

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.

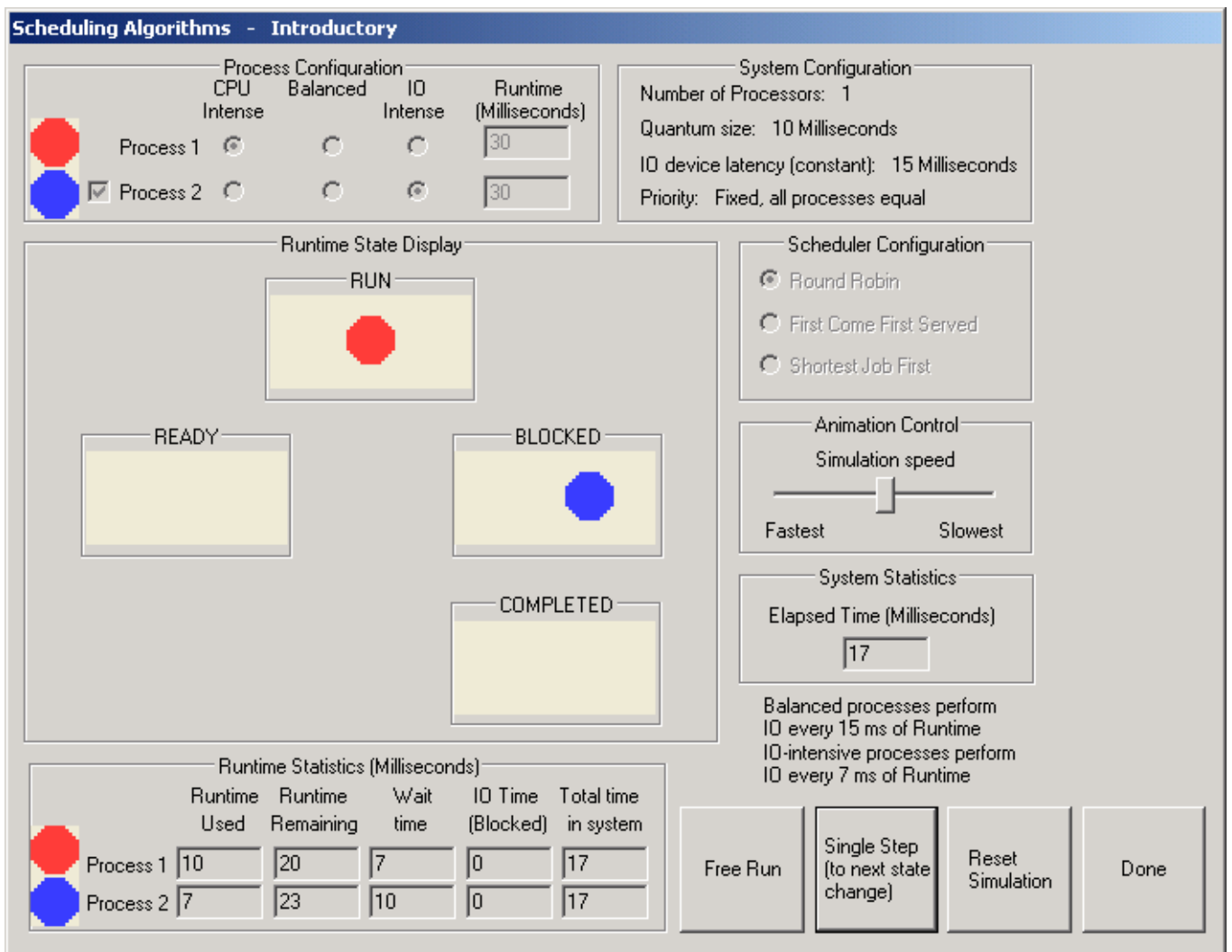


Figure 1. The Scheduling (introductory) interface.

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**.

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	30
Process 2	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	30

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☐ Round Robin

☒ First Come First Served

☐ Shortest Job First

Animation Control

Simulation speed

Fastest Slowest

System Statistics

Elapsed Time (Milliseconds)

60

Balanced processes perform IO every 15 ms of Runtime

IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	30	0	0	0	30
Process 2	30	0	30	0	60

Free Run

Pause

Reset Simulation

Done

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	300
Process 2	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	30

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☐ Round Robin
☒ First Come First Served
☐ Shortest Job First

Animation Control

Simulation speed

Fastest Slowest

System Statistics

Elapsed Time (Milliseconds)

330

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	300	0	0	0	300
Process 2	30	0	300	0	330

Free Run

Pause

Reset Simulation

Done

Questions

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

60 milliseconds

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?

The fairness of the FCFS scheduling algorithm is subjective and depends on the context of the system and the requirements of the process.

For a very short time, the impact of the scheduling algorithm may be minimal, so fairness actually doesn't matter and there may not be a significant difference in wait times.

