Compiler

Compilation system:

- hello.c (source program, text)
- hello.i (modified source program, text) preprocessor (cpp)
  - modifies according to # directives (#include<stdio.h>)
- hello.s (assembly program, text) compiler (cc1)
  - assembly
  - language program (statement equals 1 low level machine language instruction)
  - assembly language common compiler output for different high level languages
- hello.o (relocatable object programs, binary) assembler (as)
  - to binary file (relocatable object program) whose bytes encode machine language instructions
- hello (executable object program, binary) linker (ld) (linked with printf.o)
  - linker merges standard c library functions (executable object file)

sources:

- Computer Systems: A Programmers Perspective pg 4-7

Hardware Systems

Hardware Organization:

- buses
  - carry bytes of info between componenets
  - words: fixed sized chunks of bytes (4 bytes/32 bits or 8 bytes/64 bits)
- I/O devices
  - system's connection to outside world by controller or adapter
  - display, disk, mouse
  - controller: chip sets in device itself or motherboard
  - adapter plugs into slots on motherboard
- Main Memory
  - temporary storage for both program and data it manipulates
  - physically dynamic random access memory (DRAM)
  - logically organized as linear array of bytes each with its own unique address
- Processor
  - central processing unit (cpu)
  - interprets (or executes) instructions stored in main memory
  - at its core is a word sized register called program counter (PC)
    - At any point points at machine language instruction in main memory
  - executes instruction and updates pc to next instruction
  - instructions revolve around main memory, register file, and arithmetic/logic unit
    - reigter file small storage device with word sized registers each with its own unique name
    - ALU computers new data and address values
  - Simple operations CPU carries out at the request of an instruction
    - Load: Copy a byte or a word from main memory into a register, overwriting the previous contents
    - Store: Copy a byte or a word from a register to a location in main memory, overwriting the previous contents
    - Operate: Copy the contents of two registers to the ALU, perform an arithmetic operation on the two words, and store the result in a register, overwriting the previous contents
    - Jump: Extract a word from the instruction, copy that word into the program counter (PC), overwriting the previous value

sources:

- Computer Systems: A Programmers Perspective pg 7-10

Operating System

Processes:

- instance of computer program
- abstraction for a running program
- single CPU can appear to execute multiple processes concurrently by having the processor switch among them
- OS performs this using contect switching
- OS keeps track all state information for a process (context)
  - values of PC, register file, main memory
- single processor can only execute code for single process
- when OS decided to transfer control to new process performs context switch
  - save contect of current process, restore contect of new process, pass control to new context
- system call: special function, passes control to OS and saves shell context, creates new context (hello) and gives it control
- scheduling: kernel decides to preempt the current process and restart a previously preempted process
  - handled by code in the kernel called scheduler
- sum: decision: scheduling, act: context switch

Threads:

- execution unit
- process can be made up of multiple threads that execute concurrently on process context with same code and global data
- requirement for concurrency in network servers make threads important
  - easier to share data between multiple threads than multiple processes
  - threads more efficient than processes
- multithreaing a method to make programs faster with multiple processors

Virtual Memory:

- abstraction provides each process the illusion that it has exclusive use of the main memory
  - heap: follows code and data areas (which are themselves fixed in size)
    - expands and contracts dynamically at runtime (malloc, free)
  - Shared libraies: near the middle
  - stack: at the top
    - used to implement function calls
    - expands and shrinks dynamically (grows when you call/shrinks when you return from a function)
  - kernel virtual memory

sources:

- Computer Systems: A Programmers Perspective

Concurrency

General:

- concurrency refers to the general concept of a system with multiple, simultaneous activities
- parallelism refers to the use of concurrency to make a system run faster

Resources:

- each thread has its own stack
- threads in a process share heap

Approaches:

- hyperthreading (thread level concurrency)
  - multiple copies of some of the CPU hardware (program counters,register files) while having only single copies of other parts (units that perform floating-point arithmetic)
  - decides which thread to run on a cycle by cycle basis
- instruction-level parallelism: modern processors can execute multiple instructions at one time
- Application level concurrency
  - Benefits
    - overlap useful work with I/O requests
    - separate concurrent logic flow for interactions with humans (resize window)
    - reducing latency by defering work (in Dynamic Storage allocator defer coalescing to concurrent flow at lower priority in CPU spare time)
    - separate logic flows for each client
    - partitioned applications with concurrent flows run better on multiprocessor
  - Methods
    - processes
      - each logic control flow scheduled and maintained by kernel
      - separate virtual address spaces
      - need explicit interprocess communication (IPC) mechanism to communicate
    - I/O multiplexing
      - applications explicitly schedule their own logical flows in the context of a single process

        - Logical flows are modeled as state machines that the main program explicitly transitions from state to state as a result of data arriving on file descriptors
        - all flows share the same address space
    * threads
        - run in the context of a single process and are scheduled by the kernel
        - hybrid of the other two approaches
        -
        - scheduled by the kernel like process flows
        -
        - sharing the same virtual address space like I/O multiplexing flows

---

Tools for sharing:

- lock: allows only one thread to enter the part that's locked and the lock is not shared with any other processes
- Mutex Used to provide mutual exclusion
  - ensures at most one process can do something (like execute a section of code, or access a variable) at a time
  - A famous analogy is the bathroom key in a Starbucks
    * only one person can acquire it, therefore only that one person may enter and use the bathroom
    * Everybody else who wants to use the bathroom has to wait till the key is available again
- Monitor: object designed to be accessed from multiple threads
  - member functions or methods of a monitor object will enforce mutual exclusion, so only one thread may be performing any action on the object at a given time
  - if one thread is currently executing a member function of the object then any other thread that tries to call a member function of that object will have to wait until the first has finished
  - no part of the instance can be touched by more than one process at a time
  - A monitor is like a public toilet
    * Only one person can enter at a time
    * They lock the door to prevent anyone else coming in, do their stuff, and then unlock it when they leave
- Semaphore: lower-level object
  - P() and V()functions
  - You might well use a semaphore to implement a monitor
  - A semaphore essentially is just a counter

    - When the counter is positive, if a thread tries to acquire the semaphore then it is allowed, and the counter is decremented
    - When a thread is done then it releases the semaphore, and increments the counter
  - Semaphores are typically used as a signaling mechanism between processes
  - A semaphore is like a bike hire place
    - They have a certain number of bikes
    - If you try and hire a bike and they have one free then you can take it, otherwise you must wait
    - When someone returns their bike then someone else can take it
    - If you have a bike then you can give it to someone else to return, the bike hire place doesn't care who returns it, as long as they get their bike back

---

Issues:

- Deadlock is a condition in which a task waits indefinitely for conditions that can never be satisfied
  - task claims exclusive control over shared resources
  - task holds resources while waiting for other resources to be released tasks cannot be forced to relinguish resources a circular waiting condition exists
- Livelock conditions can arise when two or more tasks depend on and use the some resource causing a circular dependency condition where those tasks continue running forever
  - this blocks all lower priority level tasks from running (these lower priority tasks experience a condition called starvation)
  - A real world example of livelock occurs when two people meet in a narrow corridor, and each tries to be polite by moving aside to let the other pass, but they end up swaying from side to side without making any progress because they both repeatedly move the same way at the same time

---

Implementing in Java:

- extend Thread
  - can't extend any more classes
  - write run() function with what ever you want thread to do
    - thread.start() starts new thread (calls run within start)
    - thread.run() will run
- implement runnable
  - can implement more interfaces

  - write run() function with what ever you want thread to do and create thread with runnable in constructor

---

sources:

- Computer Systems: A Programmers Perspective
- https://www.quora.com/Semaphore-vs-mutex-vs-monitor-What-are-the-differences
- http://stackoverflow.com/questions/7335950/semaphore-vs-monitors-whats-the-difference
- http://stackoverflow.com/questions/2332765/lock-mutex-semaphore-whats-the-difference
- http://javarevisited.blogspot.com/2014/07/top-50-java-multithreading-interview-questions-html

Distributed Systems

General:

- organizing resources via network with more latency, less bandwidth, and higher error rate
- Vs parallel systems
  - Distributed has multiple (distributed users) vs parallel designed for single user or user process
  - Parallel has less security issues
  - Distributed Systems: cooperative work environment, Parallel Systems: environment designed to provide the maximum parallelization and speed-up for a single task

Definitions:

- task: collection of resources configured to solve a particular problem
  - contains not only the open files and communication channels, but also the threads (a.k.a. processes)
  - a task is a factory, all of the means of production scattered across many assembly lines

Migrating:

- Migrating Computation
  - need to move resources (memory, files)
  - move state of interaction with resources
  - need to restore communication (other side need to know new location)
- Migrating File State
  - keep track of the essential file state within the process and be ready to recreate it
  - migration system guarantees that higher
  - level file operations are atomic, operation opens the file, seeks as needed, performs a read or write, and then closes the file
  - or maintain the same file state, but only reopen the file when needed
- Migrating Communication Sessions
  - need to reestablish and map gloabl state, such as network sockets
  - higher level network name

  - edge cases, such as what happens when a process move mid
  - transmission

Networking:

- LAN: Local Area Network a homeogeneous network
  - to communicate broadcast (yell) and all stations hear the broadcast
  - station have station id or LAN Address and messages have sender and reciever id
    - everyone can hear other stations messages but ignore it unless diagnostic or malicious
  - size is self-limiting
    - longer wire the weaker the signal
    - greater the distance through the air the weaker the signal
    - network can be clogged with collision, can collapse with utilization as low as 30
- bridge
  - connect LANs
  - send message for station only to relevant LAN (hashtable)
  - bridge managed so if station changes LAN
  - bridges (through configuration) create a spanning tree of location to prevent cycles
    - if bridge fails form a different tree to get around failure
  - can't create bridges for the whole planet
- IP Address
  - IP4
    - Class A (huge): 8 bits(nework) + 24 bits (host) address begin with 0
    - Class B (big): 16 bits(nework) + 16 bits (host) address begin with 10
    - Class C (small): 24 bits(nework) + 8 bits (host) addess begin with 110
  - IP6
    - Classless, first few bits to describe the network/host division
    - 73.93.0.0/15
      - address with 15 network bits and 17 host bits

---

incomplete:

- Dynamic Host Configuration Protocol (DHCP) allows IP addresses to be assiged on a temporary or quasi-permant basis
  - DHCP is great for clients – but not for servers: Servers need to have well-known addresses

sources:

- http://www.andrew.cmu.edu/course/15-440-f14/index/lecture_index.html