array or a char pointer. A char [] is modifiable, is created within the scope of a function, and is written on the stack. A char * is different because is immutable and is a pointer to a constant string that the compiler saves in a block of memory.
- The size of a character is 1 byte but don't forget that you need a null terminator. Therefore, the smallest string is 2 bytes.
- You use a string if you want to store multiple characters in one variable. Very helpful for buffers and the like.
- If you want to access parts of the string, simply use [ ]

Unions:

- Unions are used to store different data types in the same memory location.
- It can have multiple members but only one member can have a value at a given time.
- Creation:
    - Union Data {
      Int i;
      Float f;
      Char str[20];
      } data;
- The size of the example above is 20 because it allocates space for the member that requires the most bytes. Since char str[20] requires 20 bytes, it is the most.
- To access members of the union, we use the dot operator.
    - Data.i = 10;
    - Data.f = 220.5
- If we try to print out f when i is declared, then the compiler will print out a garbage value.
- We can have pointers to unions and you can access them using the -> operator (union data * p2 = p1)
- It is useful when you want to use the same memory for two or more members.

Structures:

- Creates a data type that can be used to group items of possibly different types into a singly type.
    - Struct address {
      Char name[50];
      Char street[100];
      Char city[50];
      Char state[20];

        Int pin;
        };
- When a data type is declared, no memory is allocated for it. It is only allocated when the structure variables are created.
- You access members using the dot or -> operator.
    - -> is typically used for pointers. It is the same thing as (*foo).name

What is the difference between a structure and a union?
- Structures allocate enough bytes for its size greater than or equal to the sum of the sizes of its member. It also has each of its members within a unique storage area and you can alter the value of any of the members.
- A union is equal to the size of the largest member. It is shared by individual members of a union. You cannot alter more than one member at a time.

What are enums?
- A user defined type where you can assign name to integral constants to make a program easy to read and maintain.
    - Enum State {
      Working = 1;
      Failed = 0;
      };
- Two enums cannot have the same value.
- If values are not assigned, the compiler will assign them values such as 0, 1, etc. all the way to n.

What are pointers?
- A pointer is a variable which contains the address in memory of another variable. We can have a pointer to any variable type.
    - & gives us the address of a variable
    - * gives us the contents of an object pointed by that pointer.
    - What is a pointer to a pointer?
        - A pointer to a pointer is a form of multiple indirection. The first pointer contains the address of the second pointer, which points to the actual value.
    - What are some operations you can do on a pointer?
        - increment
        - Decrement
        - Comparisons (==, <, >)
- What are void pointers?
    - A pointer to void represent the absence of type. They point to a value that has no type. They can point to any C type.
- What are function pointers?
    - Declaration: int (*foo)(int)
        - Void (*fun_ptr) (int) = fun;
    - A function pointer points to code, not data.
    - We do not allocate or deallocate memory using function pointers.

    - It can also be passed as an argument (like with thread)
        - This is called a wrapper
    - Very useful for qsort where you pass in a comparator function and the library sort sorts an array for you based on those conditions.

What is the difference between the three placements of const?

- Const char * p - cannot change the value that p points to so (*p = ‘b’ is not allowed)
- Char const * p - cannot change the value that p points to so (*p = ‘b’ is not allowed)
- Char * const p - cannot change the address that p points to (*p = ‘b’ is allowed)
- Const char * const p - cannot change the address or the value that p points to (*p is not allowed)

What is the difference between the stack and the heap? What are the text sec, data sec, and BSS?

- The stack is used for static memory allocation (such as ints, chars, etc) while heaps are used for dynamic memory allocation (structures, unions, objects, etc.)
- Variables on the stack are stored directly to memory while on the heap, they are allocated at run time.
- Elements of a heap can be accessed at any time, whereas for a stack you have to wait till the function is returned.
- Stacks are Last In First Out Data Structures.
- The text sec contains instructions of a program.
- The data sec contains initialized global variables.
- The BSS (Block stored by symbol) contains uninitialized global variables.

