A Brief History of Operating Systems

Outline

- First generation of computing
- Second generation of computing
- Third generation of computing
- Fourth generation of computing

First Generation of Computing

- Hardware only
  - Programmers wrote, ran programs and operated system
  - Programmers could halt execution, modify programs and data, etc.
- Programming done in binary or rewiring plug boards
  - Then punch cards introduced to make life easier

Processor Utilization and Throughput

- “Open shop” approach
  - If you sign up for 1 hour, you get it — even if you do not need it all

Assume:

- Signup time is 15 min blocks
- Input time 0.3 min
- Output time 0.5 min
- Execution time 1.0 min

- *Processor utilization* = *execution time* / *total time* = 1 min / 15 min ≈ 7%
- *Throughput* = *number of jobs run* / *total time* = 1 job / 15 min = 4 jobs/hr

Second Generation of Computing

- Use of transistors
- Separation of operators, programmers
  - Programmers wrote programs on punch cards using assembly language, FORTRAN
  - Operators ran jobs

Processor Utilization and Throughput

- Now jobs can be run one after the other

Assume:

  - Input time 0.3 min
  - Output time 0.5 min
  - Execution time 1.0 min
- *Processor utilization* = *execution time* / *total time* = 1 min / 1.8 min ≈ 55%
- *Throughput* = *number of jobs run* / *total time* = 1 job / 1.8 min = 33 jobs/hr

Batch Processing

- Copy jobs from cards to tape using satellite computer
- Main computer runs tape, output collected on another tape
- Copy output from tape to paper using satellite computer

Processor Utilization and Throughput

Assume:

- Input time 0.3 min
- Output time 0.5 min
- Execution time 1.0 min

- And for the batching:
  - Delivery time of 50 jobs: 30 min
  - Transfer from cards to tape: 15 min
  - Mount tape: 5 min
  - Execution time (1 min / job): 50 min
  - Print output tape: 25 min
  - Manual separation of job outputs from printer: 15 min

Processor Utilization and Throughput

Key numbers:

- Batch execution time: 50 min
- Mounting time: 5 min
- Number of jobs executed: 50 jobs
- Execution time 1.0 min

- *Processor utilization* = *execution time* / (*mounting time + execution time*) = 50 min / (5 + 50) min = 50 / 55 ≈ 91%
- *Throughput* = *number of jobs run* / (*mounting time + execution time*) = 50 jobs / (5 + 50) min = 50 jobs / 55 min ≈ 55 jobs / hr

Brilliant Idea: Supervisory Programs

- Computers are fast at doing things people do, so *why not have them do the job scheduling*?
- First serious but informal discussion of writing such a supervisory program in 1953 in Herb Grosch's hotel room during the Eastern Joint Computer Conference
  - Intended to address idle time and work required to control I/O devices

Led to First Operating System

- The Input / Output System, for the IBM 701
  - Written at General Motors
  - Provided a set of routines for accessing I/O devices
  - If, at end of a job, the job branched back to it, it would accept and load next job
- Benefits
  - Offline operations easier
  - Changing I/O devices now just involved changing the driver in the set of routines and not in each program

Another Brilliant Idea: Buffering

- Overlap CPU, I/O
  - Idea: keep both CPU, I/O devices busy simultaneously
  - Means I/O must be done independently of CPU
- First done in SHARE operating system
  - Written by SHARE, IBM's user group, for IBM 709
  - Improved speed and automated much of operator's job
  - Operators had to load, unload cared and tapes
  - Little error recovery

Rise of Disk Operating Systems

- Resident Loader
  - Loads both user, system programs into memory; readies them for execution; passes control to them
  - Programs return control to operating system
  - Process repeats
- Job Control Language
  - Users used this to inform systems of needs of jobs (memory, printers, etc.)
- Device Independence
  - Came from device support for many different devices

Atlas System

- First computer designed to support operating system
- Several hardware innovations:
  - Extracodes: special instructions causing traps to invoke special software routines
  - Notion of virtual memory: one-level store using large disk or drum as backup memory for main store
  - Interrupts: used to determine when external event occurs