When process 1 Runtime is 300 milliseconds, the wait time for subsequent processes in the queue may become noticeable and might not be considered 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

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	300
Process 2	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	30

System Configuration

Number of Processors: 1
Quantum size: 10 Milliseconds
IO device latency (constant): 15 Milliseconds
Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☐ Round Robin
☒ First Come First Served
☐ Shortest Job First

Animation Control

Simulation speed

Fastest ————— Slowest

System Statistics

Elapsed Time (Milliseconds)

330

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	300	0	0	0	300
Process 2	30	0	300	0	330

Free Run

Pause

Reset Simulation

Done

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

Process 1:

- **Wait time: 0 milliseconds**
- **Total time in system: 300 milliseconds**

Process 2:

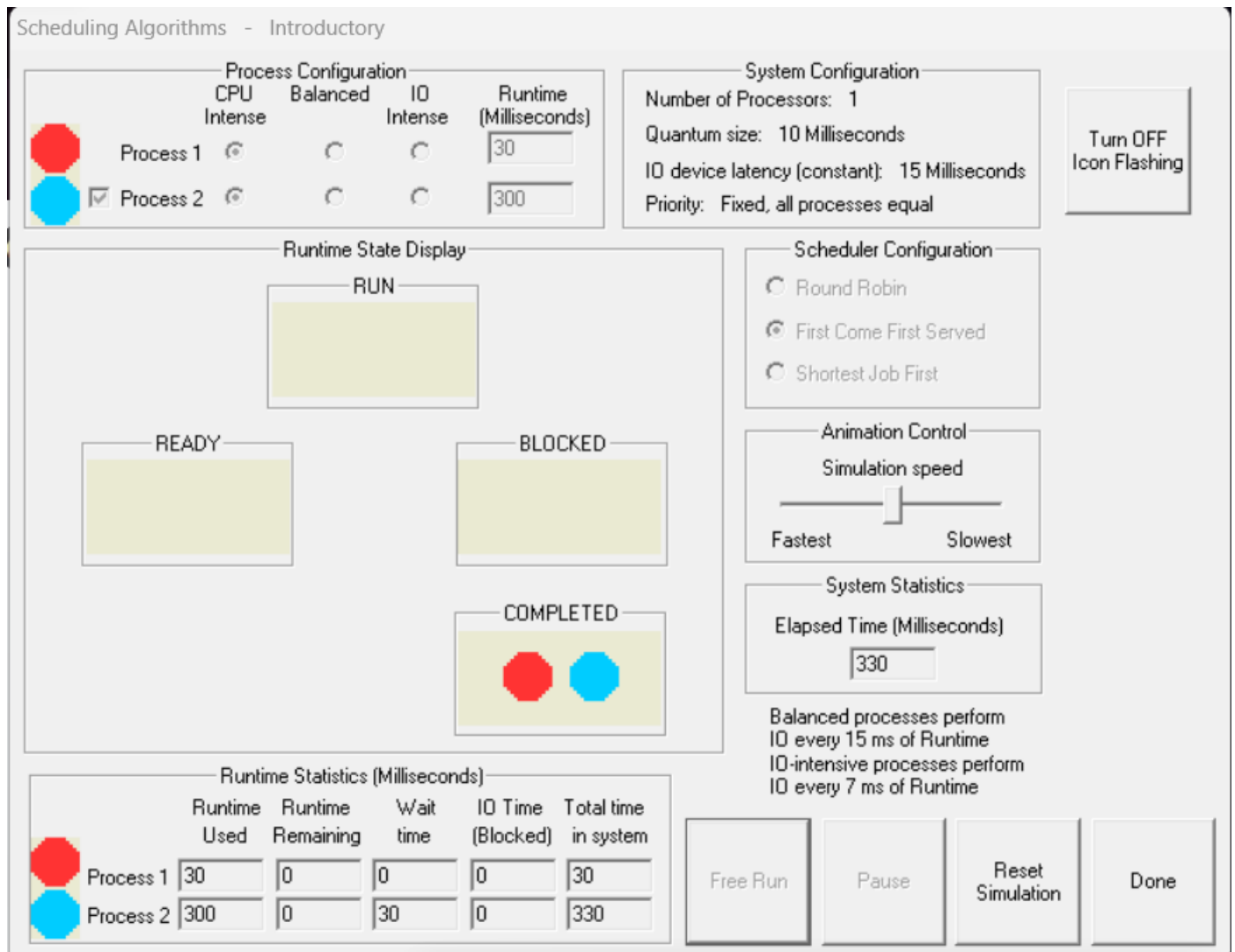
- **Wait time: 300 milliseconds**
- **Total time in system: 330 milliseconds**

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.

Process 1:

- **Wait time: 0 milliseconds**
- **Total time in system: 30 milliseconds**

Process 2:

- **Wait time: 30 milliseconds**
- **Total time in system: 330 milliseconds**

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?).

The efficiency of the FCFS scheduling algorithm is relatively straightforward, but it can be influenced by the order in which processes arrive. The concept of "luck" is more about the chance order in which processes arrive and whether that order aligns with the inherent characteristics of the workload.

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

We can use Shortest Job First (SJF) Scheduling for FCFS, combine FCFS with a SJF approach by sorting the processes in the ready queue based on their burst times. This way, shorter processes will get executed first, minimizing the convoy effect.

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.

