Operating System

Dr. Satyabrata Das

Associate Professor

Dept. of IT, VSSUT, Burla

Major issues in Operating Systems

- **structure -- how is an operating system organized?**
- **sharing -- how are resources shared among users**
- **naming -- how are resources named (by users or programs)**
- **protection -- how is one user/program protected from another**
- **security -- how to restrict the flow of information**
- **performance -- why is it so slow?**
- **reliability and fault tolerance -- when something goes wrong**
- **extensibility -- how do we add new features?**
- **communication -- how and with whom can we communicate (exchange information)**

Major issues in OS (2)

- **concurrency -- how are parallel activities created and controlled?**
- **scale and growth -- what happens as demands or resources increase?**
- **persistence -- how to make data last longer than programs**
- **compatibility -- can we ever do anything new?**
- **distribution -- accessing the world of information**
- **accounting -- who pays the bills, and how do we control resource usage?**

A brief history of operating systems

- **"in the beginning", the OS was just code to which you linked your program, loaded the whole thing into memory, and ran your program; basically, just a run-time library**
- **simple batch systems were first real operating systems:**
    - os was stored in part of primary memory
    - it loaded a single job (from card reader) into memory
    - ran that job (printed its output, etc.)
    - loaded the next job...
    - *control cards* in the input file told the os what to do
- ***Spooling* and *buffering* allowed jobs to be read ahead of time onto tape/disk or into memory.**

Early Systems - Bare Machine (1950s)

Hardware – *expensive* ; Human – *cheap*

- Structure
  - Large machines run from console
  - Single user system
    - Programmer/User as operator
  - Paper tape or punched cards
- Early software
  - Assemblers, compilers, linkers, loaders, device drivers, libraries of common subroutines.
- Secure execution
- Inefficient use of expensive resources
  - Low CPU utilization, high setup time.

From John Ousterhout slides

Simple Batch Systems (1960's)

- Reduce setup time by batching jobs with similar requirements.
- Add a card reader, Hire an operator
  - User is NOT the operator
  - Automatic job sequencing
    - Forms a rudimentary OS.
  - Resident Monitor
    - Holds initial control, control transfers to job
  - Problem
    - Need to distinguish job from job and data from program.

From John Ousterhout slides

Supervisor/Operator Control

- Secure monitor that controls job processing
  - Special cards indicate what to do.
  - User program prevented from performing I/O
- Separate user from computer
  - User submits card deck
  - cards put on tape
  - tape processed by operator
  - output written to tape
  - tape printed on printer
- Problems
  - Long turnaround time - up to 2 DAYS!!!
  - Low CPU utilization
    - I/O and CPU could not overlap; slow mechanical devices.

