

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

**Profesor:** Jorge Magallanes

Tutivén Acosta Bryan

Término 1S

Guayaquil - Ecuador

**Paralelo:** 2

**Tema:** *Proyecto MidTerm*

**Integrantes:**

*Sistemas Operativos*

Contenido

[ANTCEDENTES 2](#_Toc46521083)

[USO 2](#_Toc46521084)

[Schedsim 2](#_Toc46521085)

[Schedgen 4](#_Toc46521086)

[Schedstats 4](#_Toc46521087)

[ESTRUCTURA DEL PROYECTO 4](#_Toc46521088)

[Archivos de cabecera 4](#_Toc46521089)

[Archivos de implementaciones de funciones prototipos 7](#_Toc46521090)

Tabla de Ilustraciones

[Ilustración 1 Simulación FCFS 3](#_Toc46521301)

[Ilustración 2 Simulación JSF 3](#_Toc46521302)

[Ilustración 3 Simulación RR 4](#_Toc46521303)

[Ilustración 4 Ayuda para Schedsim 4](#_Toc46521304)

[Ilustración 5 Generador de procesos 4](#_Toc46521305)

[Ilustración 6 Salida de 3 archivos con burst promediados 5](#_Toc46521306)

[Ilustración 7 Cabecera Schedsim 6](#_Toc46521307)

[Ilustración 8 Cabecera Schedgen 7](#_Toc46521308)

[Ilustración 9 Cabecera Schedstats 7](#_Toc46521309)

[Ilustración 10 Implementación de Schedsim y parte de Schedstats 7](#_Toc46521310)

[Ilustración 11 Implementación para funciones Schedgen 8](#_Toc46521311)

[Ilustración 12 Implementación de funciones Schedstats 9](#_Toc46521312)

# ANTCEDENTES

El presente proyecto pertenece a la materia de Sistemas Operativos, el fin de este es poner en práctica los conocimientos adquiridos durante el primer parcial del semestre del 2020, estos conocimientos incluyen lo que es la implementación de algoritmos de planificación tales como: First Come First Served (FCFS) donde los procesos se ejecutan en base a cómo van llegando y se van encolando hasta que el planificador le de prioridad de ejecutarse, Shortest Job First (SJF) en modo apropiativo donde los procesos se ejecutaran con el menor burst time y donde se simulara que se quitara el cpu al procesos para darle prioridad a otro procesos y Round Robin (RR) donde el tiempo que se ejecute un proceso dependerá del quatum que se le asigne. El proyecto está desarrollado en el lenguaje de programación C. La importancia de realizar este proyecto radica en como podemos usar cierto algoritmo en base a requerimientos que deseamos, donde la elección de uno u otro dependerá ya sea por el tiempo en que llego un proceso, por el menor tiempo que tarde en ejecutarse o por darle la misma prioridad a todos por igual como es el caso de round robin.

A continuación, encontrará una breve descripción de los componentes que conforman el proyecto MidTerm tales como: uso, parámetros, estructura del programa, etc.

# USO

Se debe comenzar con la ejecución de nuestro archivo makefile a través del comando:

**make**

En caso de querer eliminar los archivos compilados por make se debe escribir el siguiente comando:

**make clean**

## Schedsim

Para ejecutar el simulador lo hacemos de la siguiente manera:

./schedsim nameFile.dat [fcfs] [sjf] [rr] [q]

* NameFile.dat: es el nombre del archivo con que se va a leer para obtener los procesos.
* fcfs -> Simulador para el algoritmo First Come First Served.
* sjf -> Simulador para el algoritmo Shortest Job First.
* rr -> Simulador para el algoritmo Round Robin.
* q -> es el quatum que se va a ejecutar en el algoritmo Roun Robin.

