L4

Parallel Computing environment

- Users should feel that they're in control of the system, even though it may be shared
- Not very important to make an illusion of a single system in a distributed computing environment
- Each computer has own processor and memory and they can talk to one-another. (it's like a graph)

Client-server architecture

- Client wants service from server
- requests server to do something on its behalf
- Server receives request and returns a response
- Two kinds of servers servernodes:
  1. compuatation servers
  2. Fileservers
     There can be others, these are simple ones
- DOS - Denial of service, happens when the server is overwhelmed with requests.
  1. Occurs when users are trying to access a popular link, or it's an attack with malicious intent.
- Server is a bottleneck here

Peer to peer computing

- doesn't distinguish clients and servers
- nodes join and may leave p2p

Bootloader program

Bootloader stored in ROM

Before bootloader

- BIOS - Basic Input/Output System, a firmware
  - special kind of software embedded into the hardware

1. Performs a Power - on -self -test (POST)
   - Informs the user about devices not working , either as a beep or as an error message
2. Initialize the hardware devices - cpu regs mem etc.

3. it loads a special register - the instruction register with a prefixed memory location
    - Memloc contains an initial bootstrap program (not the whole program, just 1 sector in size)
    - This block is called the bootblock/bootsector/mbr
    - Easy target for virus

- Any disk containing the bootsector is called a bootdisk, and the partition is called a boot-partition
- mostly the block is block #0

4. The initial bootstrap program locates the whole program and loads it
5. this bootstrap program locates the OS and loads it.

GRUB - GRand Unified Bootloader

Developed by linux

1. A bootloader that loads the Bootloaders which different OSs have
2. Allows to choose a specific kernel config of an OS or Different OSs

Chapter 3 - Process

Process

- A set of steps executed one after the other is a program

> A **program under execution** is a process

- Batch systems - jobs
    - user submits the job and doesn't wait for the output, put into the queue, and at some time it's executed
    - program not expected to execute immediately
- Time sharing systems (multitasking) - process
    - the process needs to start immediately
- Multiple parts
    - each process has an ID
        - Process 1 is init or systemd
    1. Program code - text section
    2. Program counter has the instruction number, processor registers
    3. Stack has temporary data, return address, local variables
    4. containing memory dynamically allocated during run time - when you have calloc or malloc
        - heap is a datastructure where each node has zero to two child nodes

Process concept

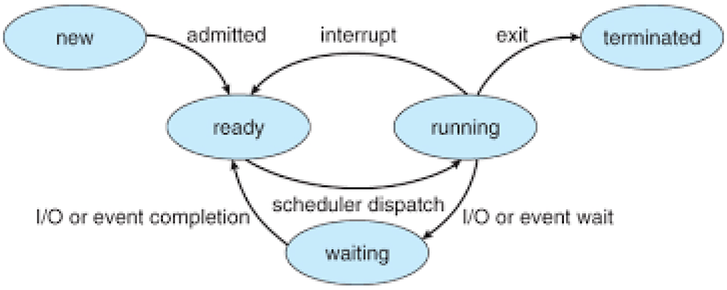
- Note: stack grows downwards, heap grows upwards

1. program is passive entity stored on disk - process is active
    - program becomes process when executable file is loaded into memory
    - execution can happen thru command line entry of name/ gui mouse clicks or sth
2. One program can have several processes of itself
   - Eg. several users can exec same program, but they have unique identifiers, os is treating them as separate stuff

State of Process

- note: the names may vary per book

1. new - the process is being created
    - the process has to wait somewhere to be executed (don't confuse with job, process has already started)
2. running - the instructions are being executed
    - when it gets a chance to execute
3. waiting - the process is waiting for some event to occur
    - sometimes the process is waiting for stuff (i/o)
4. ready: the process is waiting to be assigned to a processor
    - when the processor is put into the ready-queue
5. terminated - finished execution



- the scheduling mechanism may allow execution only in different chunks
    - once in a while to allow another process, the process should relinquish the state
- from the waiting state, it doesn't transfer **directly to the running state**
- at terminated state, all memory of the process is de-allocated

Process control block

- process state
- Program Counter reg
- CPU Regs
- CPU Sched info - priorities and pointers to queues for scheduling -- scheduling queues
- Memory management information - memory allocated to the process m pagetables

- Accounting info - cpu used, clock time etc
- I/O info - devices ready?

Process Creation

- Each process has an ID- PID
    - PID Increases by 1 for each 1
- terminated process's ID isnt recycled
- A child process can
    - get same set of resources
    - or a subset, so that parent process doesn't hog resources
    - or child will have to request the OS
- Both can exec concurrently, or parent waits for child
    - P2 will get the same address space (set of memlocs alloc to a process) as P1 (same program and data)
        - or P2 ,the child, is loaded with another program
- fork -- to create a process
    - after creating the process if you have to load a different process image into the child process you use exec() -- to let the child process do something else