Instituto Tecnológico de Costa Rica

Proyecto 1

Curso

Principios de Sistemas Operativos

Profesora

Ericka Marín Schuman

Estudiantes

Austin Edward Hakanson Hidalgo 2018231867

Ulises Rodríguez Pérez 2019380067

Semestre I, 2022

Índice

Introducción

Este proyecto consiste en la elaboración de un simulador de planificador de CPU. La implementación conlleva la creación de dos programas: un cliente y un servidor.

El programa cliente se encarga de generar la información de burst (tiempo de ejecución) y prioridad de los procesos que se desean simular en el planificador del CPU. El cliente tiene dos posibles configuraciones: manual y automática. Bajo la modalidad manual el cliente debe leer los datos para la creación de los procesos de un archivo de texto plano, mientras que con la modalidad automática genera valores aleatorios para el burst y la prioridad. Además debe crearse un hilo por cada proceso que se genera en el cliente.

Los procesos son enviados mediante la utilización de sockets al simulador del CPU que se ejecuta en el programa servidor. En este segundo programa se tiene un hilo que simula el planificador de trabajo, al recibir los procesos y colocarlos en la cola de los procesos listos para ejecutar (Ready Queue) y otro hilo para simular el planificador del CPU que selecciona los procesos según el algoritmo con el cual haya sido configurado el programa del servidor antes de ser inicializado.

Con este proyecto se busca tener un mejor entendimiento de la administración de procesos en los sistemas operativos a través de la aplicación práctica de una simulación de los conceptos teóricos estudiados en clase.

Estrategia de Solución

Para resolver la asignación se comenzó primero por realizar una investigación general del manejo de threads y sockets en C. Seguidamente se realizaron pequeños programas prototipos para probar el funcionamiento de los sockets y los threads. Una vez hecho lo anterior se comenzó a desarrollar los dos programas (cliente y servidor) e investigando en internet se encontró un repositorio de un proyecto de un simulador de CPU cuya estructura y organización fueron tomadas como base para la elaboración de este proyecto. Se tomó el código y se ajustó para satisfacer las necesidades específicas de uso de threads y sockets que requería el desarrollo de este proyecto. La estrategia funcionó y dio resultados bastante positivos.

Análisis de Resultados

Las tareas realizadas para llevar a cabo este proyecto así como su porcentaje de realización se describen en la siguiente tabla.

Tarea	Porcentaje de realización	Observaciones
Lectura de archivo para cliente manual	100%	
Creación de procesos	100%	

aleatorios para cliente automático		
Creación de threads para cada proceso en el cliente	100%	
Envío de datos de los procesos por sockets del cliente al servidor	100%	
Creación del job scheduler	100%	
Creación del cpu scheduler	100%	
Desarrollo de los algoritmos de planificación del cpu scheduler	100%	
Generación de estadísticas de los procesos	80%	Falta la cantidad de tiempo en que el CPU estuvo ocioso
Impresión de la cola del ready	100%	
Impresión de los mensajes tanto en el cliente como del servidor para observar la simulación	100%	
Detención del servidor sólo cuando el usuario lo pide	50%	Se puede detener por comando, pero si el cliente se desconecta o termina (como en el caso del cliente manual) el servidor también.

Lecciones aprendidas

Algunas lecciones aprendidas durante el desarrollo de este proyecto incluyen:

1. Se deben empezar todos los proyectos con tiempo, y no subestimar su dificultad.
2. Se aprendió sobre programación en el lenguaje C, el manejo de la librería pthread y el uso de sockets como mecanismo de comunicación entre programas de bajo nivel.
3. Se aprendió que debe haber una comunicación constante y regular entre los miembros del equipo para asesorar el avance de cada uno y determinar qué bloqueos tiene cada uno y tratar de avanzar juntos.
4. Se aprendió sobre la administración y planificación de procesos en los sistemas operativos mediante la aplicación práctica de este simulador.