What is malloc?

- It allocates a requested size of bytes and returns a pointer to the first byte in the memory block.
    - What is memory saturation?
        - All the memory on a machine is used for dynamic allocation and cannot be used for future mallocs.
    - What is fragmentation?
        - When most of your memory is allocated in a large number of non contiguous blocks, leaving a good amount of memory unallocated. When a request for a large block is made, the request fails even though the total amount of free memory is larger than the requested block size. There are ways to deal with this problem. First, one part of your memory resource can be reserved for small blocks, leaving the rest available for larger blocks. Second, large blocks can be allocated from one end of your memory resource and small blocks can be allocated from the opposite end.
            - Best fit - traverse the list until you find the block with the least wasted space (closest to the amount asked for allocation)
            - Worst fit - traverse the list until you find a block with the most space.
            - First fit - look for the first free block to be allocated.

What are the dangers of realloc?

- Realloc changes the size of the the previously allocated space by freeing the pointer. This is dangerous because old contents can sometimes be left somewhere in memory. Also, since it moves memory around, old pointers to that memory become invalid and could cause the program to crash.

What is errno?

- It is short for error number. It is used to report and retrieve error conditions through error codes stored in a static memory location. You can use strerror to get a human-readable string for the error number and perror to print that.

What are segmentation faults?

- They occur when the program is trying to read or write to an illegal memory location.
    - They are caused by dereferencing null pointer (memory management)
    - Attempting to access a nonexistent memory address
    - Attempting to access memory the program does not have the rights to
    - Attempting to write read-only Memory
    - Dereferencing or assigning to an uninitialized pointer
    - Stack or buffer overflow
    - Array out of bounds error

What are the GCC Stages of Compilation?

- Preprocessing
    - it expands macros and modifies the source code (directives, constants, and macros).
    - If we have any header files, the compiler will include the entire code for that library like stdio.h.
    - Comments are removed.
    - All macro names will be replaced by their values.
- Compilation
    - Source code gets converted to assembly code.
- Assembler
    - Assembly code gets converted to machine code (like 0s and 1s)
- Linking
    - An executable is created where all the code of the functions and addresses of those functions will be included.

What are makefiles?

- Makefiles are used to execute commands that builds an object file.
- An object file is the real output from the compilation phase and it takes all the symbols and allows a linker to see what symbols are used. The linker combines object files and gets it ready to create an executable.

What are macros?

- Text substitution with variables
- For example, #define SQUARE(x) (x)*(x) will replace all instance of SQUARE(x) with (x)*(x). If we don't include the parenthesis then the correct response will not be outputted.

What are some important Linux Terminal Commands?

- Man - gives you access to system's manual pager or reference manuals
    - Man ls - displays the manual page for ls
    - Man -a intro - looks for all available intro manuals
    - Man -k printf and man -f printf - finds short descriptions of any of the commands found with a name like that
- Top - gives an interactive command line application that contains statistics on processes and resource usage and currently running processes.
    - It shows the current time, uptime for the system running, memory usage, tasks, and CPU usage.
- Ps - produces a list of currently running processes on your computer (not real time)
    - It is commonly used with grep, more, and less commands
    - PID means the process ID which identifies the running process
    - TTY is the terminal type (eg. bash)
- Ls - lists the contents of the current directory.
- More - view text files or other output in a scrollable manner and displays text one screenful at a time.
    - Space - scrolls the display one screen full of data at a time
    - Enter - scroll one line at a time
    - B - scroll one line at a time
    - / - search for test
    - You can use it with grep which can search through a file for you and use more to see instances of that search
- Less - does not have to read in the input file like more
    - Faster and you can scroll backwards and forwards
- Grep - searches one or more input file or a stream of input and can be used to search for patterns
- Cat - create single or multiple files, view contents of file, concatenate files, and redirect output in terminal
- Tail - shows you the last 10 lines of a file by default or a specified value
    - Tail -<number of lines> <file name>
- Head - shows you the first 10 lines of the file by default or a specified number
    - Head -<number of lines> <file name>

