Lecture 2 - Processes

Karl R. Wilcox
Karl@cs.rhul.ac.uk

Royal Holloway University of London

Overview

- In this class we will discuss
  - The process model
  - Race Conditions
  - Three unsatisfactory attempts to avoid race conditions
  - Process Scheduling
    - Round Robin Scheduling
    - Priority Scheduling
    - Multiple Queues
    - Shortest Job First
    - Two-level Scheduling

Royal Holloway University of London

Operating System Tasks

- **Processes**
  - Run them smoothly, assign CPU time
- **Memory**
  - Organise main memory, allocation
- **Files**
  - Allocation of files stored on disk, security
- **Input / Output management**

What Is a Process?

- A process is a program in execution
- A process consists of:
  - An executable program
  - Data
  - program counter
  - stack pointer
  - some registers
  - Other information
- If a process is stopped / interrupted, to be able to proceed with the process one has to store all the information about it
  - Often stored in a Process Table
- What is the difference between a program and a process?
- Why does the operating system interrupt a running process to give CPU time to another?
- Software interrupts: Signals

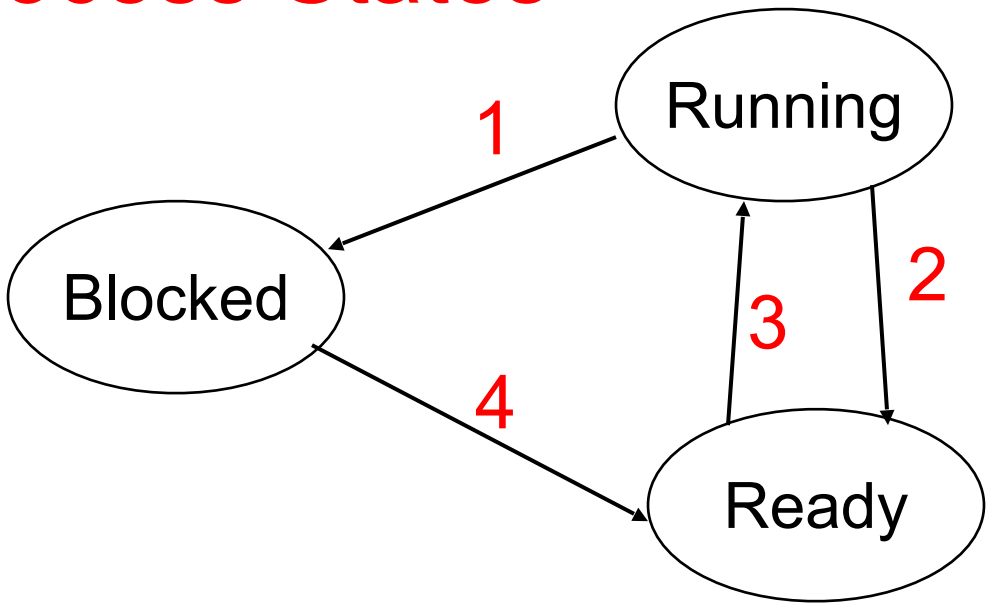
Royal Holloway University of London

The Process Model

- Pseudo parallelism vs. true hardware parallelism
- The CPU assigns units of time to the processes running
- The processes shouldn't have built in time assumptions
- What processes have built in time assumptions?

Royal Holloway University of London

Process States



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

Royal Holloway University of London

Race Conditions

- Processes often share files or resources
- Example Race condition – Printer spool slots
- How can we avoid race conditions?
- Critical sections
  - No two processes may be simultaneously inside their critical sections
  - No assumptions may be made about speeds or the number of CPUs
  - No process running outside its critical section may block other processes
  - No process should have to wait forever to enter its critical section

Royal Holloway University of London

Attempts to Avoid Race Conditions

- Disabling Interrupts
  - too much power for user processes
- Lock Variables
  - same problem as in example
- Strict Alternation
  - busy waiting

Royal Holloway University of London

Scheduling

- When more than one process is runnable the operating system must decide which one to run
- The part of the operating system concerned with this decision is called the **scheduler**
- the algorithm it uses is called the **scheduling algorithm**
- Complications
  - Every process is unique and unpredictable
  - Clock interrupts
  - pre-emptive scheduling or run to completion?