Scheduling Algorithms - Introductory

Process 1

☒ CPU Intense
 ☐ Balanced
 ☐ IO Intense

Runtime (Milliseconds)

300

Process 2

☒ CPU Intense
 ☐ Balanced
 ☐ IO Intense

Runtime (Milliseconds)

30

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☐ Round Robin
 ☐ First Come First Served
 ☒ Shortest Job First

Animation Control

Simulation speed

Fastest

Slowest

System Statistics

Elapsed Time (Milliseconds)

330

Balanced processes perform IO every 15 ms of Runtime

IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	300	0	30	0	330
Process 2	30	0	0	0	30

Free Run

Pause

Reset Simulation

Done

Scheduling Algorithms - Introductory

Process 1

☒ CPU Intense
 ☐ Balanced
 ☐ IO Intense

Runtime (Milliseconds)

30

Process 2

☒ CPU Intense
 ☐ Balanced
 ☐ IO Intense

Runtime (Milliseconds)

300

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☐ Round Robin
 ☐ First Come First Served
 ☒ Shortest Job First

Animation Control

Simulation speed

Fastest

Slowest

System Statistics

Elapsed Time (Milliseconds)

330

Balanced processes perform IO every 15 ms of Runtime

IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	30	0	0	0	30
Process 2	300	0	30	0	330

Free Run

Pause

Reset Simulation

Done

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).

	FCFS	SJF
Advantage	Simple to understand and implement.	Generally provides a shorter average waiting time and turnaround time compared to FCFS.
Disadvantage	May lead to a phenomenon known as "convoy effect," where shorter processes get stuck behind longer ones, causing potential delays.	Requires knowledge of the burst time, which is often not known in advance.

=> **SJF** is superior in terms of minimizing waiting and turnaround times on average. However, its effectiveness relies on accurate predictions of process burst times, which might be challenging in dynamic environments.

Circumstances:

FCFS:

- Suitable for systems with a mix of short and long processes where simplicity is valued over optimal performance.
- When burst times are relatively equal.

SJF:

- Effective in scenarios where there is a good estimate of process burst times.
- Best suited for systems where minimizing average waiting time and turnaround time is a priority.

Conclusion:

There is no universally superior algorithm; the choice depends on the specific requirements of the system and the characteristics of the workload. In practice, other scheduling algorithms, such as priority scheduling or round-robin, might be considered to balance simplicity and performance.

Lab Activity: Scheduling: Introductory: FCFS 2

Types of process – as defined by their behavior

1. Configure the simulation as follows:

Process 1: Type = CPU Intense, Runtime = 30 milliseconds

Process 2: Not selected

Scheduler Configuration = First Come First Served

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	30
Process 2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	300

System Configuration

Number of Processors: 1
Quantum size: 10 Milliseconds
IO device latency (constant): 15 Milliseconds
Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☐ Round Robin
☒ First Come First Served
☐ Shortest Job First

Animation Control

Simulation speed

Fastest Slowest

System Statistics

Elapsed Time (Milliseconds)

30

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	30	0	0	0	30
Process 2	0	0	0	0	0

Free Run

Pause

Reset Simulation

Done

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?

Yes

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

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	30
Process 2	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	300

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☐ Round Robin

☒ First Come First Served

☐ Shortest Job First

Animation Control

Simulation speed

Fastest Slowest

System Statistics

Elapsed Time (Milliseconds)

45

Balanced processes perform IO every 15 ms of Runtime

IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	30	0	0	15	45
Process 2	0	0	0	0	0