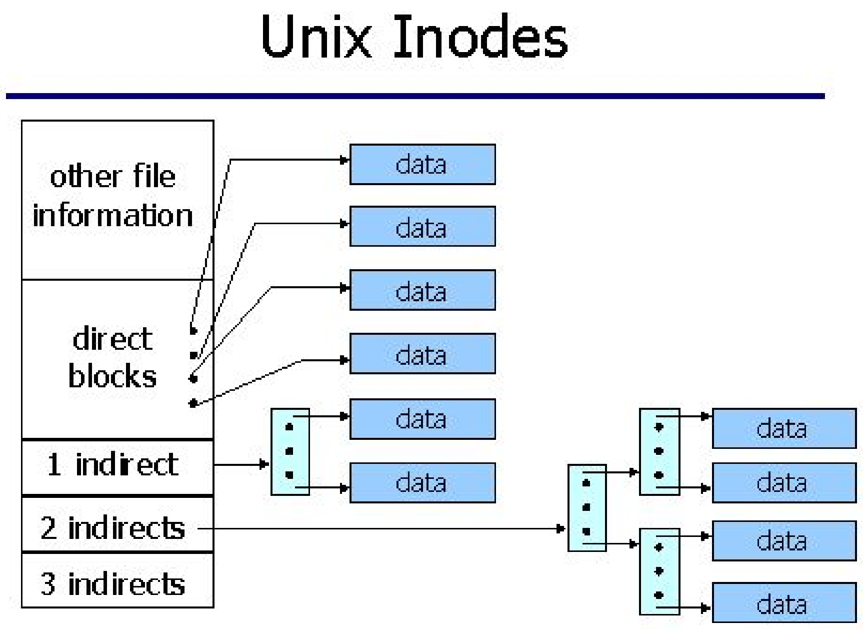
Post Midterm Material

What is segmented paging?

Segmentation refers to dynamically size chunks and paging refers to statically size chunks. The segmentation part is the chunks and the chunks are of size page.

- What are i-nodes?
    - A data structure in a Unix file system that describes a filesystem object such as a file or directory.
    - This includes metadata on the following
        - Mode/permission
        - Owner ID

        - Group ID
        - Size of the file
        - Number of hard links of the file
        - Time last accessed
        - Time last modified
        - Time inode last modified
    - The associated inode table contains a listing of all inode numbers and when a user searches for a file, the file system looks through this table to find the file.
    - Directories are just files that have a few additional settings in their inodes. It's basically a file with another file in it.
    - Inodes store 15 data block addresses or pointers
        - 12 are direct pointers to the first 12 data blocks of a file.
        - The 13th is an indirect pointer to a data block that contains addresses to the next data blocks of a file. This is a layer of indirection.
        - The 14th is an indirect pointer with two layers of indirection.
        - The 15th is an indirect pointer with three layers of indirection.
        - This massively increase the number of bytes you can store.



    -
    - Why do we need inodes?
        - Data for a file may be stored in different locations on the hard drive so this gives a direct mapping to where the data for the files are since a file is essentially just a bunch of bytes.

File Permissions
- What are the three different types of permission groups?

