Operating Systems Workbench V2.1
Scheduling (Introductory) Activities and Experiments

Richard Anthony January 2005

This laboratory sheet accompanies the '***Scheduling Algorithms – Introductory***' application within the Operating Systems Workbench.

1. Prerequisite knowledge

You should have a basic understanding of the following concepts: Operating System, Processor, Process.

2. Introduction

This simulation has been designed to introduce you to the fundamentals of scheduling whilst keeping the complexity low. The simulation shows scheduling behaviour in terms of the movement of processes between the RUN, READY and BLOCKED states.

You can investigate the way in which three different scheduling algorithms work with either one or two processes. The processes can be configured to be one of: CPU intensive (does no Input / Output); Balanced (does some Input / Output); or IO intensive (does lots of Input / Output) – you will be able to investigate how the tasks' characteristics affect their behaviour in the system.

Figure 1 shows the Scheduling (introductory) interface during a simulation of Round Robin Scheduling with two processes.

The display is divided into a number of sections. Each section is briefly explained:

- **Process Configuration** – this enables configuration of the number of processes and the type of processes used in the simulation. In the example shown both processes are enabled. Process 1 is configured to be CPU intensive (that is, it does no I/O) whilst Process 2 is configured to be I/O intensive (that is, it does lots of I/O). Both processes have been configured to have a runtime of 30 milliseconds (this is, the amount of CPU processing time required is 30 milliseconds, the actual time they will spend in the system can be more than this).
- **System Configuration** – This tells you the configuration settings for the scheduler itself. These settings are fixed for the introductory Scheduling application.
- **Scheduler Configuration** – this enables you to select one of three different scheduling algorithms.
- **Animation Control** – this enables you to select the speed of the simulation.
- **System Statistics** – this provides a real-time display of the elapsed time since the start of a simulation.
- **Runtime Statistics** – this provides a real-time display of various statistics for each process during the simulation.
- **Runtime State Display** – This shows real-time scheduling behaviour in terms of the movement of the processes between the RUN, READY and BLOCKED states. A process enters the COMPLETED state when it has satisfied its processing requirement (as defined by its runtime value).
- **Free Run button** – this causes the simulation to run at the selected speed until all processes have completed, or the simulation is paused by pressing the Pause / Single-Step button.
- **Single-Step button** – this causes the simulation to run until the next process state-change and then stop. During free-run, this button's label changes to 'pause' and if pressed will stop the simulation at the next process state-change.
- **Reset Simulation button** – this button can be pressed at any time during a simulation. The process states and statistics are all reset but the simulation configuration settings are preserved. This is to permit unlimited repeat runs of the same simulation, without having to wait for completion.
- **Done button** – this button exits the application immediately without preserving statistics or configuration settings. Control is returned to the top-level menu.

