1B11 Operating Systems

Processes

Prof. Steve R Wilbur
s.wilbur@cs.ucl.ac.uk

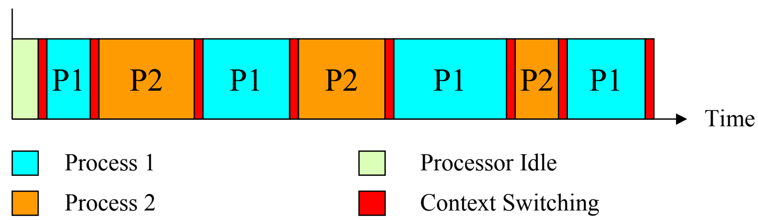
Lecture Objectives

- How do we make best use of the processor cycles?
- How do we do Multiprogramming?
- What is a Process?
- How are they managed?

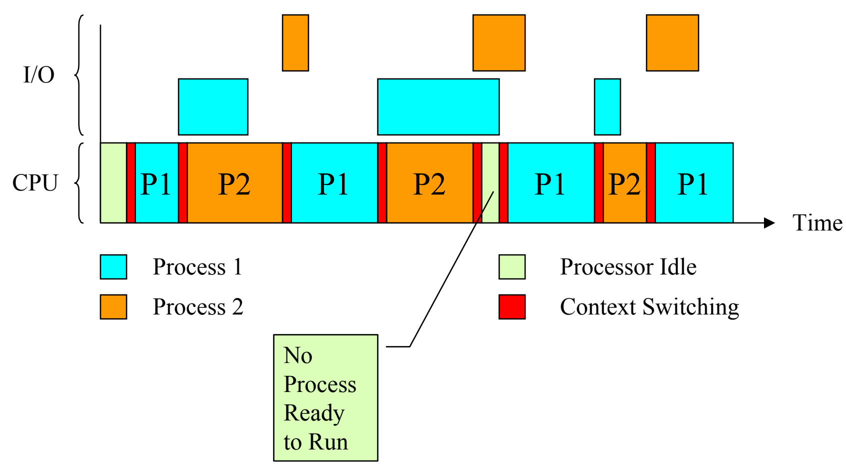
1B11-2 1999 Slide 2

Why have Processes?

- Computers (were) expensive
- CPU idle while waiting for I/O (polled)
- Could have several programs in memory
- Switch to another as they engage in I/O
- Now important management concept



In More Detail



A Process

- A Process is a running program - the program that the CPU is currently fetching and executing instructions for
- A process exists in a virtual machine consisting of:
  - memory
  - a schedule
  - a set of I/O channels that it is allowed to use

1B11-2 1999 Slide 5

Process Life Cycle

- To start a program, the OS must:
  - Find the binary/executable on disk
  - Allocate storage for the code and data
  - Allocate swap space on disk
  - Map the pages in memory and swap space
  - Copy the code into the physical pages
  - Start execution by saving current context (see later) and jumping (loading the PC) to the start address of the code!

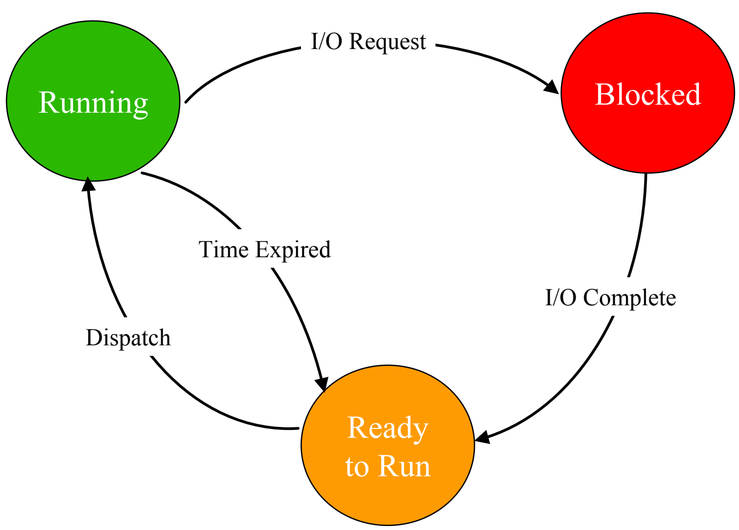
1B11-2 1999 Slide 6

Important Features

- Code and data for process must be in memory
- When process is suspended from running need to save its *current state*
- Saving, restoring this state and deciding which process to run is known as *context switching* - *"wasted" time* or *overhead*
- Needs to be *policy* for choosing next process to run - high level *scheduler*

1B11-2 1999 Slide 7

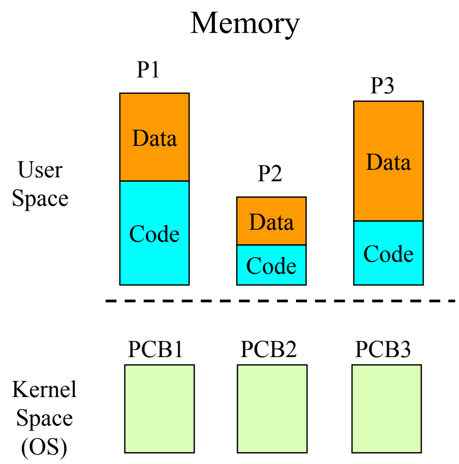
Process State Transitions



1B11-2 1999 Slide 8

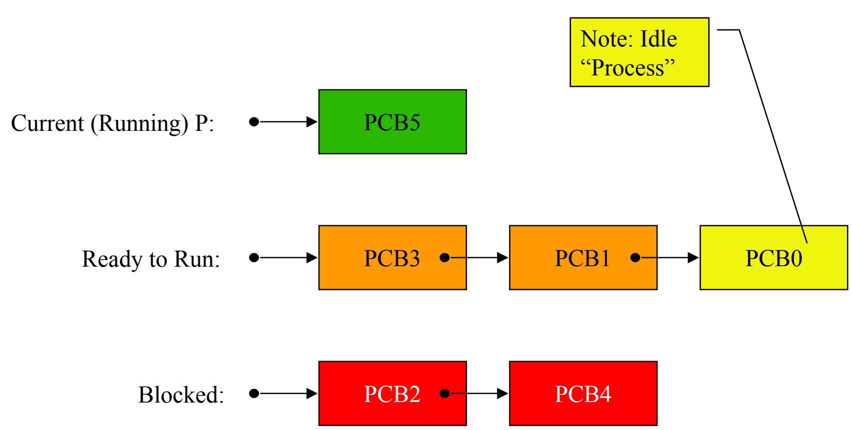
Process Representation

- OS keeps *Process Control Block (PCB)* for each process
- Contains management information
  - Location of P in memory
  - P's state
  - P's owner (user)
  - etc.



1B11-2 1999 Slide 9

Process Representation - 2



1B11-2 1999 Slide 10

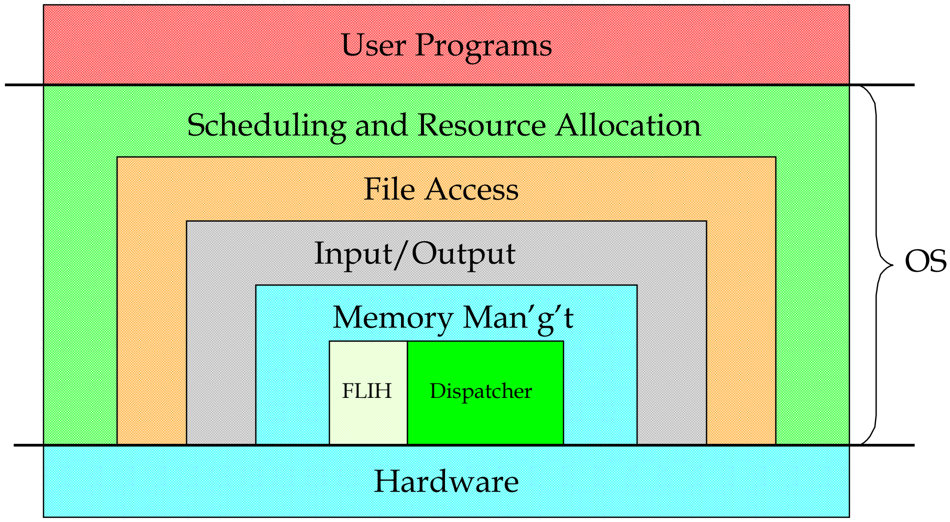
Process State Transitions - 2

Running-> Ready	- Time slot expires - Scheduler determines time slot
Ready->Running	- Current P-> Blocked or *pre-empted* - Highest priority runnable-> running - Null/Idle process if Q empty
Running->Blocked	- P makes *synchonous I/O request* - or waits for earlier *asynchronous I/O request*
Blocked->Ready	- I/O or events complete

Process State Transitions - 3

- Null/Idle Process
  - May do something useful: machine diagnostics, compute prime numbers, etc
  - or just wait in kernel loop for event (I/O)
- Scheduling vs Dispatching
  - Dispatcher just takes next P from Q and sets it running
  - Scheduler decides on ordering of Ps on Q on basis of policy, eg. shortest job, premium payment, real-time activity, etc.

Operating System Model



1B11-2 1999 Slide 13

Context...