        - Owner - only the owner of the file or directory, they will not impact the actions of the other user
        - Group - only to the group that has been assigned to the file or directory
        - All users - anyone on the system has permissions to the file
    - What are the three different permission types?
        - Read (r) - user's capability to read the contents of the file
        - Write (w) - user's capability to write or modify a file
        - Execute (x) - user's capability to execute or view a file
    - How can you check permissions from the command line?
        - You can use the "ls -l" command from the terminal and that will list all the files and permissions displayed as: _rwxrwxrwx 1 owner:group
            - What does the permissions string mean?
                - The underscore is the special permission flag (varies)
                - The first rwx is for the owner's permissions
                - The next rwx is for the group's permissions
                - The last rwx is for all users permissions
                - The number (in this case 1) is the number of hardlinks to that file
    - How can you modify the permissions of a file?
        - Chmod can be used to edit permissions from the command line
        - There are two ways to do this:
            - Explicitly defining permissions
                - Use Assignment operators + and - to take add or remove specific permissions
                    - U = owner, g = group, o = other, a = all users
                    - R = read, w = write, e = execute
                    - If you type in chmod a-rw <filename>, that removes permissions
                    - If you type in chmod a+rw <filename>, that adds permissions
            - Using binary references to set permissions
                - rwx, rwx, rwx is the same as 111, 111, 111, etc.
                - You would do chmod xyz, where:
                    - X = binary to decimal of owner's permission
                    - Y = binary to decimal of group's permission
                    - Z = binary of other user's permission

What are the 4 File I/O System Calls?

- Open - int open(const char * Path, int flags)
    - used to open a file for reading, writing, or both
    - Path to the file
        - Absolute paths (if not working in same directory) - /
        - Relative paths (if working in the same director) - filename with extension
    - Flags:
        - O_RDONLY - read only

        - O_WRONLY - write only
        - O_RDWR - read and write
        - O_CREAT - create file if it doesn't exist
        - O_EXCL - prevent creation if it already exists
    - Returns the file descriptor
- Read - size_t read(int fd, void * buf, size_t cnt)
    - Read data from one buffer to fd, read size bytes from the file into memory location
    - File descriptor
    - Buffer to read data from
    - Length of the buffer
    - Returns number of bytes on success
    - Returns 0 on reaching EOF
    - Returns -1 on error or signal interrupt
- Write - size_t write(int fd, void * buf, size_t cnt)
    - Write data from fd into buffer, write the bytes stored in buffer to fd
    - File descriptor
    - Buffer that holds the right data
    - The length of the buffer
    - Returns the number of bytes written on success
    - Returns -1 on error or signal interrupt
- Close - int close(int fd)
    - tells the OS you are done with a file descriptor and it closes the file
    - File descriptor
    - Returns 0 on success
    - Returns 1 on failure
- How do open and close work on a system level?
    - Open finds the existing file, makes an entry for it in the file table, and sets the first available fd to point to the entry in the file. Close destroys that file table entry as long as no other process is being pointed to it and it will fd to NULL. For read and write, the buffer and fd given must be valid. If not, it will return -1.
- Dirent struct
- Opendir
- Readdir

What are file descriptors?

- File descriptors are an integer that identifies an open file of the process
- What are file descriptor tables?
    - File descriptor tables are a collection of integer array indices that are file descriptors in which elements are pointers to file table entries. One unique file descriptors table is provided in OS for each process.
    - File table entry - a structure in memory that describes an open file, which is created when process request to open a file
    - What are the standard file descriptors?

        - 0 means read from stdin - used whenever we write any character from the keyboard
        - 1 means write to stdout - output to the terminal or screen in general
        - 2 means write to stderr
- What is the difference between blocking and nonblocking?
    - Blocking IO is when the system is set to wait state until your IO operation is completed. Nothing else can be done until the operation is done being serviced. Nonblocking IO is when your system won't let you go into wait state and will perform operations independent of the IO.

What are processes?

- A process is a program in execution. It is running code that has its own address spaces (in main memory). It is sometimes known as the text section.
    - What are some attributes of a process?
        - Process id - a unique identifier assigned by the operating system
        - Parent Process ID (PPID)
        - STAT - state of the process
            - New
            - Ready - after creation and ready for execution
            - Run - currently running process in CPU (only one process at a time can be under execution)
            - Wait - when process request for I/O request.
            - Complete
            - Suspended run - ready queue becomes full so some processes are suspended.
            - Suspended Block - waiting queue becomes full
        - TIME - CPU time used by the process
        - TT - control terminal of the process
        - COMMAND - user command that started the process
        - What is a context switch?
            - The process of saving the context of one process and loading the context of the other process. Like unloading one process and loading another.
                - This happens when a high priority process comes to ready state, an interrupt occurs, or preemptive CPU scheduling used.
        - CPU Registers
        - I/O status information - devices allocated to process, open files, etc.
        - CPU Scheduling information - priority
        - These attributes are known as the context of the process. They contain information that lets the OS know when to schedule the process again and the state.
    - What is a Program Control Block?

