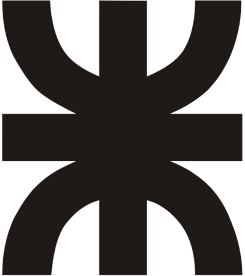
**UNIVERSIDAD TECNOLOGICA NACIONAL**



**FACULTAD REGIONAL RESISTENCIA**

**SISTEMAS OPERATIVOS**

**Proyecto:**

**Simulador para la Administración de Memoria y Planificación de Procesos**

**Equipo docente:**

* Ing. Cuenca Plestch, Liliana
* Ing. Gramajo, Sergio
* Ing. Ristoff, Alberto
* Ing. Roa, Jorge Alejandro

**Integrantes del equipo:**

* Arias, Leandro Exequiel
* Florentín, Cristian Alexis
* Imfeld, Facundo Nicolas
* Matto, Pablo José
* Nasir, Khalil Abdul

**Indice**

**Introducción**

**Consideraciones**

**Algoritmos de asignación de Memoria y de planificación de procesos**

**Algoritmos de asignación de Memoria**

**Algoritmos de planificación de procesos**

**Códigos**

**Instructivo de uso**

**Introducción**

El objetivo del desarrollo del siguiente simulador es permitir visualizar los aspectos de la planificación de procesos a corto plazo, empleando los algoritmos de planificación FCFS, Prioridades, Round Robin, Colas Multinivel. Además, se representará la gestión de la memoria con particiones Fijas, aplicando algoritmos First-Fit y Best-Fit; y particiones Variables, utilizando algoritmos Best-Fit y Worst-Fit para un esquema de un solo procesador, mostrando el ciclo de vida completo de un proceso desde su ingreso al sistema hasta su finalización. El objetivo de este documento es brindar información acerca del funcionamiento del simulador, en el mismo se detallarán las consideraciones que se tienen que tener en cuenta a la hora de cargar los datos requeridos para la simulación. Se detallará y adjuntará el funcionamiento de los diferentes algoritmos de planificación y algoritmos de intercambio.

**Consideraciones**

Memoria:

* Tamaño Total: Se definirá una memoria mínima de 50kb.
* Tamaño ocupado para el Sistema Operativo: Este simulador reservará siempre el 10% de la memoria total para el sistema operativo, esta memoria no estará disponible para los procesos.
* Particiones Fijas:
* Algoritmos de intercambio: First Fit, Best Fit. Estos algoritmos se encargarán de administrar la asignación de particiones a los procesos.
* Cantidad de particiones: Se establece un mínimo de 2 particiones
* Al seleccionar el tipo de partición "fija" se deberá ingresar el tamaño de la partición separando por "-", cada tamaño separado por "-" nos indica una partición.
* Tamaño de cada partición: Se puede optar por crear particiones en las cuales todas tengan el mismo tamaño, o definir particiones con tamaños variados, en ambos casos la suma de todos los tamaños no deberá superar el tamaño total de memoria. En caso de que se supere el tamaño máximo de memoria no podrá continuar con la simulación y deberá reiniciar el simulador
* Particiones Variables
* Algoritmos de intercambio: Best Fit, Worst Fit

Algoritmos de Planificación:

Estos algoritmos se encargarán de organizar los procesos de manera eficiente en cuanto al uso del procesador.

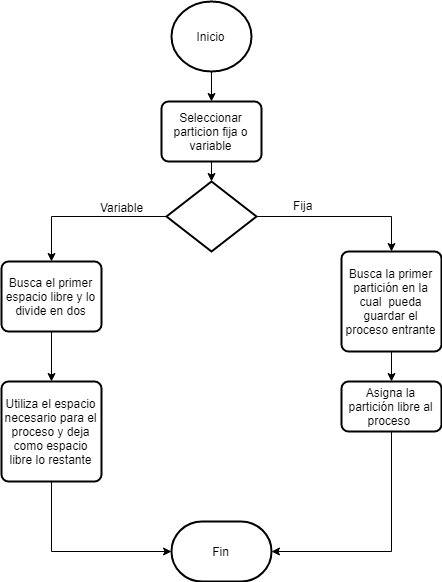
* FCFS (First Come First Served)
* Por prioridades: Cuando se seleccione este algoritmo, se habilitará en la pantalla de carga un cuadro en el cual podremos indicar la prioridad del proceso
* Round Robin: Cuando se seleccione este algoritmo, se habilitará en la pantalla de carga un cuadro en el cual podremos indicar el quantum con el cual se desea trabajar.
* Colas Multinivel: Se utilizarán tres niveles de colas, el primer nivel con Round Robin de quantum = 2, donde se tratarán procesos de prioridad 0 a 5 (prioridad más alta); el segundo nivel con Round Robin de quantum = 3, donde se tratan procesos con prioridad de 6 a 10 (prioridad media); y el tercer nivel con FCFS donde se tratarán procesos con prioridad mayor a 10 (prioridad más baja).

Procesos

* Cantidad de procesos: Se podrá cargar un mínimo de 2 procesos
* Tamaño de cada proceso: Se debe considerar que el tamaño de proceso no sea superior al tamaño de partición en caso de particiones fijas. No se debe permitir el ingreso de un proceso de tamaño superior al total de memoria disponible.
* Tiempo de arribo: Instante en el cual el proceso ingresa a la cola de nuevos.
* Tiempo de irrupción
* Ráfaga de CPU: Tiempo que el proceso hará uso del procesador, considerando que si el proceso posee ráfaga de E/S se especificará una segunda ráfaga de CPU.
* Ráfaga de E/S: Tiempo que el proceso atiende E/S.
* Política adoptada para la ejecución de los procesos en caso de que llegaran en el mismo instante: Si sucede que dos procesos llegaran en el mismo instante de tiempo, ya sea que se tratase de un nuevo proceso que arriba o de un proceso que ha finalizado su E/S, se tomará como prioritario en la ejecución de la CPU a aquel que ha arribado en dicho instante, quedando al aguardo de su finalización el proceso que finalizó su E/S. Éste quedará encolado hasta que haya finalizado sus quantum de tiempo el proceso arribado.
* Prioridad de los procesos: Se toma la prioridad de los procesos considerando de forma ascendente. A medida que los procesos aumenten en su valor, baja su prioridad. Ejemplo, un proceso con prioridad 0 tiene más prioridad que uno que tenga prioridad 7.

**Importante: Cualquier ingreso indebido de datos provocará el bloqueo del simulador.**

**Algoritmos de asignación de Memoria y de planificación de procesos**



**Algoritmos de asignación de Memoria**

En el presente simulador, la memoria se podrá particionar de dos formas: particiones fijas o particiones variables. Las particiones fijas podrán ser con todas sus particiones del mismo tamaño, o bien ser de diferentes tamaños.

Algoritmo First Fit

Este algoritmo consiste en recorrer la memoria hasta encontrar una partición que sea lo bastante grande para asignarse al proceso entrante. En el caso de particiones variables, se le asigna el primer espacio libre que esté disponible (en el caso de ser el primer proceso, en memoria tendremos todo el espacio disponible excepto el espacio asignado al sistema operativo) y lo divide en dos; una de esas partes será todo el espacio requerido por el proceso y la otra quedara como espacio libre, disponible para otros procesos que puedan hacer uso del mismo.

En ambos casos, tanto para particiones fijas como particiones variables, se presentan inconvenientes; en particiones fijas puede ocurrir que el tamaño del proceso sea muy inferior al tamaño de la partición, produciendo fragmentación interna (espacio que no se podrá liberar ni usar hasta que finalice el proceso o bien termine la simulación. En el caso de las particiones variables puede ocurrir que entre dos procesos quede espacio sin asignar (fragmentación externa) puesto que ningún proceso que ingrese a memoria pueda ser asignado, esto podría optimizarse mediante un proceso de compactación de memoria.

Algoritmo Best Fit

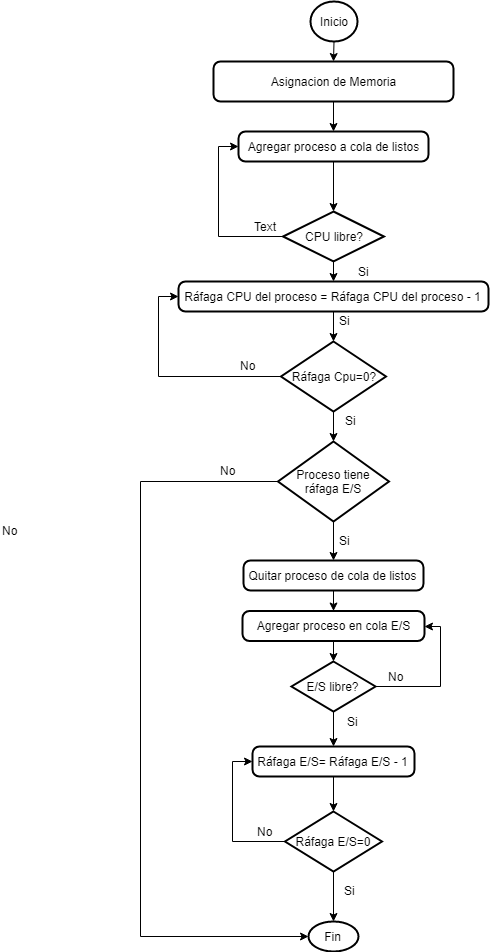
Este algoritmo busca en toda la memoria, de principio a fin y toma la partición más pequeña que se ajuste más al tamaño del proceso. El algoritmo Best Fit es más lento que el First Fit, ya que debe buscar en toda la memoria cada vez que se le llama y comparar cada partición con el tamaño del proceso solicitado.

Algoritmo Worst Fit

Este algoritmo debe recorrer toda la memoria buscando el espacio libre mas grande disponible y se la asigna al actual proceso, con esto se busca que el espacio libre restante pueda ser utilizado por otro proceso.

**Algoritmo FCFS**

Este algoritmo consiste en asignar la CPU al primer proceso que la solicite. La implementación de la política FCFS se puede gestionar mediante una cola FIFO, pues los procesos a medida que van ingresando a la cola de listos van quedando encolados en el orden en el cual fueron llegando. Es el algoritmo de planificación de CPU más simple, pero no así el mas efectivo puesto a que presenta inconvenientes tales como la situación en la que un proceso se este ejecutando, no podrá ingresarse ningún otro proceso hasta haber finalizado en su totalidad el proceso que esté haciendo uso de la CPU, es un algoritmo no apropiativo, en el cual se produce un tiempo de espera en cola de listos muy extenso e ineficiente, ya que los procesos en la mayoría de las veces no llegarán de forma ordenada y los que tengan tiempo de ejecución muy largos harán que el tiempo de respuesta de procesos cortos se vuelva muy prolongada.



**Algoritmo Por Prioridades**

Este algoritmo basa su funcionamiento en asignarle prioridades a los procesos para competir por el uso del procesador. Las prioridades pueden definirse interna o externamente. Las prioridades definidas internamente utilizan algún valor como puede ser el tiempo de CPU que demanda un proceso; las prioridades definidas externamente se establecen en función de criterios externos al sistema operativo, como ser la importancia del proceso.

La planificación de procesos por prioridades puede ser apropiativa, en la cual si llega un nuevo proceso a la cola de listos y tiene una prioridad mayor a la del proceso que está haciendo uso de la CPU, se expulsa al proceso en ejecución y se le asigna el recurso al nuevo proceso; también puede ser cooperativa, en este caso el algoritmo colocara al nuevo proceso al comienzo de la cola de listos, evitando la expulsión de un proceso en ejecución. Un problema importante de este algoritmo es que se puede producir un bloqueo indefinido o muerte por inanición, es decir, que se atiendan procesos con prioridades altas y nunca llegue a ejecutarse procesos con bajas prioridades. Una solución al problema del bloqueo indefinido de los procesos de baja prioridad consiste en aplicar mecanismos de envejecimiento, que consiste en luego de un tiempo ir aumentando la prioridad de los procesos que lleven mucho tiempo en espera por hacer uso de la CPU.

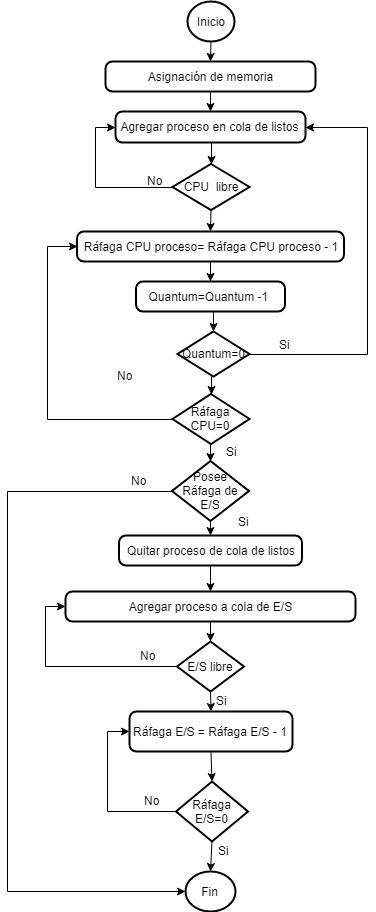


**Algoritmo Round Robin**

También conocido como planificación cíclica o turno rotatorio, este algoritmo basa su funcionamiento en interrupciones de reloj cada cierto intervalo de tiempo. Cuando sucede la interrupción el proceso actual en ejecución se sitúa en la cola de listos, y se selecciona el siguiente trabajo según la política FCFS. Este método permite seleccionar todos los elementos de la carga de trabajo de manera equitativa, mediante un recorrido de la cola de listos desde principio a fin y empezando nuevamente por el primer proceso de la cola.

Una desventaja de la planificación Round Robin es que trata de forma desigual a los procesos limitados por el procesador y a los procesos limitados por la E/S. Generalmente, un proceso limitado por la E/S tiene ráfagas de procesador más cortas (cantidad de tiempo de ejecución utilizada entre operaciones de E/S) que los procesos

limitados por el procesador. Si hay una mezcla de los dos tipos de procesos, sucederá lo siguiente: un proceso limitado por la E/S utiliza el procesador durante un periodo corto y luego se bloquea; espera a que complete la operación de E/S y a continuación se une a la cola de listos. Por otra parte, un proceso limitado por el procesador generalmente utiliza su quantum de tiempo completo mientras ejecuta e inmediatamente vuelve a la cola de listos o finaliza. De esta forma, los procesos limitados por el procesador tienden a recibir un quantum no equitativo de tiempo de procesador, lo que conlleva un mal rendimiento de los procesos limitados por la E/S, uso ineficiente de los recursos de E/S y un incremento en la variación del tiempo de respuesta.



**Algoritmo de colas multi nivel sin Retroalimentación**

Este tipo de algoritmo, divide la cola de procesos preparados en varias colas distintas. Los procesos se asignan permanentemente a una cola, generalmente en función de alguna propiedad del proceso, como por ejemplo el tamaño de memoria, la prioridad del proceso o el tipo de procesos. Cada cola tiene su propio algoritmo de planificación.