Free Run

Pause

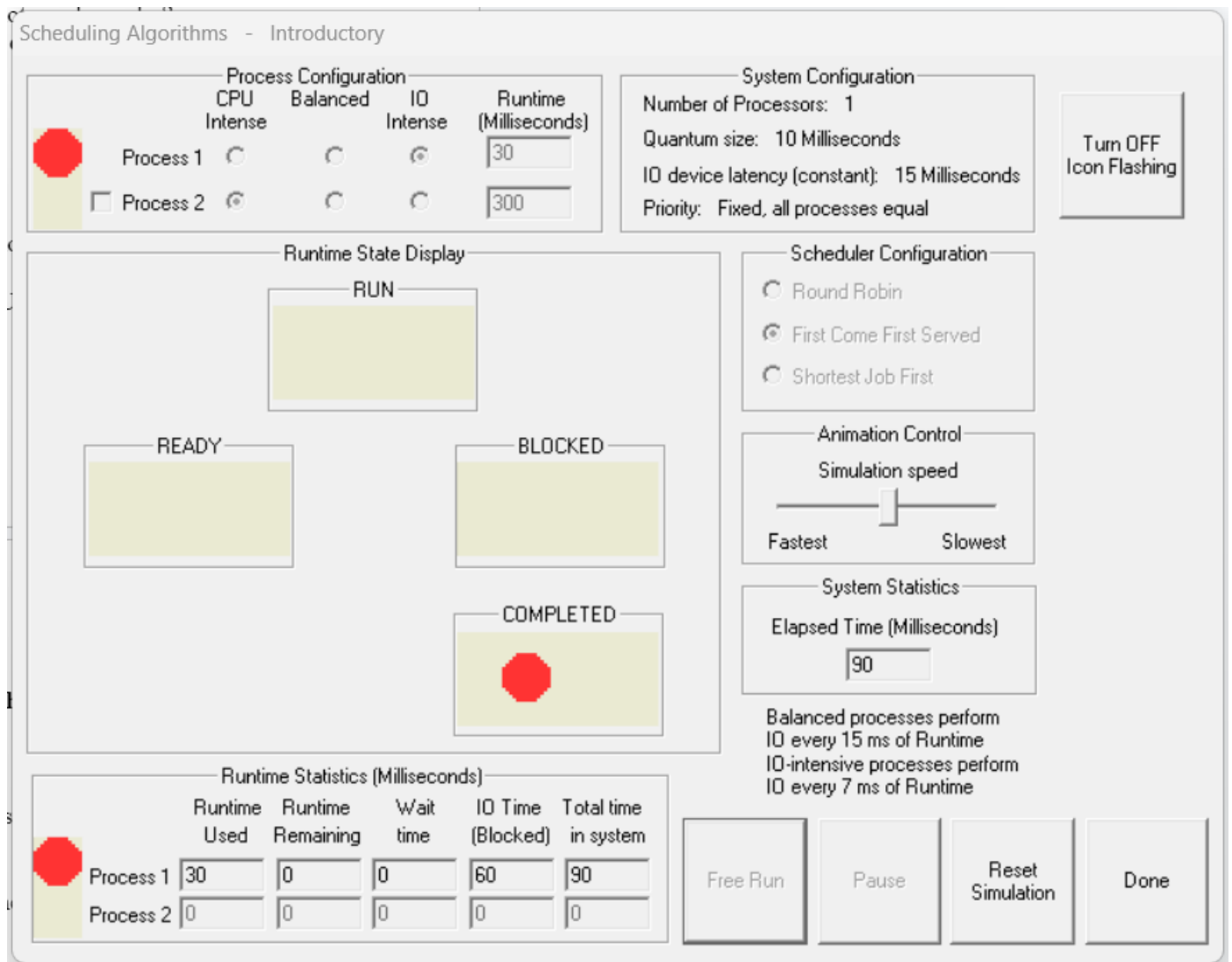
Reset Simulation

Done

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

No. Because it performed IO during the execution, which is 15 milliseconds.

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?

No. Because it performed 4 IOs during the execution, so it added up to 60 milliseconds IO time (Blocked)

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?

- **CPU Intense: Runtime and Total Time in System may be close when waiting time is minimal.**
- **Balanced: Total Time in System is influenced by both runtime and waiting time, with waiting time potentially impacting it more than CPU Intensive processes.**
- **IO Intense: Total Time in System is usually significantly higher than runtime due to substantial waiting time for I/O operations**

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**.

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	30
Process 2	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	130

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☐ Round Robin

☒ First Come First Served

☐ Shortest Job First

Animation Control

Simulation speed

Fastest ————— Slowest

System Statistics

Elapsed Time (Milliseconds)

220

Balanced processes perform IO every 15 ms of Runtime

IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	30	0	0	60	90
Process 2	130	0	90	0	220

Free Run

Pause

Reset Simulation

Done

Questions

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

220 milliseconds

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?

Yes, the CPU was always busy. When it isn't busy, the CPU might not be fully utilized during I/O operations, but it can still efficiently handle other tasks.

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

IO performance parallel with CPU Intense.

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 the FCFS algorithm is preemptive or non-preemptive? How can you tell?

FCFS is a non-preemptive scheduling algorithm because it doesn't preemptively interrupt the execution of a process; it allows the currently running process to complete its execution before moving on to the next one in the queue.

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?

- **Lower Throughput**
- **Increased Response Time**
- **Resource Underutilization**
- **Poor System Responsiveness**
- **Increased Latency**

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

The choice between preemptive and non-preemptive scheduling often depends on the specific use case. Real-time systems, where timely responses are critical, often benefit from preemptive scheduling. On the other hand, non-preemptive scheduling might be suitable for systems with predictable workloads and where minimizing context-switching overhead is a priority.

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**.

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	30
Process 2	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	30

System Configuration

Number of Processors: 1
Quantum size: 10 Milliseconds
IO device latency (constant): 15 Milliseconds
Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☐ Round Robin
☐ First Come First Served
☒ Shortest Job First

Animation Control

Simulation speed

Fastest ————— Slowest

System Statistics

Elapsed Time (Milliseconds)

120

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	30	0	0	60	90
Process 2	30	0	90	0	120

Free Run **Pause** **Reset Simulation** **Done**

Questions

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

120 milliseconds

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?

Yes, the CPU was always busy. When it isn't busy, the CPU might not be fully utilized during I/O operations, but it can still efficiently handle other tasks.

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

IO performs in Process 1 and SJF algorithm.

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

The Shortest Job First (SJF) scheduling algorithm can be either preemptive or non-preemptive,

depending on its implementation. The key difference lies in whether the scheduler is allowed to interrupt a running process to start another one with a shorter burst time (preemption) or not (non-preemption).

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**.

Scheduling Algorithms - Introductory

Process Configuration

	CPU	Balanced	IO	Runtime (Milliseconds)
Process 1	<input checked="" type="radio"/> Intense	<input type="radio"/> Balanced	<input type="radio"/> Intense	30
Process 2	<input checked="" type="radio"/> CPU	<input type="radio"/> Balanced	<input type="radio"/> Intense	130

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☒ Round Robin

☐ First Come First Served

☐ Shortest Job First

Animation Control

Simulation speed

Fastest ————— Slowest

System Statistics

Elapsed Time (Milliseconds)

160

Balanced processes perform IO every 15 ms of Runtime

IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	30	0	20	60	110
Process 2	130	0	30	0	160

Free Run

Pause

Reset Simulation

Done

Questions

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

160 milliseconds

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?

