COS 126 Lecture 19: Operating Systems

Modern operating systems support
* software tools for creating programs (lect. 18)
* running multiple programs (multiprogramming)
* saving/acessing data (files, virtual memory)
* user interaction (window system)
* interaction with other systems (networking)
* core applications programs

software libraries
multiprogramming
file system/virtual memory
window manager
    single-user multiprograming
    graphic user interface
applications
networking

EXECUTION CONTROL
    OS keeps track of state of CPU, devices
EXTERNAL DEVICES
    display, keyboard, mouse, disks, CD, network
VIRTUAL MACHINES
    OS implements abtract devices
        simpler for user than real hardware
        easier to change to new hardware

Libraries and separate compilation

Programs use LIBRARIES or other modules
    ex: printf
Recompile library program when it is used?

Compiler produces OBJECT MODULE
* code
* addresses for variables defined within
* external variable references

RELOCATE: add start address to each
    instruction address field
RESOLVE: find actual addresses for
    each external variable, fill in
    at each place marked by the compiler

LINKER (link editor)
* concatenates input object modules
    to make a single output file
* resolves external references
* merges relocation tables

a.out = code + relocation tables

LOADER translates object code to executable code
    by adding start address to each reference
    requiring relocation (info in merged table)

Multiprogramming

Operating system "kernel"
    keeps track of several programs

INTERRUPT:
    stop
    save PC somewhere "special"
    change PC

Interrupts are
    necessary to manage input-output devices
    useful for multiprogramming

OS allows several programs to "share" CPU by
    keeping table of "current" PC's for programs
    setting clock to interrupt periodically

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
    (only load one copy of program; e.g. emacs)

UNIX file system layout

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

Each disk has
    "index" information (i-nodes)
    "data" (data blocks)

Superblock (block 1)
    size and number of data blocks
    size and number of i-nodes
    free list of data blocks

I-nodes (one per file)
    accounting info
    pointers to disk blocks
        (tree structure for big files)

Data blocks: just data

Directory: list of file names/i-node addrs

File: list of data blocks

forms a TREE
    traverse the tree for sequntial 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 files:

add a third level

Ex: 10*256*256*1K = 655.36 M

Typical medium-sized file

Tradeoff on data block size

Too small: Large files excessively fragmented

Too large: Excess waste in small files

Virtual Memory

Problem 1:

several programs need to share same memory

direct solution: divide up the memory

Problem 2:

program needs more memory than machine has

direct solution: "overlays"

Problem 3:

it's all just memory

why should file system look more complicated?

"Better" solution to all three problems:

all programs assume access to all memory

each program actually uses a small portion

VIRTUAL MACHINES

simulate multiple copies of a machine on itself

Ex: can debug OS

physical address space

how much memory is there?

limitation: $/bit cost

virtual address space

maximum amount of memory

an instruction could directly reference

limitation: address size (bits/instruction)

Paging

Widely-used method for implementing VM

Design hardware to "trap" all addresses
Keep virtual memory (for each program) on disk

Divide into PAGES
Keep table with
- flag indicating if page is in memory
- relative position of page in memory

Make page size a power of two,
    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
Each page brought in has to REPLACE another
    "page replacement strategies"
        still being studied, invented

Basic principles
    MEMORY HIERARCHY
        local: fast, small, expensive
        remote: slow, huge, cheap
    CACHE recently accessed information
        to give illusion that all info is local

Size of Virtual Memory

16 bits is not enough

24 bits is not enough

32 bits is not enough

Is 64 bits enough?

18446744073709551616 > 10^19 addresses

Need more sophisticated paging strategy, primarily
because page table would be too big
multilevel
associative

Some big numbers
10^20: number of grains of sand
10^27: number of oxygen atoms in a thimble
2^256: number of electrons in the universe

512 bits is certainly enough
no need for relocation,
use a galactic address dispenser

Window Manager

VIRTUAL TERMINALS

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
- flexibility vs. standardization
- other ways of interacting with computer?

Client-Server Model

System divided into two distinct parts

DISPLAY SERVERS
- present display
- monitor keyboard and mouse input
- (implement virtual display)

CLIENTS
- applications programs
- (use virtual display)

Server is interface between
- client program and display hardware

Model generalizes beyond display management
- client: request service
- server: do the work
- Ex: file service

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"
  - not 128.112.128.43 (/etc/hosts)

(many details omitted!)

Operating systems/network issues

Network applications

- communication (mail, news)
- remote login (telnet)
- file transfer (ftp)
- publishing (html)
- browsing (netscape)
- 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
- * the 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??