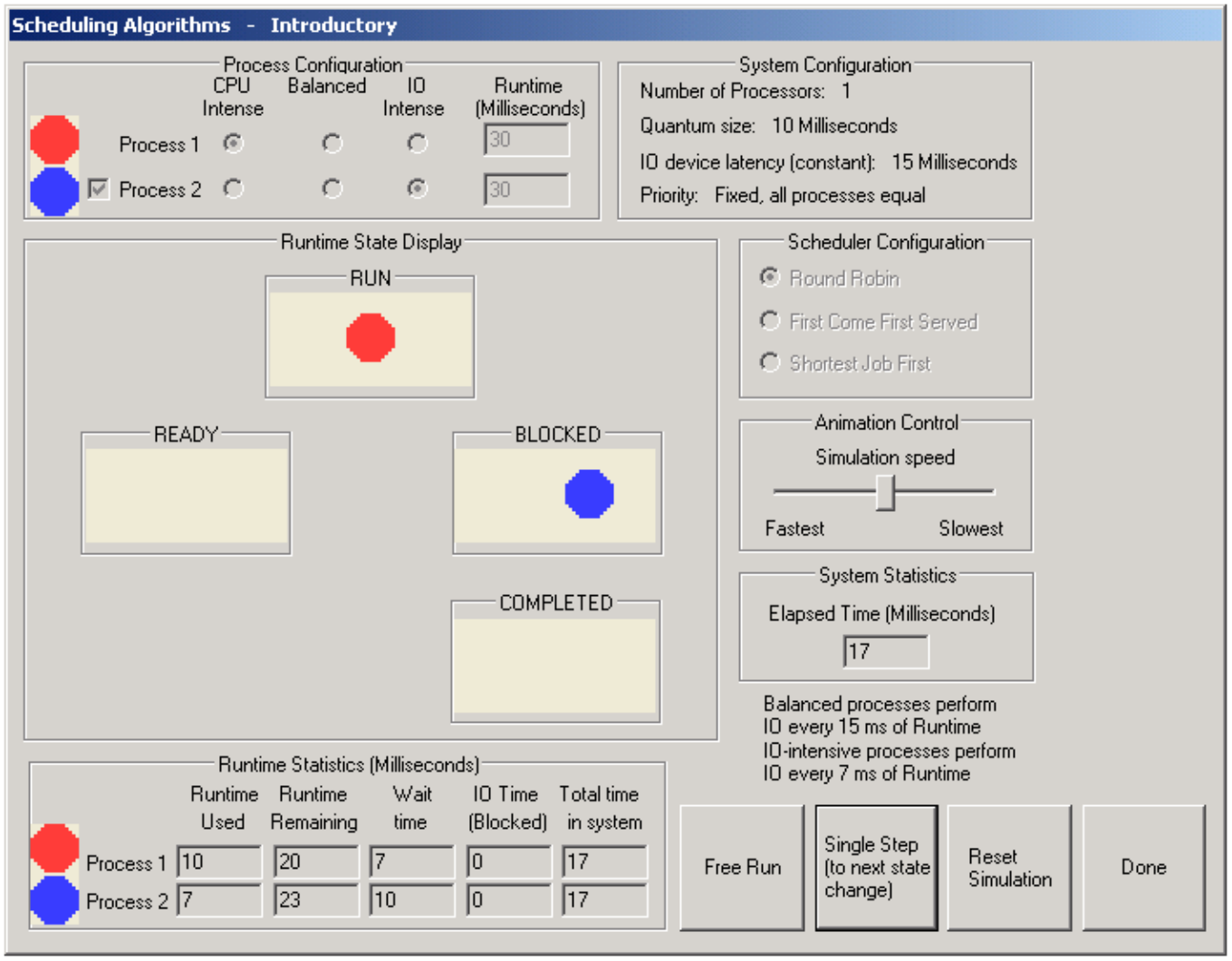

Figure 1. The Scheduling (introductory) interface.

The following 'lab activities' sections describe specific experiments and investigations to help you maximise the benefit of the software.

Please take care to read the instructions carefully for each step and try to follow instruction sequences carefully. Try to predict the outcome of experiments in advance if possible. If the outcome is not as expected try to determine why not. You can repeat experiments as often as required and can work at your own pace. Try repeating experiments with slight changes in the parameters – sometimes just changing one parameter slightly can lead to big differences in the results and can shed light on the relative importance of a particular aspect of the simulation.

For each activity the configuration settings are explained. In each case it is assumed that you have already started the 'Introductory Scheduling Algorithms' simulation application. To do this:
1. Start the **Operating Systems Workbench**.
2. From the main menu bar, select **Scheduling Algorithms**.
3. From the drop-down menu, select **Introductory**.

Lab Activity: Scheduling: Introductory: FCFS 1
Introduction to the FIRST COME FIRST SERVED (FCFS) Scheduling Algorithm

1. Configure the simulation as follows:
   Process 1: Type = CPU Intense, Runtime = 30 milliseconds (default)
   Process 2: Type = CPU Intense, Runtime = 30 milliseconds (default)
   Scheduler Configuration = First Come First Served

2. Press the **Start Configuration** button and pay attention to the sequence of states through which the processes pass.