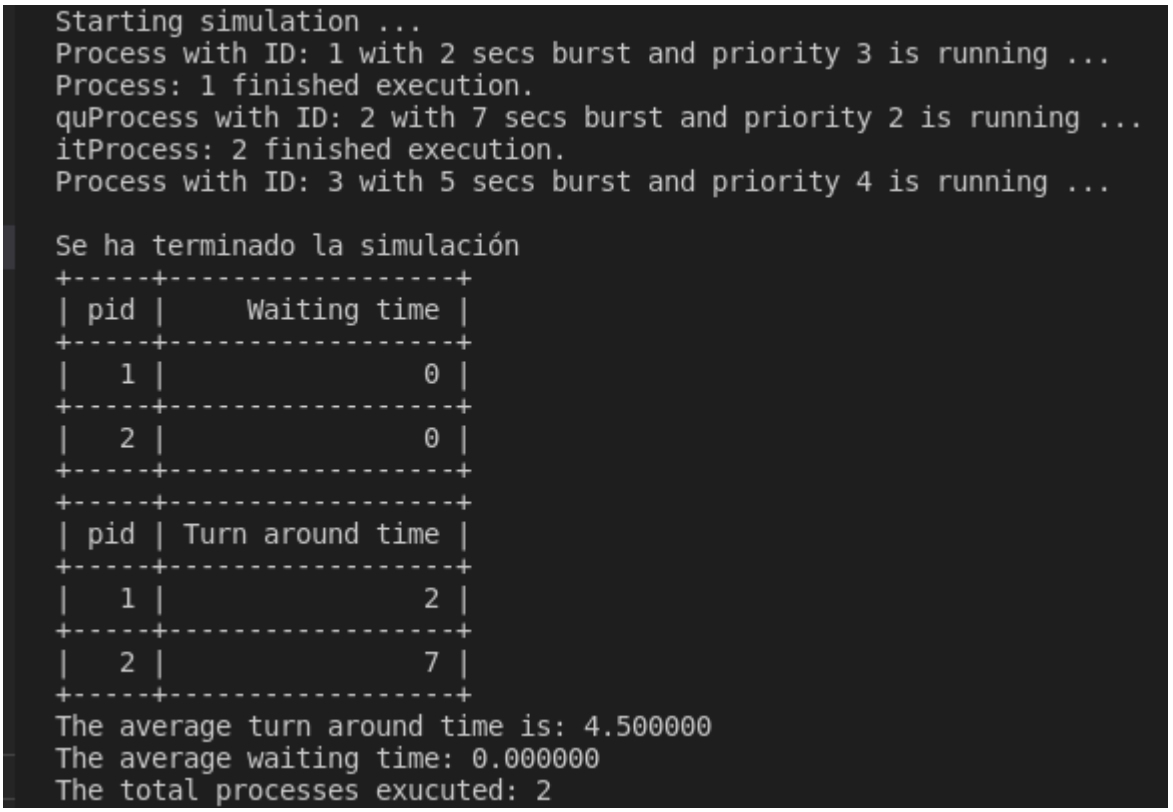
Casos de prueba

A continuación se muestra la ejecución de algunos casos de prueba del simulador bajo diferentes configuraciones. Se muestra la ejecución de la simulación por algoritmo de planificación de procesos seleccionado y el tipo de cliente utilizado.