Para este simulador, se ha adoptado la siguiente condificación de colas multinivel sin retroalimentación:

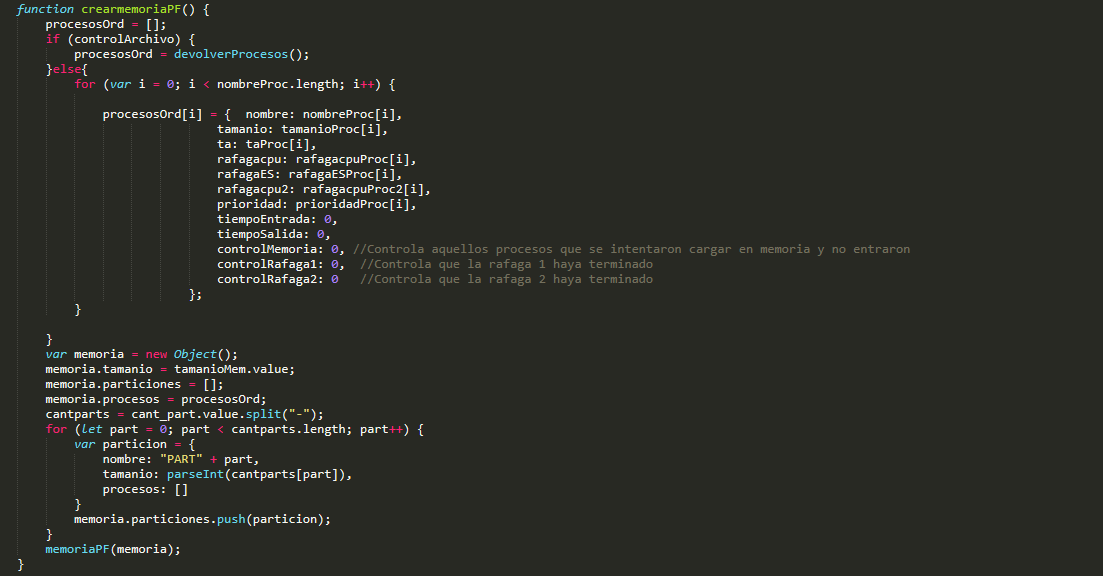
Cantidad de colas: 3.

* Cola Nivel 0: La más prioritaria.
  + Algoritmo de planificación: Algoritmo Round Robin con quantum de 2 unidades de tiempo;
  + Prioridades en los procesos: En este nivel se ejecutarán aquellos procesos con prioridad que van desde 0 a 5;
* Cola Nivel 1: Media prioridad.
  + Algoritmo de planificación: Algoritmo Round Robin con quantum de 3 unidades de tiempo;
  + Prioridad en los procesos: En este nivel se ejecutarán aquellos procesos con prioridad que van desde 6 a 10;
* Cola Nivel 2: Menor prioridad.
  + Algoritmo de planificación: FCFS;
  + Prioridad en los procesos: Contendrá todos aquellos procesos con prioridades superiores a 10.

**Códigos**

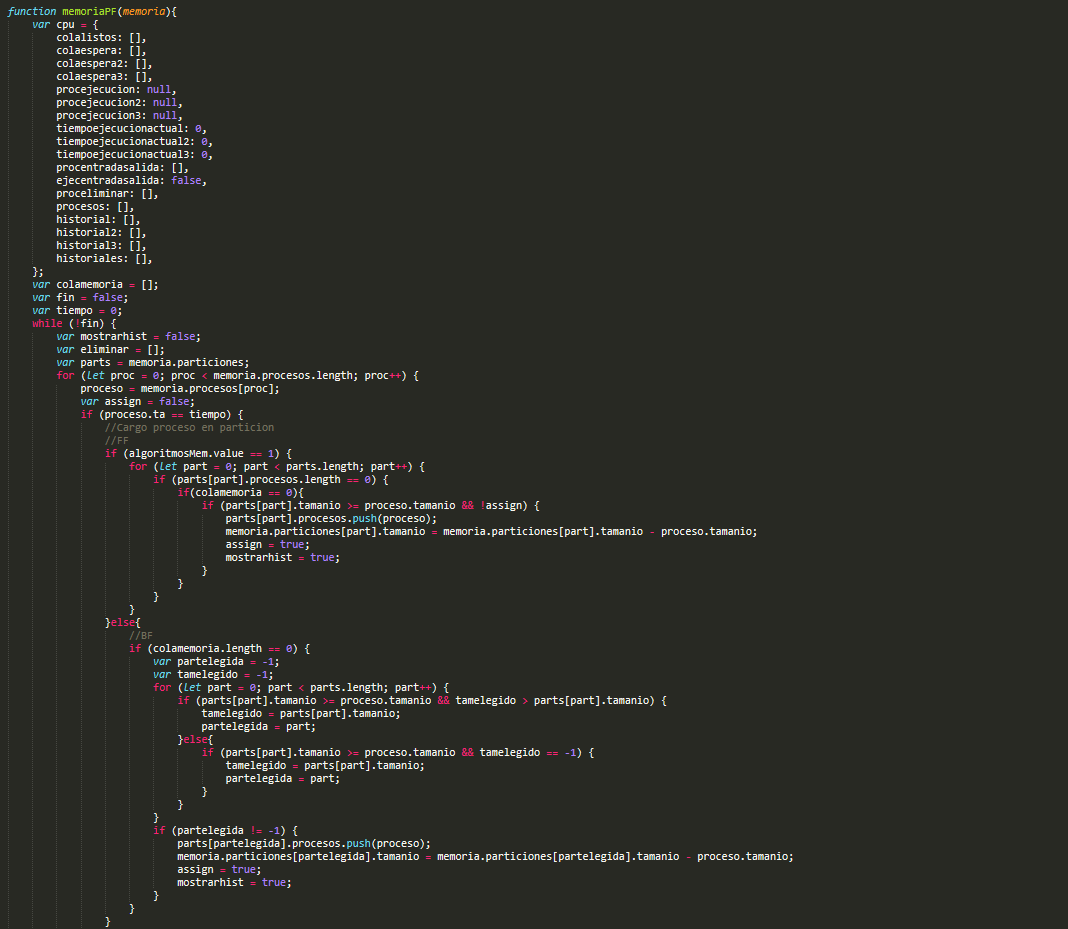
**Particiones Fijas**

Function crearmemoriaPF()



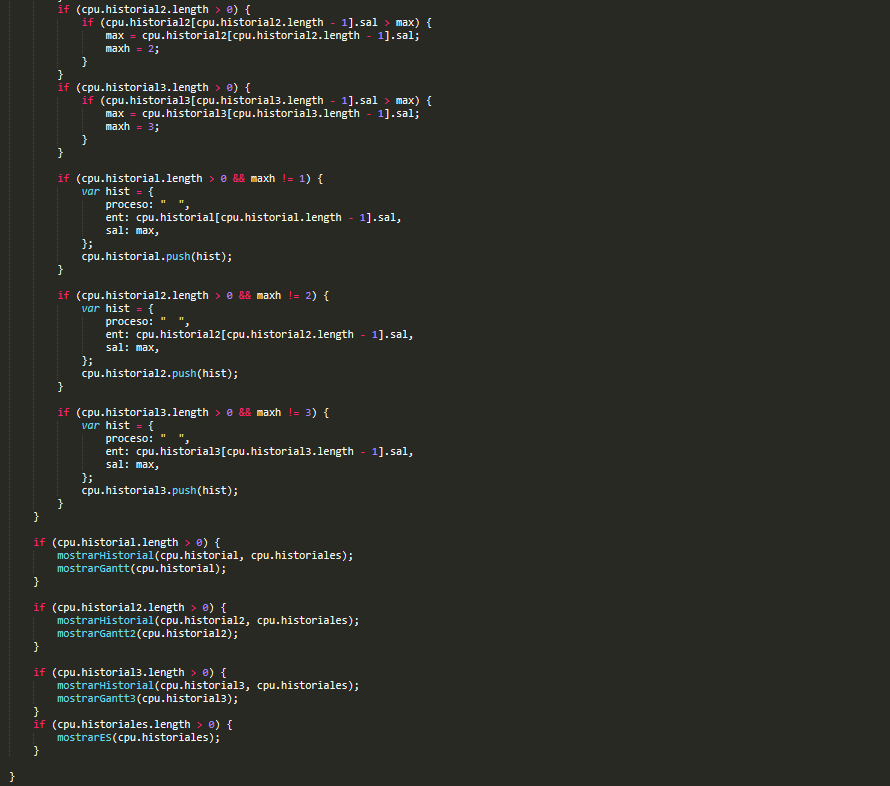
Genera las particiones que hayamos ingresado, comprobando que cada “-” ingresado sea una partición.

Function memoriaPF(memoria)









Esta función se encarga de atender la lógica de asignación de memoria, ya sea first fit o best fit. También realiza el control de qué procesos pueden hacer uso de la memoria dependiendo de las particiones que se encuentren libres y si los tamaños de partición permiten alojar un proceso entrante. En este bloque se toman los datos para realizar las representaciones gráficas y los historiales de memoria y de proceso.

Function CPU(cpu,tiempo)