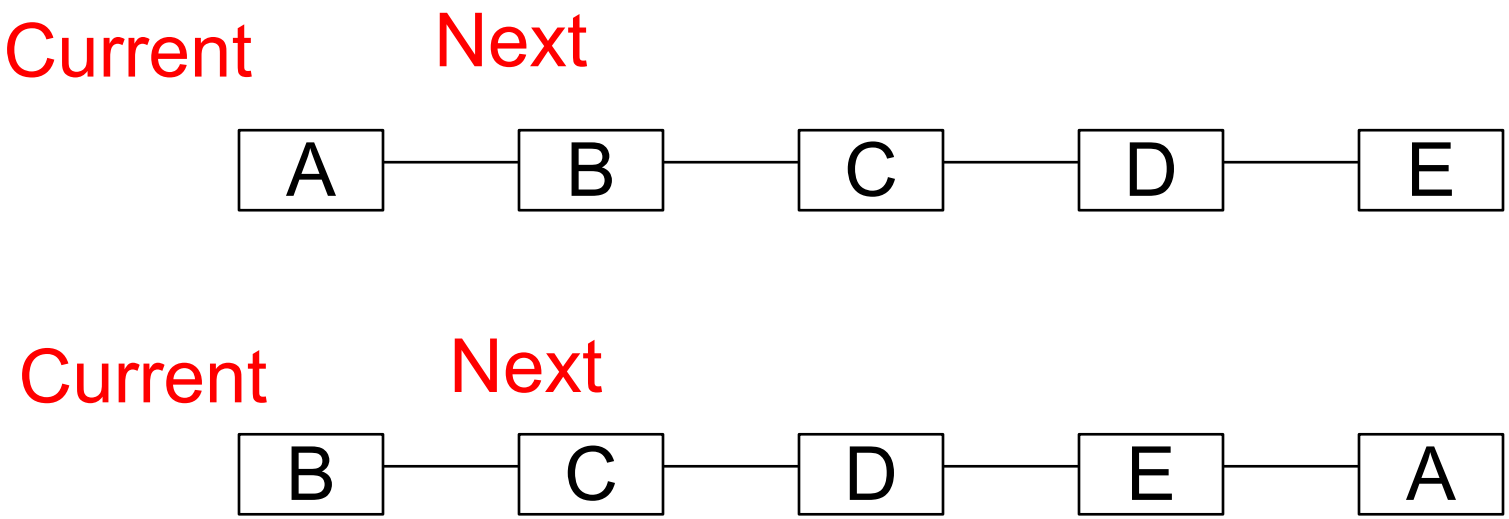
Royal Holloway University of London

Aims of the Scheduling Algorithm

- Fairness
  - make sure each process gets its fair share of the CPU
- Efficiency
  - keep the CPU busy 100 percent of the time
- Response time
  - minimise response time for interactive users
- Turnaround
  - minimise the time batch users must wait for output
- Throughput
  - maximise the number of jobs processed per hour

Round Robin Scheduling

- Each process is assigned a time interval called its quantum which it is allowed to run
- If the process is still running at the end of the quantum the CPU is given to another process



Round Robin Quantum Size

- Interesting question: Length of the quantum?
- Switching from one process to the next one *(process switch)* requires a certain amount of time
- Suppose switch takes 5 msec
- What %-age of time is used in process switches if:
  - quantum set to 20 msec
  - quantum set to 100 msec