FIFO

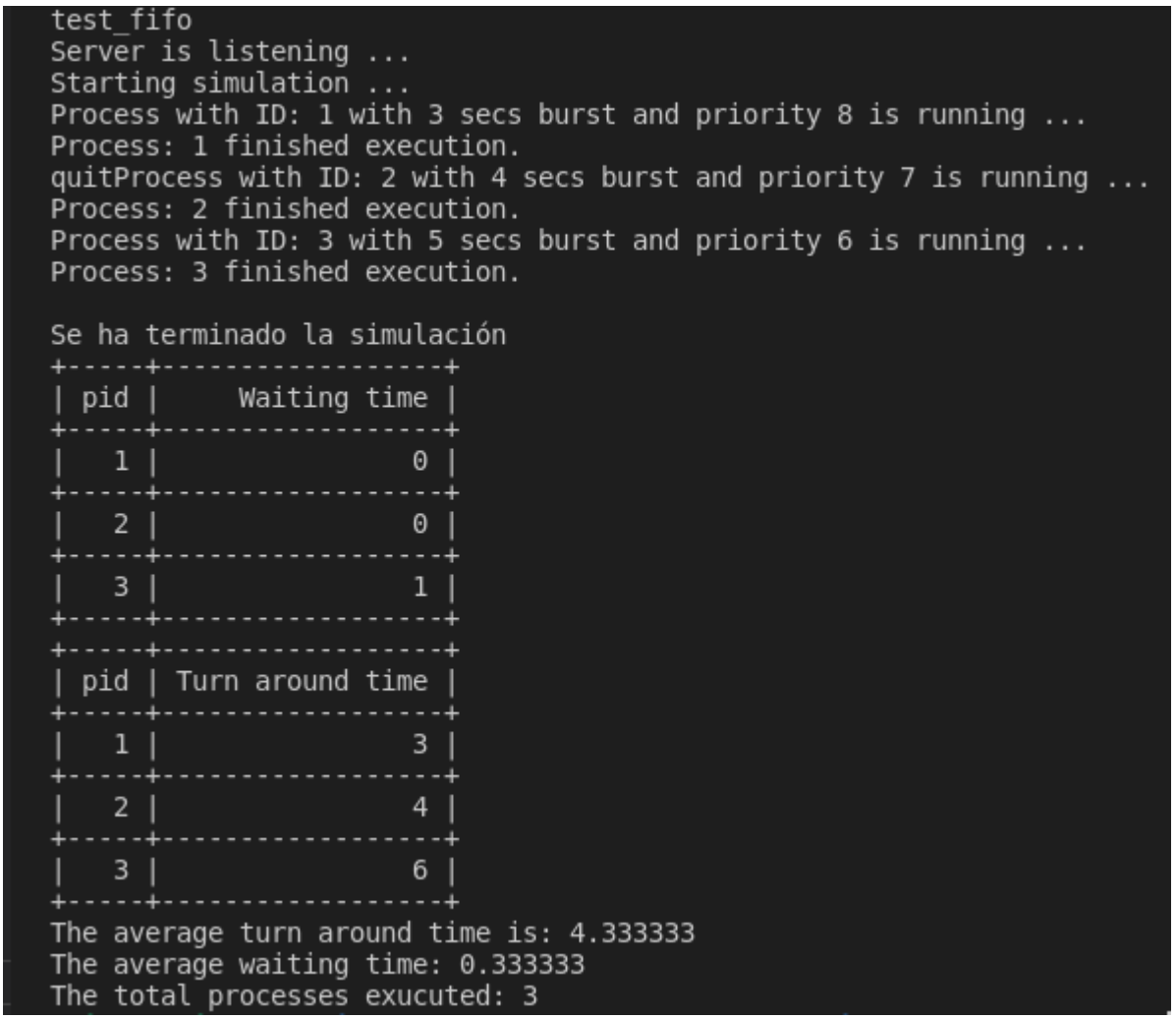
AUTO:

Esta prueba se hizo con los siguientes valores (minBurst 2, maxBurst 8, creationMin 2, creationMax 5) desde el cliente y el resultado fue el siguiente:



MANUAL:

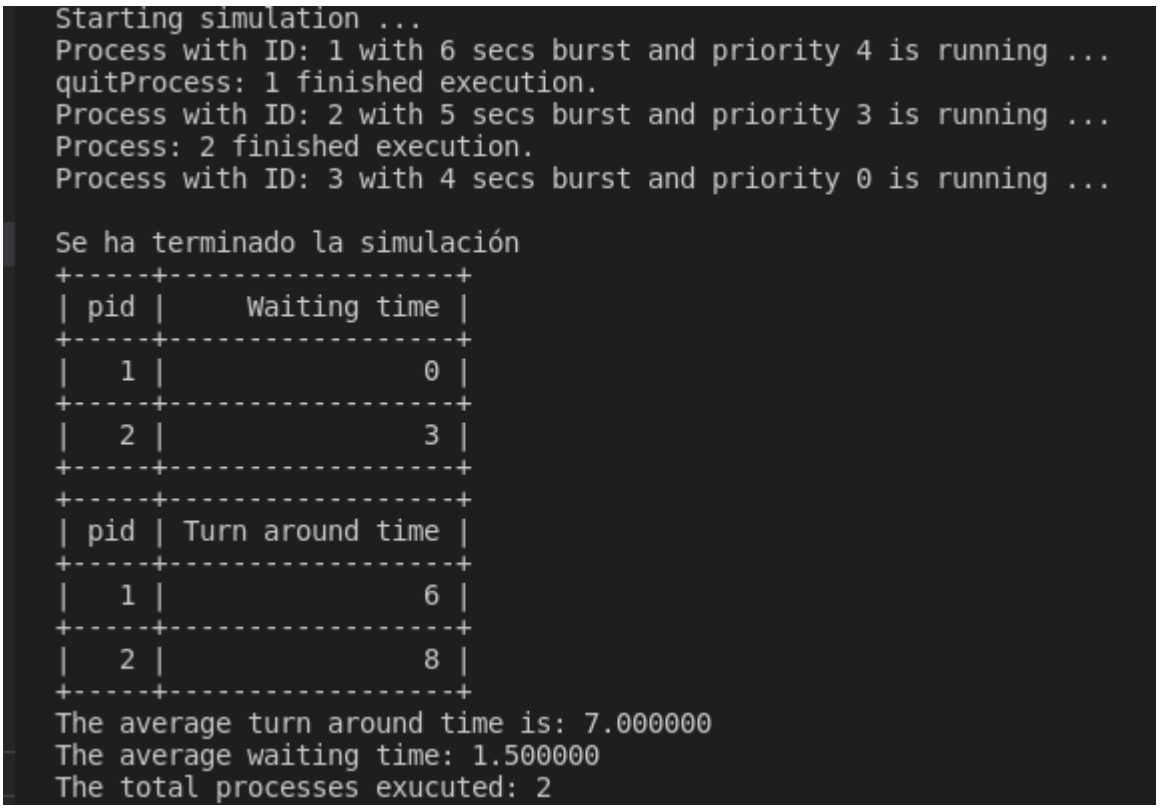
Esta prueba se hizo con los siguientes valores (Burst 3, Prioridad 8, Burst 4, Prioridad 7, Burst 5, Prioridad 6) desde el cliente y el resultado fue el siguiente:



SJF

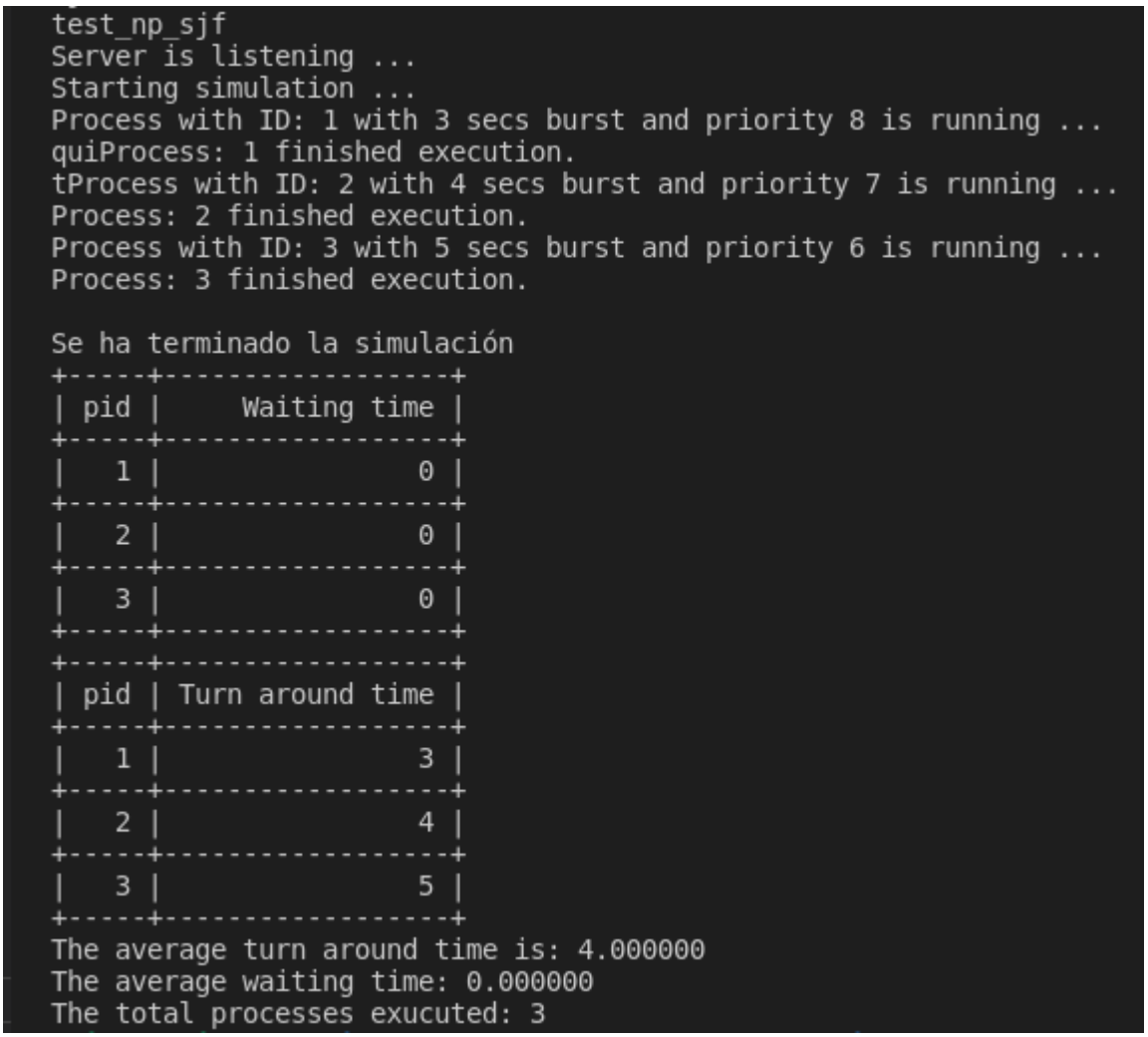
AUTO:

Esta prueba se hizo con los siguientes valores (minBurst 2, maxBurst 8, creationMin 2, creationMax 5) desde el cliente y el resultado fue el siguiente:



MANUAL:

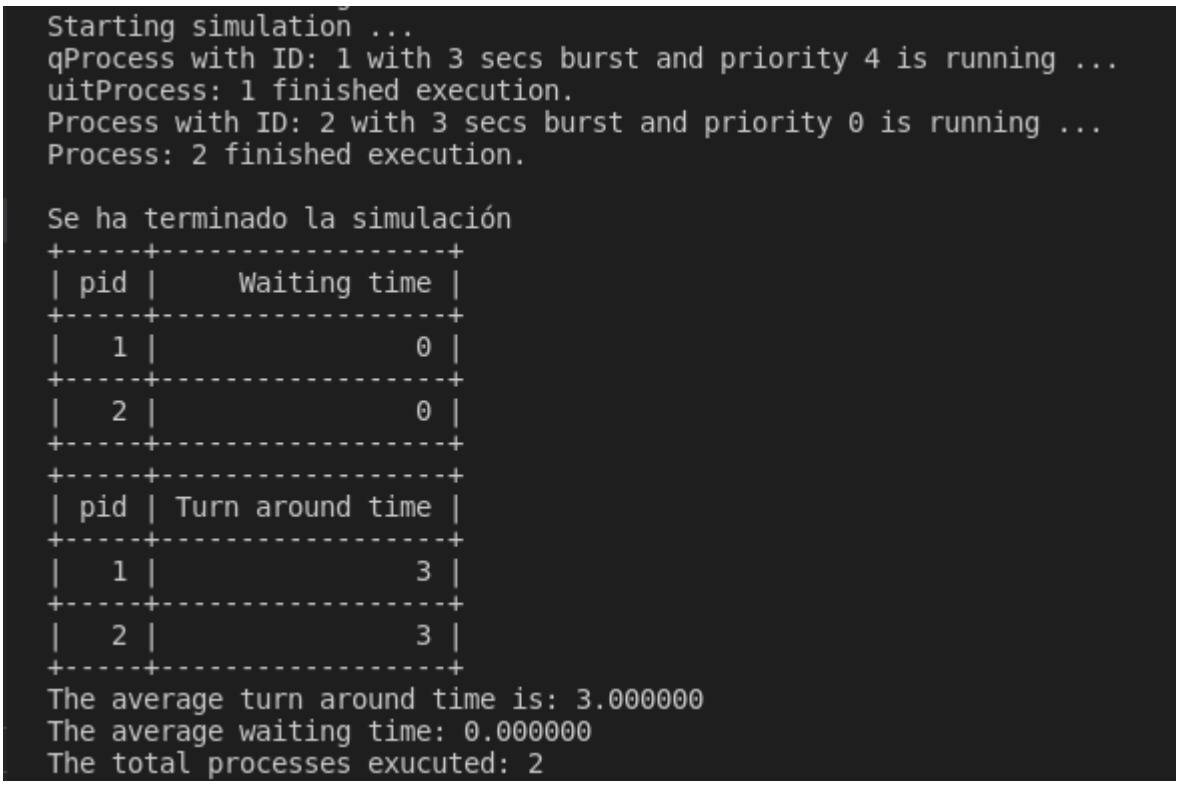
Esta prueba se hizo con los siguientes valores (Burst 3, Prioridad 8, Burst 4, Prioridad 7, Burst 5, Prioridad 6) desde el cliente y el resultado fue el siguiente:



HPF

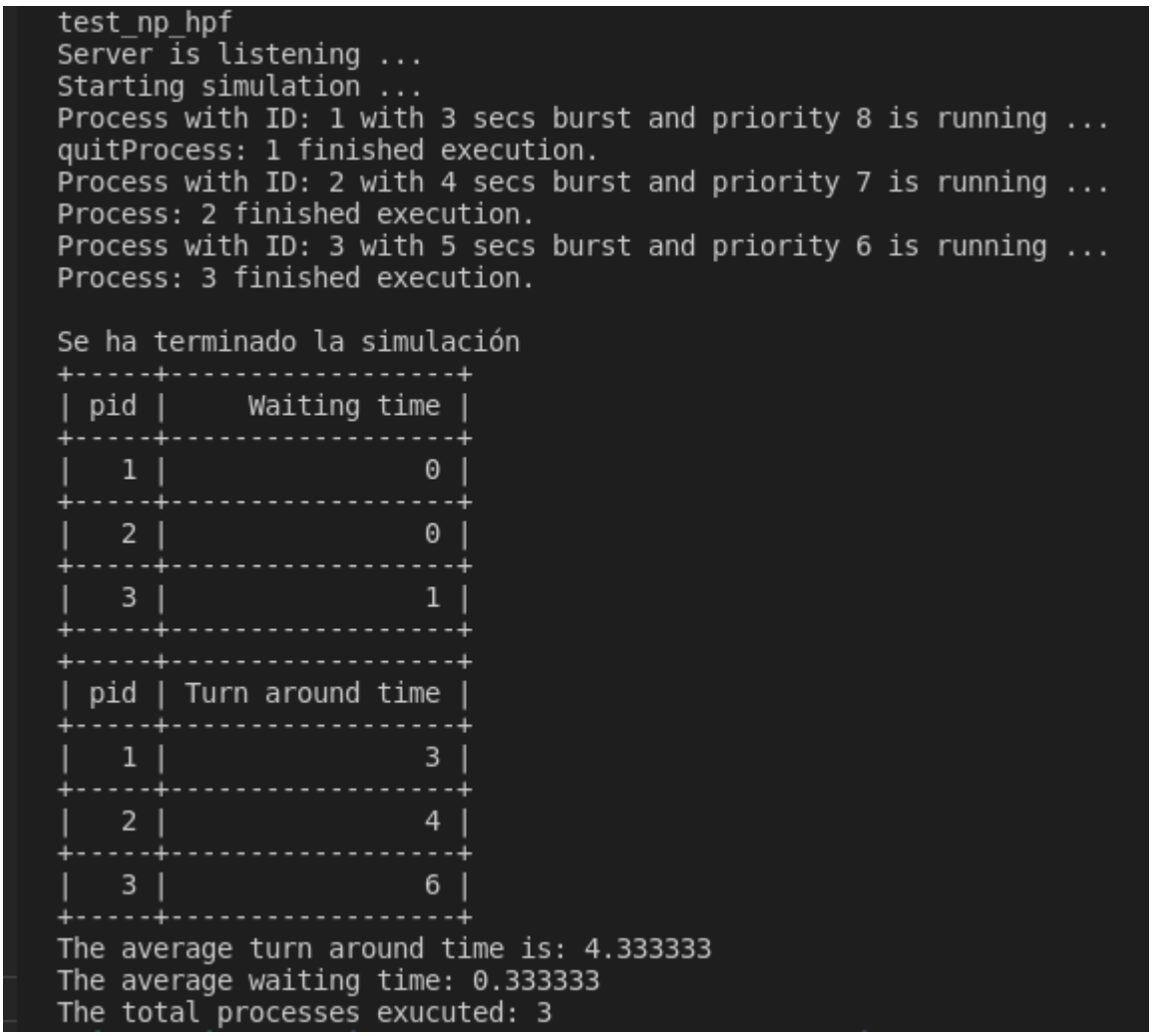
AUTO:

Esta prueba se hizo con los siguientes valores (minBurst 2, maxBurst 8, creationMin 2, creationMax 5) desde el cliente y el resultado fue el siguiente:



MANUAL:

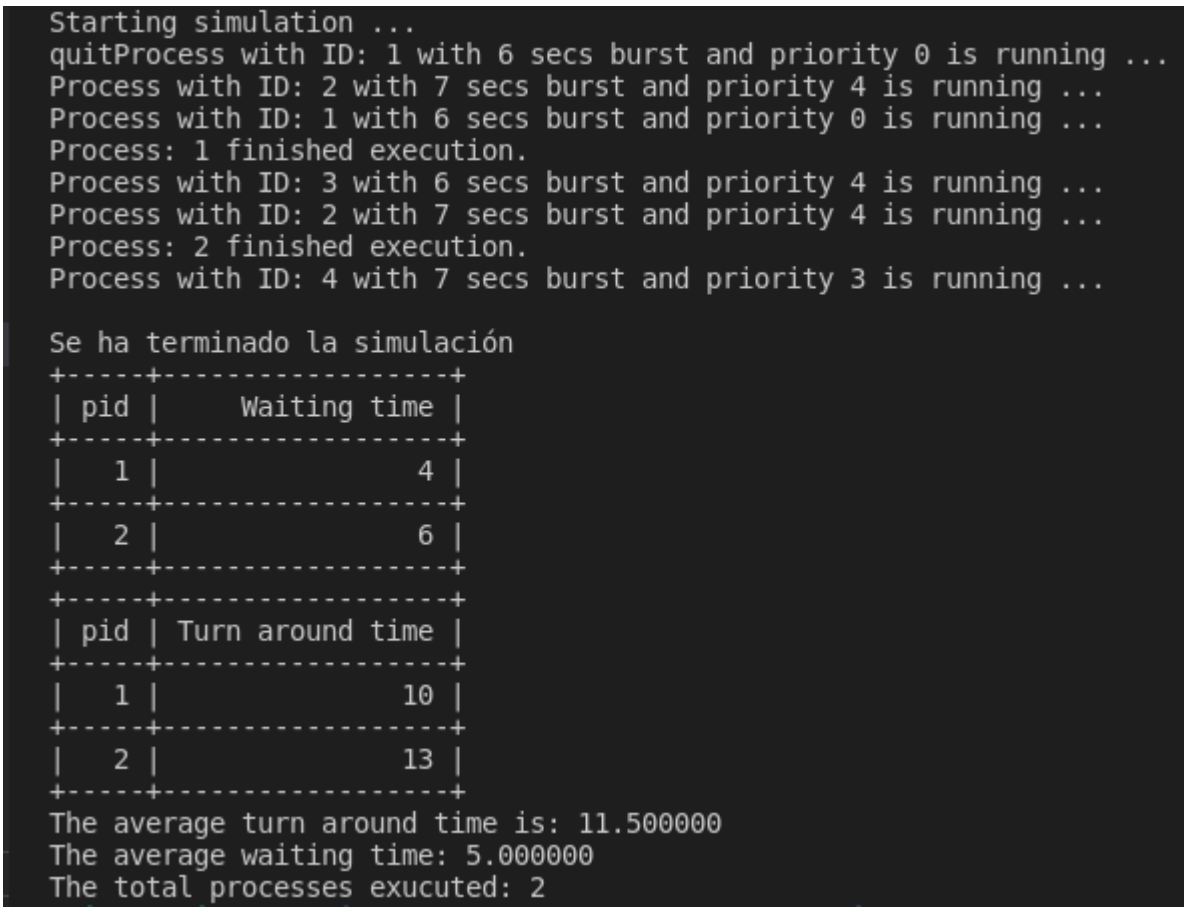
Esta prueba se hizo con los siguientes valores (Burst 3, Prioridad 8, Burst 4, Prioridad 7, Burst 5, Prioridad 6) desde el cliente y el resultado fue el siguiente:



Round Robin

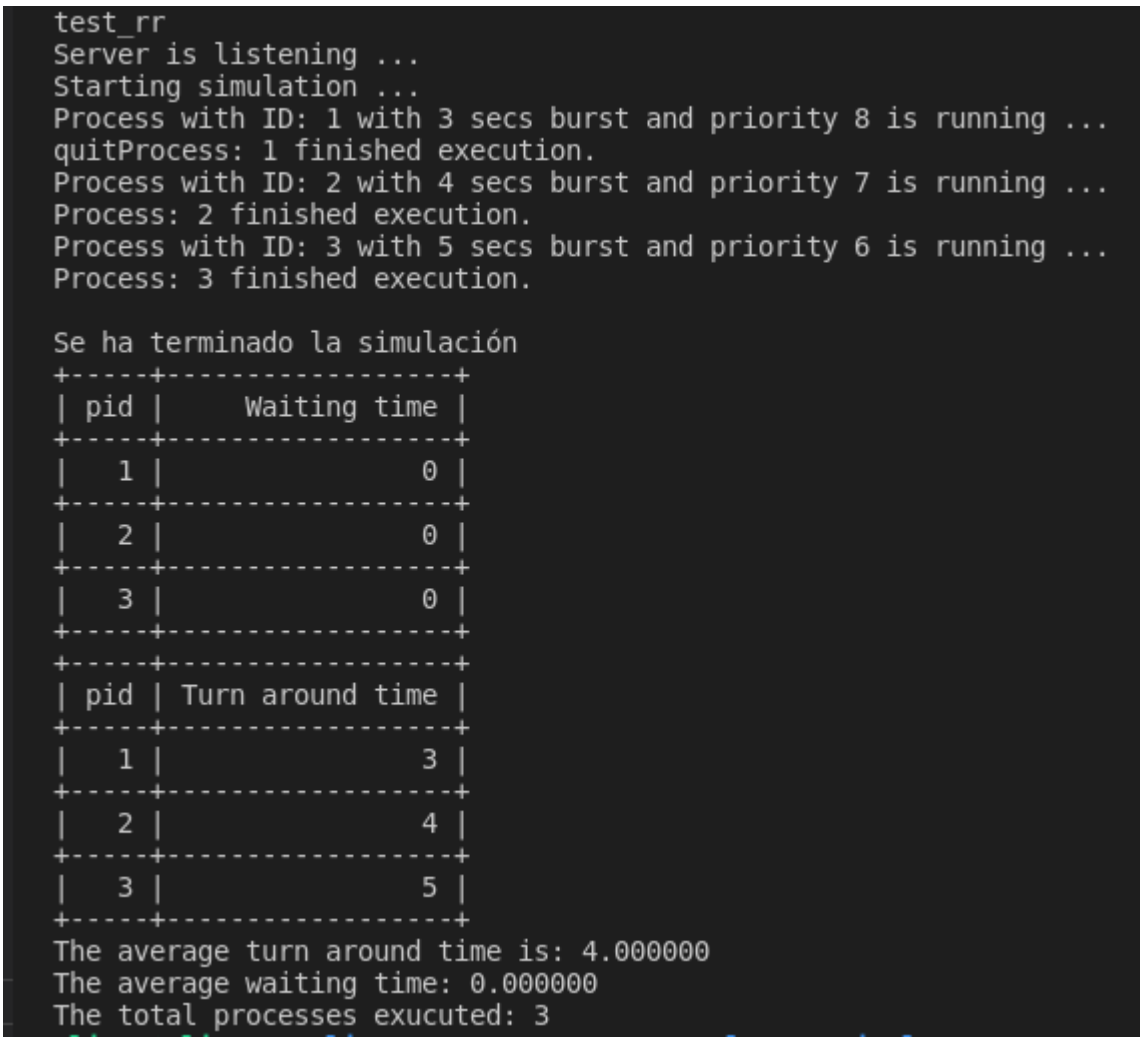
AUTO:

Esta prueba se hizo con los siguientes valores (minBurst 2, maxBurst 8, creationMin 2, creationMax 5) desde el cliente, un quantum de 4 y el resultado fue el siguiente:



MANUAL:

Esta prueba se hizo con los siguientes valores (Burst 3, Prioridad 8, Burst 4, Prioridad 7, Burst 5, Prioridad 6) desde el cliente, un quantum de 4 y el resultado fue el siguiente:



Manual de usuario

Cliente:

1. Debe de haber levantado el servidor antes de ejecutar el cliente
2. Para ejecutar desde el lado del cliente debe dirigirse a la carpeta src/client
3. Desde una consola en la carpeta ya mencionada ejecutamos el comando “make” para compilar la solución.
4. Una vez compilada la solución ejecutamos los siguientes comando según las preferencias de uso:
    a. Manual: “./main --manual --file processes.txt”
    b. Auto: “./main --auto --minBurst n --maxBurst n --creationMin n --creationMax n”
    c. “n” representa al valor entero que será asignado.

Servidor:

1. Para ejecutar desde el lado del servidor debe dirigirse a la carpeta src/server
2. Desde una consola en la carpeta ya mencionada ejecutamos el comando “make” para compilar la solución.

3. Una vez compilada la solución ejecutamos los siguientes comando según las preferencias de uso:
    a. Round Robin: ./main --rr n (donde n es el quantum del algoritmo)
    b. FIFO: ./main --fifo
    c. HPF: ./main --hpf
    d. SJF: ./main --sjf
4. Una vez levantado el servidor y al menos un cliente podremos ejecutar los siguientes comandos:
    a. print: Imprime la cola de procesos en ready.
    b. quit: Finaliza la ejecución y muestras las estadísticas.

Bitácora

En la siguiente tabla se da una descripción general de las actividades realizadas y los momentos en los cuáles se hicieron.

Fecha	Actividad	Responsables
29 de marzo	Leer el enunciado del proyecto y conversar sobre la distribución de las tareas	Austin Hakanson Ulises Rodríguez
30 de marzo - 2 de abril	Investigación sobre sockets en C y la librería pthreads	Austin Hakanson Ulises Rodríguez
2 de abril - 8 de abril	Desarrollo de cliente (manual y automático)	Austin Hakanson
2 de abril - 8 de abril	Desarrollo del job scheduler	Ulises Rodríguez
8 de abril - 12 de abril	Desarrollo del cpu scheduler	Austin Hakanson Ulises Rodíguez
12 de abril	Documentación	Austin Hakanson Ulises Rodíguez

Bibliografía

Korea University Programming Club, (2018) CPU Scheduler Simulator (Version 0.61.2) [Source Code]. https://github.com/kukosmos/cpu-scheduler-simulator