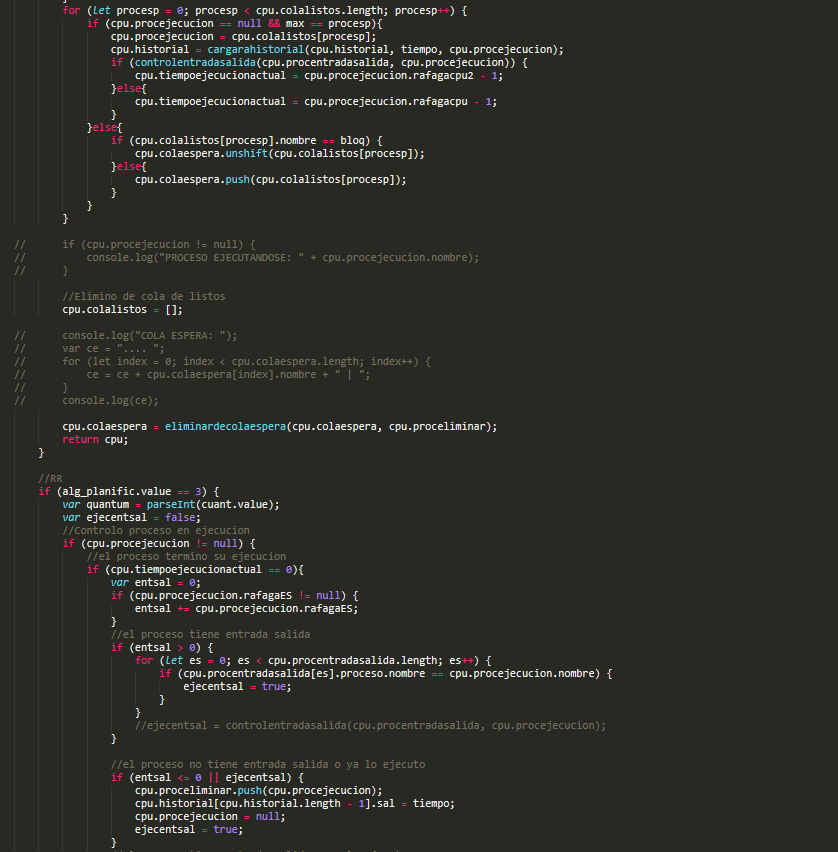


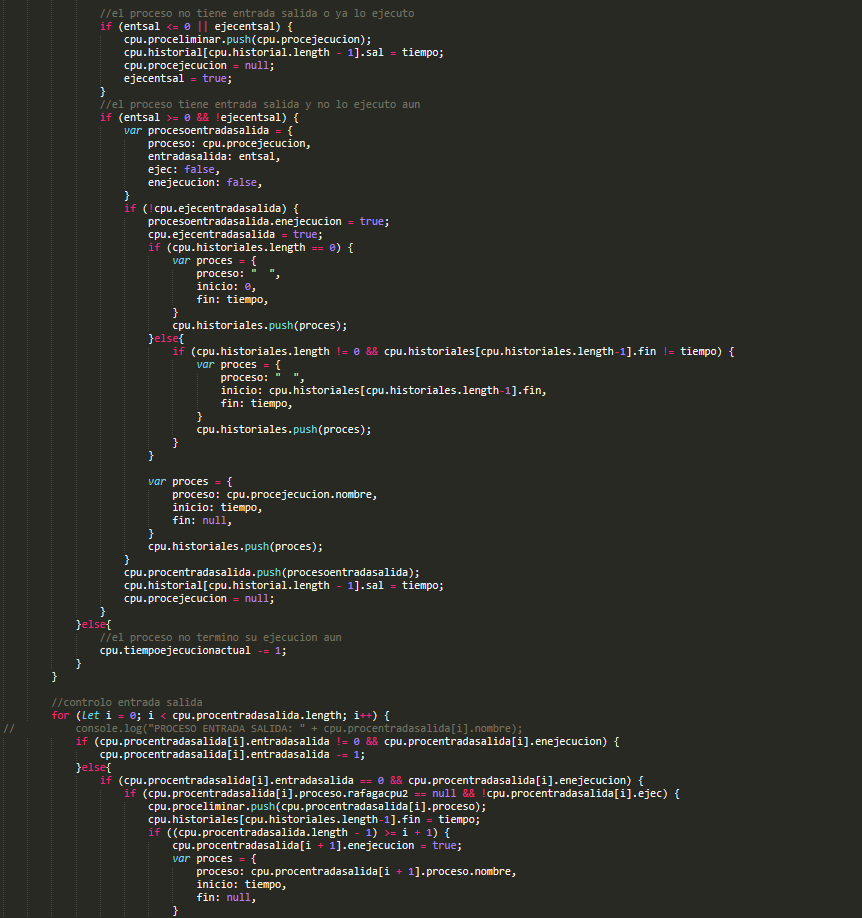


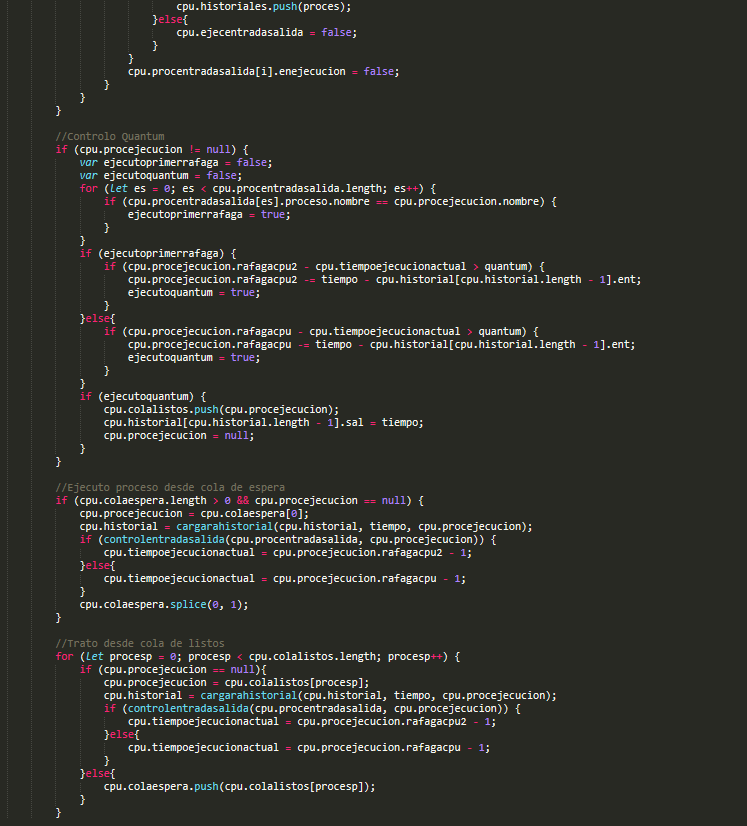




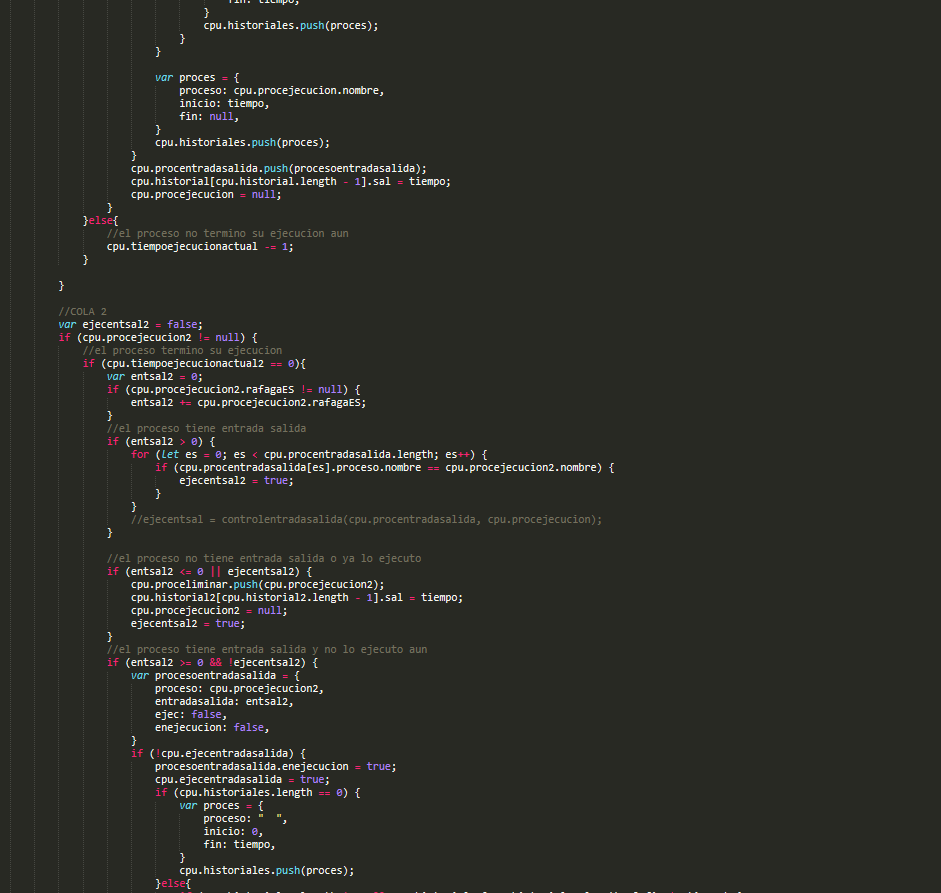




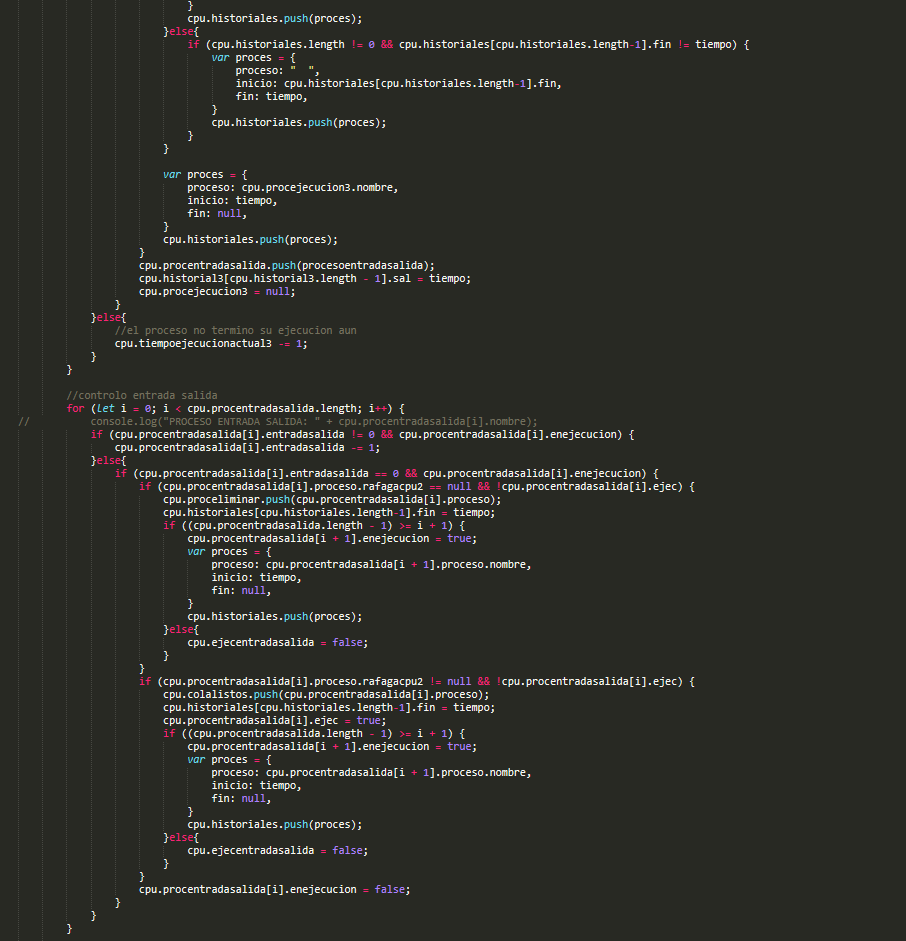




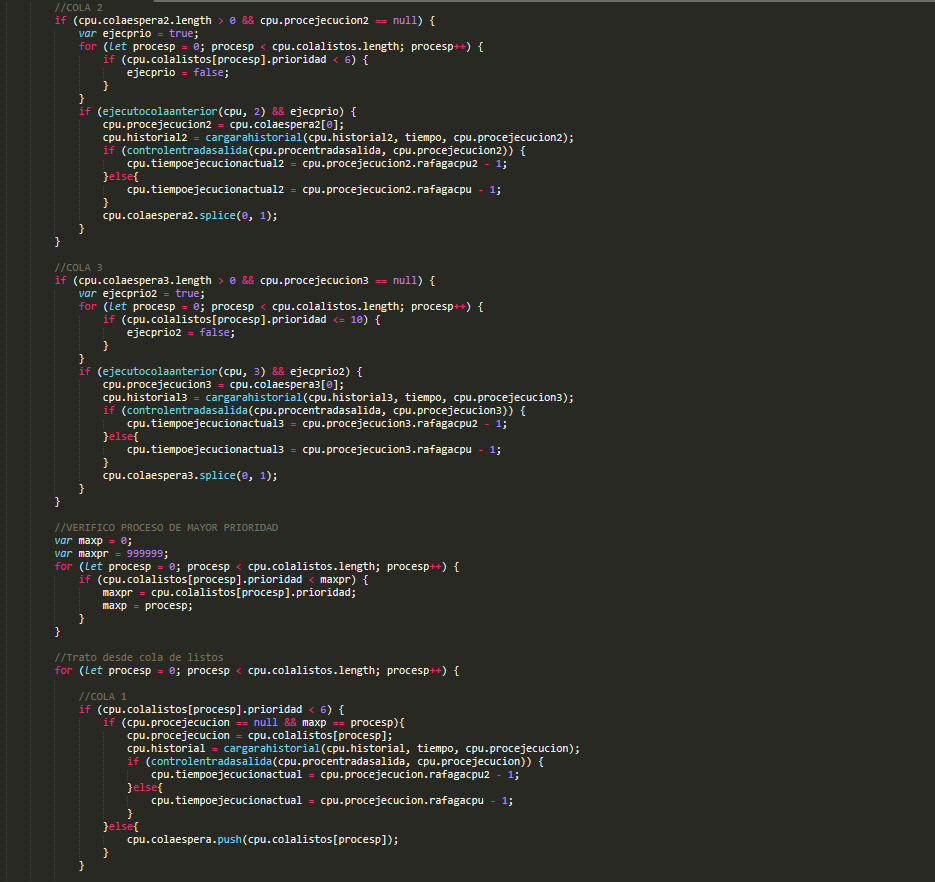








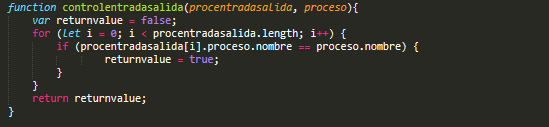




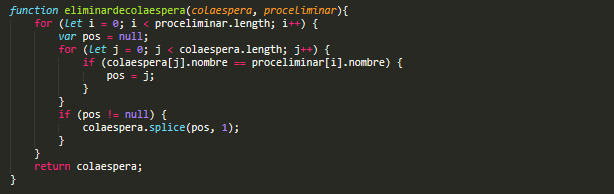


En este bloque de código se tratará el manejo y asignación de la CPU a los procesos según el algoritmo seleccionado. Se hacen verificaciones sobre las colas de espera y segun el planificador elegido se ordenarán por tiempo de arribo y prioridades(en caso del planificador por prioridades o colas multinivel).

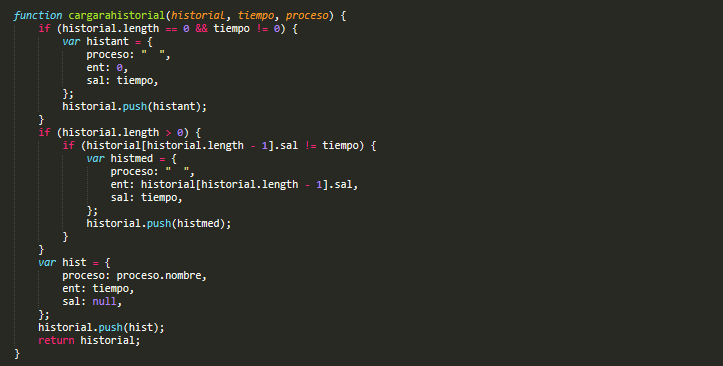
Function controlentradasalida(procentradasalida,proceso)



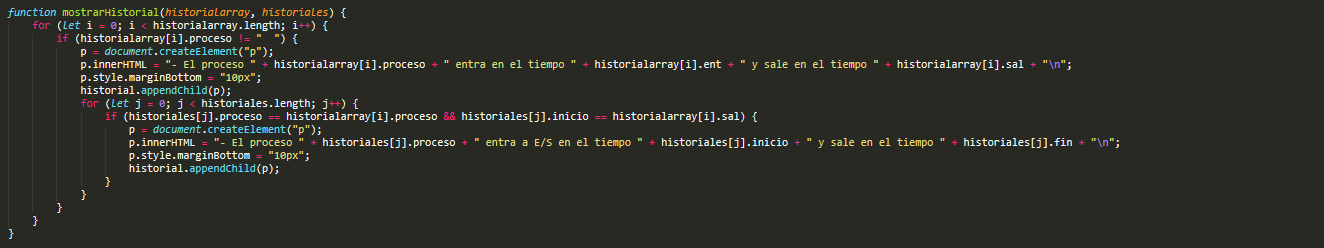
Function eliminardecolaespera(colaespera,proceliminar)



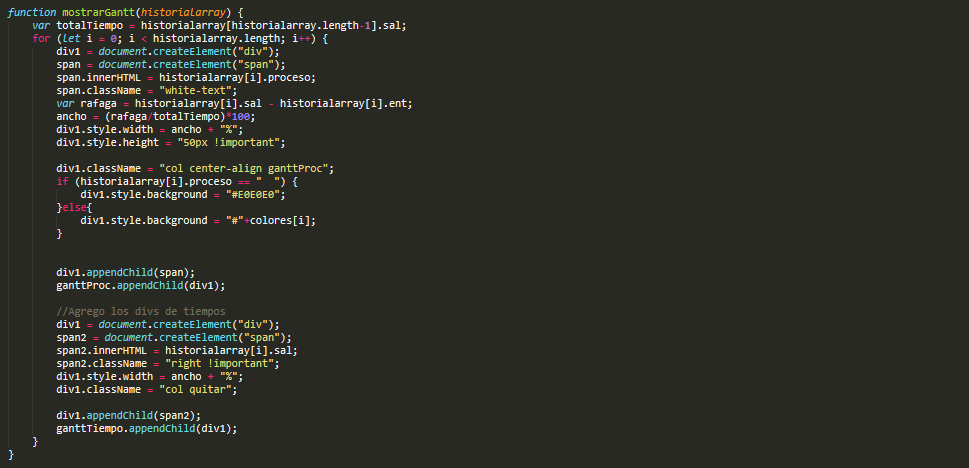
Function cargarahistorial(historial,tiempo,proceso)



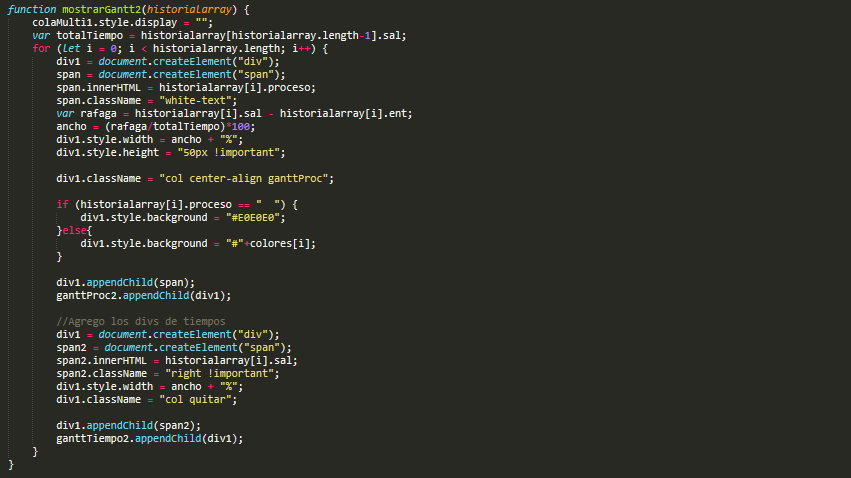
Function mostrarHistorial(historialarray)



Function mostrarGantt(historialarray)