Yes, the CPU was always busy. When it isn't busy, the CPU might not be fully utilized during I/O operations, but it can still efficiently handle other tasks.

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

IO performs in Process 1 and Round Robin algorithm.

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

Round Robin is a preemptive scheduling algorithm because it allows the CPU scheduler to interrupt a currently running process after a fixed time quantum and switch to the next process in the ready queue.

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?

FCFS: 220 milliseconds

SJF: 120 milliseconds

RR: 160 milliseconds

=> FCFS's total time in the system can be affected by the order of arrival, SJF aims to minimize total processing time by prioritizing shorter jobs, and RR introduces preemption with a fixed time quantum, which can influence waiting times for both short and long jobs.

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**.

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	30
Process 2	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	30

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☒ Round Robin

☐ First Come First Served

☐ Shortest Job First

Animation Control

Simulation speed

Fastest Slowest

System Statistics

Elapsed Time (Milliseconds)

60

Balanced processes perform IO every 15 ms of Runtime

IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	30	0	20	0	50
Process 2	30	0	30	0	60

Free Run

Pause

Reset Simulation

Done

Questions

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

No, because of the wait time in each process in the Round Robin algorithm.

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?

Process 1: It isn't the same. Because Process 1 only took two turns waiting for Process 2 to run and each turn took 10 milliseconds.

Process 2: It's the same. Because Process 2 took three turns waiting for Process 1 to run and each turn took 10 milliseconds.

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

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	60
Process 2	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	60

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☒ Round Robin

☐ First Come First Served

☐ Shortest Job First

Animation Control

Simulation speed

Fastest Slowest

System Statistics

Elapsed Time (Milliseconds)

120

Balanced processes perform IO every 15 ms of Runtime

IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	60	0	50	0	110
Process 2	60	0	60	0	120

Free Run

Pause

Reset Simulation

Done

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

No, because of the wait time in each process in the Round Robin algorithm.

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?

Process 1: It isn't the same. Because Process 1 only took five turns waiting for Process 2 to run and each turn took 10 milliseconds.

Process 2: It's the same. Because Process 2 took six turns waiting for Process 1 to run and each turn took 10 milliseconds.

The pattern is that which Process starts first will have 1 less wait time.

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?

RR is generally considered fair because it provides each process with a regularly allocated time quantum. However, fairness can be subjective and depend on specific system requirements. Meanwhile, FCFS may lead to unfairness due to the convoy effect, while SJF may not be fair to longer jobs. The choice of a scheduling algorithm depends on the desired trade-offs between fairness, response time, and overall system efficiency.

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**.

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	50
Process 2	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	30

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☒ Round Robin

☐ First Come First Served

☐ Shortest Job First

Animation Control

Simulation speed

Fastest ————— Slowest

System Statistics

Elapsed Time (Milliseconds)

95

Balanced processes perform IO every 15 ms of Runtime

IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	50	0	0	45	95
Process 2					

Free Run

Pause

Reset Simulation

Done

Questions

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

95 milliseconds

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

50 milliseconds

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

155 milliseconds

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

Lab Activity: Scheduling: Introductory: RR 4

Familiarization 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**.

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	50
Process 2	<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	50

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☒ Round Robin

☐ First Come First Served

☐ Shortest Job First

Animation Control

Simulation speed

Fastest Slowest

System Statistics

Elapsed Time (Milliseconds)

130

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	50	0	30	45	125
Process 2	50	0	35	45	130

Free Run

Pause

Reset Simulation

Done

Questions

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

130 milliseconds

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

Because of the difference in wait time, Process 1 performs less 1 wait turn than

Process 2 and it took 5 milliseconds.

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

130 milliseconds

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**).

The Round Robin scheduling algorithm provides a fair allocation of CPU time to processes, but the interleaved execution and context switching overhead mean that the time taken to execute two processes is not necessarily double the time taken to execute one of them.

Lab Activity: Scheduling: Introductory: RR 5

Familiarization 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

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	50
Process 2	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	50

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

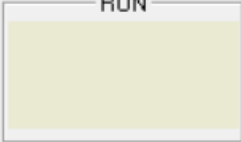
IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

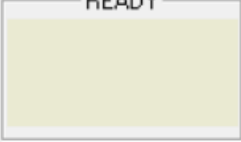
Turn OFF Icon Flashing

Runtime State Display


RUN




READY



BLOCKED



COMPLETED



Scheduler Configuration

☒ Round Robin

☐ First Come First Served

☐ Shortest Job First

Animation Control

Simulation speed

Fastest Slowest

System Statistics

Elapsed Time (Milliseconds)

50

Balanced processes perform IO every 15 ms of Runtime

IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	50	0	0	0	50
Process 2	0	0	0	0	0