3. When the process has completed, note the **System Statistics** and **Runtime Statistics**.

Questions
Q1. What is the total execution time of the processes (how long between starting and finishing)?

Q2. Do you think the FCFS scheduling algorithm is fair? Does fairness actually matter if all processes run for a very short time only? Try setting the process 1 Runtime to 300 milliseconds and repeat the experiment. Look at the wait time for each process – does this seem reasonable?

Lab Activity: Scheduling: Introductory: SJF 1
Introduction to the SHORTSET JOB FIRST (SJF) Scheduling Algorithm

1. Configure the simulation as follows:
   Process 1: Type = CPU Intense, Runtime = 300 milliseconds
   Process 2: Type = CPU Intense, Runtime = 30 milliseconds
   Scheduler Configuration = First Come First Served

2. Note the 'wait time' and 'total time in system' accumulated by each process.

3. Configure the simulation as follows:
   Process 1: Type = CPU Intense, Runtime = 30 milliseconds
   Process 2: Type = CPU Intense, Runtime = 300 milliseconds
   Scheduler Configuration = First Come First Served

4. Note the 'wait time' and 'total time in system' accumulated by each process.

Q1. From your observations in steps 2 and 4, what can be said about the efficiency of the FCFS scheduling algorithm (does it depend on luck?).

Q2. Can you think of a way to improve the behaviour of the FCFS scheduling algorithm?

5. Repeat steps 1- 4 but use the Shortest Job First scheduling algorithm in place of the FCFS algorithm. Once again pay attention to the sequence of states through which the processes pass, and the process statistics after each simulation run.

Q3. What is the fundamental difference between the FCFS and SJF scheduling algorithms? Which is superior? (such a judgment requires justification, if you think one of the algorithms is superior you must state why this is so, and under what circumstances).

Lab Activity: Scheduling: Introductory: FCFS 2
Types of process – as defined by their behaviour

1. Configure the simulation as follows:
    Process 1: Type = CPU Intense, Runtime = 30 milliseconds
    Process 2: Not selected
    Scheduler Configuration = First Come First Served

2. Press the **Start Configuration** button and pay attention to the sequence of states through which the process passes.

3. When the process has completed, note the **System Statistics** and **Runtime Statistics**.

The 'Runtime Used' is the amount of time the process actually used the CPU.
The 'Total Time in System' is the time between starting the process and its completion.

Questions
Q1. Is the Runtime Used the same as the Total Time in System? If not, explain why?

IO means Input / Output. IO devices (printers, disk drives, network interfaces etc.) tend to operate much slower than the CPU. Thus, there is considerable delay to the running of a process when it performs IO. When a process performs IO we say that the process is 'Blocked' whilst waiting for the IO device to do its work. The more often a process performs IO, the more time it will spend 'Blocked'.

A process that never performs IO will use the CPU to the fullest extent that the scheduling algorithm allows. We say that this type of process is CPU intensive (or Compute intensive). A process that performs a lot of IO is said to be IO intensive.

4. Repeat the experiment, using the modified configuration:
    Process 1: Type = Balanced, Runtime = 30 milliseconds
    Process 2: Not selected
    Scheduler Configuration = First Come First Served

Q2. Is the Runtime Used the same as the Total Time in System? If not, explain why?

5. Repeat the experiment, using the modified configuration:
    Process 1: Type = IO Intense, Runtime = 30 milliseconds
    Process 2: Not selected
    Scheduler Configuration = First Come First Served

Q3. Is the Runtime Used the same as the Total Time in System? If not, explain why?

Q4. Can you see a pattern to the relationship between the Runtime Used and the Total Time in System for the three types of process?

Lab Activity: Scheduling: Introductory: FCFS 3
Introduction to the problems associated with non-preemptive scheduling (part 1)

1. Configure the simulation as follows:
   Process 1: Type = IO Intense, Runtime = 30 milliseconds
   Process 2: Type = CPU Intense, Runtime = 130 milliseconds
   Scheduler Configuration = First Come First Served

2. Press the **Start Configuration** button and pay attention to the sequence of states through which the processes pass.