Royal Holloway University of London

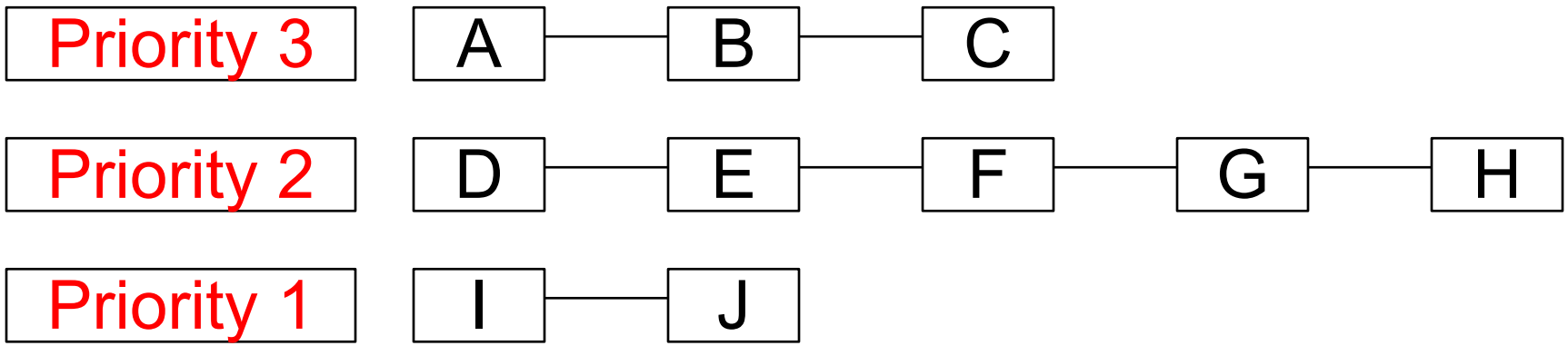
Priority Scheduling

- The processes are not equally important some will get higher priority than the other
- But high priority processes should not be run forever! Therefore decrease the priority of the currently running process *after some time*
- E.g. decrease the priority at each clock tick. If the priority dropped below that of the next highest process, then do a process switch
- Priorities can be assigned to processes statically or dynamically

Royal Holloway University of London

Priority Classes

- It is convenient to group processes into priority classes
- Among the classes priority scheduling but -
- within each class Round Robin Scheduling



Multiple Queues

- Depending on the time a process switch takes it could be more efficient to give processes big amounts of CPU time once in a while instead of small quanta frequently
- On the other hand, giving all processes a large quantum would mean poor response time
- Solution - Priority classes where in the highest priority class, each process gets one quantum, in the second class processes run for 2 quanta etc

Multi-queue Example

- Suppose a process needs to compute continuously for 7 quanta
- Using multiple queues this process would only need 3 swaps (1 -> 2 -> 4)
- With Round Robin Scheduling it would have needed 7 swaps
- Suppose a process needs the CPU continuously for 100 quanta
  - How many swaps do we need for it with multiple queues?
  - And with Round Robin?

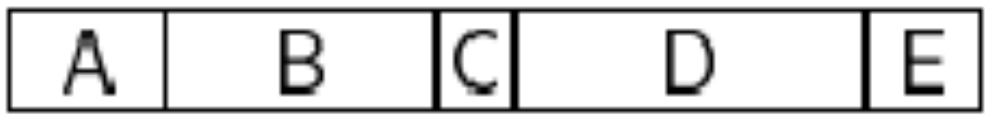
Royal Holloway University of London

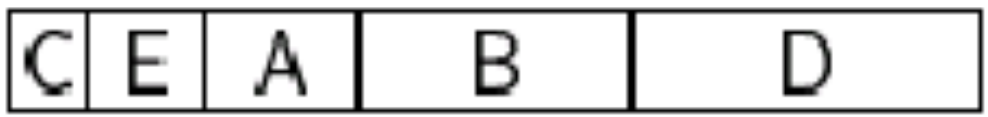
Shortest Job First

- If we have more information about processes
  - e.g. the run times of the processes
- we can apply a more specialised algorithm
- Suppose we know the run times of the incoming jobs
- several of (equally important) jobs are waiting for the CPU
- We should run the Shortest Job First

Shortest Job First example

- processes A through E are waiting to be run
- Their run times are
  - A - 4 minutes
  - B - 7minutes
  - C - 2minutes
  - D - 9 minutes
  - E - 3 minutes





Features of Shortest Job First

- Always minimum average time. It would be nice if it could be used for interactive processes as well
- Estimates could be made based on past behaviour Then run the process with the shortest estimated running time
- Suppose T_0 estimated time for a terminal
- T_1 length of next run
- $T = aT_0 + (1 - a)T_1$ new estimate for the terminal
- The parameter *a* regulates the memory of the estimation
  - Aging
- Problem with processes not arriving simultaneously

Royal Holloway University of London

About Scheduling

- Guaranteed scheduling
- Policy versus Mechanism

Royal Holloway University of London

Two Level Scheduling

- In the situation where the main memory is insufficient some of the runnable processes will be kept on the disk
- Two level scheduler
  - lower level scheduler only choosing processes which are in main memory
  - periodically the higher level scheduler removes processes that have been in memory long enough and loads processes that have been on the disk too long