Free Run

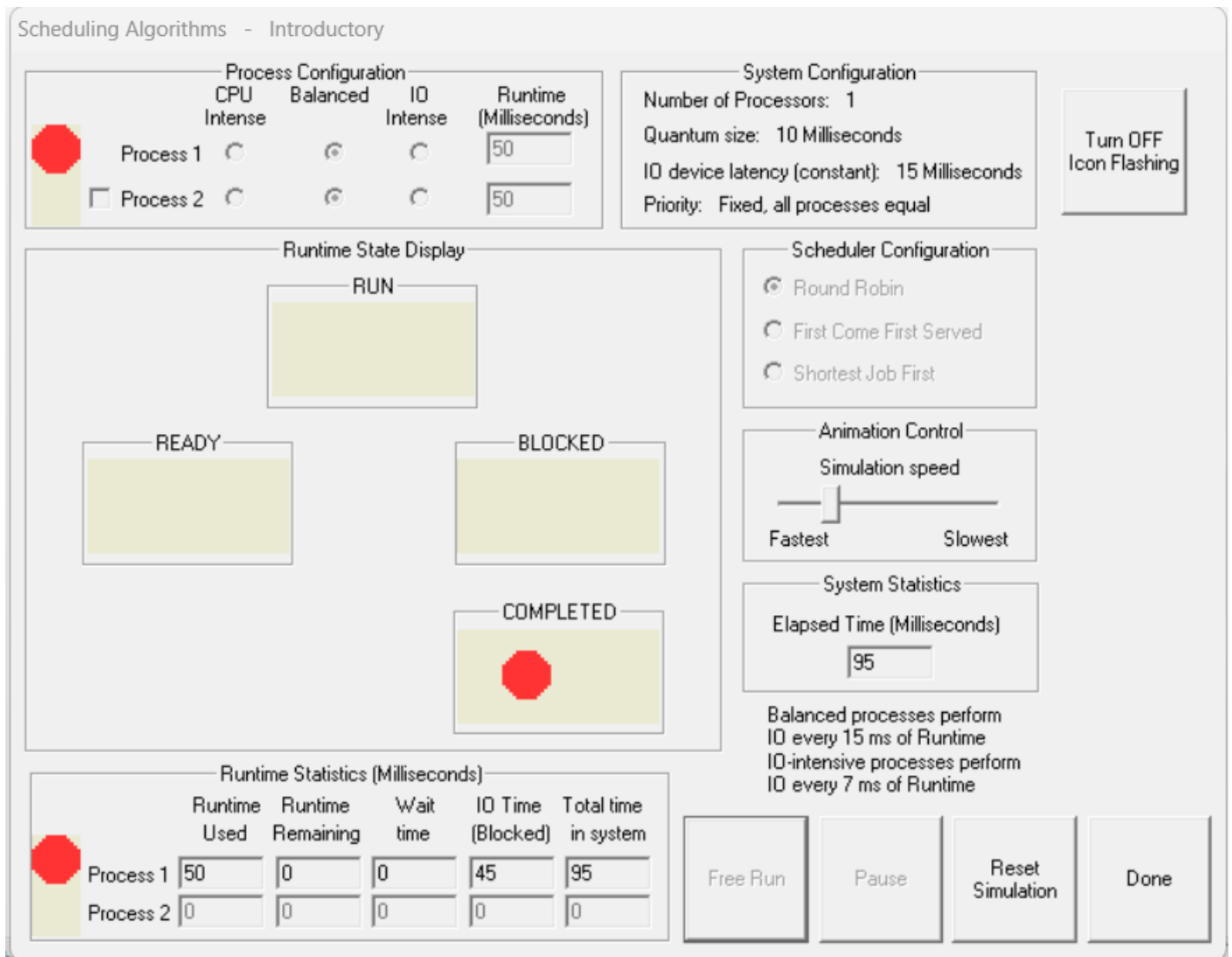
Pause

Reset Simulation

Done

Ready => Run => Completed

- ii. Run a single balanced process



Ready => Run => Blocked (each 15 milliseconds) => Completed

- iii. Run a single IO intensive process

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	50
Process 2	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	50

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☒ Round Robin

☐ First Come First Served

☐ Shortest Job First

Animation Control

Simulation speed

Fastest Slowest

System Statistics

Elapsed Time (Milliseconds)

155

Balanced processes perform IO every 15 ms of Runtime

IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	50	0	0	105	155
Process 2	0	0	0	0	0

Free Run

Pause

Reset Simulation

Done

Ready => Run =>Blocked (each 7 milliseconds) => Completed

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.

	IO Time (Blocked)	Total time in system	Elapsed Time
CPU Intense	0	50	50
Balanced	45	95	95
IO Intense	105	155	155

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

CPU Intense: IO doesn't perform

Balanced: IO performs each 7 milliseconds

IO Intense: IO performs each 15 milliseconds

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

CPU Intense is the fastest and IO Intense is the slowest because of the appearance of IO performance.

Q3. Explain why the slowest process is so slow?



Because it performs IO every 7 milliseconds so when the runtime is large, the total time in the system will perform more IO and make it the slowest process.