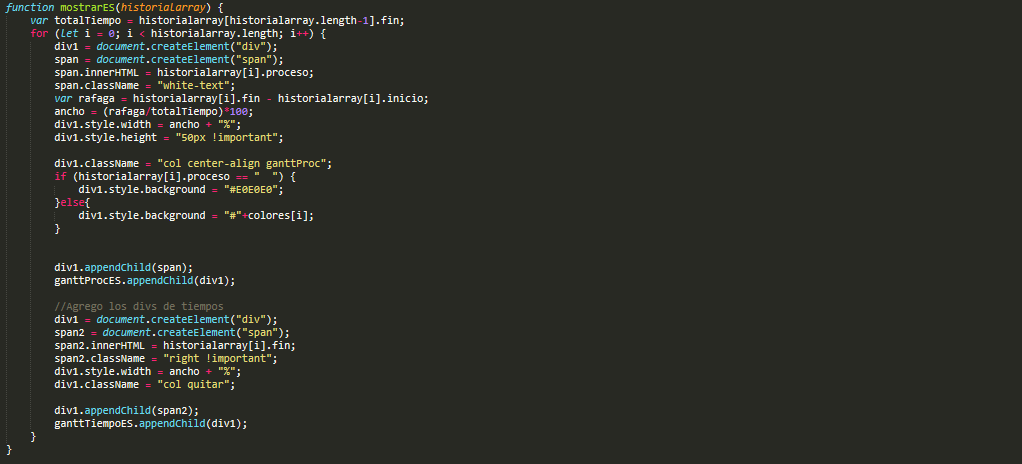
Function mostrarGantt2(historialarray)



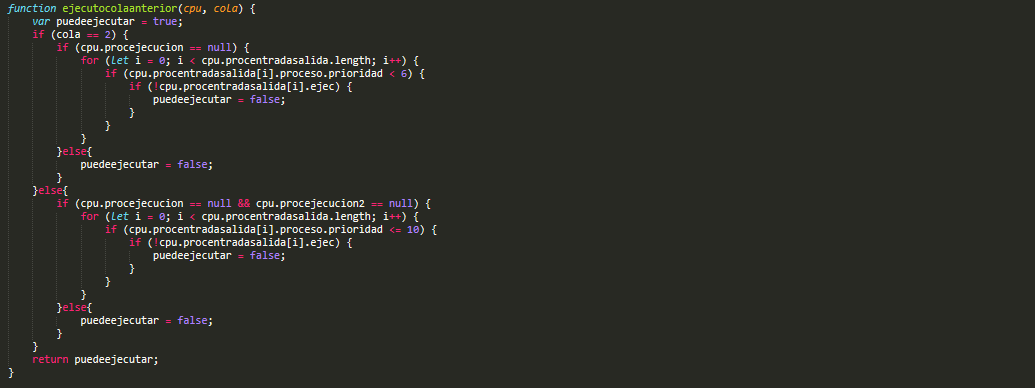
Function mostrarGantt3(historialarray)



Function mostrarES(historialarray)



Function ejecutocolaanterior(cpu,cola)



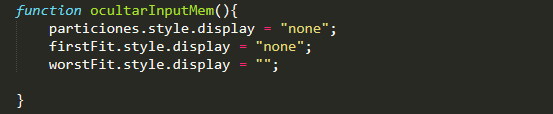
function mostrarMemoria(memoria, tiempo)



**PARTICIONES VARIABLES**

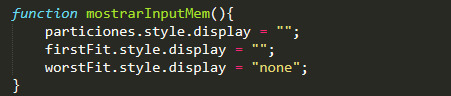
Funciones

function ocultarInputMem()



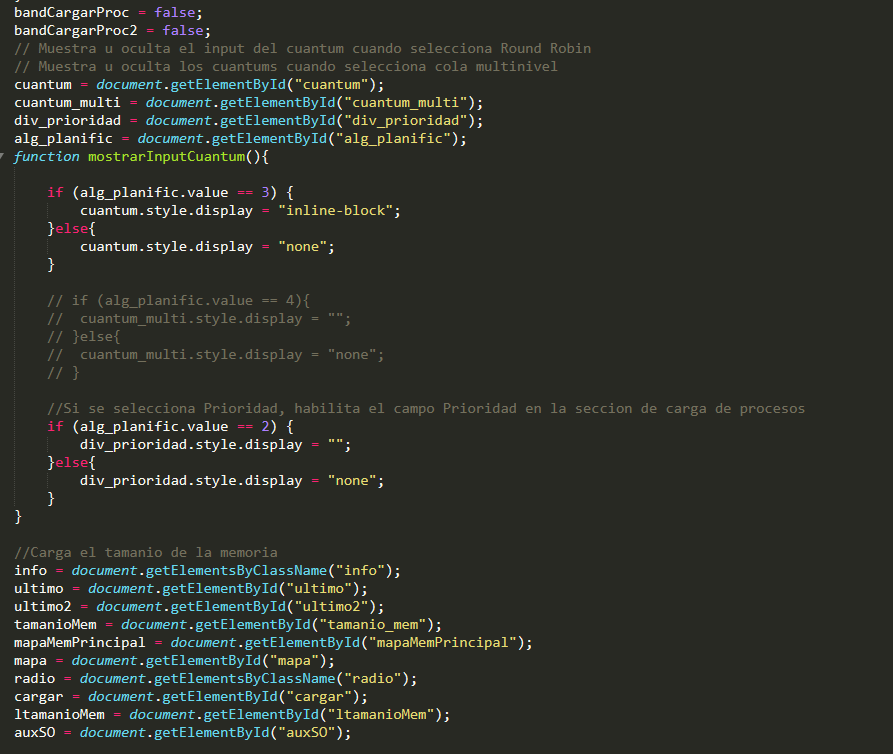
Administran la parte grafica;

function mostrarInputMem()



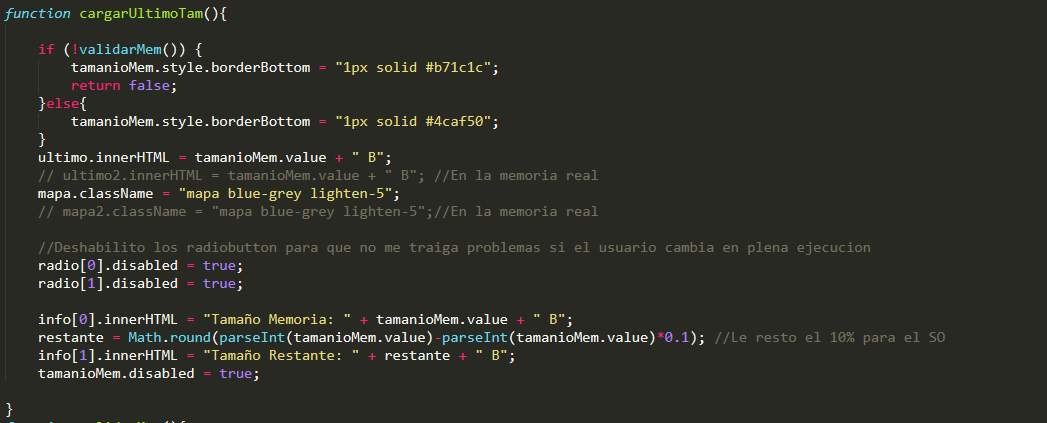
Administran la parte grafica;

function mostrarInputCuantum()



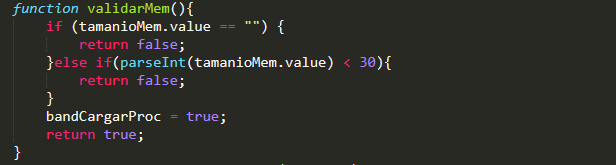
Despliega la opción para colocar quantum en alg RR

function cargarUltimoTam()



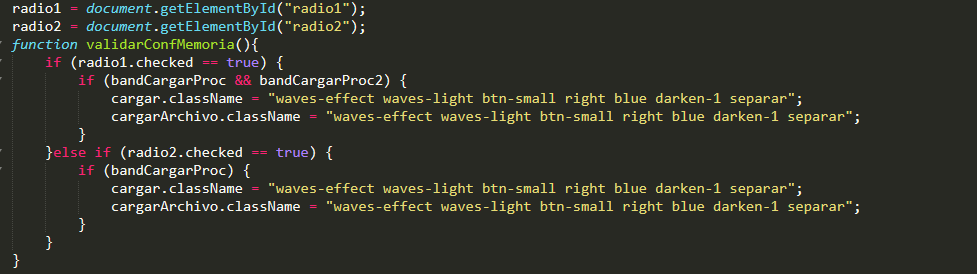
Verifica si hay espacio en memoria para seguir asignando;

function validarMem()



Utilizado para verificar tipo de particion utilizado. Buton1 se trata de PF, buton2 BF;

function validarConfMemoria()



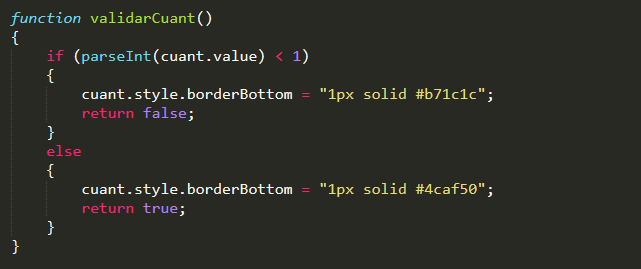
Valida la memoria ingresada.

function cargarProcesoMem()



Lo que esta funcion hace es que al llamar a la funcion cargarProcesosPlanificador(), se tome los datos del proceso, de acuerdo al valor del id se le antepone la letra “P” y se resta el tamaño total de memoria con el tamaño del proceso.

function validarCuant()

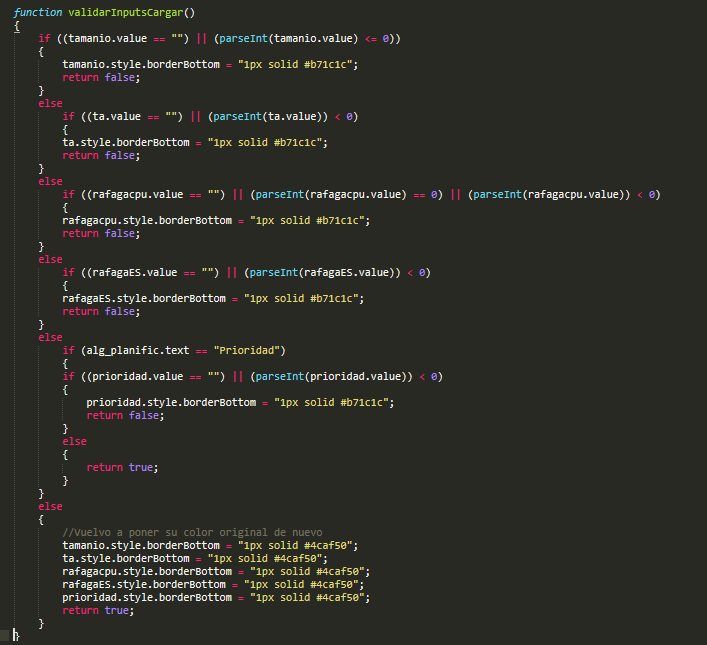


function cargarProcesosPlanificador()



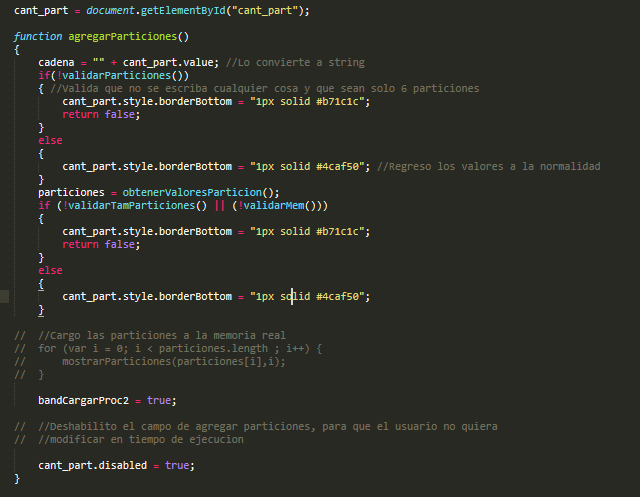
Esta función trabaja sobre el ingreso manual de los procesos. Por cada tipeo en los campos correspondientes, la función irá armando la tabla de procesos a utilizar.

function validarInputsCargar()



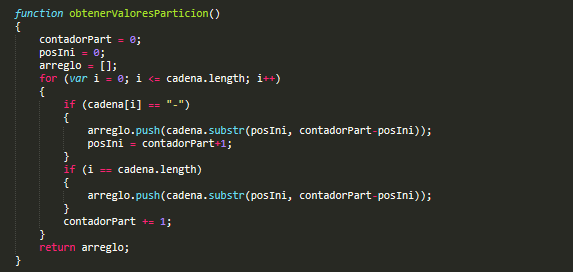
Comprueba que los campos ingresados sean correctos, de otro modo en los campos donde se valida un error, dicho campo se marca con una línea color rojo.

function agregarParticiones()



Hace una verificación de las particiones ingresadas. En caso de no corresponder, marca el campo con la línea roja.

function obtenerValoresParticion()



Esta función lo que hace es armar un arreglo con las particiones que se ingresa, cada una de estas separadas por el símbolo “-”.

function validarParticiones()



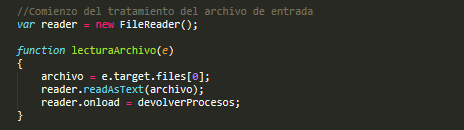
Lo que esta función hace es verificar que el símbolo “-” sea correctamente puesto para que se puedan procesar las particiones que se asignan. También corrobora que existan al menos dos particiones y menos de siete, ya que se ha adoptado como política que sólo se puedan ingresar hasta 6 particiones.

function validarTamParticiones()



Controla que la suma de las particiones no sea mayor que la memoria y que el valor de las particiones no sean menores a 5. Por último, actualiza el valor de la partición Se utilizan las variables globales definidas anteriormente tamanioMem y particiones y dos locales la cual una contiene el tamaño de la memoria (tamanioMem10) y otra que acumula el valor de cada particion (acumulador).

function lecturaArchivo(e)



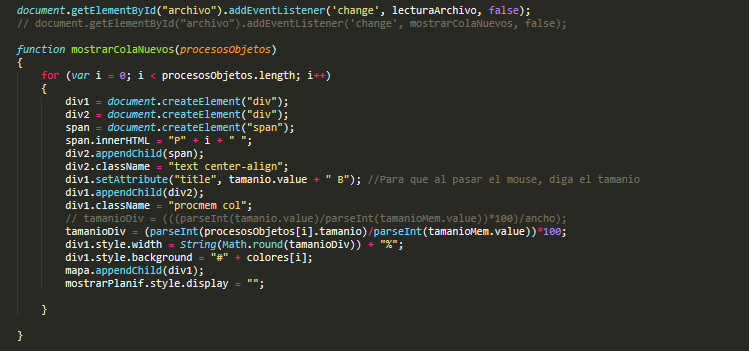
Función que permite leer el archivo a tomar para la carga de los procesos.

function devolverProcesos()



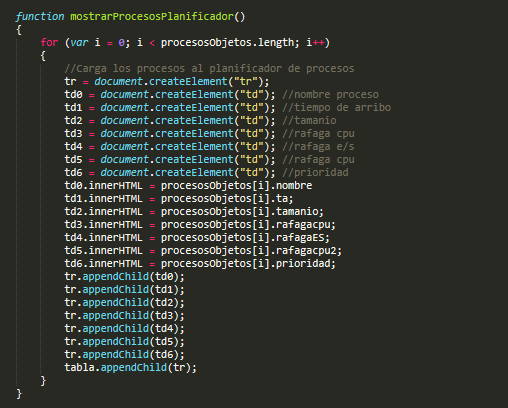
La función muestra por pantalla la carga de los procesos mediante archivo y también de forma gráfica la cola de nuevos.

function mostrarColaNuevos(procesosObjetos)



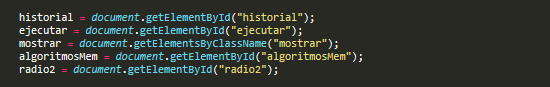
Realiza la el diagrama de la Cola de Nuevos.