        - A data structure in the OS kernel that contains the information needed to manage the scheduling of a particular process.
- What are the systems calls associated with a process?
    - fork()
        - Used to create a process. It takes no arguments and it returns the process ID. This becomes the child process of the caller process. After a call, both processes will execute the same instructions following the fork system call.
        - A negative return value indicates unsuccessful
        - 0 indicates that a new child process was created successfully.
        - A positive value indicates the process ID of the child process. It is of type pid_t. You can use getpid to get the process ID assigned to a process.
        - What is happening underneath when fork is called?
            - The OS will make two identical copies of address spaces, one for the parent and the other for the child. Both will start their execution after the call to fork. Any modification to any variables will be independent of each other.
    - exec()
        - Used to replace the current process with the new process. If it fails it returns 1. Else, it never returns. This is useful if you want the program to execute another binary file.
    - wait() - pid_t wait(int * stat_loc)
        - It blocks the calling process until one of its child processes exits or a signal is received. After it is terminated, the parent continues execution. This may happen if exit is called, an int is returned from main, or the OS tells it too.
        - If a process has more than one child, then parent process has to wait wait until all children are done.
        - If only one child is terminated, then process ID of child is returned. If more than one, then any PID will be returned. Unless you use waitpid(child_pid, &status, options)
        - If any process has no child, then wait returns immediately (-1).
    - exit() - void _exit(int status)
        - terminates the execution of the process immediately. Any open file descriptors belonging to the process are closed. Any child of the process are inherited by process 1. The parent's process is sent SIGCHILD.
            - What is a zombie process?
                - A zombie process is a process that has completed execution (via exit) but still scheduled to run even though it is now. The child process has simply not been reaped from the process table.
                - It will remain a zombie if the parent does not call wait(), which leaves the zombie process in the process table.

        - To remove a zombie process, SIGCHILD can be sent to the parent using the kill command.
            - If the parent process still refuses to reap the zombie, terminate the parent process.
    - What is an orphan process?
        - An orphan process is a process that is still executing but whose parent has died (via exit).
        - They are usually reparented to another parent process.
        - Sometimes it is necessary, especially if a longer job needs running.
        - Solutions to the problem
            - Exterminate the orphan.
            - Reincarnate the parent process.
            - Expiration in which each process is given a certain time to finish before being killed.
    - What is a zombie orphan process?
        - A zombie process whose parent is no longer in the system.
        - If the same PID is taken on by a different process, then that is a problem.

What are signals?

- Signals are a form of inter process communication (IPC). It is an asynchronous notification sent to a process or to a specific thread within the same process in order to notify it of an event that occurred.
    - It will interrupt the target process' normal flow of execution to deliver it.
    - If the process has its own signal handler, that will be used; else, the default one.
        - The signal can be ignored unless SIGKILL or SIGSTOP is sent to a process.
        - The signal can be caught. If done, then the process registers a function with kernel.
    - What are some useful signals?
        - SIGINT - user interrupt
        - SIGSTOP - terminate process
        - SIGKILL - terminate process with extreme prejudice
        - SIGCHILD - child process has been terminated

What are interrupts?

- Interrupts are generated by devices and they usually mean a device needs attention.
- They cause the hardware to transfer control to a fixed location in memory where an interrupt handle is located.
    - Interrupt handlers are part of the kernel.

What are exceptions?

- Exceptions are conditions that occur during the execution of a program instruction such as arithmetic overflows, illegal instructions, or page faults.

- The CPU handles exceptions like interrupts and the control is transferred to the exception handler.