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

i. Run both as CPU intensive process

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
 Process 1	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text" value="50"/>
 Process 2	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text" value="50"/>

System Configuration

Number of Processors: 1
Quantum size: 10 Milliseconds
IO device latency (constant): 15 Milliseconds
Priority: Fixed, all processes equal

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☒ Round Robin
☐ First Come First Served
☐ Shortest Job First

Animation Control

Simulation speed



Fastest Slowest

System Statistics

Elapsed Time (Milliseconds)

Balanced processes perform IO every 15 ms of Runtime
IO-intensive processes perform IO every 7 ms of Runtime


Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
 Process 1	<input type="text" value="50"/>	<input type="text" value="0"/>	<input type="text" value="40"/>	<input type="text" value="0"/>	<input type="text" value="90"/>
 Process 2	<input type="text" value="50"/>	<input type="text" value="0"/>	<input type="text" value="50"/>	<input type="text" value="0"/>	<input type="text" value="100"/>

Ready => Processes run alternately (10 milliseconds each) after previous Process is back to Ready State => Completed

ii. Run both as balanced process

Scheduling Algorithms - Introductory


 Process 1

☐ CPU Intense

☐ Balanced

☐ IO Intense

Runtime (Milliseconds)

 Process 2

☒

☐ CPU Intense

☐ Balanced

☐ IO Intense

Runtime (Milliseconds)

System Configuration

Number of Processors: 1
 Quantum size: 10 Milliseconds
 IO device latency (constant): 15 Milliseconds
 Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☒ Round Robin
 ☐ First Come First Served
 ☐ Shortest Job First

Animation Control

Simulation speed



Fastest Slowest

System Statistics

Elapsed Time (Milliseconds)

Balanced processes perform IO every 15 ms of Runtime
 IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
<div>  Process 1 </div>	<input type="text" value="50"/>	<input type="text" value="0"/>	<input type="text" value="30"/>	<input type="text" value="45"/>	<input type="text" value="125"/>
<div>  Process 2 </div>	<input type="text" value="50"/>	<input type="text" value="0"/>	<input type="text" value="35"/>	<input type="text" value="45"/>	<input type="text" value="130"/>

Free Run

Pause

Reset Simulation

Done

Ready => Processes run alternately (10 milliseconds each) after previous Process is back to Ready State and process is in Blocked State every 15 milliseconds run time => Completed

iii. Run both as IO intensive process

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
Process 1	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	50
Process 2	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	50

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Turn OFF Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☒ Round Robin

☐ First Come First Served

☐ Shortest Job First

Animation Control

Simulation speed

Fastest Slowest

System Statistics

Elapsed Time (Milliseconds)

162

Balanced processes perform IO every 15 ms of Runtime

IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
Process 1	50	0	0	105	155
Process 2	50	0	7	105	162

Free Run

Pause

Reset Simulation

Done

Ready => Processes run alternately after previous Process is back to Ready State and process is in Blocked State every 7 milliseconds run time => Completed

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.

	Wait time	IO Time (Blocked)	Total time in system	Elapsed Time
CPU Intense	40/50	0/0	90/100	100
Balanced	30/35	45/45	125/130	130
IO Intense	0/7	105/105	155/162	162

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

Process 1 always takes less time to run than Process 2 and the time consumed is based on the wait time of Process 2 subtracts to Process 1.

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?

The Balanced and IO Intense is blocked and have to wait for another process to run. They were blocked because IO performed at a different time.

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

difference.

CPU Intense is the fastest and IO Intense is the slowest because of the appearance of IO performance.



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).

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
 Process 1	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text" value="50"/>
 Process 2	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="text" value="50"/>

System Configuration

Number of Processors: 1

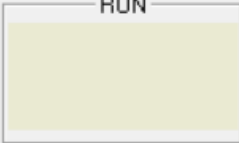
Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

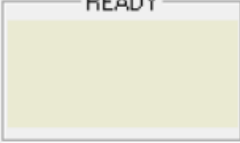
Priority: Fixed, all processes equal

Runtime State Display


RUN



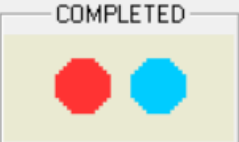
READY



BLOCKED



COMPLETED



Scheduler Configuration

☒ Round Robin

☐ First Come First Served

☐ Shortest Job First

Animation Control

Simulation speed

Fastest Slowest



System Statistics

Elapsed Time (Milliseconds)

Balanced processes perform IO every 15 ms of Runtime



IO-intensive processes perform IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
 Process 1	<input type="text" value="50"/>	<input type="text" value="0"/>	<input type="text" value="14"/>	<input type="text" value="0"/>	<input type="text" value="64"/>
 Process 2	<input type="text" value="50"/>	<input type="text" value="0"/>	<input type="text" value="20"/>	<input type="text" value="105"/>	<input type="text" value="175"/>

Scheduling Algorithms - Introductory

Process Configuration

	CPU Intense	Balanced	IO Intense	Runtime (Milliseconds)
 Process 1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="text" value="50"/>
<input checked="" type="checkbox"/>  Process 2	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="text" value="50"/>

System Configuration

Number of Processors: 1

Quantum size: 10 Milliseconds

IO device latency (constant): 15 Milliseconds

Priority: Fixed, all processes equal

Turn OFF
Icon Flashing

Runtime State Display

RUN

READY

BLOCKED

COMPLETED

Scheduler Configuration

☐ Round Robin
☒ First Come First Served
☐ Shortest Job First

Animation Control

Simulation speed



Fastest Slowest

System Statistics

Elapsed Time (Milliseconds)