Salida con algoritmo FCFS:

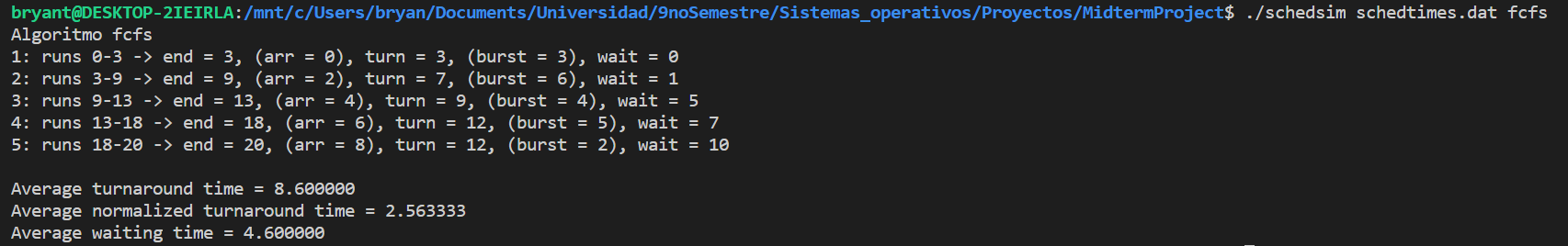


Ilustración Simulación FCFS

Salida con algoritmo JSF:

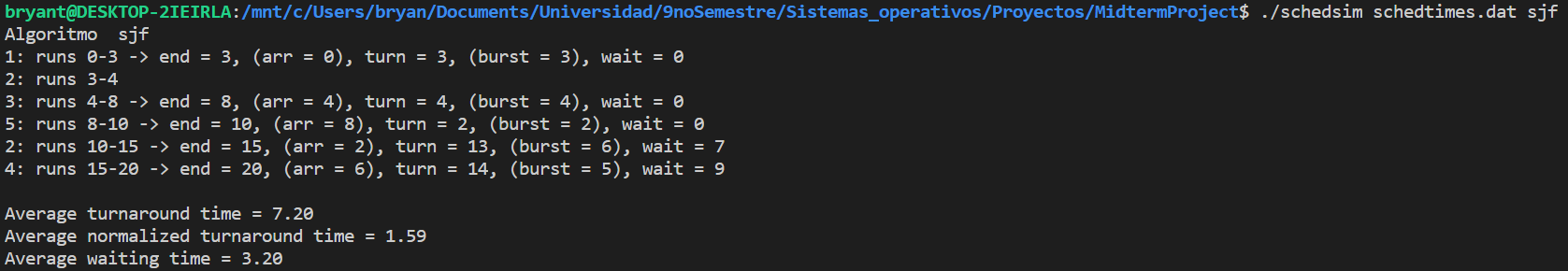


Ilustración Simulación JSF

Salida con algoritmo RR y q = 1:

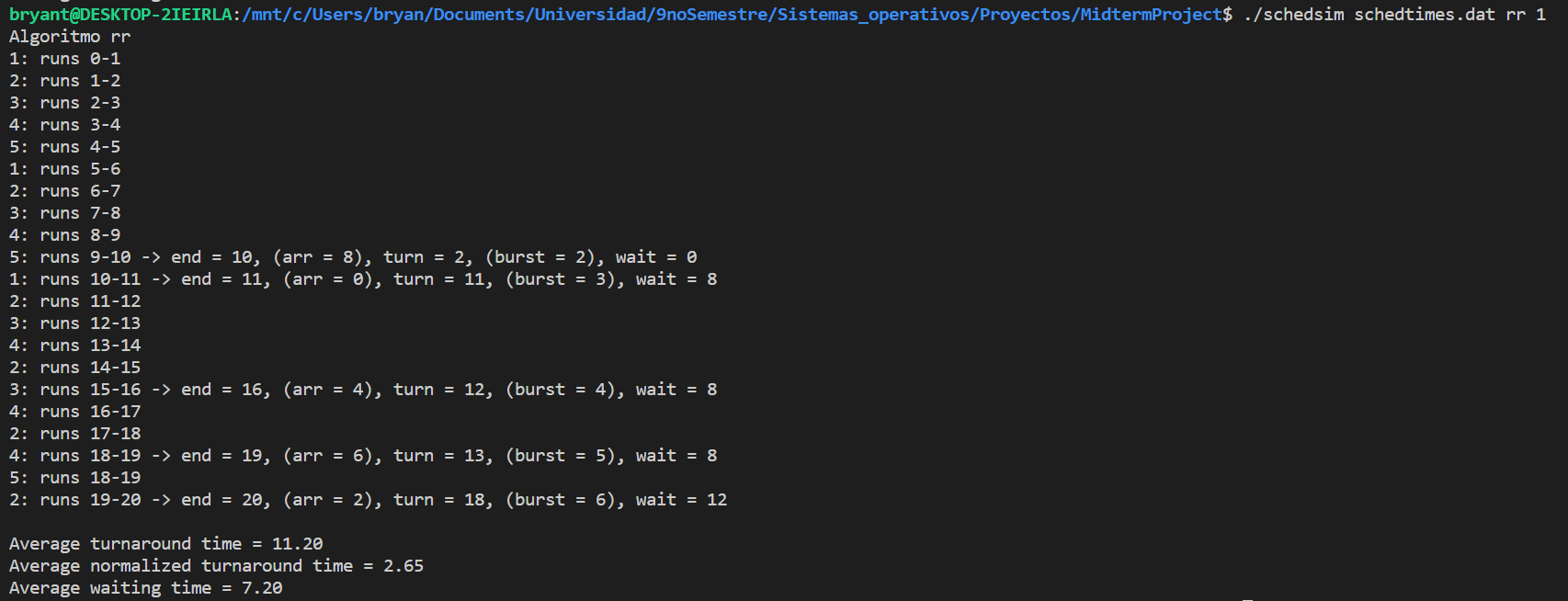


Ilustración Simulación RR

./schedsim -h

* Permite pedir ayuda al programa, este genera una salida de como deberíamos ejecutar el programa schedsim.

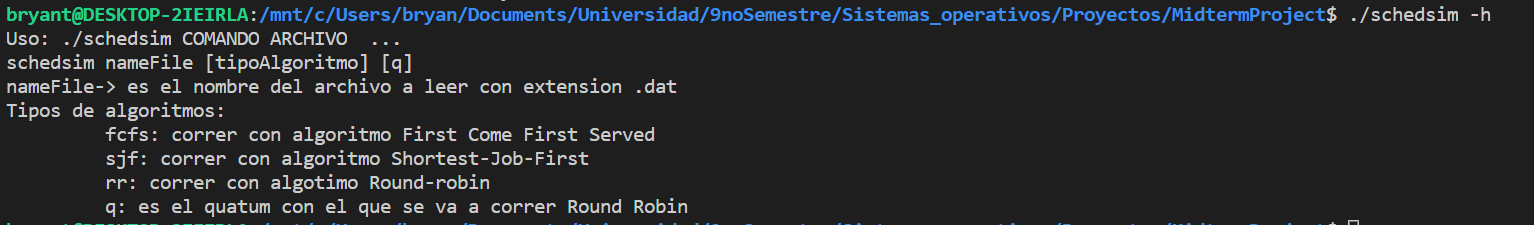


Ilustración Ayuda para Schedsim

## Schedgen

Para ejecutar este programa se realiza de la siguiente manera:

./schedgen N pa pb nameFile.dat

* N-> es el número de procesos que se desean generar.
* pa-> probabilidad para generar un valor para arrive time en un proceso.
* pb-> probabilidad para generar un valor para burst time en un proceso.
* nameFile-> es el nombre del archivo con que se va a crear para guardar la generacion de los procesos.

Salida del programa generador de procesos:



Ilustración Generador de procesos

## Schedstats

Para ejecutar este programa se realiza de la siguiente manera:

./schedstats nameFile.dat

* nameFile-> es el nombre del archivo de donde se van a leer los procesos, puede usarse cualquier extenxion que tenga el archivo de lectura tal como txt.