What are Threads?

- A thread is a path of execution within a process. It is typically used to divide a process into multiple threads. They have multiple execution contexts in one process.
- What is the difference between a kernel thread and a user thread?
    - A kernel thread is a thread in which the kernel creates and schedules. It has a pretty good scheduler and it has true simultaneity (if you have multiple cores). However, the user is not in control, the only way to switch from one thread to another is by running a kernel schedule, and you need a context switch to a different process.
        - The OS does recognize them.
        - If one kernel thread performs a blocking operation, then another thread can continue.
    - A user thread is one that the user creates and schedules. The user is in control and if you want to swap between processes you just need to jump to the a different address in memory. However, the usermode schedulers are notoriously sketchy and you can not have true simultaneity.
        - The OS does not recognize them.
        - Context switch time is less.
        - If one user level thread performs a blocking operation, then the entire process is blocked.
- What is the race condition?
    - This occurs when two or more threads can access shared data and they try to change it at the same time. Because you can swap between threads at anytime, you don’t know what order the threads accessed the data.
        - What is thread synchronization?
            - A mechanism that ensures that two or more concurrent processes or threads do not simultaneously try to execute the critical section of the code (the section where the race condition occurs).
            - How can we implement thread synchronization?
                - Mutual Exclusion (mutexes) - if a process is executing in its critical section, then no other process is allowed to execute in the critical section. This happens at runtime. They are resolved in call order and are sensitive to the number of times they are called.
                    - Test-and-set - instruction used to modify content of a memory location and return its old value as a single atomic operation.
                        - See if a lock can be acquired repeatedly by checking the memory location.
                    - Compare and swap - compares the contents of a memory location with a given value, and if they are

the same, modifies the contents of the memory location to a new given value.

- Fetch-and-increment - atomically increments the contents of a memory location by a specific value. If this operation is executed by one process, no other process will ever see an intermediate result.
- What are deadlocks?
    - A situation in which a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. For example, process 1 is holding resource 1 and is waiting for resource 2 but process 2 is waiting for resource 1.
    - How is deadlock caused?
        - Mutual exclusion - one or more than one resource is non-shareable (one at a time)
        - Hold-and-wait - a process is holding at least one resource and is waiting for resources.
        - No preemption - a resource cannot be taken from a process unless the process releases the resource.
        - Circular wait - a set of processes are waiting for each other in a circle.
    - How can we control a deadlock?
        - Prevention or avoidance - not let the system into a deadlock state.
            - Obtain the locks in a fixed order and do not obtain locks again after you start unlocking.
        - Detection and recover - let deadlock occur, then do preemption.
        - Ignore it. Reboot the system.
        - Global ordering of mutex locking
        - Rising and falling permission phases
- Semaphores - a variable used to solve critical section problem in process synchronization. It limits the number of simultaneous calls to an application or critical section. It is a signaling mechanism that signals the call processing task to wakeup.

            - Counting semaphores - non-negative integer values.
            - Binary semaphores - 0 and 1 only.
- What is a condition variable?
    - An explicit queue that threads can put themselves on when some state of execution is not desired. It can use signaling to wake the thread up and change the state.

How would you compare and contrast a process and a thread?

- A process is created using fork(). The parent process continues and calls wait() on child to wait for it. The child process often execs, replaces itself with new code, and runs to completion. In threads, we use pthread_create, and the parent process continues on unless join() is called. The new thread loads the function and runs that function to completion.
- The big difference is that a thread is a new stack on the text segment. You can pass in a function pointer, which runs the function and allows the previous execution context to keep going. If you call join, you will stop running and wait till it is done.
- A process uses a binary executable whereas a thread just calls a function to be run.
- Threads are fast, no need to find more memory to switch between them. They also have the same address space. This makes it easy to share data.
- Processes are nice when you want different address spaces and unmodifiable code.

What are IP Addresses?

