**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)

Ignore resource utilization

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 too 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

Hardware – cpu, memory, I/O

OS – resource allocator, control program

Manages execution of user programs

Middleware – software framework to provide additional services to app devs

Computer System Operation

Bootstrap program – ROM/EEPROM

(Firmware)

System process / daemons – run all the time

Interrupt – event occurred

Hardware sends by bus

Software sends by system call

Storage Structure

RAM – Dynamic random access

Volatile

Von neumann architecture

Fetch instruction, store in instruction registr

I/O Structure

SCSI – Small computer-systems interface

Devices connect to the controller

Device driver – provide OS with uniform interface

Multiprocessor Systems

Increased throughput, economy of scale, increased reliability

Graceful degradation – survive failure

Fault tolerant

Multicore – on chip communication is fast

Clustered Systems – multiple systems together

High-availability of service

SAN – Storage area network

Pools of storage

Operating-System structure

Time sharing – processes are switched between on the CPU to make it look real time

Requires interactive system

Process – program loaded into memory

Swapping – process in and out of memory

Trap / exception – software interrupt

Running modes - User, kernel

OS and processes – scheduling processes / threads, creating / deleting user and system processes, suspending / resuming processes, process synchronization, process communication

Memory Management – which parts of memory are currently used and by who, deciding which process and data to move into / out of memory, allocate and deallocate memory

File System management – create / delete files, create / delete directories, support primitive manipulation of files, mapping files onto secondary storage, backup files

Mass-Storage – free-space management, storage allocation, disk scheduling (tertiary storage devices)

WORM (write-once, read many times)

Cache – fast temporary storage in between CPU registers and main memory

I/O System – buffering, caching, spooling, general device-driver interface, drivers for specific hardware

Protection – any mechanism for controlling access

Security – defend system from attacks

UserID or SecurityID

GroupID and effective UID

Kernel Data Structures

Linked list – single, doubly, circular

Stack – sequentially ordered (push / pop)

Queue – FIFO

Tree – data hierarchically

Hash function – input data -> returns numeric value

Bitmaps – binary digits to represent the status

Computing Environments

Portals – access to internal servers

TCP/IP – network protocol

Virtualization – OS to run app like it is in a different OS

Emulation – Source CPU is different from target CPU