3. When the process has completed, note the **System Statistics** and **Runtime Statistics**.

Questions
Q1. What is the total execution time of the processes (how long between starting and finishing)?

Q2. Did you notice if the CPU was always busy? (you can run the simulation again to check)? Does it matter if the CPU is not busy?

Q3. Look at the wait time for each process - what has caused this?

Scheduling that releases the CPU to another process when the current process performs IO operations is called preemptive scheduling. Non-preemptive scheduling does not release the CPU until the current process has completed.

Q4. Do you think FCFS algorithm is preemptive or non-preemptive? How can you tell?

When the CPU is kept busy we say that the CPU is used efficiently. When the CPU is not busy and processes are waiting we say that the scheduling is inefficient.

Q5. What is the fundamental effect of having inefficient scheduling?

Q6. Which do you think is the most efficient: preemptive scheduling or non-preemptive scheduling?

Lab Activity: Scheduling: Introductory: SJF 2
Introduction to the problems associated with non-preemptive scheduling (part 2)

1. Configure the simulation as follows:
   Process 1: Type = IO Intense, Runtime = 30 milliseconds
   Process 2: Type = CPU Intense, Runtime = 130 milliseconds
   Scheduler Configuration = Shortest Job First

2. Press the **Start Configuration** button and pay attention to the sequence of states through which the processes pass.

3. When the process has completed, note the **System Statistics** and **Runtime Statistics**.

Questions
Q1. What is the total execution time of the processes (how long between starting and finishing)?

Q2. Did you notice if the CPU was always busy? (you can run the simulation again to check)? Does it matter if the CPU is not busy?

Q3. Look at the wait time for each process - what has caused this?

Q4. Do you think SJF algorithm is preemptive or non-preemptive? How can you tell?

Lab Activity: Scheduling: Introductory: ROUND ROBIN (RR) 1
Introduction to the problems associated with non-preemptive scheduling (part 3)

1. Configure the simulation as follows:
   Process 1: Type = IO Intense, Runtime = 30 milliseconds
   Process 2: Type = CPU Intense, Runtime = 130 milliseconds
   Scheduler Configuration = Round Robin

2. Press the **Start Configuration** button and pay attention to the sequence of states through which the processes pass.

3. When the process has completed, note the **System Statistics** and **Runtime Statistics**.

Questions
Q1. What is the total execution time of the processes (how long between starting and finishing)?

Q2. Did you notice if the CPU was always busy? (you can run the simulation again to check)? Does it matter if the CPU is not busy?

Q3. Look at the wait time for each process - what has caused this?

Q4. Do you think RR algorithm is preemptive or non-preemptive? How can you tell?

Q5. How does the 'Total Time in System' differ, between the three scheduling algorithms (FCFS, SJF and RR)? Can you explain why this is so?

Lab Activity: Scheduling: Introductory: RR 2
Sharing the CPU

1. Configure the simulation as follows:
   Process 1: Type = CPU Intense, Runtime = 30 milliseconds
   Process 2: Type = CPU Intense, Runtime = 30 milliseconds
   Scheduler Configuration = Round Robin

2. Press the **Start Configuration** button and pay attention to the sequence of states through which the processes pass.

3. When the process has completed, note the **System Statistics** and **Runtime Statistics**.

Questions
Q1. For each process, is the Runtime Used the same as the Total Time in System? If not, explain why?

Q2. Is the total Runtime Used (for **both** processes), the same as the Total Time in System for **each** process? Explain individually for each process?

4. Repeat the experiment, using the modified configuration:
   Process 1: Type = CPU Intense, Runtime = 60 milliseconds
   Process 2: Type = CPU Intense, Runtime = 60 milliseconds
   Scheduler Configuration = Round Robin

Q3. For each process, is the Runtime Used the same as the Total Time in System? If not, explain why?