- IP Address are addresses that have information about how to reach a specific host, especially outside the LAN. They use a 32 bit unique address, which means there are 2^32 number of different unique addresses.
    - 00000000.00000000.00000000.00000000
- What are the 5 different subclasses of IP addresses?
    - Class A: 192.XXXXXX
        - Network ID is 8 bits long, Host ID is 24 bits long.
        - Usually used for a large number of hosts
        - The first bit of netid is set to 0
        - 126 network IDs
        - 16,777,214 host IDs
    - Class B: 192.168.XXX
        - Network ID is 16 bits long, Host ID is 16 bits long.
        - Usually used for medium to large networks.
        - The first two bits are 10.
        - 16384 network addresses.
        - 65534 host addresses.
    - Class C: 192.168.1.XX
        - Network ID is 24 bits long, Host ID is 8 bits long.
        - The first three bits are set to 110
        - 2097152 network addresses.

        - 254 host addresses.
    - Class D: 192.168.1.123
        - Reserved for multicasting - send to one address and it is automatically forwarded to many. So if something needs to be sent to the router, it may be sent on 224.0.0.2.
        - Defined by 1110.
        - Used at link or internet layer of ISO OSI
    - Class E: 240.0.0.0 - 255.255.255.254
        - They are set to 1111.
    - Host IDs are used to identify a host within a network.
    - The naming problem occurs because many addresses are wasted in classes A-D. There are N names and M identities. Some devices share the same IP address. To avoid this, they used Dynamic Host Configuration Protocol (DHCP) in which a service give you
- What is a direct Memory Access (DMA)?
    - DMA allows an input/output device to send and receive data directly to or from main memory, bypassing the CPU to speed up memory operations.
- What is Programmed IO?
    - A method of transferring data between the CPU and a peripheral, such as a network adapter or ATA storage device.
- What is a Network Address Translation (NAT)?
    - NAT is a method of remapping on IP address space into another by modifying network address in the IP header of packets while they are in transit across a traffic routing device.
- What is an internet protocol (IP)?
    - Higher level address for accessing for accessing a device over the internet.
- What is a Local Area Network (LAN)?
    - A computer network that interconnects computers within a limited area via ethernet. All machines see each other in the network and have 2 addresses: a MAC address for hardware and an IP address for software.
- What is the Domain Name Service (DNS)?
    - DNS translates from hostnames to IP addresses, in which each piece of hardware has its own unique MAC address.
    - If a domain is requested, check DNS caches for IP address
    - If not in there
        - Get up address
        - Search DNS authority.
        - Error DNS not found
- What is UDP?
    - User Datagram Protocol allows for devices to communicate quicker at the cost of error correction.
    - It is unreliable. It may not deliver the packet. If link failure occurs or interference occurs, UDP will not deliver the packet

    - If a packet gets dropped, no requests for sending it again will be placed.
- What is TCP?
    - Transmission Control Protocol.
    - TCP allows simulated connections between machines. It is reliable because it ensures that both sides of connections are getting all the packets. When a packet reaches a port, any application that is listening to the port, will know that this package is theirs.
    - If a packet is dropped, the machine requested will request it again.
- What is the ISO OSI Layer Stack?
    - A networking framework to implement protocols in seven layers called the Open System Interconnection.
    - This goes from the least abstract to the most abstract layer
    - Physical - transmits bits by converting them to some electrical impulse, light, or radio signal. It provides the hardware means of sending and receiving data.
    - Data Link - Data packets are encoded and decoded into bits. It furnishes transmission protocol and handles errors in the physical layer, flow control, data integrity and data identity.
    - Network - provides switching and routing technologies, creating logic paths, and transmitting data from node to node. It manages/discovers/forwards messages along routes to other hosts that maintain a routing table that manages routes when congested.
    - Transport - transparent transfer of data between end systems, or hosts, and is responsible for end-to-end error recovery and flow control. This means managing reliability, retransmission, integrity, and size.
    - Session - establishes, manages, and terminates connections between applications. It sets up, coordinates, and terminates conversations, exchanges, etc. between applications. It synchronizes data streams and multiplex streams.
    - Presentation - it translates from application to network format. It will transform data into the form that the application layer can accept. It formats and encrypts data sent across a network.
    - Application - provides services for file transfers, email, and other network software services. It supports application and end-user processes.
    - Important note: sometimes messages will accumulate in a bubble because very few bytes are sent over and metadatas are so costly. Therefore, flush is used to prevent bytes from being stuck in the data stream.
- What are sockets?
    - Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket listens on a particular port at an IP, while the other reaches out to form a connection. It is a way to talk to computers using standard Unix file descriptors.
        - What steps does the server take to set up a socket?
            - Int sockfd = socket(domain, type, protocol)
                - Domain - integer, communication domain (AF_INET).

