Process Management

Process Concepts

Frédéric Haziza <daz@it.uu.se>

Department of Computer Systems
Uppsala University

Spring 2007

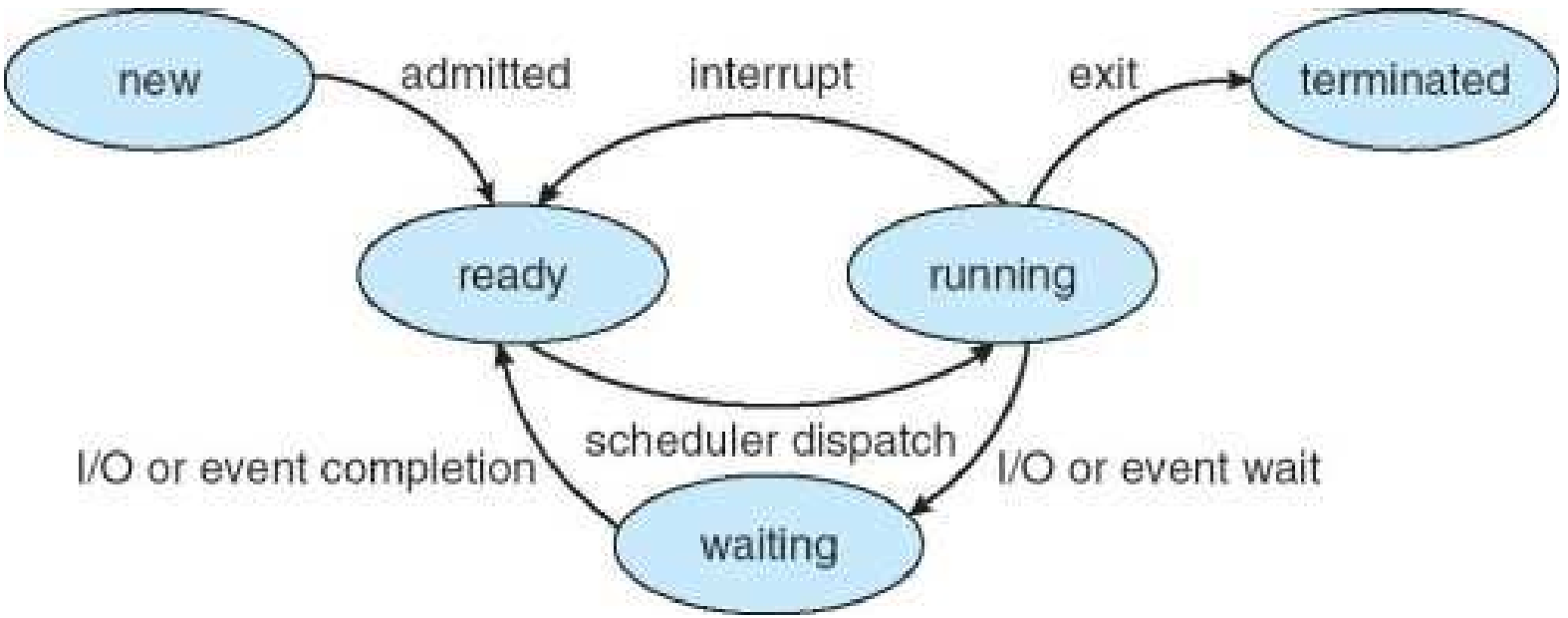
Outline

What characterizes a process?

- Program in execution
- Stack (Temporary data, function parameters,...)
- Heap
- Data section (Global variables)
- CPU Registers
- Program Counter (PC)

- Program code = Text section
- Program in execution = text section (executable file) *loaded in memory*

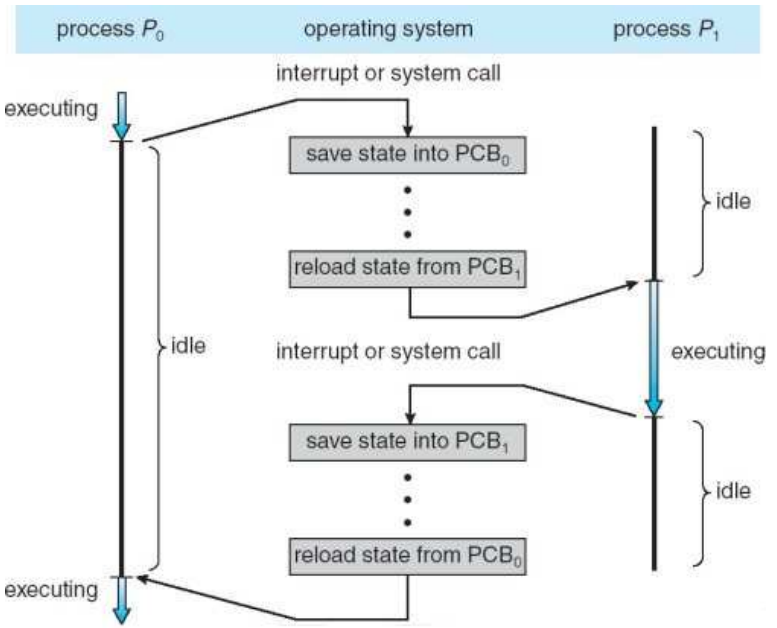
States


new
admitted
interrupt
exit
terminated
ready
running
scheduler dispatch
I/O or event completion
I/O or event wait
waiting

Process Control Block (PCB)

PCB
process state
process ID (number)
PC
Registers
memory information
open files
other resources

- Interrupts
  - Generated asynchronously by external devices and timers
  - Example: The I/O is complete or timers have expired
- Traps (software errors, illegal instructions)
- System calls



Scheduling Queues

Job Queue

Linked list of PCBs

- (main) job queue
- ready queue
- device queues

Scheduling

- **Job scheduler** (loads from disk)
- **CPU scheduler** (dispatches from ready queue)

Context Switch

Context switch

PCB swap

- Cost?
- 10ms switch for 100ms work => 9% wasted

Process creation and termination

See the lab 1 ...

Interprocess Communication (IPC)

2 models

- Message Passing
- Shared Memory

See black board...

Benefits

- Small amount to exchange
  => Message Passing, because no conflict to avoid
- Shared Memory
  =>Working at the speed of memory – faster

Shared Memory

Recall that the OS prevents processes to share memory => Agreement on relaxing restriction

Example (Producer-Consumer)

Unbounded buffer and bounded buffer (book p98)

Requires:

- Synchronisation
  (No consumption of non produced items)
- Waiting

Message Passing

No shared space.
Can be distributed accross network

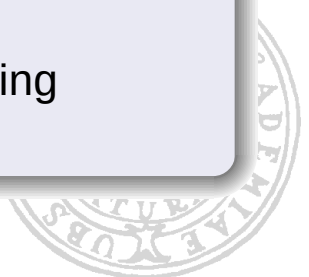
Example

Chat program

- send(m)
- receive(m)

Requires a communication link

- direct or indirect
  (mailbox/ports)
- synch. or asynch.
  (blocking or non-blocking)
- automatic or explicit buffering
  (info on the link)



Threads

Heavy-weight *vs* Light-weight...

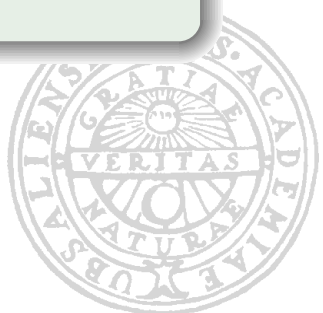
Example (Web server)

We want to serve more than one client at a time

- 1 process. If incoming request, new process created => costly!
- 1 process. If same task as other one, why overhead => better to multithread

On Solaris:

- Time for creating a process = 30 x time for creating a thread
- Time for context switching = 5 x time for switching a thread



Benefits

- Responsiveness
- Resource sharing
- Economy
- Utilization of multiprocessor architectures

Multithread Models

Deals with correspondance between

- threads in user space
- threads in kernel space

One to One **Many to One** **Many to Many**

Issues

- *fork()* creates a copy of a process with all threads
  or
  just the one which calls the fork?
- Cancellation
- Signal handling: Read in book page 139
- Thread pool (limit in system, pre-create threads)
- Thread specific data (sharing data?)
- about the models themselves