Slide 1

Operating System Overview

COMP3231/COMP9201 Operating Systems

2004/S2

Slide 2

What are the objectives of an Operating System?

➜ convenience & abstraction

- the OS should facilitate the task of application and system programmer
- hardware details should be hidden, uniform interface for different I/O devices provided

➜ efficiency

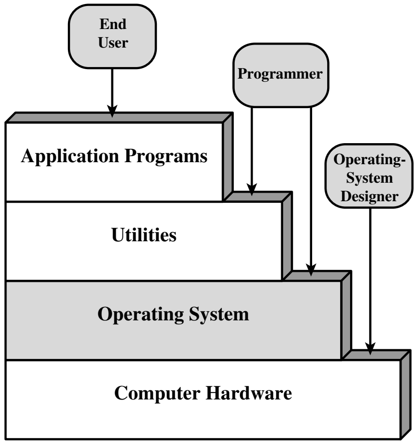
should take up few resources, make good use of resources, and be fast

➜ protection

fairness, security, safety

Slide 3

Layers of Computer System



Slide 4

Services Provided by the Operating System

➜ Program execution

- load instructions and data into main memory
- initialise I/O devices, etc

➜ Access to I/O devices

- provides a uniform interface for various devices

➜ Controlled access to files

- abstracts over structure of data on I/O device
- provides protection mechanisms

Slide 5

Services Provided by the Operating System

➜ System access: provides protection of

- data
- system resources; and
- resolves access conflicts

➜ Program development

- Editors, compilers, and debuggers: not part of the core, but usually supplied with the OS.

Slide 6

Services Provided by the Operating System

➜ Error detection and response

Possible errors:

- internal and external hardware errors
  - memory error
  - device failure
- software errors
  - arithmetic overflow
  - access forbidden memory locations
- operating system cannot grant request of application

the OS has to

- clear error condition
- minimise effect on other applications

Slide 7

Services Provided by the Operating System

➜ Accounting

- collect statistics
- monitor performance
- used to anticipate future enhancements
- used for billing users

Slide 8

Operating System

The operating system controls the

- movement, storage, and processing of data

but it is not always 'in control':

➜ functions same way as ordinary computer software

- it is just a program (or a set of programs) that is executed
- relinquishes control of the processor to execute other programs
- must depend on the processor to regain control

Slide 9

Kernel

- Portion of operating system that is running in privileged (or "kernel" or "supervisor") mode
- Usually resident in main memory
- Implements protection
- Contains fundamental functionality required to implement other services
- Also called the nucleus or supervisor

Slide 10

Evolution of an Operating System

OS have to evolve over time because of

- hardware upgrades and new types of hardware
- changing performance and costs leading to changing trade-offs
  - hardware gets cheaper, bigger, faster
  - people get more expensive
- New services
  - graphical user interfaces
  - file systems
- Fixes

Slide 11

Evolution of Operating Systems

Serial Processing: late 1940s to mid 1950s

- No operating system
- Machines run from a console with display lights and toggle switches, input device, and printer
- Manual schedule
- Setup for each user included
  - loading the compiler, source program,
  - saving compiled program,
  - loading and linking

Improvements: libraries of common functions, linkers, loaders, compilers, debuggers available to all users.

Slide 12

Evolution of Operating Systems

Simple Batch Systems: mid 1950s, by GM for IBM 701

- The monitor controls the execution of programs:
  - it batches jobs together
  - the program branches back to monitor when finished
  - resident monitor is in main memory and available for execution
- Instructions to monitor via Job Control Language (JCL)
  - the monitor contains a JCL interpreter
  - each job includes instructions in JCL to tell the monitor
    - what compiler to use
    - what data to use
  - predecessor of shell

Monitor takes up main memory and CPU time but improves utilization of computer

Slide 13

Hardware Features

New hardware features support development of OS features

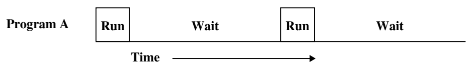
- ➜ Memory protection
  - do not allow the memory area containing the monitor to be altered
- ➜ Timer
  - prevents a job from monopolizing the system
- ➜ Privileged instructions
  - for example, I/O instructions
- ➜ Interrupts
  - relinquishing control to and gaining control from user program

Slide 14

Uniprogramming

Problem:

- ➜ Processor must wait for I/O instruction to complete before preceding
- ➜ I/O instructions are very slow compared to computations



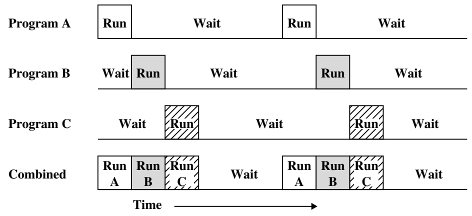
Solution: Interleave the execution of multiple jobs!

Slide 15

Multiprogramming

When one job needs to wait for I/O, the processor can switch to the other job

- ➜ Increased throughput
- ➜ Increased utilisation

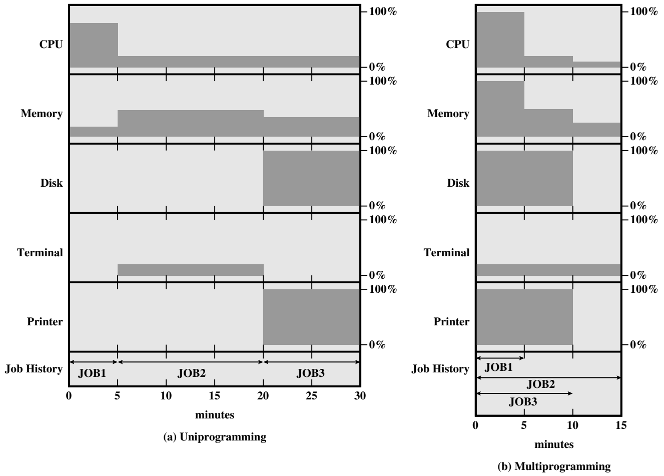


Slide 16

Example

	Job 1	Job 2	Job 3
Type of Job	CPU bound	I/O bound	I/O bound
Duration	5 min	15 min	10 min
Memory req't	50k	100k	80k
Disk?	No	No	Yes
Terminal?	No	Yes	No
Printer?	No	No	Yes

Slide 17



Slide 18

Effects of Multiprogramming

	Uniprogramming	Multi-programming
Processor utilis.	22%	43%
Memory utilis.	30%	67%
Disk utils.	33%	67%
Printer utilis.	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/h	12 jobs/h
mean resp. time	18 min	10 min

Slide 19

Time Sharing

Batch multiprogramming improves the utilisation of static jobs, but what about interactive jobs?

- ➜ Using multiprogramming to handle multiple interactive jobs
- ➜ Processor's time is shared among multiple users
- ➜ Multiple users simultaneously access the system through terminals

Slide 20

Batch Multiprogramming versus Time Sharing

Different requirements for interactive execution

	Batch Multiprogramming	Time Sharing
Principal objective	Maximise CPU utilisation	Minimise response time
Control	JCL with job	Interactive commands

One of the first systems: Compatible Time-Sharing System (CTSS), 1961, IBM 709 & IBM 7094

- ➜ a system clock creates interrupts in regular intervals
- ➜ system switches to a new user
- ➜ old user's program and data saved to disk

Primitive Time Sharing (CTSS)

Job1: 15,000 Job3: 5000

Job2: 20,000 Job4: 10,000

Slide 21