Dakota Harkins

University of Technology Sydney  - 97116899

RTOS Assignment 3

Contents

[Introduction 2](#_Toc514789319)

[Theory of Operation 3](#_Toc514789320)

[Operating conditions 4](#_Toc514789321)

[Prg\_1: 4](#_Toc514789322)

[Prg\_2: 4](#_Toc514789323)

[Implementation 6](#_Toc514789324)

[Method 6](#_Toc514789325)

[Normalizing code 6](#_Toc514789326)

[Documentation standards 6](#_Toc514789327)

[Deadlock detection example: 7](#_Toc514789328)

[SRTF example: 7](#_Toc514789329)

[Flow Chart 8](#_Toc514789330)

[Prg\_1: 8](#_Toc514789331)

[Prg\_2: 9](#_Toc514789332)

[Experiments 10](#_Toc514789333)

[Hypothesis 11](#_Toc514789334)

[Results 11](#_Toc514789335)

[Conclusion 12](#_Toc514789336)

# 

# Introduction

This report details the design, implementation and testing of a program (Prg\_1) that applies a shortest time remaining algorithm (SRTF) to a set of processes, pass the average time waiting and turnaround time through a FIFO and then print that data to a text file named “output.txt”. It was tasked that this program must involve the use of threads and a FIFO.

A secondary program Prg\_2 read process information from a text file “Topic2\_Prg\_2.txt and detects potential deadlocks in these processes. It then prints a relevant message to “output\_topic2.txt”. If there was a deadlock this message will include the locked processes and if there was not a deadlock it will print the order in which the processes are completed. Prg\_2 also alerts the user through the terminal that the data has been written to file.

# Theory of Operation

There were several technical requirements for this assignment. There were also several other key concepts used in the solution. For the sake of the report these have been defined briefly below:

Threads & Multi-threading: Multithreading is when a full program is broken up into multiple small sequences of programmed instructions that can be executed simultaneously. A thread of execution in one set of these instructions.

FIFO: a mechanism that allows two threads to communicate with each other. One thread can write data into the pipe and another can read from the other ‘end’. The crucial concept here is that the first data to be placed into the FIFO will be the first data to be consumed.

Makefile: this is a file used to run various compiling tasks with one simple command.

Deadlock: a point at which no progress can be made on any process due to resource limitations.

User defined signal: a software generated interrupt that is sent to a process by the OS due to another process triggering it.

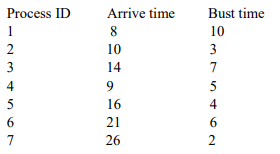
Shortest Time Remaining First Algorithm (STRF): scheduling method that ensures that at any time the process with the smallest amount of time remaining until completion is selected to execute.

# Operating conditions

The operating requirements given for this assignment included:

Prg\_1:

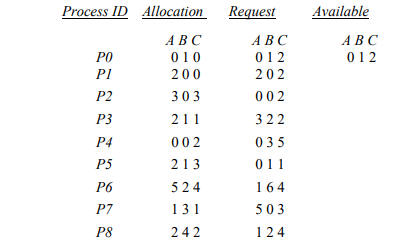
* The STRF algorithm will be applied to the hard-coded data provided in the assignment:



* Two threads would be used to implement the program:
  + Thread 1: simulate CPU scheduling by applying the STRF algorithm. It will measure the average waiting time and turnaround time in the CPU scheduling. It will create a FIFO and then write the average waiting time and turnaround time to that FIFO.
  + Thread 2: waits for signal from thread 1 to say that there is something new in the FIFO: reads that data and writes it to a text file named “output.txt”. If this file does not exist in the current directory it will be created, otherwise it will be overwritten.
* Waiting time and turnaround time are presumed to be in the same units as the arrival time and bust time (ms)
* If an error occurs at any point this will print the error to the user and terminate the program including:
  + If no “output.txt” was unable to be opened
  + If FIFO initialization failed

Prg\_2:

* Data stored within “Topic2\_Prg\_2.txt” will include only 9 processes, and 3 resources and will be in the format shown below:



* E.g. the first two lines contain the table header. The third line will also contain the available data. Data will be stored as integers.
* Data shall be written to a text file named “output\_topic2.txt”. If this file does not exist in the current directory it will be created, otherwise it will be overwritten.
* If a deadlock is detected the output will be a list of deadlocked processes, otherwise the output will be process sequence order.
* Once the output has been written to the file the program will send a user defined signal SIGUSR1 which will send a message to the user “Writing to output\_topic2.txt has been completed”
* If an error occurs at any point this will print the error to the user and terminate the program including:
  + If “Topic2\_Prg\_2.txt” was unable to be opened
  + If “output\_topic2.txt” was unable to be opened

# Implementation

## Method

The design of the code was almost exactly dictated by the assignment constraints as written above. The only real decision was in the use of mkfifo, a c library component to develop the FIFO. This was chosen because it was required we demonstrated our knowledge of FIFOs in the assignment and also it allowed for a simple solution.

Furthermore the code was also developed to ensure reusability and sustainability (it is easy to understand) through the two following methods:

### Normalizing code

All code should be broken down or “normalized” into packages of functionality that are easy to understand and may be reused by various programs. In this project I broke up the functionality into 5 sections:

* Processes: this includes the initialization and analysis of the processes required for thread 1 in Prg\_1
* Threads: this includes the implementation of the two threads including the FIFO
* Prg\_1: this contains all the function calls from the various source files to implement the Prg\_1 logic
* Prg\_2: this contains the function calls from the various files to implement the Prg\_2 logic

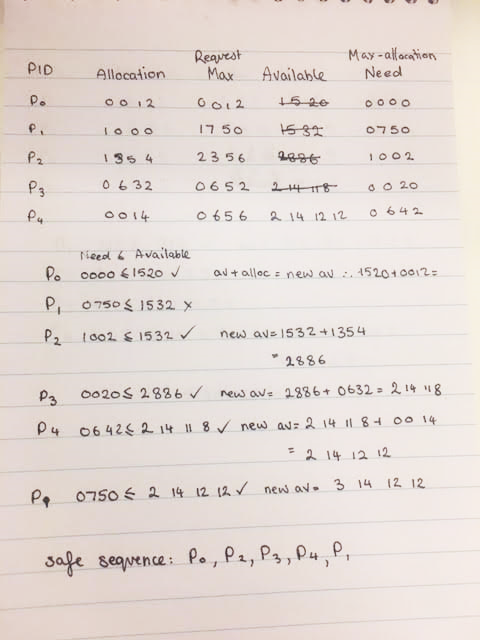
### Documentation standards

It is important for reusability and maintainability that all coding documentation is kept up to date and in a standard format. For these reasons:

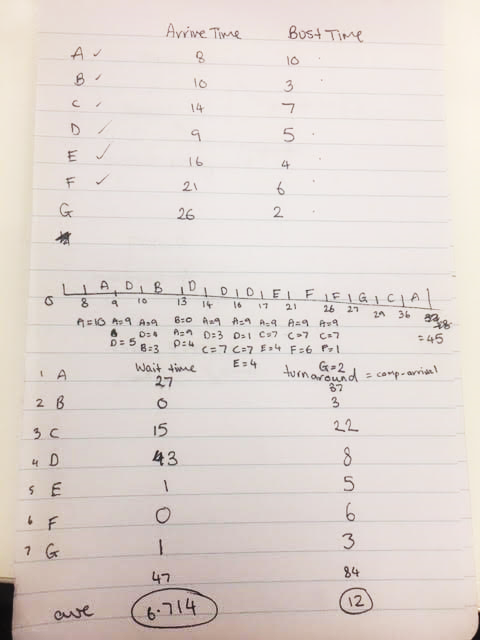
* The small reusable packages all have header files to make them simpler to utilize in the future.
* All files include developer headers including date developed and creator and a brief description of what this file contains
* A readme file is also provided which includes further compiling information

To ensure the accuracy of the code I also completed all the required calculation tasks by hand.