Salida del programa generador de archivos schedturns, schednturns y schedwaits:

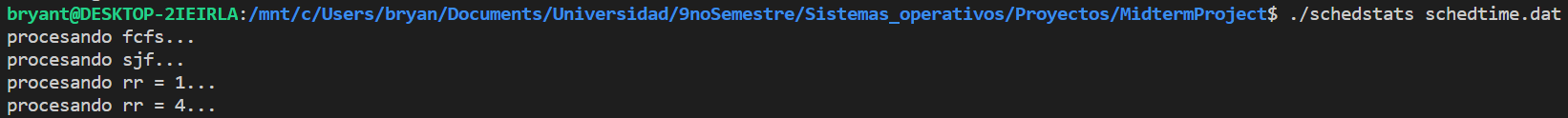


Ilustración Salida de 3 archivos con burst promediados

# ESTRUCTURA DEL PROYECTO

## Archivos de cabecera

El proyecto MidTerm esta conformado por archivos de caberas “.h” tales como:

* **funcionesShedsim.h:** Este archivo de cabecera contiene las funciones prototipo como:
* *help()* que me permite obtener una ayuda de parte del programa, donde dará indicaciones.
* *validacionArchivo()* permite identificar si un archivo existe o no.
* *cargarProcesos()* permite cargar los procesos que están en un archivo en una estructura malloc.
* *func\_fcfs()* se encarga de simular el algoritmo first come first served con los procesos cargados en la estructura.
* *func\_sjf()* se encarga de simular el algoritmo shortets Job First con los procesos cargados en la estructura.
* *func\_rr()* se encarga de simular el algoritmo Round Robin con los procesos cargados en la estructura.

Los parámetros como nombre indican nombre del archivo a cargarse los procesos, modo para indicar si se ejecutaran los algoritmos en modo “o” oculto para ejecutar schedstats y “d” detallado para ejecutar schedsim.

* *func\_max\_min()* se encarga de encontrar el burst time máximo y mínimo para poder agrupar los procesos en otra función.
* *Generar\_promedio\_rafaga()* se encarga agrupar los burst time, acumular los turn around time, waiting time, normalized turn around time y hacer un contador de burst time, al final generar los 3 archivos requeridos schednturns.dat, schedturns.dat y schedwaits.dat.
* *clonarProcesos()* se encarga de copiar una estructura de procesos en otra, esto permite trabajar sin colas procesos.

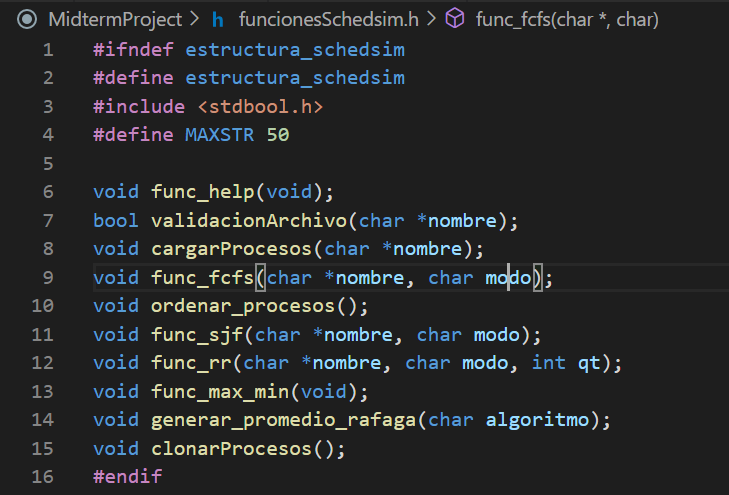


Ilustración Cabecera Schedsim

* **funcionesSchedgen.h:** Este archivo de cabecera contiene las funciones prototipos como *help()* que me permite obtener una ayuda de parte del programa, donde dará indicaciones y *generar\_procesos(params..)* me permitirá generar un archivo con procesos a partir de ciertos parámetros.