Interrupt Examples

- Alarm clock: at certain time, invoke the following software routine:

```
Disable alarm clock interrupt
Save program status
Invoke requested routine
Reset alarm clock interrupt
Reset program status
Resume normal processing
```

- Device: input occurs; invoke the following software routine:

```
Disable device interrupt
Save program status
Invoke appropriate routine
Reset device interrupt
Reset program status
Resume normal processing
```

Third Generation of Computing

- Use of integrated circuits
- Multiprogramming: where many jobs run interleaved on one machine
  - Spooling: buffering jobs (just like I/O); put jobs on disks to enable quick input and moving data and instructions between the disks and CPU
  - Monitor can schedule jobs as disks can be accessed in random order
    - Better than tapes, which must be accessed sequentially
  - Computation of one job, I/O of another, can overlap

Examples

- First implemented in EXEC II, the operating system for UNIVAC 1107
  - Ran faster than users could load cards
  - Performance in time between submission, resubmission of a job, 33% circulated in under 5 min, processor utilization 90%
- Burroughs 5000 Master Control program
  - Used priorities to determine which job to run
  - All user programs written in ALGOL or COBOL
    - No assemblers available

Customer Service and Compatibility

- IBM System/360 family used these
  - Extensive customer service, support
  - Very powerful (for the time) operating system, OS/360
    - See Fred Brooks' book *The Mythical Man-Month*
  - Upward compatibility for all systems in IBM 360 family
  - Powerful job control language

Problems Introduced

- Protection required
  - To prevent one job from overwriting others
  - To prevent job from executing illegal instructions, causing system to crash
  - To prevent infinite use of the CPU
  - To prevent one job from interfering with I/O of another job

Hardware Solutions

- Illegal instructions cause trap to prevent system crash
- Special fence register separates operating system from jobs
  - Crossing it triggers trap
- Upper, lower bounds registers delimit memory allocated to current job
  - Crossing beyond either triggers trap

Software Solutions

- Use timer to interrupt jobs that hog CPU too long
- To prevent jobs from interfering with I/O of each other, define at least 2 modes of execution
  - Kernel (aka supervisor, system, monitor) mode executes privileged instructions
  - User mode is mode use jobs run in; attempting to execute privileged instructions causes trap
  - To invoke kernel mode from user mode, issue system code (the extracodes of Atlas)

System Calls

computer	system call opcode
IBM 370	SVC
DECSystem-10	UUO
DECSystem-20	JSYS
PDP-11	TRAP
VAX-11	CHMK, CHMS, CHME

- Load code for system call into register
- Instruction causes trap
- Operating systems checks legality of request, acts accordingly
- Operating system can do things not related to current job
  - Example: spooling

Time Sharing

- Proposed by Strachy in 1959
- MIT's CTSS (Cambridge Time Sharing System), SDC's Q-32 were earliest time sharing systems
  - Reduced time between job submission, getting results
  - Guaranteed quick response to short requests
  - Many users shared computer
- MIT's Multics combined time sharing with many Atlas features such as virtual memory, protection, device independence
- Batch often merged with time sharing on various systems (BBN TOPS-10, DEC TOPS-20, VAX VMS)
- Time sharing added to some batch systems (IBM OS/360 with TSO)

Layered Machines

- Archetype is Dijkstra's THE system
- As you go up the layers, each defines a more developed system
  - Called "layers of abstraction"
  - Processes at level *j* ignore all issues at layer *j*–1, invoking process at that layer when the services it provides are needed
- Each layer forms an "abstract" or "virtual" machine

THE Layers

- *Level 0*: hardware layer
  - Real time clock interrupt to prevent CPU hogging
  - Processor management
- *Level 1*: segment controller process layer
  - Manages storage
  - Higher layers see only segments, their actual locations being hidden
- *Level 2*: Operator console (message interpreters)
  - Handles traffic to, from operator at system console
  - Higher levels see their own console

  Separate process because first part of message must be processed to figure out which process the message is to be sent to