        - Type - communication type (SOCK_STREAM)
        - Protocol - protocol value for IP, = 0.
    - Int setsockopt(int sockfd, int level, int optname, const void * optval, socklen_t optlen)
        - Manipulates options for socket referred by the file descriptor, sockfd. Prevents address already in use error.
    - Int bind(int sockfd, const struct sockaddr * addr, socklen_t addrlen);
        - Binds the socket to the address and port number specified in addr. Localhost or a specific IP.
    - Int listen(int sockfd, int backlog);
        - Puts socket in passive mode, where it waits for client to approach the server to make a connection. Backlog is the length of the queue of pending connections for sockfd is.
    - Int new_socket = accept(int sockfd, struct sockaddr * addr, socklen_t * addrlen);
        - Extracts the request on the queue of pending connections for the listening socket, sockfd, creates a new connected socket, and returns a new file descriptor to that socket.
- What steps does the client take?
    - Int sockfd = socket(domain, type, protocol)
        - Domain - integer, communication domain (AF_INET).
        - Type - communication type (SOCK_STREAM)
        - Protocol - protocol value for IP, = 0.
    - Int connect (int sockfd, const struct sockaddr * addr, socklen_t addrlen)
        - Connects the socket referred to by the file descriptor sockfd to the address specified by addr.
- What are some other important structures?
    - Gettaddressinfo - returns one or more addrinfo structures, each of which contains an internet address that can be specified in a call to bind or connect
    - Gethostbyname - it searches the hosts database for information about a particular host and returns information about the hostname past into it.
    - Hostent - a structure that represents an entry in the host database
        - Char *h_name - official hostname
        - Char ** h_aliases - alternative names for the host
        - Int h_addrtype - host address type
        - Int h_length - length of each address
        - Char ** h_addr_list - addresses for the host
        - Char * h_addr - the first host
    - sockaddr _in

            - Sin_family - AF_INET
            - Sin_port - htons(17892)

What are distributed systems?

- Some system that is made up of multiple different subsystems and all the subsystems and processing is asynchronous. They all cooperate to give you the illusion of a single system. They are all located on networked hardware and communicate by passing messages.
- Why use distributed systems?
    - Resilience
    - Availability
    - Performance
    - However, they scale poorly, use a lot of bandwidth and are fragile.
- Fragility due to replication of state
    - Centralize it
    - Two phase commit - send an abort message before synch message and stop the update.
        - Send lock message to everyone
        - Once everyone replies
        - Send update to everyone
        - Once everyone replies
        - Send a synch message
        - Make sure everyone replies
    - read/write quorum
        - Presume you have n nodes
        - If I write to Nw < N nodes, how many do you need to read from to be sure you see the read?
            - If Nr is the number read, then Nr+Nw > N
- Error States and Failures
    - Failstop - operation immediately halts with some status or error code
    - Crash - node/operation does not respond
    - Byzantine - sometimes down something kind of wrong … for no discernable reason
- Update is slow or uncertain
        - Deferred update
            - Hold onto the change until it can become canonical
        - Update in place
            - Presume the update is just fine, but retain a copy or some other way to roll back to previous state