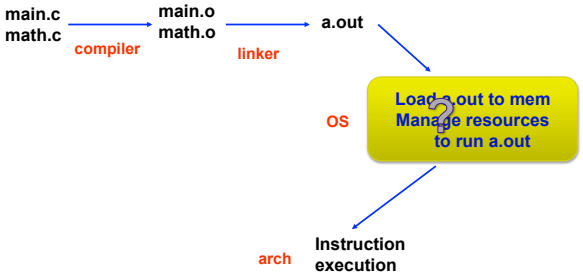
ECE595: Introduction to Operating Systems

Y. Charlie Hu

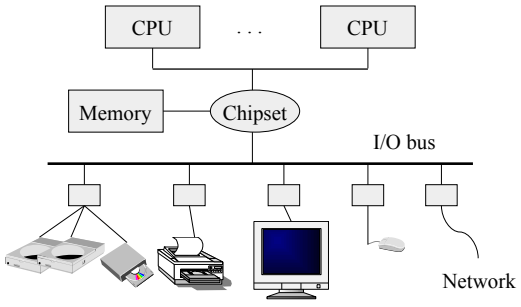
1/9/2017

Missing piece of the puzzle



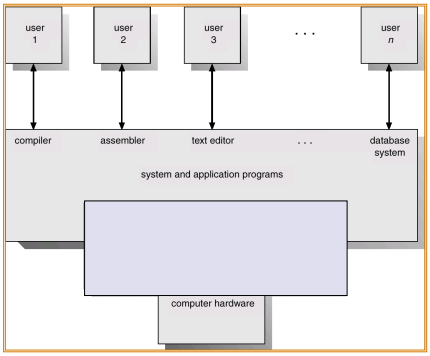
2

A Typical Computer from a Hardware Point of View



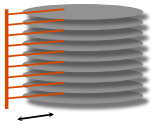
3

Computer System Components



4

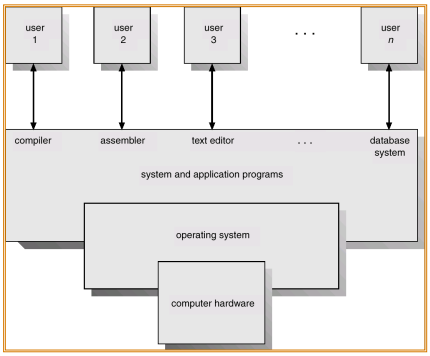
Example: programming hard drive



- Physical reality
  - Block oriented (e.g. 512 bytes)
  - Physical sector numbers
  - No protection among users of the system
  - Data might be corrupted if machine crashes
  - Programming:
    - Loading values into special device registers

"I will save prog. assignment 1 solution on platter 5, track 8739, sector 3-4."

Computer System Components



Example: programming hard drive

- Physical reality
  - Block oriented
  - Physical sector numbers
  - No protection among users of the system
  - Data might be corrupted if machine crashes
  - Programming:
    - Loading values into special device registers
- File system abstraction
  - Byte oriented
  - Named files
  - Users protected from each other
  - Robust to machine failures
  - Programming
    - open/read/write/close

"I will save prog. assignment 1 solution on platter 5, track 8739, sector 3-4."

"My assignment 1 solution is in ~ychu/proj1/process.c."

Brief History of Computer Systems (1)

- In the beginning, 1 user/program at a time (by operator)
- Simple batch systems were 1st real OS
  - Spooling and buffering allowed jobs to be read ahead of time
- Multiprogramming systems provided increased utilization (throughput)
  - multiple runable jobs loaded in memory
  - overlap I/O with computation
  - benefit from asynchronous I/O devices (interrupt, DMA, …)
  - 1st instance where the OS must schedule resources
    - CPU scheduling
    - Memory management
    - Protection

Brief History of Cmputer Systems (2)

- Timesharing systems support interactive use
  - Logical extension of multiprogramming
  - Permits interactive work
    - Each user feels he/she has the entire machine
  - Optimize response time by frequent time-slicing multiple jobs
  - More complex than multiprogramming OS
    - In addition to CPU scheduling, memory management, protection
    - Virtual memory to allow part of the job be in memory
    - File system (needed by interactive use)
    - Job communication, synchronization
    - Handling deadlocks
  - Most systems today are timesharing systems (focus of this class)

9

What is an OS?

