Lecture S2: Operating Systems

What is an Operating System?

Modern operating systems support:

- Software tools for creating programs (Lecture S3).
  - libraries, compilers
- Running multiple programs.
  - multiprogramming
- Saving/accessing data.
  - files, virtual memory
- User interaction.
  - window system
- Interaction with other systems.
  - networking
- Core applications programs.
  - client-server

What is an Operating System?

Execution Control.

- OS keeps track of state of CPU, devices.

External Devices.

- Display, keyboard, mouse, disks, CD, network.

Virtual Machines.

- Pretend machines that each person/program can use.
- OS implements abstract devices.

Multiprogramming

Operating system "kernel" keeps track of several programs.

- CPU does 1 thing at a time.
- Goal: illusion of multiple machines.

INTERRUPT:

- Part of hardware of real machines (not discussed with TOY).
  - stop
  - save PC somewhere "special"
  - change PC
- Necessary to manage input-output devices.
  - mouse click, keyboard
- OS allows several programs to "share" CPU by keeping table of "current" PC's for programs setting clock to interrupt periodically.
  - arizona
  - round-robin or user priorities

Multiprogramming: Two Useful Properties

RELOCATABLE program.

- Can be moved while it is executing.
  (useful if OS rearranges memory a la `malloc`)

REENTRANT program.

- Can be executed while it is executing.
  - same program running for multiple users
- Only load one copy of program.
  - emacs, gcc

Virtual Memory

Problem 1: several programs need to share same memory.

- Direct solution: apportion up the memory.

Problem 2: program needs more memory than machine has.

- Direct solution: "overlays."
  - program shuffles its own data in and out of memory to disk

It's all just memory, why should file system look more complicated?

"Better" solution: VIRTUAL MEMORY (1960's).

- All programs assume access to all memory.
- Each program actually uses a small portion.

Virtual Memory

VIRTUAL MACHINES.

- Simulate multiple copies of a machine on itself.
- Ex: can debug OS.

Physical address space.

- How much real memory is there?
- Limitation: $ per bit cost.

Virtual address space.

- Maximum amount of memory an instruction can directly reference.
- Limitation: address size (bits / instruction).

Size of Virtual Memory

How many bits is enough?

- 16 bits is not enough.
- 32 bits is not enough.
- 64 bits?
  - 2^{64} = 18,446,744,073,709,551,616 > 10^{19} addresses
- 512 certainly enough.

Some big numbers.

- 2^{70}: number of grains of sand on beach at Coney Island.
- 2^{93}: number of oxygen atoms in a thimble.
- 2^{256}: number of electrons in the universe.

More sophisticated paging strategies needed.

Paging

Paging: widely-used method to implement virtual memory.

- Design hardware to "trap" all addresses.
- Keep virtual memory (for each program) on disk.
  - only part that CPU is currently accessing is in main memory

Divide into PAGES. Keep table with:

- Flag indicating if page is in memory.
- Relative position of page in memory.

Make page size = 2^x, use leading bits of address for page name

Each memory reference:

- Check if page is in memory.
- Get it from disk if not.
- Use page table to reset upper address bits.

Paging

Each page brought in has to REPLACE another.

- Page replacement strategies.
  - Ex. least recently used
- Still being studied, invented.

Basic principles.

- MEMORY HIERARCHY
  - local: fast, small, expensive
  - remote: slow, huge, cheap
- Tradeoff speed for cost.
- CACHE recently accessed information.

Window Manager

Virtual Terminals.

- Each program has its own virtual display.
- Ex. X-terminal: complex, customizable, virtual!
- Just another simulation program.
- Commonplace today, rare in 1985
- Ingenious design meets accelerating technology.

History.

- Xerox PARC (Alto), Macintosh, Windows NT, X-terminal, Netscape.

Problem or opportunity?

- Truly "virtual."
- Moving away from grounding in reality.
  - harder for programmers to understand what is happening
- Flexibility vs. standardization.
- Other ways of interacting with computer?

Client-Server Model

System divided into two distinct parts.

- Ex: display server (implement virtual display).
  - draw stuff on screen
  - monitor keyboard and mouse input
- Ex: Client (use virtual display).
  - applications programs

Server is interface between client program and display hardware.

Model generalizes beyond display management.

- Client: request service.
- Server: do the work.

Advantages.

- Single server can handle multiple clients.
- Keeps kernel simple, adaptable.
- Smooth transition to DISTRIBUTED SYSTEM.

The Network

"Ultimate" distributed system.

INTERNET

- "All the cooperating networks."

Circuit switched network

- Phone system.

Packet switched network

- Network system.

IP: Internet protocol.

- Packet.
  - 1-1500 bytes
  - from address
  - to address
- Address.
  - Ex. 128.112.128.43

ROUTERS

- Move packets across network.

TCP: Transmission control protocol.

- Break big messages into packets.
- Collect received packets into messages.
- Check for errors.

Domain Name System.

- Distribute authority/responsibility for name service.
- Can use "phoenix.princeton.edu" instead of 128.112.128.43.

(many details omitted!)

Operating System / Network Issues

Network applications.

- Communication (mail, news).
- Remote login (telnet).
- File transfer (ftp, Napster, Gnutella).
- Publishing (html).
- Browsing (Netscape, IE).
- E-commerce.

Modern rendition of ancient tradeoffs.

- Personal computer or Network computer.
- ONE huge virtual machine?!?

Compare/contrast.

- Computer center, phone system, Post office (snail mail), Libraries.

Current network ethics:

- Honor and foster individualism.
- Network is good and must be preserved.

Should hackers or the government "run" the net?

- Can commercial apps trust an "open net"?
- Does a "closed net" violate individual rights?

Security/Privacy/Copyright.

Who owns? Who pays?

Unix File System Layout

Goal: provide simple abstraction (sequence of bytes) for user programs.

Each disk has:

- I-nodes (one per file).
  - indexing information
  - pointers to disk blocks
- Data blocks.
  - just data

Superblock (block 1).

- Catalog of disk layout.
- Size and number of data blocks.
- Size and number of i-nodes.
- Free list of data blocks.

File.

- List of data blocks.

Directory.

- List of file names.
- i-node addresses

Forms a TREE structure.

- Traverse the tree for sequential access.

File Layout Examples

Small file.

- I-node lists data blocks.
- Ex: 10 i-node entries, 1K data blocks.
  - handles files < 10K

Medium-sized file.

- i-node lists blocks that list data blocks.
- Ex: 10 i-node entries.
  - 256 data block pointers/block
  - handles files < 2.56 M

Large file.

- Add a third level.
- Ex: 10*256*256*1K = 655.36 M.

Tradeoff on data block size.

- Too small: large files are excessively fragmented.
- Too large: excess waste in small files.