Operating System Issues

- ref: [Tanembaum, sect 1.2, (6.1) & 6.4] [O'H&Bryant, sect 8.2]
- operating system (O/S) functions
- a brief history of operating systems
- key concepts:
  - process management
    - scheduling, process state, multiprocessing
  - storage management (later: O2-O4)
    - memory and input/output
  - distributed system management (later: N1, COMP2410, COMP3310)
    - networking and distributed file systems
  - protection and security (COMP2410)

Operating Systems

- essential to the operation of modern computers!
- greatest achievement: gives users access to multiple virtual computers from a single machine

  (servers like `iwaki` can support $\leq$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, fairness, ...
- main O/S functions:
  - define 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

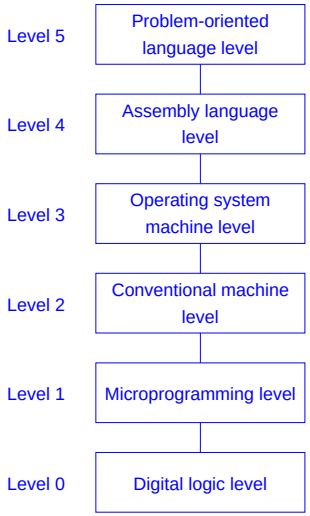
A Brief History of Operating Systems (1)

- 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)
- 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)

A Brief History of Operating Systems (2)

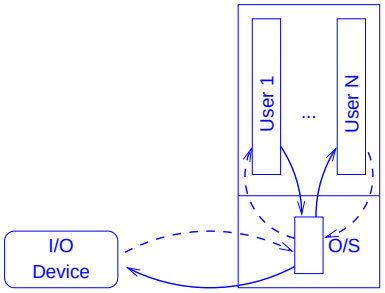
- Third Generation (mid-60's to mid-70's)
  - development of general purpose systems (attempted to make the computer a truly universal machine)
  - the O/S were large and often cumbersome (provided thick layer(s) of software between the user and the hardware)
- Fourth Generation (mid-70's to present)
  - emphasis on O/S user interface (window systems, mouse)
  - strong support for machine networking ($\rightarrow$ internet) & distributed computing
  - security $\Rightarrow$ 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. VMware, Xen, KVM

Review of Abstract Machine Levels



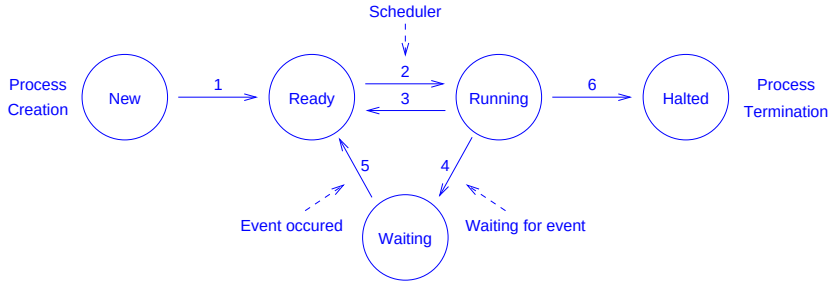
Processes

- we can think of a process as being the execution of a single program
  - program code, program counter, contents of processor registers, data allocated on stack, global data
- why have multiple processes?
  - a single process may not utilise the CPU well
  - much time may be wasted with I/O (see diagram)
  - can let other processes use CPU while waiting for external activities, like I/O
  - enables a computer to be *multi-functional* (even a modern PC)!



Fair CPU Usage and Scheduling

- how can the CPU be shared fairly between various processes?
- is the scheme of allowing other users in when waiting on I/O fair?
  - depends on the frequency and nature of such I/O requests
- better: enforce limits on processes using the CPU by using time slicing
  - typically 1–10 ms Q: what considerations for this choice?
  - O/S (with hardware support) generate a special timer interrupt when the time is up to change process (other processes wait in a queue)



- note: processes can be assigned priorities

Process State

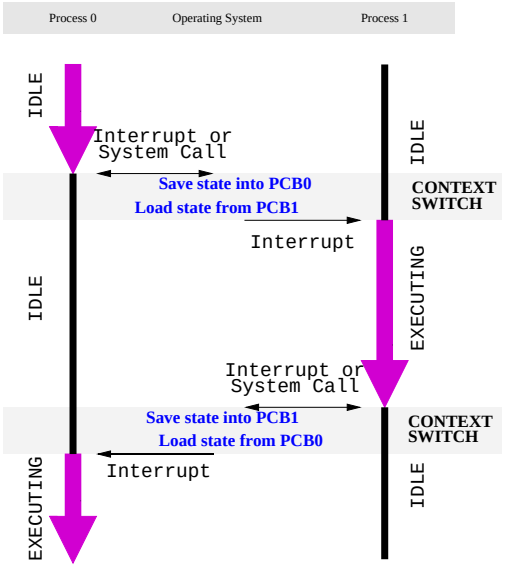
- O/S must be able to interrupt a process and later restore it to exactly the same state
- the O/S must save the process's state information
  - kept in the process control block
- the memory associated with one process must not be modified by another
- upon stopping a process, the O/S must save the values of all registers
- prior to restarting a process, the O/S must
  - recover the values of all registers
  - ensure that its code/data is in (or can be paged into) main memory

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

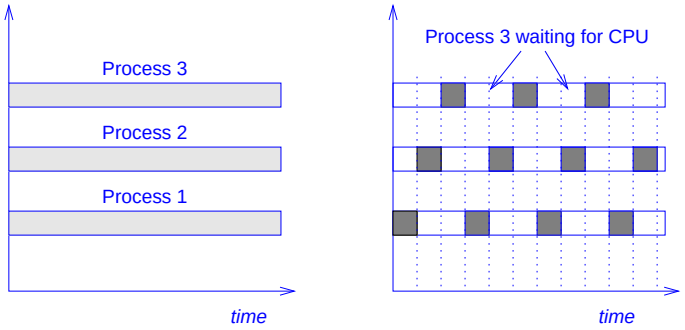
process control block:

Process State
Process Number
Program Counter
Registers
List of Open Files
Memory Limits
possible other info

Context Switching: Process State Manipulation



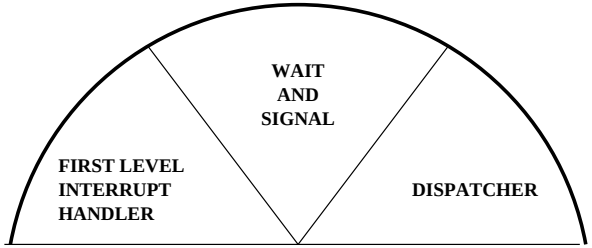
Multiprocessing and Multiusers



- effect of multiprocessing is that a single processor can create the (virtual) effect of multiple processors
- several users may simultaneously use the same machine and consider themselves to be the sole user
- memory protection: must ensure that user1 cannot 'clobber' (processes of) user2 or the operating system (example)

Core Operating System Functions: Process Management

- within the level 3 (O/S level) abstract machine, OS functions may be arranged at several levels
- process management is largely handled at the innermost level



Operating System Nucleus