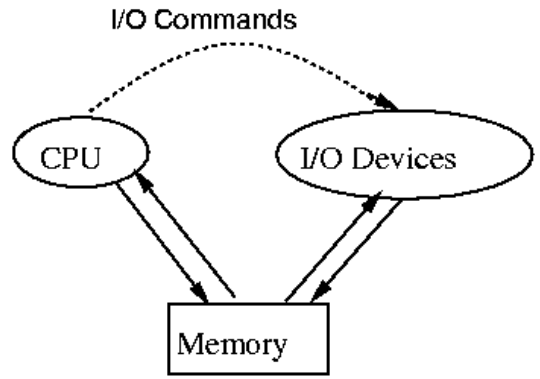
From John Ousterhout slides

Batch Systems - Issues

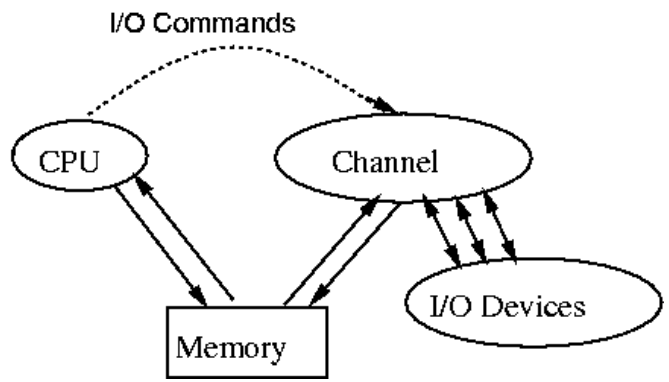
- Solutions to speed up I/O:
- Offline Processing
  - load jobs into memory from tapes, card reading and line printing are done offline.
- Spooling
  - Use disk (random access device) as large storage for reading as many input files as possible and storing output files until output devices are ready to accept them.
  - Allows overlap - I/O of one job with computation of another.
  - Introduces notion of a job pool that allows OS choose next job to run so as to increase CPU utilization.

Speeding up I/O

- Direct Memory Access (DMA)



- Channels



Multiprogramming

- *Multiprogramming* systems provided increased utilization
  - keeps multiple runnable jobs loaded in memory
  - overlaps I/O processing of a job with computes of another
  - benefits from I/O devices that can operate asynchronously
  - requires the use of interrupts and DMA
  - tries to optimize *throughput* at the cost of response time

Timesharing

- ***Timesharing*** **supported *interactive* use of**
  - each user feels as if he/she has the entire machine (at least late at night!)
  - timesharing tries to optimize *response time* at the cost of throughput
  - based on time-slicing -- dividing CPU equally among the users
  - permitted active viewing, editing, debugging, participation of users in the execution process
- **MIT Multics system (mid-late 1960s) was first large timesharing system**

Personal Computing Systems

Hardware – *cheap* ; Human – *expensive*

- Single user systems, portable.
- I/O devices - keyboards, mice, display screens, small printers.
- Laptops and palmtops, Smart cards, Wireless devices.
- Single user systems may not need advanced CPU utilization or protection features.
- Advantages:
  - user convenience, responsiveness, ubiquitous

Distributed Operating Systems

- **distributed systems facilitate use of geographically distributed resources**
  - **machines connected by wires**
- **supports communication between parts of a job or different jobs**
  - **interprocess communication**
- **sharing of distributed resources, hardware and software**
  - **resource utilization and access**
- **permits some parallelism, but speedup is not the issue**

Parallel Operating Systems

- Support parallel applications wishing to get speedup of computationally complex tasks
- Needs basic primitives for dividing one task into multiple parallel activities
- Supports efficient communication between those activities
- Supports synchronization of activities to coordinate sharing of information
- It’s common now to use networks of high-performance PCs/workstations as a parallel computer

Embedded Operating Systems

- The decreased cost of processing makes computers ubiquitous. Each "embedded" application needs its own OS or control software:
  - cell phones
  - PDAs (Palm Pilot, etc.)
  - "network terminals" (internet interfaces)
- In the future, your house will have 100s of these things in it (if it doesn't already)

Real-time systems

- Correct system function depends on timeliness
- Feedback/control loops
- Sensors and actuators
- Hard real-time systems -
  - Failure if response time too long.
  - Secondary storage is limited
- Soft real-time systems -
  - Less accurate if response time is too
  - Useful in applications such as multimedia, virtual reality.

A personal computer today

interaction

- Super AMOLED display
- Capacitive touchscreen (multitouch)
- Audio (speaker, microphone)
- Vibration
- S pen

- 4G LTE
- NFC
- WiFi
- Bluetooth
- Infrared

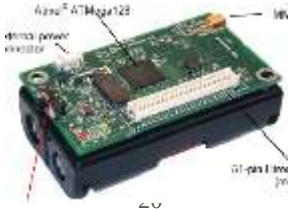
- 64 GB internal storage (extended by microSD)
- Adreno 330 GPU
- Hexagon DSP
- Multimedia processor

- 13 MP front camera
- 2 MP back camera
- Accelerometer
- Gyroscope
- Proximity sensor
- Compass
- Barometer
- Temperature sensor
- Humidity sensor
- Gesture Sensor
- GPS

Operating systems are everywhere

Operating systems are everywhere

Systems Today



Irvine Sensorium