Deadlock detection example:



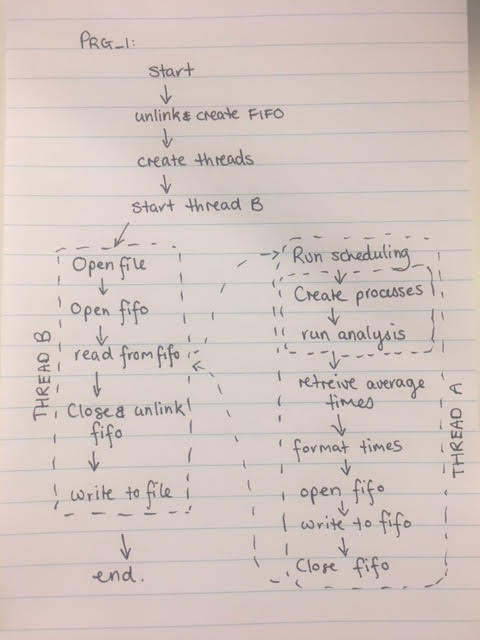
SRTF example:



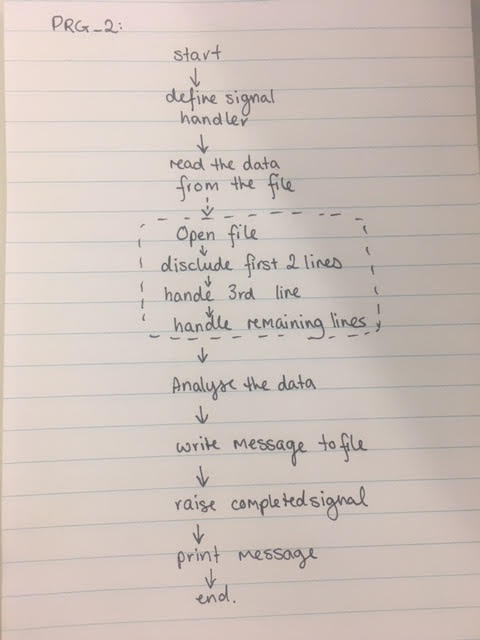
## Flow Chart

The developmental flow chart is as follows:

Prg\_1:

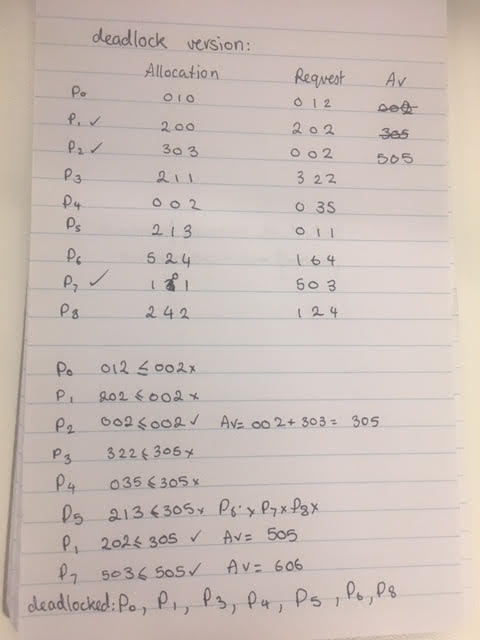


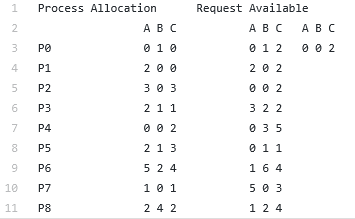
Prg\_2:



# Experiments

Aside from the original data I also utilized several of my own tests in order to confirm that the programs ran completely as expected including input processes in Prg\_2 that would deadlock and processes in Prg\_1 that had a very complicated SRTF algorithm e.g. two processes arrived with the same time remaining.

I calculated a set of processes that would deadlock and implemented them as follows: 



## Hypothesis

There would be no issue raised in normal running of both of these tests. The STRF algorithm would execute the first process.

## Results

Both programs ran successfully as expected. The deadlocking detection output was “Deadlocking occurred due to processes: P0 P3 P4 P5 P6 P8” which is correct.

# Conclusion

The programs worked as expected and were able to handle various test cases. The design and implementation as successful and correctly in line with the assignment specifications. The investigation into deadlocks and STRF, user defined signals and FIFOs was informative. Conclusively this was a successful project.