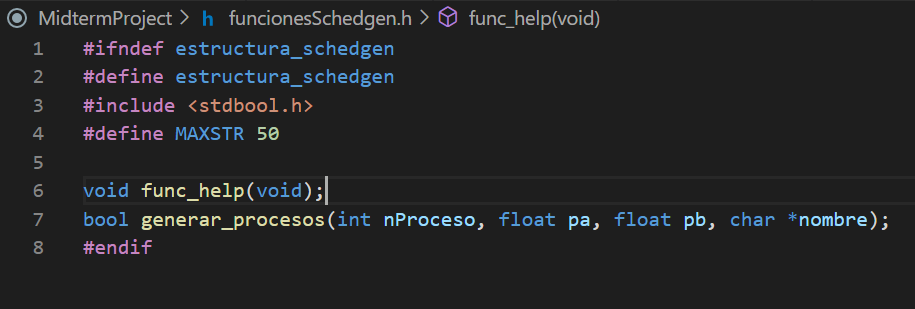


Ilustración Cabecera Schedgen

* **funcionesSchedstats.h:** Este archivo de cabecera contiene la función prototipo como *help()* que me permite obtener una ayuda de parte del programa, donde dará indicaciones.

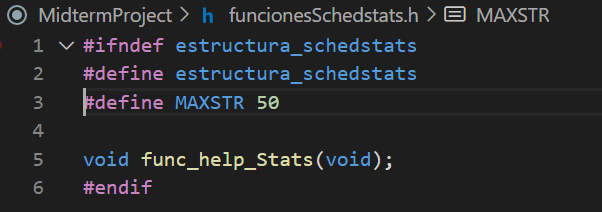


Ilustración Cabecera Schedstats

## Archivos de implementaciones de funciones prototipos

Además, el proyecto está conformado por archivos donde se implementan las funciones prototipos especificadas en los archivos de cabeceras “.h”, a continuación, una breve explicación:

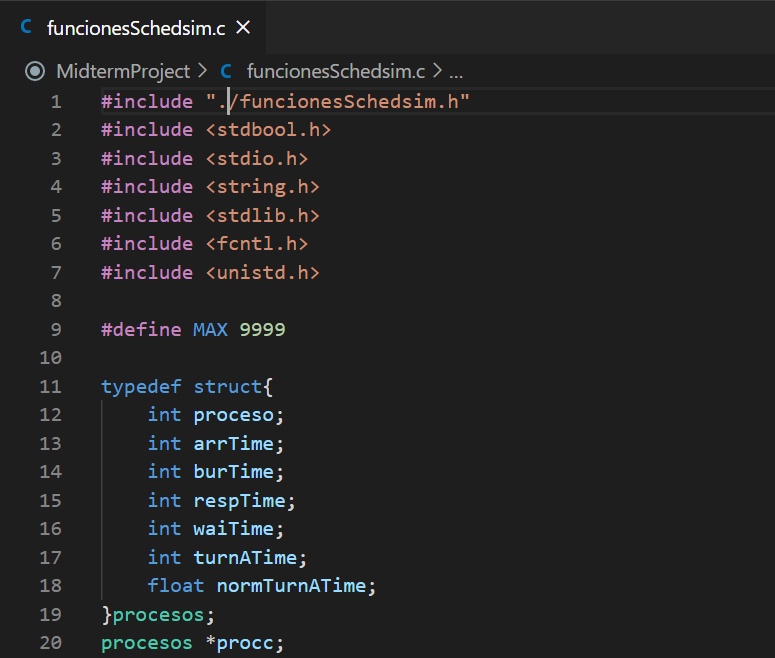
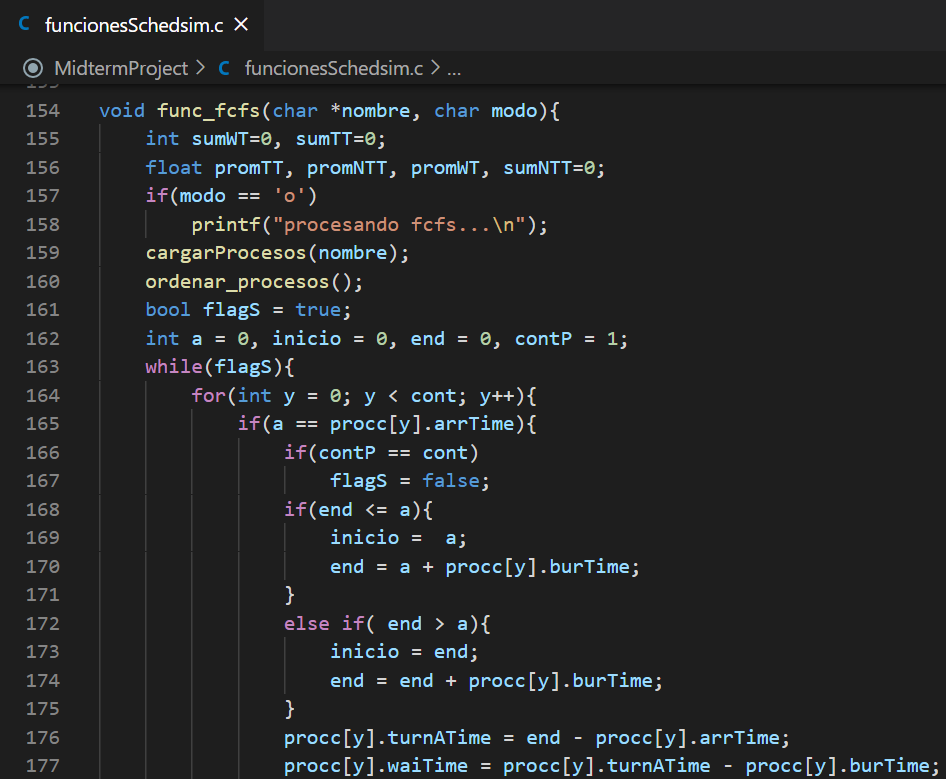
* **funcionesShedsim.c:**