Balanced processes perform
IO every 15 ms of Runtime
IO-intensive processes perform
IO every 7 ms of Runtime

Runtime Statistics (Milliseconds)

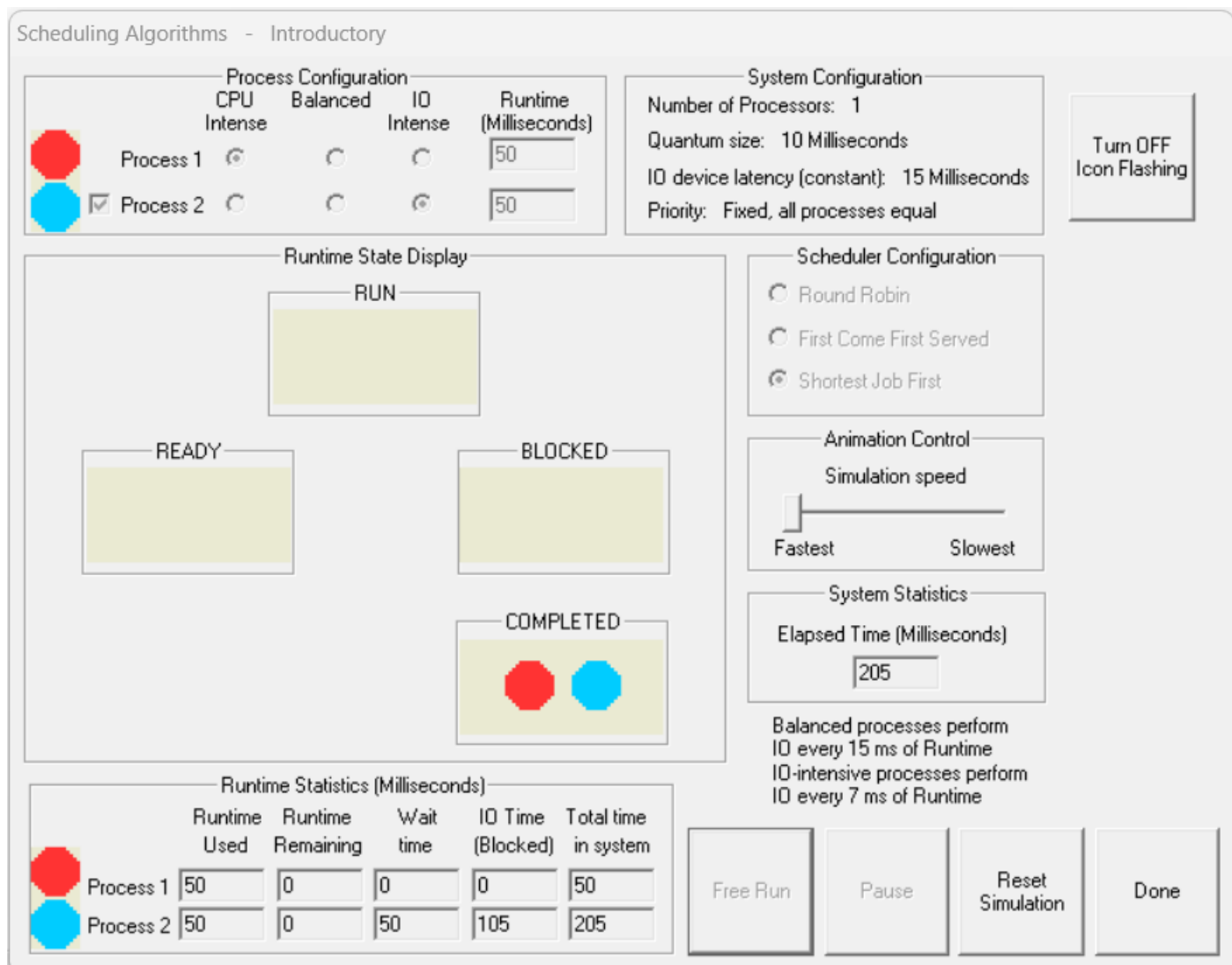
	Runtime Used	Runtime Remaining	Wait time	IO Time (Blocked)	Total time in system
 Process 1	<input type="text" value="50"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="50"/>
 Process 2	<input type="text" value="50"/>	<input type="text" value="0"/>	<input type="text" value="50"/>	<input type="text" value="105"/>	<input type="text" value="205"/>

Free Run

Pause

Reset Simulation

Done



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

The discrepancy between total time in two processes is changed based on different types of processes running at the same time.

4. Observe the effect of the different scheduling algorithms.

The discrepancy between total time in two processes is increasing.

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

The change in each type of time is change responsively to scheduling behavior and the total time taken for the simulation.

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

The type of process that best shows differences between scheduling algorithms is CPU Intense, as it emphasizes the algorithms' ability to minimize total processing time. I/O Intense processes may show less variation between scheduling algorithms, as their execution is often dominated by I/O wait times, which are not influenced as directly by scheduling decisions. The choice of the "best" scheduling algorithm depends on the specific workload characteristics and system requirements.