THE Layers

- *Level 3*: I/O handlers
  - Buffer input, unbuffer output
  - Higher levels see logical device units, not registers

  Above message interpreter because, if a device malfunctions, system must be able to inform operators
- Level 4: User processes
  - Each has complete virtual machine with separate I/O devices, operator console, segmented storage, CPU
  - Other than communication via primitives, processes completely isolated
- Level 5: operator (not implemented)

First True Virtual Machine

- IBM's CP/CMS, aka OS/370
  - Gave users access to all machine features including illusion of private address space, CPU, I/O devices ("minidisks")
- How it works
  - Underlying VM is real monitor
  - 3 modes: virtual user, virtual monitor, real monitor
  - All traps, interrupts on VM cause trap, interrupt to real monitor
  - Real monitor handles it, modifies virtual monitor to make it appear virtual monitor had handled the request, and restarts virtual monitor

Virtual Machine

- Speed
  - How much you have to so in software
  - Example: IBM VM/370 only simulates privileged instructions, so speed acceptable
- Advantages
  - You have complete user isolation
  - You can use it for operating system development; if your operating system crashes, others can continue working and you need not restart the machine
- Disadvantages
  - Sharing hardware can be painful
  - How do you share 3 disks among 7 users?
    - One way: give each user a smaller, “virtual” disk

Minicomputers

- Initially, referred to price; gradually became somewhat related to size
- Some examples through the decades follow
- 1950s: Burroughs E-101, Bendix G-15
  - Price: under $50,000
  - Characteristics: large size; vacuum tubes; slow

Minicomputers

- 1960s: CDC 160, IBM 1620 had 12-bit instruction size
  - Introduced relative, indirect addressing modes

  PDP-1, PDP-8
  - Price: under $18,000
  - Characteristics: PDP-8 introduced real-time clock, DMA, etc. beginning real-time control for minicomputers

  DDP 116, DATA 620, IBM 1130, IBM 1800
  - Characteristics: 16-bit architecture, vectored interrupt, multiple accumulators
  - DDP 116 had I/O Selector, a beginning operating system for minicomputers
  - IBM 1800 introduced disk operating systems
  - All accepted commands from a terminal, could monitor real-time devices

Minicomputers

- 1970s: PDP-11
  - Characteristics: planned as family of compatible computers
  - RT-11 operating system was single user, had notion of foreground, background jobs
  - RSTS operating system was time sharing system
  - RSX-11 operating system was real-time operating system

  UNIX
  - Thompson, Ritchie at Bell Labs
  - Used ideas from CTSS, Multics but made their operating system cleaner and simpler in part due to space constraints
  - Originally written in assembly but later rewritten in C for maintainability, portability

Fourth Generation of Computing

- Use of microprocessor, contained on a single large-scale integrated(LSI) or very large-scale integrated (VLSI) chip
- Made true minicomputers, microcomputers available

Microcomputers

- Intel 4004: 4-bit CPU
- Intel 8008:8-bit CPU; Intel 8080 improved it
- Many hobbyists built these; some companies also made them
  - Apple II based on 6502 chips
- Digital Research created CP/M (Control Program for Microprocessors)

Workstations

- Started at Xerox PARC
  - Alto was first computer to use mouse
  - Star was to be basis for “office of the future” that Xerox was developing
    - Did badly, but inspired others
- Sun Microsystems began marketing Sun workstations, which quickly became widely accepted in technical community
  - Used 2 versions of UNIX; initially SunOS, then also Solaris

Personal Computers

- Apple Lisa computer was not very successful
- Successor, the Apple Macintosh, introduced in 1984, was
- IBM PCs, introduced in 1981, became popular
- Microsoft introduced MS-DOS in 1981
  - Began as a CP/M clone called 86-DOS purchased from Seattle Computer Products
- Microsoft Windows released in 1985
  - Graphical front-end for MS-DOS
  - Windows NT released in 1993, based on MS-DOS
  - Windows 95 released in 1995; compatible with MS-DOS