Ilustración Implementación de Schedsim y parte de Schedstats

* **funcionesShedgen.c:**

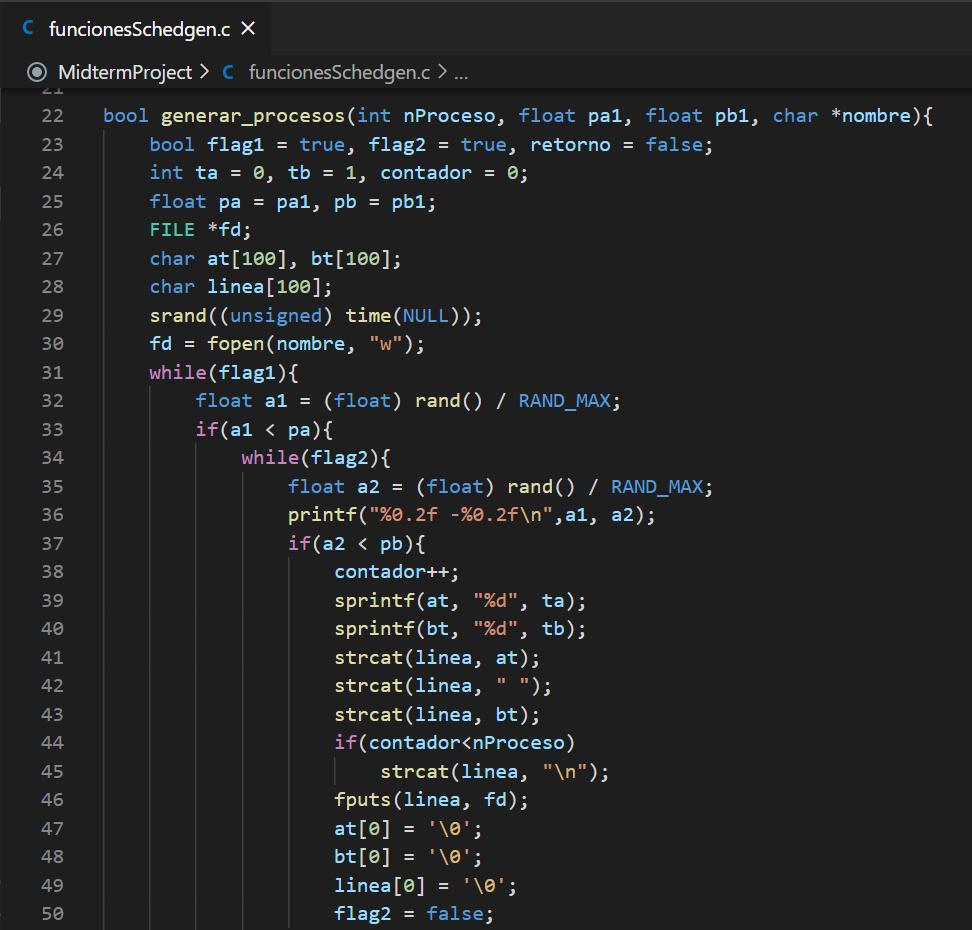


Ilustración Implementación para funciones Schedgen

* **funcionesShedstats.c:**

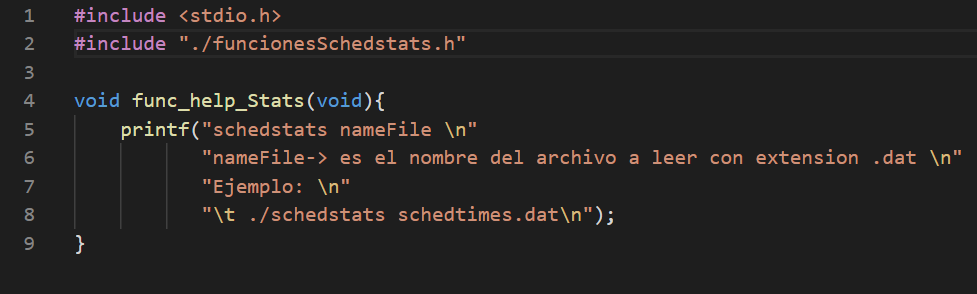
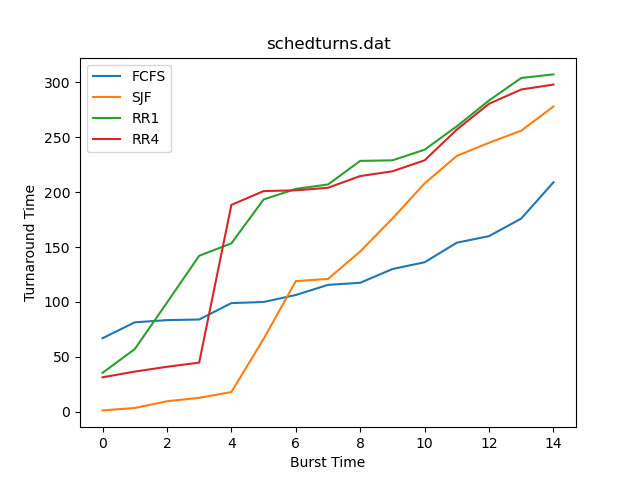




Ilustración Implementación de funciones Schedstats

# GENERADOR DE GRÁFICOS CON PYTHON

El archive llamado generadorGraficosStats necesita la **librería matplotlib** para ejecutarse y generar los gráficos.

Al ejecutarse se tendrán gráficos de salida como:

