Processes Threads and Synchronisation

Comp 305 Lecture 2

What's New

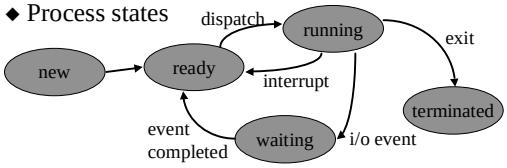
- Homework: Chap 4 prob 2, 4, 5, 7

Today's Lecture

- Processes
- Threads
- Concurrency
- Synchronisation
- Chapters sequence 4, 6, 7 and 5.

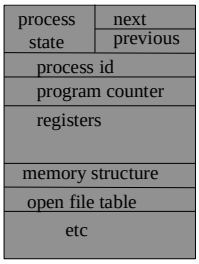
What is a Process?

- Program in execution
- Several processes may execute the same program
- Process states

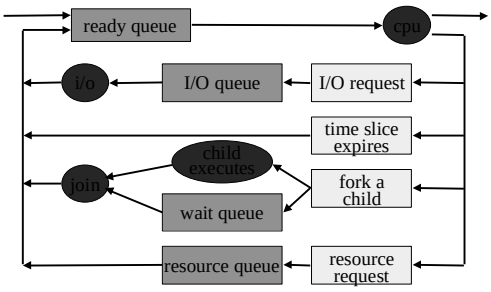


Process Control Block

- Representation of a process
- Used to save and restore state
- Exact contents are system dependent.

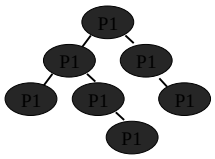


Process Queues



Operations on Processes

- Process creation (fork)
  - Parent/child relationship
  - Wait or continue in parallel
  - Child is copy of parent or has own address space
- Process termination
  - Exit or abort or kill
- Suspend
  - Internal (wait) or

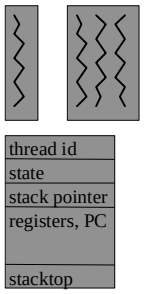


Context Switch

- Switching from one process to another
- Often tens of microseconds
- Save executing process in its PCB
  - Registers, program counter, state, ...
- Load new process from its PCB

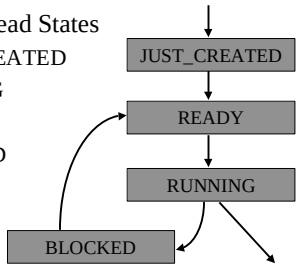
Threads

- A thread is an execution sequence
- Each process has at least one thread
- Each threads has its own control block
- Shares process state:
  - memory, open files, etc



Thread States

- Nachos Thread States
  - JUST_CREATED
  - RUNNING
  - READY
  - BLOCKED

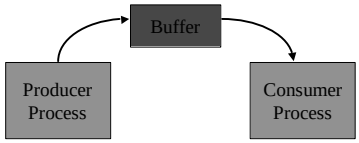


Thread Operations

- Fork
  - Creates a new thread sharing global structures
- Yield
  - Yields processor, enters ready queue
- Sleep
  - Yields processor, enters a wait queue
- Finish

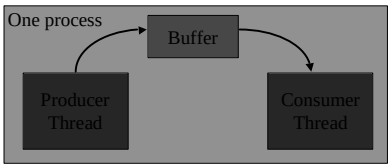
A Buffering Problem

- Two Process Solution



A Buffering Problem

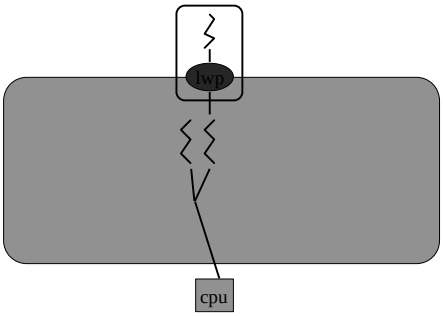
- Two Thread Solution



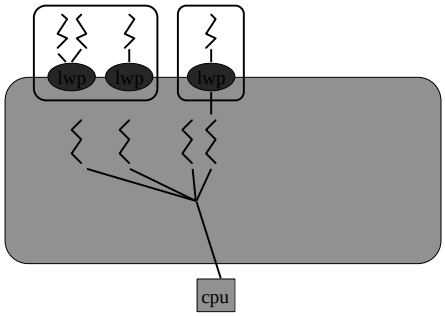
Types of Threads

- Kernal Threads
  - In the kernal only, no impact on application
- User Threads
  - Scheduled within user process
  - Not seen by kernal
- Lightweight Process
  - Seen by user; schedulded by kernal

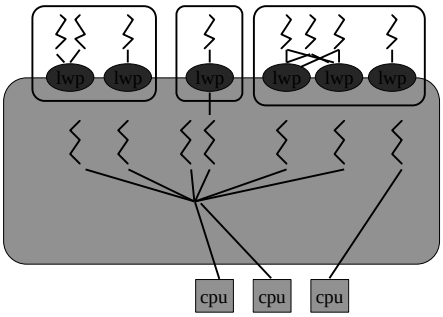
Solaris 2 Threads



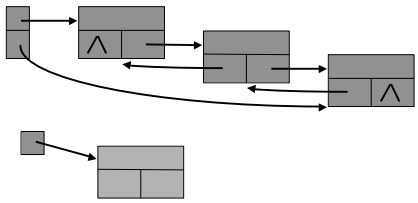
Solaris 2 Threads



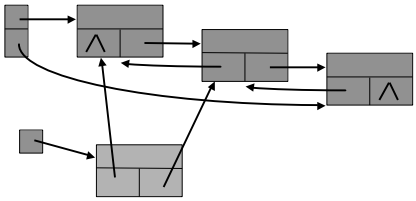
Solaris 2 Threads



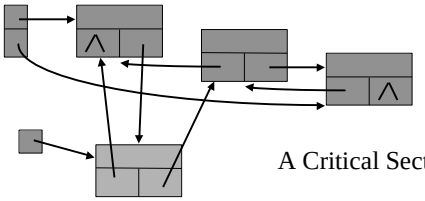
The Concurrency Problem



The Concurrency Problem



The Concurrency Problem



A Critical Section!

Critical Sections

repeat

entry section ⇐

critical section

exit section ⇐

remainder section

until *false*