

Operating system issues

- Operating system (O/S) functions
- A brief history of operating systems
- Processes and Process Management
- References:
 - Tanenbaum, chapters 1.2, (6.1) and 6.4
 - Specification of the PeANUt computer (Section 3)
 - “Operating System Concepts with Java”, A. Silberschatz, P.B. Galvin, and G.Gagne
 - Bryant & O’Hallaron, chapter 8.2

Main operating system functions

- Provide support mechanisms for the user interface (command line, GUI)
- Sharing hardware and other resources among users
- Allowing users to control sharing of data amongst themselves
- Facilitating input/output (I/O)
- Recovering from errors

Operating systems

- Essential to the operation of modern computers!
- Greatest achievement: Gives users access to multiple **virtual** computers from a single machine (**iwaki** supports ≤ 100 users at once, each with the illusion of using a separate machine)
- What is an operating system? Software or firmware programs, that
 - make the computer more usable, convenient, secure
 - manage the hardware carefully to give good performance

A brief history of operating systems (1)

- Zeroth Generation (1940's)
 - No real O/S: Users hand-coded all functions in machine language
 - Programs loaded via a bootstrap program
 - I/O facilitated by subroutine libraries
- First Generation (1950's)
 - Job-oriented machines; O/S provided easy transition between them
 - O/S initialised the machine, allowing the batching of jobs (thus increasing the efficiency of computer usage)

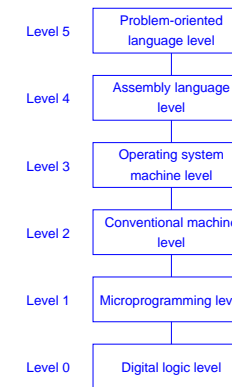
A brief history of operating systems (2)

- Second Generation (early 1960's)
 - Development of multiprocessing (the illusion that many jobs run simultaneously), background jobs became possible
 - Virtual memory developed
 - Standardisation of memory media (disks/tapes), so that data became more portable
 - Development of real-time systems (allowed computers to control industrial processes)
- Third Generation (mid-60's to mid-70's)
 - Development of general purpose systems (attempted to make the computer a truly universal machine)
 - The O/S were large and often cumbersome (provided thick layer(s) of software between the user and the hardware)

5

COMP2300, 2006

Review of abstract machine levels



7

COMP2300, 2006

A brief history of operating systems (3)

- Fourth Generation (mid-70's to present)
 - Emphasis on O/S user interface (window systems, mouse)
 - Strong support for machine networking (Internet), & distributed computing
 - Security ⇒ more significant; data encryption used widely
- Current operating systems
 - UNIX (Solaris, Linux, etc.)
 - MS-DOS, Windows-95, Windows-NT, Windows 2000, ...
 - VMS, System 7 / Mac Os X (Macintosh)
- New trend: **virtualization**
e.g. Xen <http://www.cl.cam.ac.uk/Research/SRG/netos/xen/>

6

COMP2300, 2006

Key Concepts

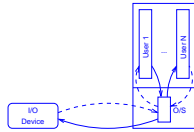
- Process Management
- Storage Management
 - Memory and Input/Output
- Distributed System Management
 - Networking and Distributed File Systems
- Protection and Security

8

COMP2300, 2006

Processes

- We can think of a process as being the execution of a single program
 - Program code, program counter, content of processor registers, data allocated on stack, global allocated data.
- Why having **multiple processes**?
 - A single process may not utilise the CPU well
 - Much time is wasted with I/O
Example:



- Can let other processes use CPU while waiting for external activities, like I/O

9

COMP2300, 2006

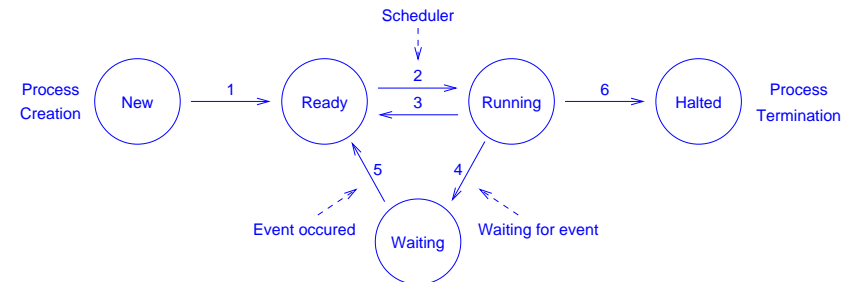
Fair CPU usage

- How can the CPU be shared fairly between various processes?
- Is the scheme of allowing other users in when waiting on I/O fair?
 - It depends entirely on the frequency and nature of such I/O requests
 - Can a fairer scheme be devised?
- Enforcing limits on processes using the CPU by using **time slicing**
 - O/S (with hardware support) generate a special timer interrupt when the time is up to change process
 - Other processes wait in a **queue**
- Processes can be assigned **priorities**

10

COMP2300, 2006

Process scheduling



11

COMP2300, 2006

Process context

- O/S must be able to **interrupt** a process and later restore it to exactly the same state
- The O/S must save the process's state information
- The memory associated with one process must not be modified by another
- Upon stopping a process, the O/S must save the values of all registers
- Prior to restarting a process, the O/S must
 - recover the values of all registers
 - ensure that its code is in main memory

(What's the difference between an interrupt and trap?)

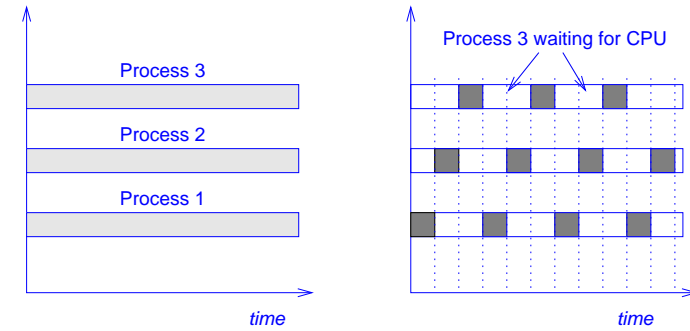
12

COMP2300, 2006

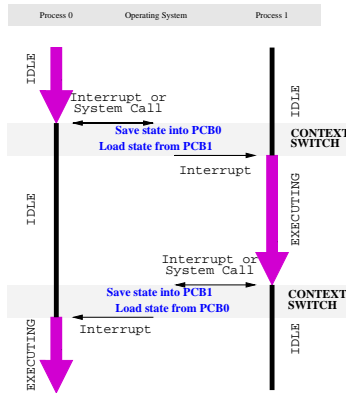
Process Control Block

Process State
Process Number
Program Counter
Registers
List of Open Files
Memory Limits
Possible Other Info

Multiprocessing



Context Switching

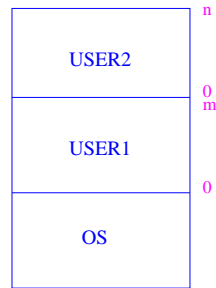


Multiprocessing, multiusers

- Virtual level 3 (O/S level) machines
- Effect of **multiprocessing** is that a single processor can create the effect of multiple processors
- Several users may simultaneously use the same machine and consider themselves to be the sole user

Protection

- Must ensure that **user1** cannot clobber (processes of) **user2** or **operating system**



Core Operating System Requirements

