**How a modern computer works**

CPU gets data from machine instructions and data

DMA - direct memory access

Operating System – program that manages a computer’s hardware (convenient, efficient)

polling - waiting for a response from a device

interrupt - cpu goes to sleep until device responds

Modes

Process

bases memory locations from "0"

MMU - memory management unit

Maps process memory locations to actual real memory

Extremely fast

Holds memory rights - R or RW

text - read only, shared between multiple processes

program instructions

string literals

data - read / write

global variables

FILE \* stdin, stdout, stderr

stack - all local variables

.v

.^

Heap

data

text

Kernel - basic guts of OS

load parts of processes to memory

How memory changes between multi process or multi thread

process thread

text same same

data new same

stack new new

heap new same

read( from, dest, size)

if the kernel is in virtual memory the returned data

must be changed to work with virtual memory

makefile

CC = gcc

CFLAGS = -ggdb

argc - count of arguments

argv - array of arguments

exec converts the arguments to argc and argv

What the OS does for a program

compiling with debug will include the symbol table / function names

printf code is added when .o files are linked

sharing library - each printf uses the same code (typically true)

lock / keep in memory - OS keeps it in memory (typically false)

.dll - dynamic load library (not stored in memory)

only loaded into memory when first called

Virtual Memory - allows each thread / process to think it has its own computer

memory starts at "0x0"

bios starts at 0

Processes loaded into memory in segments

2, 4k bytes = pages

Page Fault - kernel must find the segment of code on the hard disk and load into memory

Swapping - moving real memory to hard disk to create room

MMU uses expensive equipment

digital circuitry for every element of the array

associative memory - addressed by the contents no index

cannot use MMU in machine langauge / real memory

takes to long, it must be instantaneous

OS keeps track of every timer

Interrupt fires every 1/1000s and updates each timer

OS removes memory from processes not currently running

Programs

Ready -> CPU

Most of the time programs leave the CPU for blocked purposes (I/O Request)

CPU will prefetch instructions from memory