Q4. Is the total Runtime Used (for **both** processes), the same as the Total Time in System for **each** process? Explain individually for each process? Can you see a pattern forming?

Q5. Based on your observations, do you think that the RR scheduling algorithm is fair to all processes? How does the fairness of RR compare to that of FCFS? Or SJF?

Lab Activity: Scheduling: Introductory: RR 3
Familiarisation with the Round Robin Scheduling Algorithm (part 1)

1. Configure the simulation as follows:
   Process 1: Type = balanced, Runtime = 50 milliseconds (default)
   Process 2: Not selected (default)
   Scheduler Configuration = Round Robin

2. Press the **Start Configuration** button and pay attention to the sequence of states through which the process passes.

3. When the process has completed, note the **System Statistics** and **Runtime Statistics**.

Questions
Q1. What is the total execution time of the process (how long between starting and finishing)?

Q2. How much of this time is due to actual processing (using the CPU)?

Q3. How much of this time is due to waiting for I/O devices?

Q4. If there are no other processes present, how much CPU time is unused while the process is executing?

Lab Activity: Scheduling: Introductory: RR 4
Familiarisation with the Round Robin Scheduling Algorithm (part 2)

1. Configure the simulation as follows:
   Process 1: Type = balanced, Runtime = 50 milliseconds
   Process 2: Type = balanced, Runtime = 50 milliseconds
   Scheduler Configuration = Round Robin

2. Press the **Start Configuration** button and pay attention to the sequence of states through which the process passes.

3. When the process has completed, note the **System Statistics** and **Runtime Statistics**.

Questions
Q1. What is the total execution time of each process (how long between starting and finishing)?

Q2. Why are the execution times different for two identical processes?

Q3. What is the overall time taken to execute both the processes?

Q4. Why is the time taken to execute two processes not double the time taken to execute one of them? (refer to the results of **Lab Sheet: Scheduling: Introductory: RR3**).

Lab Activity: Scheduling: Introductory: RR 5
Familiarisation with the Round Robin Scheduling Algorithm (part 3)

1. Configure to use one process.
Try each of the following cases:

i. Run a single CPU intensive process
ii. Run a single balanced process
iii. Run a single IO intensive process

In all cases set the Runtime to 50 milliseconds.

For each of the above cases note the differences in the way the process runs.

In each case - Observe the various states that each process is in prior to completion.
Collect the statistics shown at the end of the simulation for each case (i to iii)
Tabulate your results.

Q1. What are the differences in way the three types of processes run?

Q2. Which process runs the fastest? Which one runs the slowest? Explain why there is this difference.

Q3. Explain why the slowest process is so slow?

2. Configure to use two processes.
Try each of the following cases:

i. Run both as CPU intensive process
ii. Run both as balanced process
iii. Run both as IO intensive process

In all cases set the Runtime to 50 milliseconds.

In each case - Observe the various states that each process is in prior to completion.
Collect the statistics shown at the end of the simulation for each case (i to iii)
Tabulate your results.

Q4. Comment on the differences you can observe for the **total time** taken for the simulation

Q5. Using the various statistics collected, comment on the way that the processes ran. Were any blocked and unable to run? Why were they blocked?

Q6. Which simulation runs the fastest? Which one runs the slowest? Explain why there is this difference.

Lab Activity: Scheduling: Introductory: Scheduling Algorithms 1
Comparison of the different Scheduling Algorithms

1. Configure to use two processes (try various combinations of process type, running one as CPU intensive and the other as IO intensive, etc.).

2. For each different process configuration, repeat the simulation for each type of scheduling algorithm (First Come First Served, Shortest Job First and Round Robin).

3. Observe the effect of having different types of processes running at the same time.

4. Observe the effect of the different scheduling algorithms.

Q1. What difference does the 'process-mix' make to the overall scheduling behaviour and the total time taken for the simulation?

Q2. What type of process best shows up the differences between the scheduling algorithms? What type of process least shows up the differences? Why is this?