"Code" that *sits between*:

- programs & hardware
- different programs
- different users

But what does it do/achieve?

10

What is an OS?

- Resource manager
- Extended (abstract) machine

  Makes computers *efficient* and *easy* to use

- (will have a 3rd def based on pragmatics)

11

What is an OS?

Resource manager (answer1)

- Allocation
- Reclamation
- Protection

12

What is an OS?

Resource manager
- Allocation
- Reclamation
- Protection

Finite resources
Competing demands

Examples:
- CPU
- Memory
- Disk
- Network

13

What is an OS?

Resource manager
- Allocation
- Reclamation
- Protection

"The OS giveth
The OS taketh away"

Implied at termination
Involuntary at run time
Cooperative (yield cpu)

14

What is an OS?

Resource manager
- Allocation
- Reclamation
- Protection

"You can't hurt me
I can't hurt you"

Implies some degree of safety & security

15

What is an OS?

Extended (abstract) machine (answer 2)

- Much more ideal environment than the hardware
  - Ease to use
  - Fair (well-behaved)
  - Supporting backward-compatibility
  - Reliable
  - Secure
- Illusion of infinite, private (reliable, secure) resources
  - Single processor → many separate processors
  - Single memory → many separate, larger memories

16

Is there a perfect OS?
(resource manager, abstract machine)

Efficiency
Fairness

Portability
Interfaces

Security
Robustness

- Conflicting goals
  - Fairness vs efficiency
  - Efficiency vs portablity
  - …
- Furthermore, …

17

Hardware is evolving…

- 60's-70's - Mainframes
  - Rise of IBM
- 70's - 80's – Minicomputers
  - Rise of Digital Equipment
- 80's - 90's – PCs
  - Rise of Intel, Microsoft
- 90's - 00's – handheld/portable systems (laptops)
- 2007 - today -- mobile systems (smartphones), Internet of Things
  - Rise of iPhone, Android

18

Implications on OS Design Goals: Historical Comparison

	Mainframe	Mini	Micro/ Mobile
System $/ worker	10:1 – 100:1	10:1 – 1:1	1:10-1:100
Performance goal	System utilization	Overall cost	Worker productivity
Functionality goal	Maximize utilization	Features	Ease of Use

19

Hardware is evolving (cont) …

- New architectures
  - Multiprocessors
  - 32-bit vs. 64-bit
  - Multi-core

20

May You Live in Interesting Times…

- Processor speed doubles in 18 months
  - Number of cores per chip doubles in 24 months
- Disk capacity doubles every 12 months
- Global bandwidth doubles every 6 months

→ Performance/cost "sweet spot" constantly decaying

** Does human productivity ever double?*

21

Applications are also evolving…

- New applications
  - Scientific computing
  - Computer games
  - Java
  - WWW (web servers, browsers)
  - Networked games
  - Peer-to-peer
  - Web 2.0 (search, youtube, social network, …)
  - Mobile apps (1.5+ million iPhone, Android apps each)
  - …

22

Implications to OS Design

- Constant evolution of hardware and applications continuously reshape
  - OS design goals (performance vs. functionality)
  - OS design performance/cost tradeoffs
- Any magic bullet to good OS design?

23

There is no magic in OS design

This is Engineering

- Imperfection
- Tradeoffs (perf/func)
- Constraints
  - hardware, cost, time, power
- Optimizations

Nothing's Permanent

- High rate of change
  - Hardware
  - Applications
- Cost / benefit analyses
- One good news:
  - Inertia of a few design principles

24

Separating Policies from Mechanisms

A fundamental design principle in Computer Science

Mechanism – tool/implementation to achieve some effect

Policy – decisions on what effect should be achieved

Example – CPU scheduling:

- All users treated equally
- All program instances treated equally
- Preferred users treated better

Separation leads to flexibility!

25