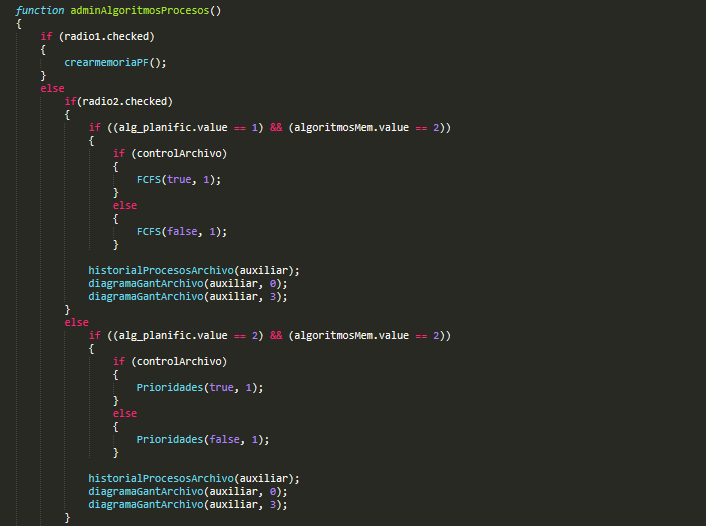
function mostrarProcesosPlanificador()



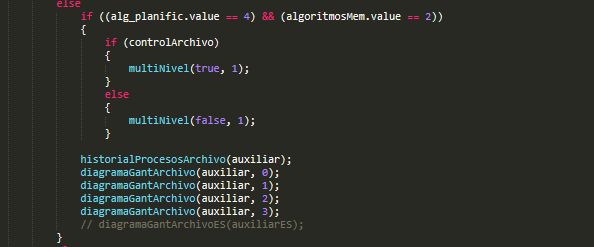
Muestra los datos de los procesos en forma de tabla.

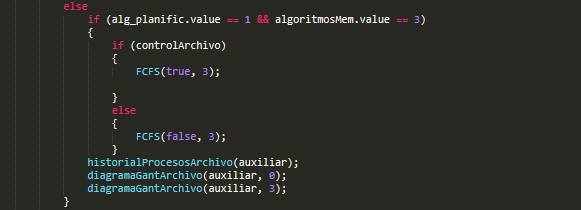
function adminAlgoritmosProcesos()

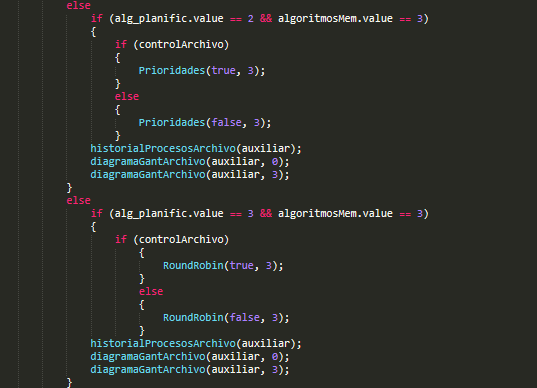








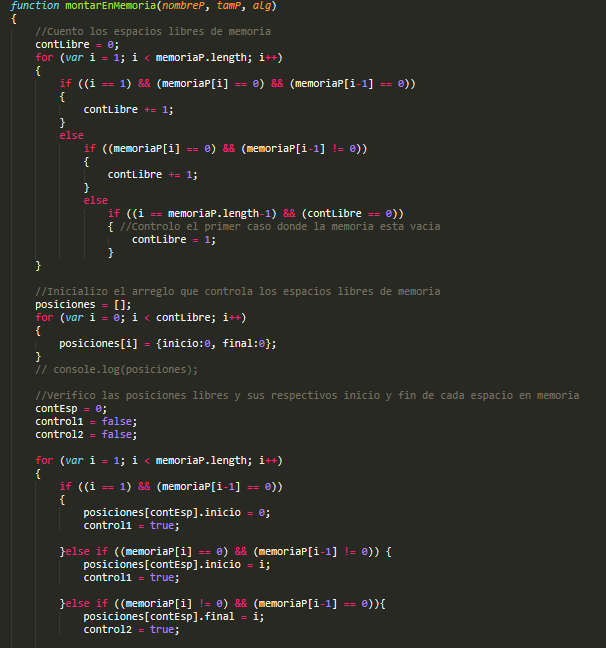


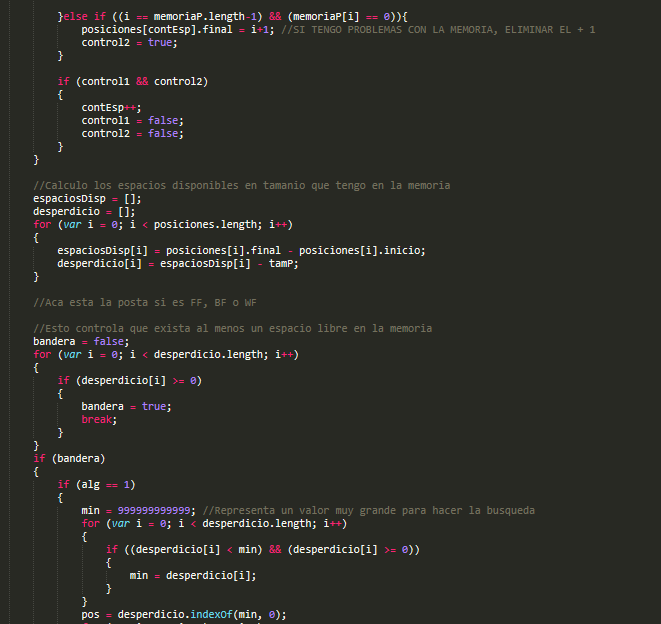


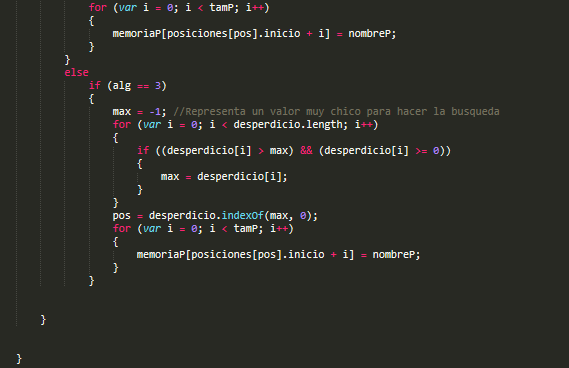


De acuerdo al tipo de algoritmo de planificación y de acuerdo a los algoritmos best fit y worst fit de memoria, se pasan los parámetros para diagramar los gantt.

function montarEnMemoria(nombreP, tamP, alg)

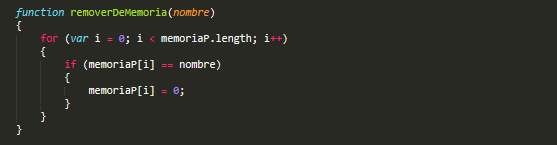






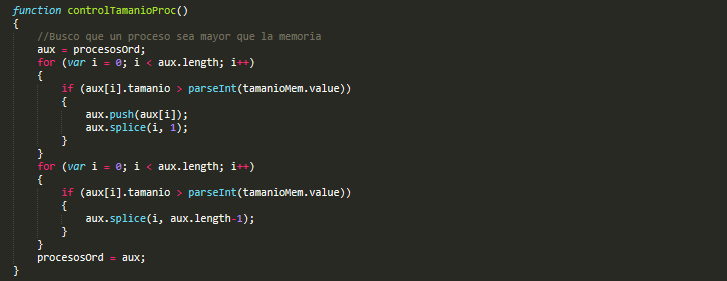
Toma como parámetros tanto el nombre proceso, tamaño del proceso y algoritmo. Se tiene un control de los espacios libres mediante un contador de espacios libres (contLibre).

function removerDeMemoria(nombre)



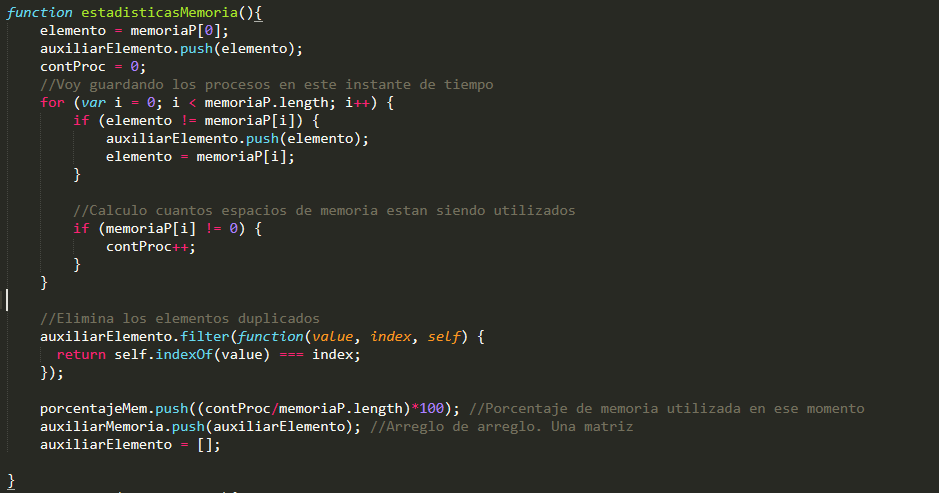
Remueve de memoria el proceso. Al encontrar el nombre pasado como parámetro coloca 0 en la posición del arreglo memoriaP.

function controlTamanioProc()



Realiza la búsqueda de aquellos procesos que superen el tamaño de la memoria.

function estadisticasMemoria()



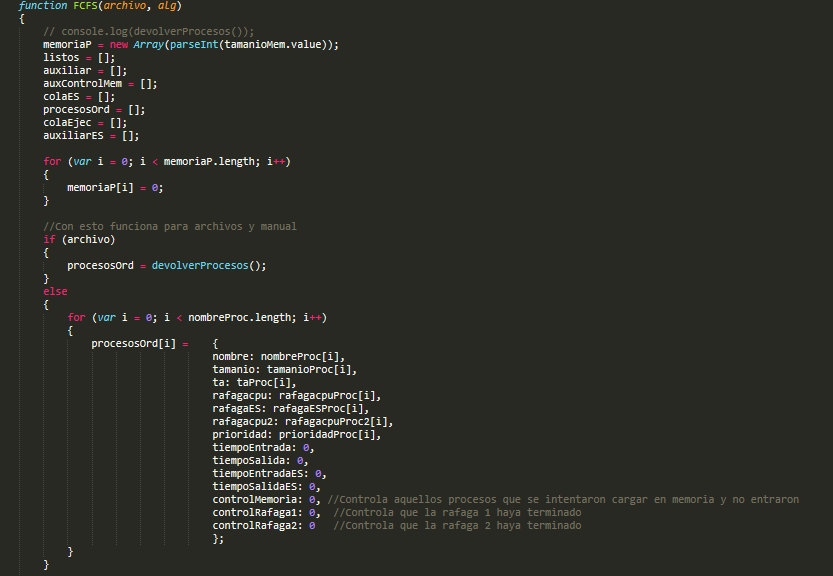
Muestra el porcentaje de utilización de la memoria en cada tiempo con respecto a los procesos que han sido cargados.

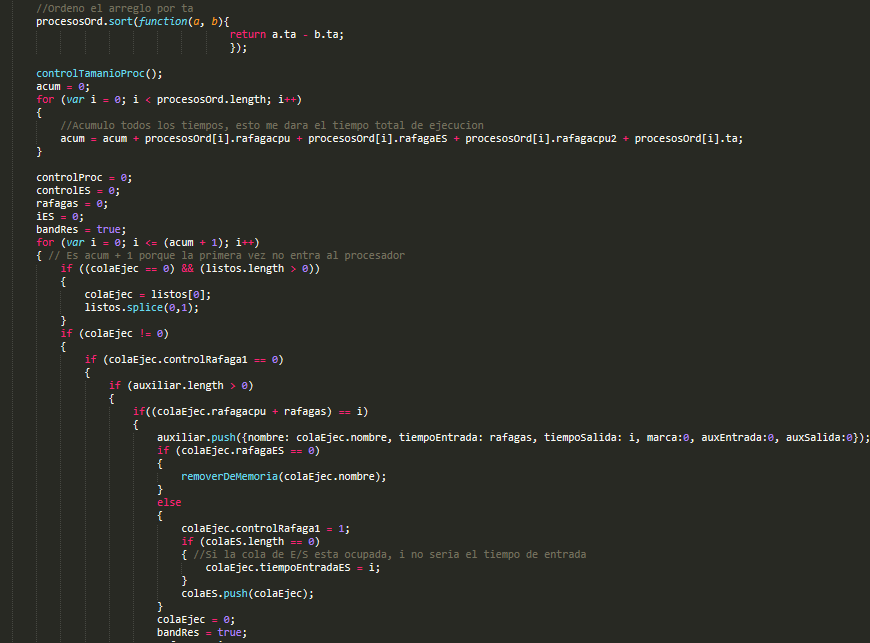
**Funciones dedicadas a los algoritmos de planificación:**