- Processes are defined by their code, data, and state
- What does the state consist of:
  - Program Counter
  - Stack Pointer
  - Register Contents
  - Page/MMU register Contents
  - Anything else?
- How do we suspend one program, to start another (i.e. give the illusion of concurrency)?

1B11-2 1999 Slide 14

Process Control Block

- Contains state information such as:

o	Process State	Ready, etc
o	Process ID	to relate I/O, events inter-process communications (IPC), etc
o	Priority	or time slot, time to run etc.
o	Memory pointers	location of process code and data
o	Resources Allocated	terminals, devices
o	Register Save Area	processor registers, stack pointer etc. User ID
o	Owner	Parent process ID
o	Parent	

1B11-2 1999 Slide 15

Process Control Block - 2

- When P *swapped out*:
  - Not all of PCB needs to be retained in kernel
  - Need to also store swap device address of Process image in PCB

1B11-2 1999 Slide 16

Cost of Context Switch?

- A context switch entails at least 1 trap (call to scheduler or I/O or clock interrupt)
- Then it involves a ***full save of registers and page table***
- What if we knew we were only going to do a bit of work? Couldn't we do something quicker?
- Yes: threads…...

1B11-2 1999 Slide 17

Threads - bare processes...

- Threads are also known as ***"lightweight processes"***
- Assume we have a "safe" language, then we can have "concurrency" and scheduler, but avoid saving all registers and so on - let threads run in ***same virtual address space and so on***
- All the same except that scheduler is part of library not a system call (cheaper) and much less state saved and restored in a context switch (minimum is stack, register and PC)

1B11-2 1999 Slide 18

Scheduling Algorithms

- What is the order in which a scheduler chooses the next process when another one suspends or stops?
  - Random, round robin (fair) priority, mix
- Can try to balance between I/O and CPU intensive based on past behaviour
- Can have a real-time requirement (e.g. must respond before the reactor blows up).
- Complex- especially in newer multiprocessor or distributed systems!

1B11-2 1999 Slide 19

Communicating Processes

- Normally in a time-sharing multi-tasking system processes are unrelated. But, sometimes…we want them to communicate with each other.
- Communicating processes in different machines or address spaces use messages and protocols.
- Communicating processes in the same address space can share memory, and therefore can communicate by sharing variables (memory)

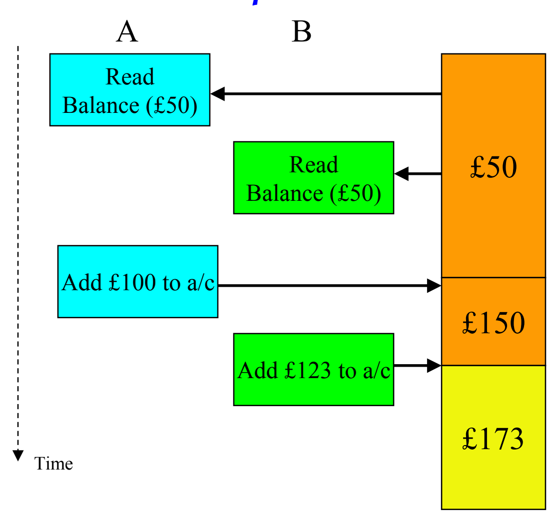
1B11-2 1999 Slide 20

Shared Variable Comms

- There are *interesting* problems with concurrent access to shared variables:
  - Run process A, which reads bank account v
  - Switch to process B, which reads same value
  - Switch to process A which adds £100 to it and writes it
  - Switch to process B which adds £100 to it and writes it
  - Result is wrong (only 1 update had an effect)
  - ***"Lost update"*** problem

1B11-2 1999 Slide 21

Lost Update Problem



- Result should be £273
- One update lost because balance read a second time before first update complete
- Need to ***bind*** two steps of update together
- A could ***lock*** data until it has finished with it

1B11-2 1999 Slide 22

Shared Variable Comms - 2

- Solutions include:
  - Locks
  - Queues
  - Monitors - lots of others - will be covered later in course - a difficult subject

1B11-2 1999 Slide 23

Problems with message passing

- Concurrent systems that pass messages or share variables can ***deadlock***
  - Process a is waiting for process c
  - Process c is waiting for b
  - Process b is waiting for a
  - (consider problems that can occur at a road junction when each car blocks the exit of another)
- or ***livelock*** or be unfair and so on

1B11-2 1999 Slide 24

Summary

- Process - running program
- Scheduler picks between processes to run
- Scheduler saves and restores contexts in process table
- Virtual Machine hides processes from each other
- Shared memory and messages can allow communication between processes
- Concurrency causes problems

Further Reading

- Ritchie C, "Operating Systems inc. UNIX and Windows", 3rd ed, 1997, Letts, 1 85805 302 1, £13.95
- and many more