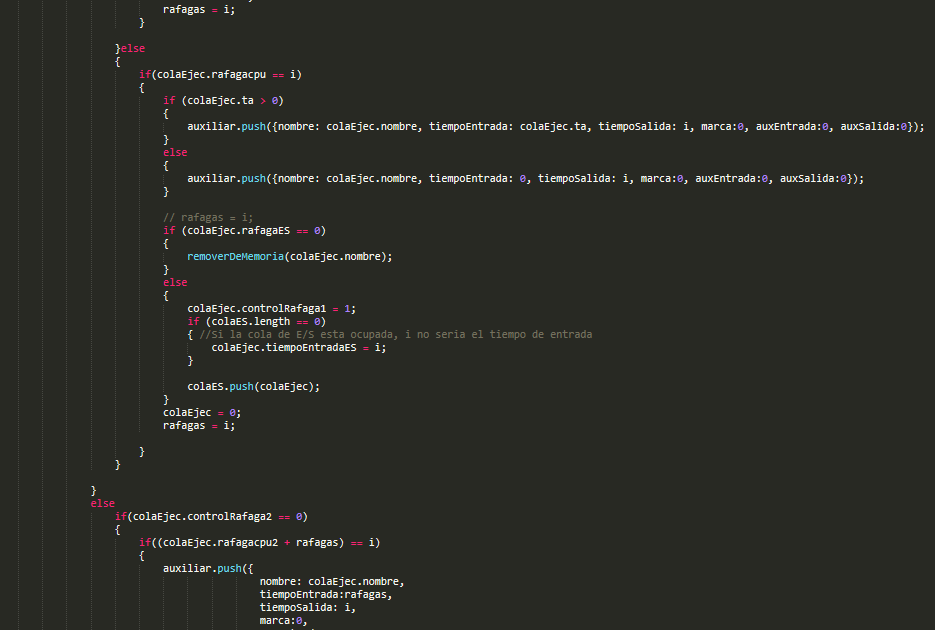
El tratamiento de todos estos algoritmos tienen la siguiente composición:

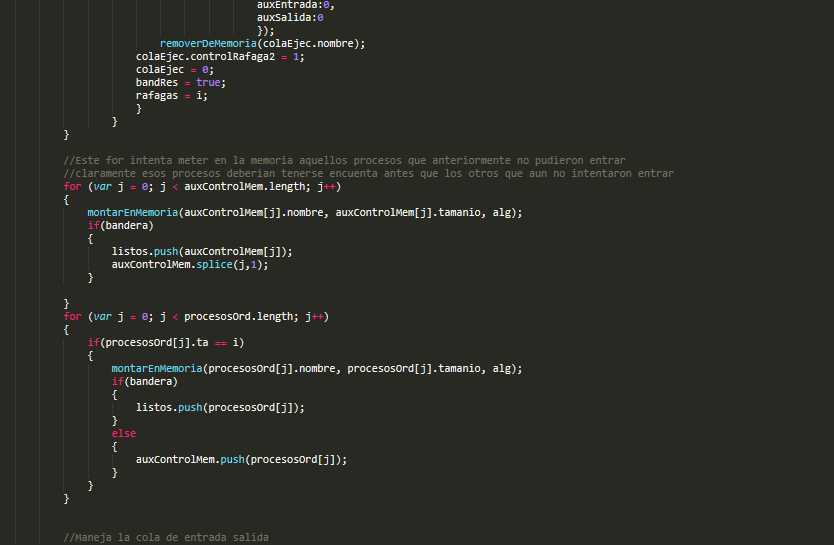
* Reciben los procesos, ya sea en forma de archivo o en forma manual;
* Llama a la función validarProcesos() para verificar las ráfagas, tamaño, etc;
* Ordena el arreglo que contiene a los procesos de acuerdo a: sus prioridades (si se tratase de por Prioridad o MultiNivel), por tiempo de arribo (si se tratase del FCFS o RR);
* Llama a la función controlTamanioProc() que, como explicamos más arriba, realiza un testeo entre el tamaño de la memoria con los tamaños de los procesos;
* Ocupan un acumulador (acum) en el cual se almacena el total de tiempo de los procesos (ráfaga cpu, ráfaga e/s, ráfaga cpu), el cual será el tiempo total de ejecución;
* Realiza una serie de if anidados en los cuales se comprueban los arreglos de listos, el arreglo colaES y las ráfagas de cada proceso. Para el tratamiento, los procesos que intervienen se irán colocando en el arreglo de listos, para que estos luego sean tratados. En el caso de que el proceso que se encuentra en listos haya finalizado sus ráfagas, remueve el proceso (en caso de no tener más ráfagas de cpu) mediante la función removerDeMemoria(). En caso de que cuente con E/S, pasa el proceso al arreglo colaES. En caso de que esta cola esté vacía, lo trata mediante el if dedicado a la colaES, caso contrario lo encola.
* Luego de estas comprobaciones mediante if anidados, realiza un for que trabaja en aquellos procesos que no han podido ingresar a memoria por no disponer de ella.
* Por último trata la colaES;
* Muestra las estadísticas de memoria con la función estadísticasMemoria().

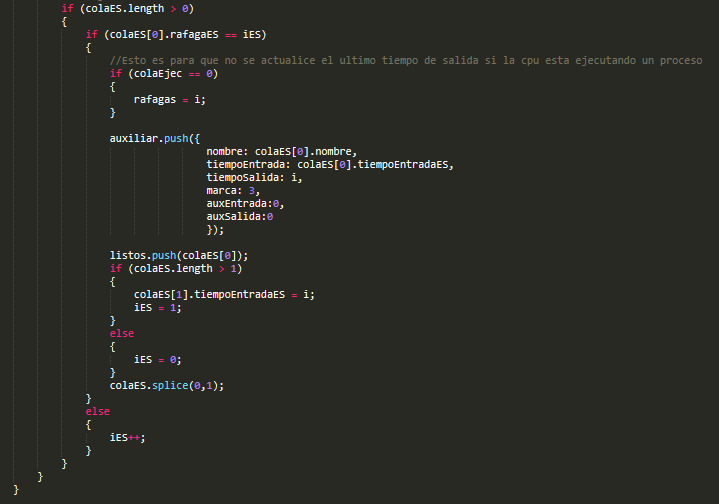
function FCFS(archivo, alg)



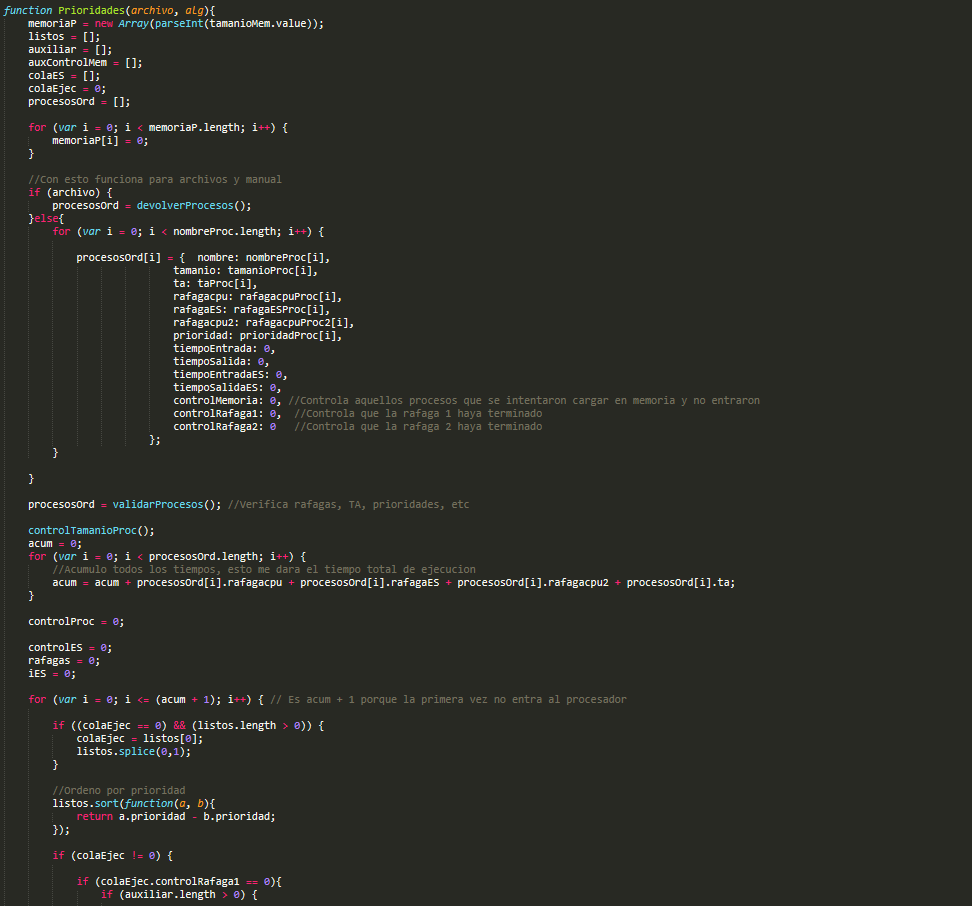




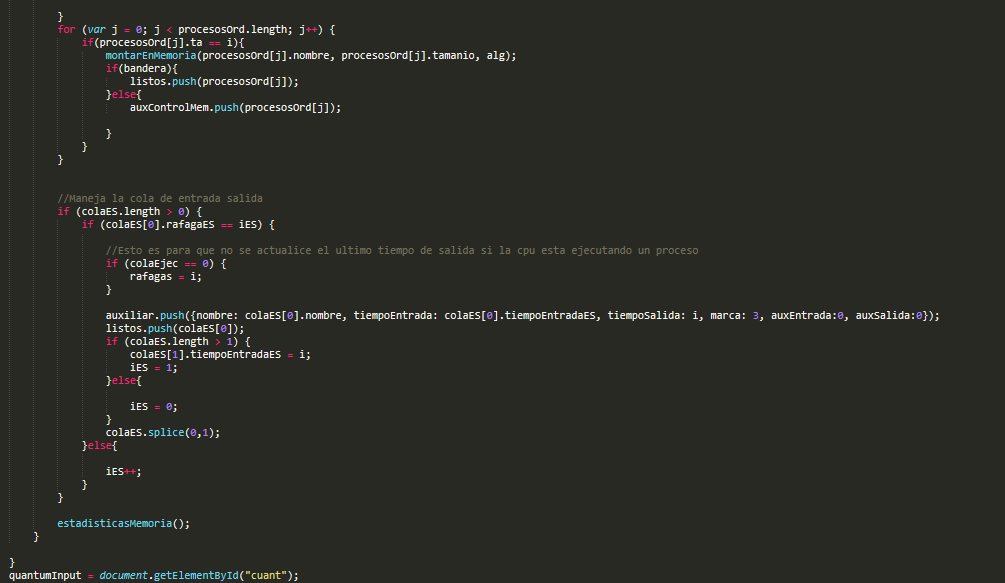




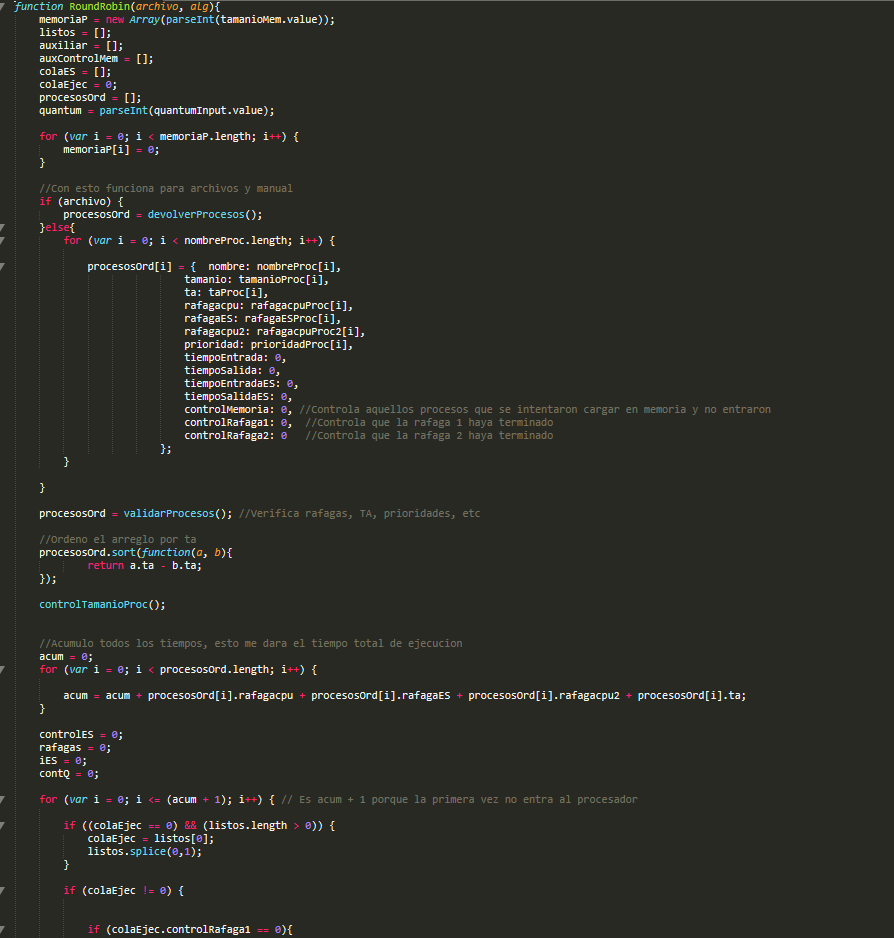
function Prioridades(archivo, alg)



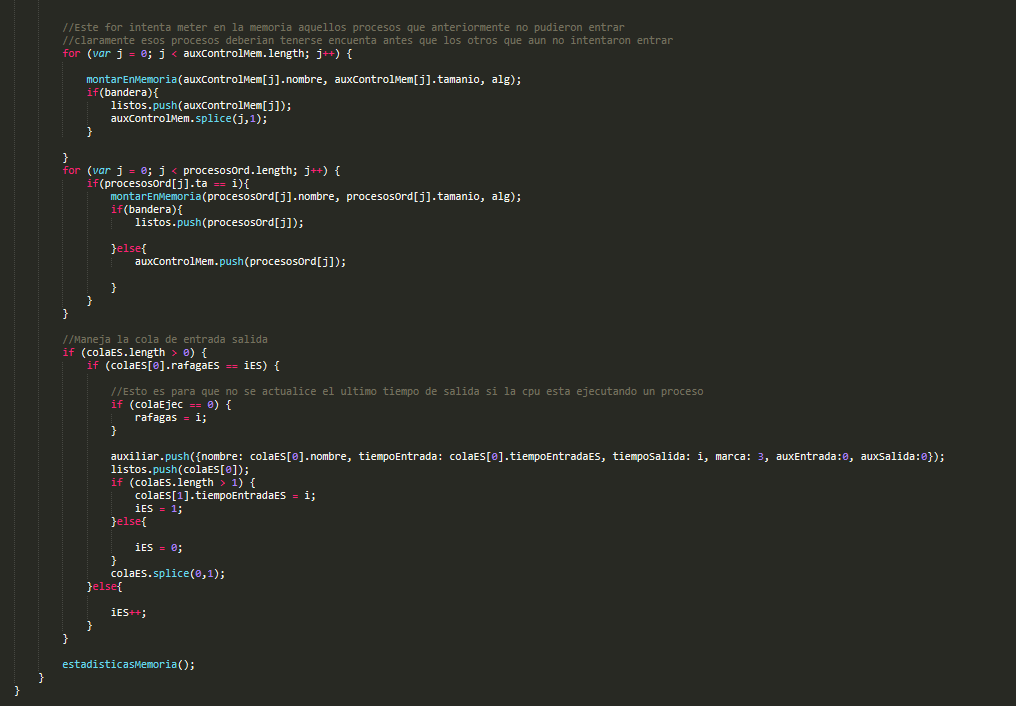




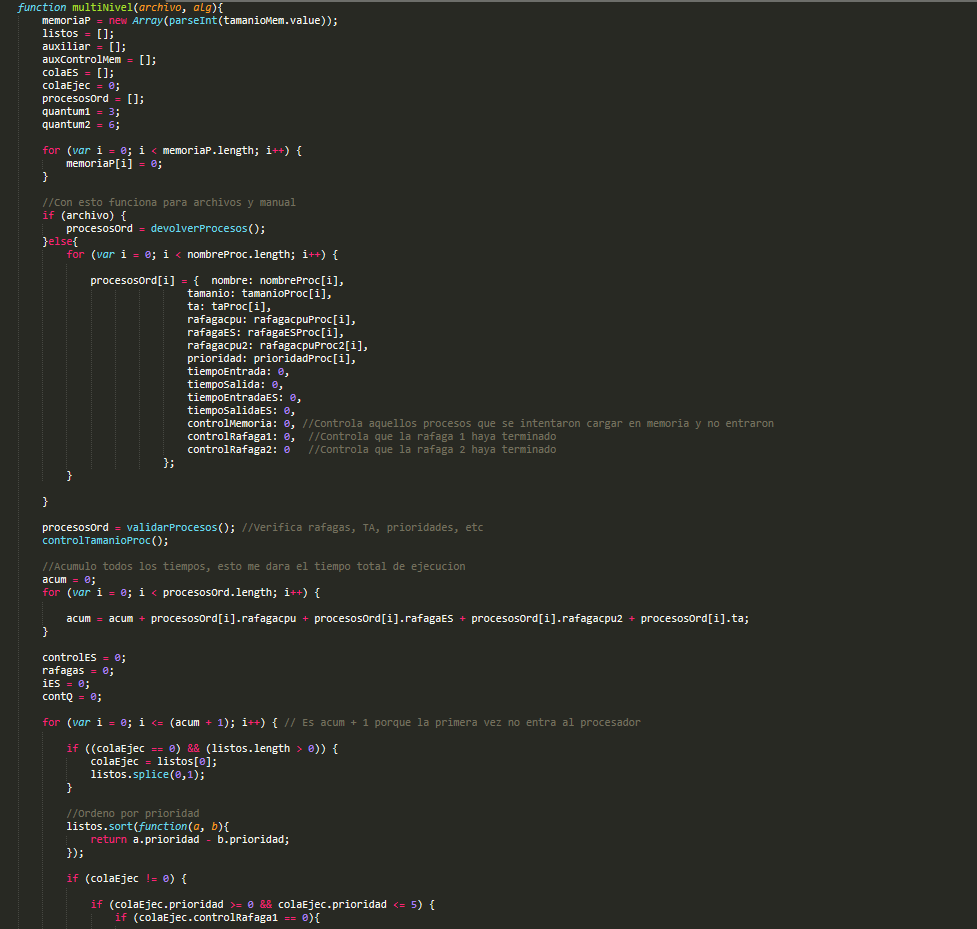
function RoundRobin(archivo, alg)





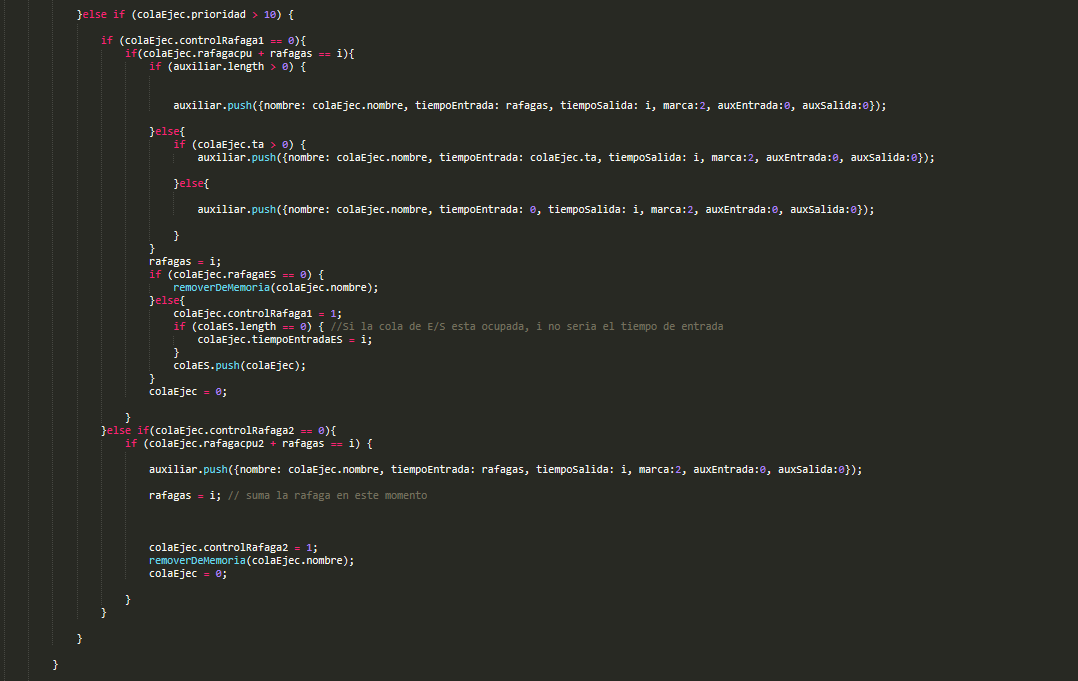


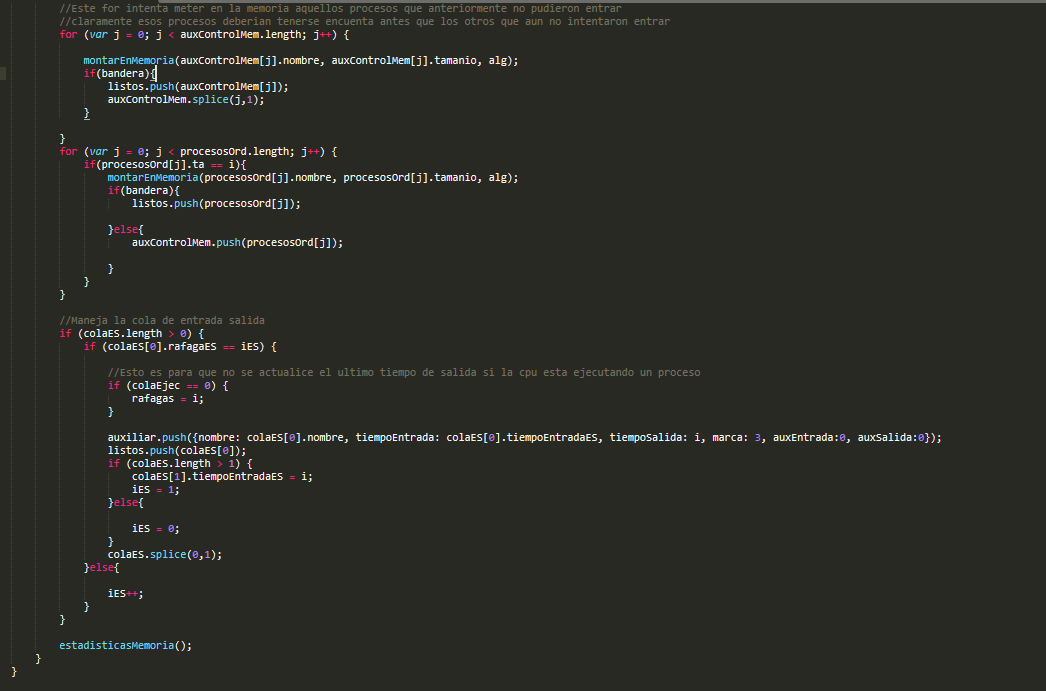
function multiNivel(archivo, alg)







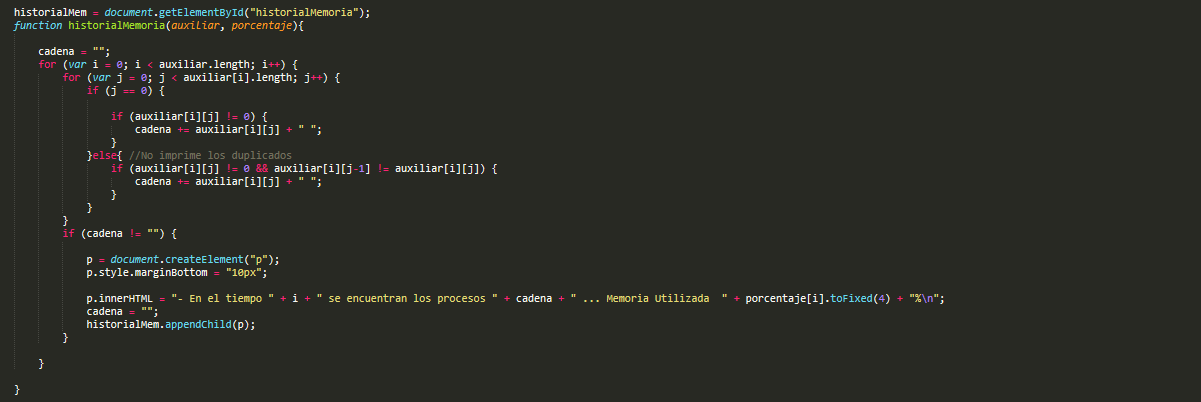




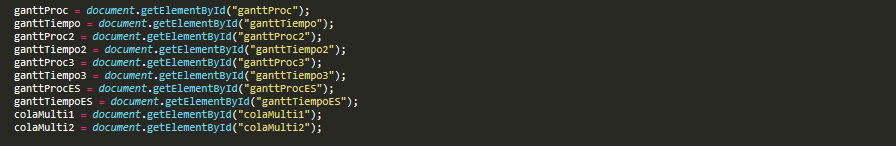
function historialProcesosArchivo(auxiliar)



function historialProcesosArchivoaux(auxiliar)



function diagramaGantArchivo(auxiliar, marca)



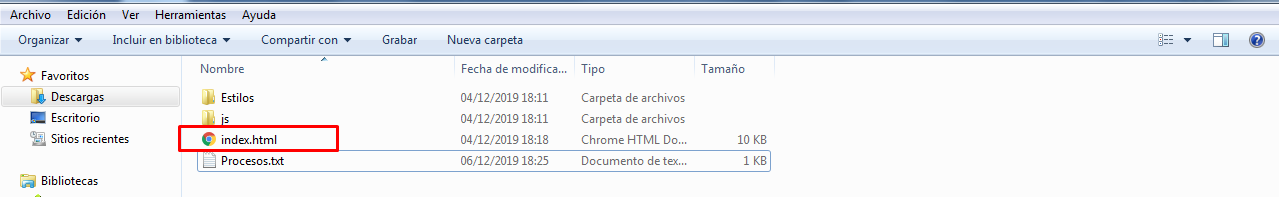




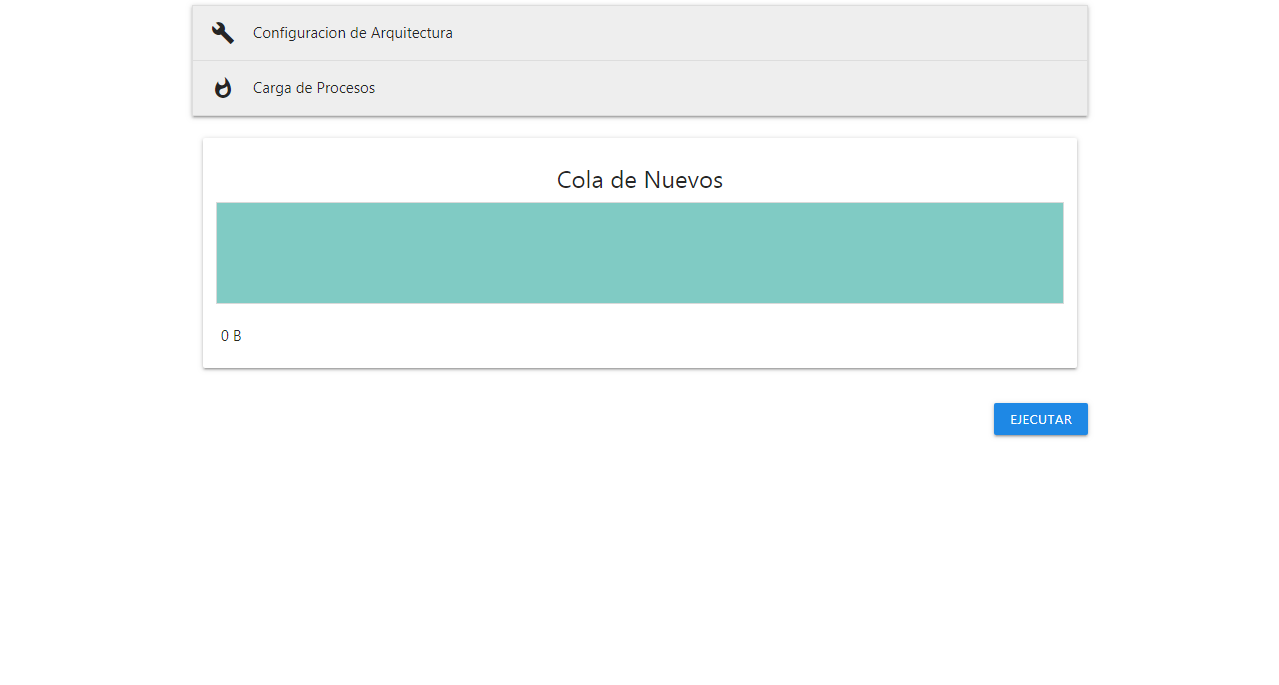
**Instructivo de uso**

**Introducción**

Proceda a abrir el archivo index.html



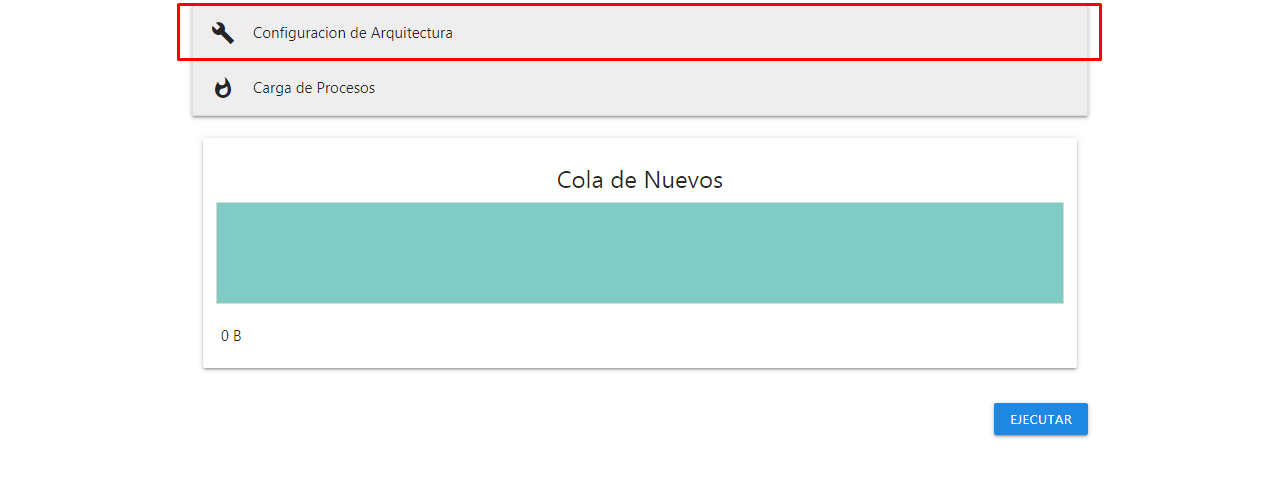
seguidamente debería de aparecer ante usted en pantalla, la siguiente imagen:



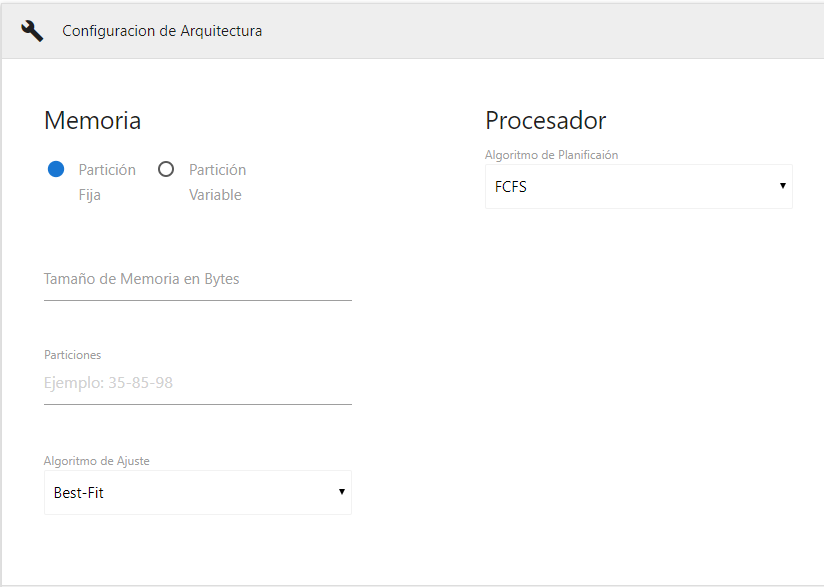
**Primer Carga de Trabajo**

* **Configuración de la Arquitectura**

Iniciaremos la primera carga de trabajo, para ello seleccione “Configuración de la Arquitectura”



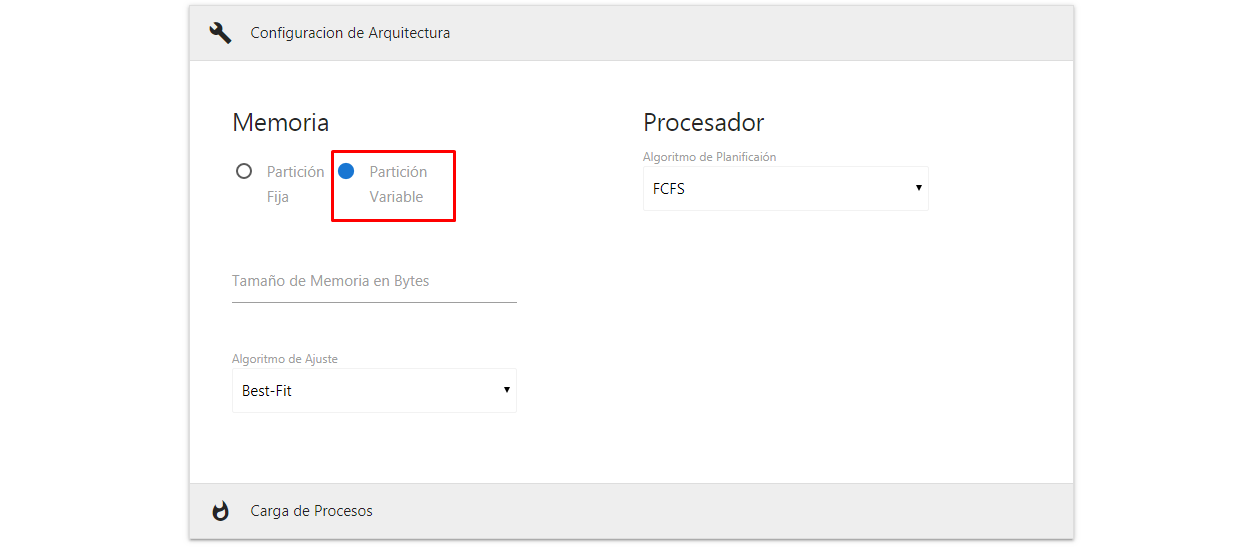
Debería de aparecer ante usted la siguiente pantalla



Deberá detallar a continuación, los parámetros de carga de entrada del simulador, para esta primera carga de trabajo seleccione “Partición Variable”. A continuación, realice el ingreso del tamaño de memoria, para lanzar la primera prueba, ocuparemos un valor de 1024.

Seguidamente, seleccione de la sección de algoritmo de ajuste, el algoritmo “Best Fit”. Luego, dirija el cursor a la sección de Procesador para elegir el algoritmo de Planificación. Para este caso, seleccione FCFS

Si ha seguido de manera correcta los pasos, debería de aparecer ante usted la siguiente imagen:

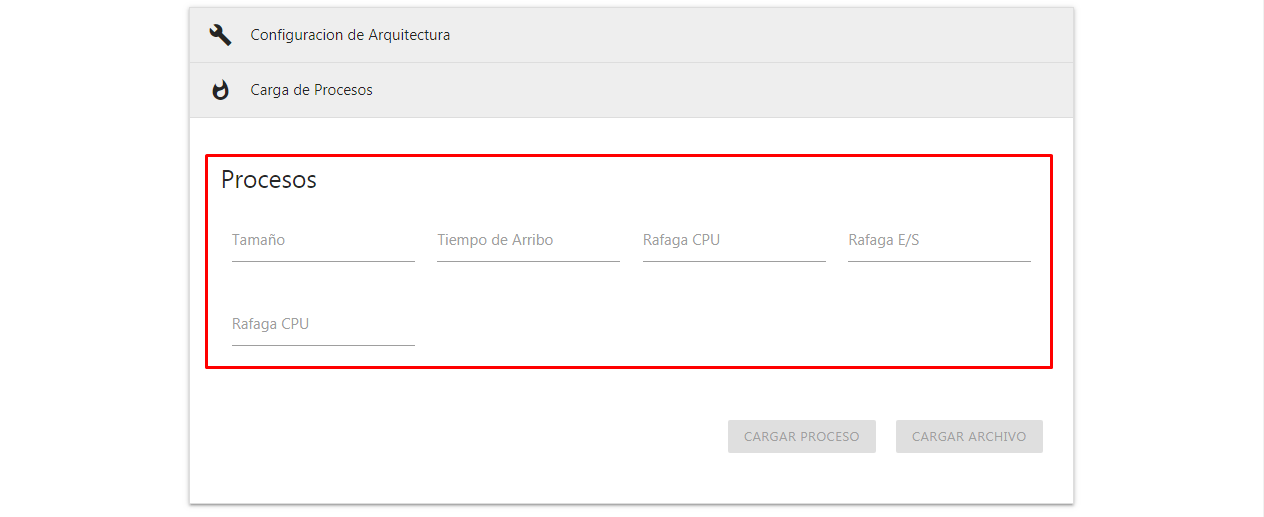


**Carga de Procesos en forma Manual:**

Para cargar sus primeros procesos, seleccione el apartado de “Carga de Procesos”

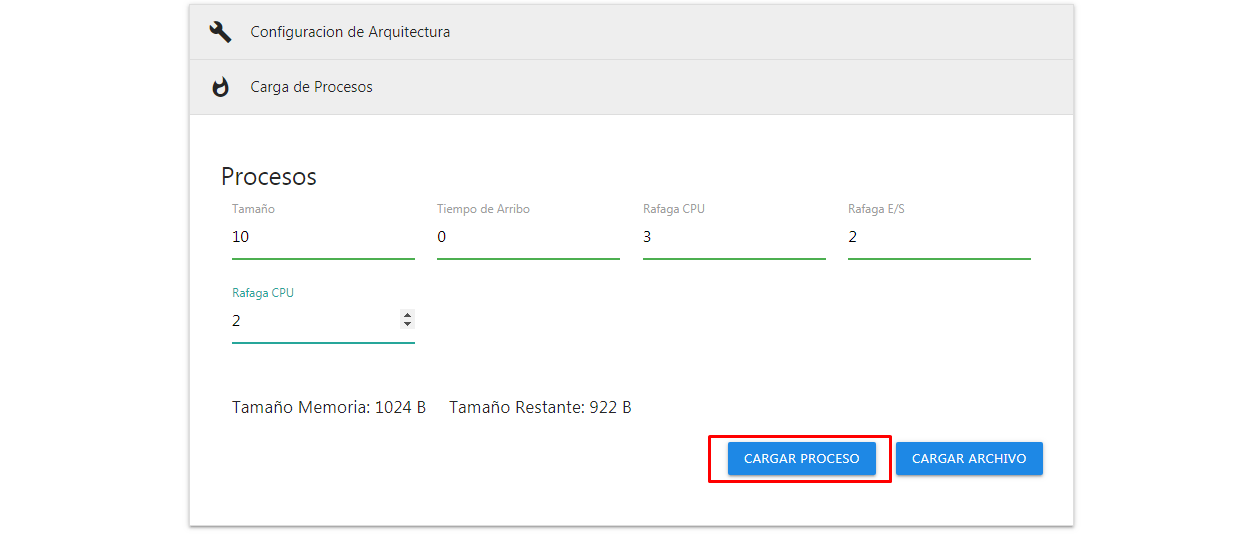


Se desplegará ante usted los siguientes campos:



**Nota: El SO ocupa el 10% de la memoria con la que se trabaje, por eso mismo se observa en la foto que hay 922 B de tamaño restante**

Complete los campos teniendo en cuenta los siguientes datos suministrados como ejemplo y cuando termine de completar cada fila, presione en “Cargar Proceso” como se muestra en la siguiente pantalla:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tamaño | Tiempo de Arribo | Ráfaga de CPU | Ráfaga E/S | Ráfaga CPU |
| 10 | 0 | 3 | 2 | 2 |
| 4 | 5 | 1 | 2 | 5 |
| 14 | 2 | 3 | 5 | 1 |

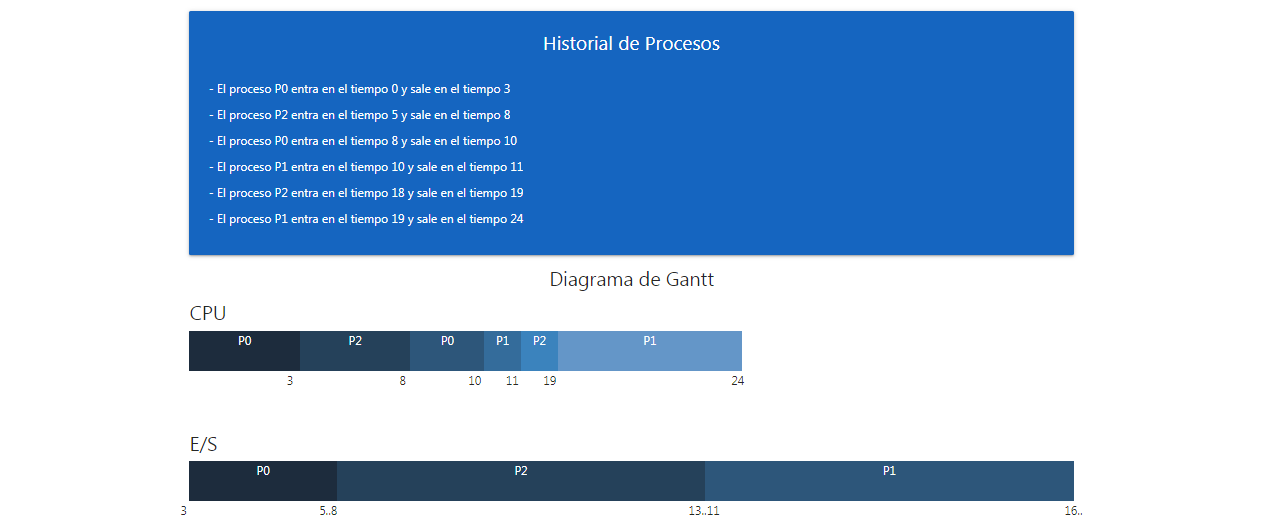
Datos de ejemplo

Cuando finalice la carga, podrá visualizar de manera ilustrativa, una cola de Nuevos con los procesos representados en diferentes colores en la cola, además de una tabla donde se aprecian los mismos datos ingresados más un campo que indica la prioridad del proceso, que, como estamos trabajando bajo planificación FCFS, tendrán valor “Ninguno”.



**Ejecución:**

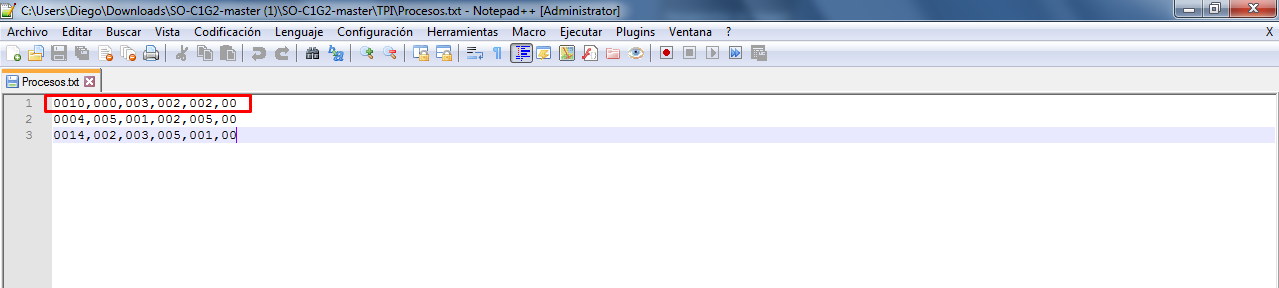
Para su ejecución, presione el botón “Ejecutar” el cual visualizará el historial de procesos y así se observará el movimiento de los mismos en torno al momento que entran y salen del procesador, además de los diagrama de Gantt tanto de CPU como de E/S que aporta una mejor representación.



Si ha seguido todos los pasos descritos en este instructivo sin omitir por alto ninguno, ahora debería ser capaz de poder realizar cualquier carga de trabajo de forma “Manual”.

**Carga de Procesos desde un archivo de texto**

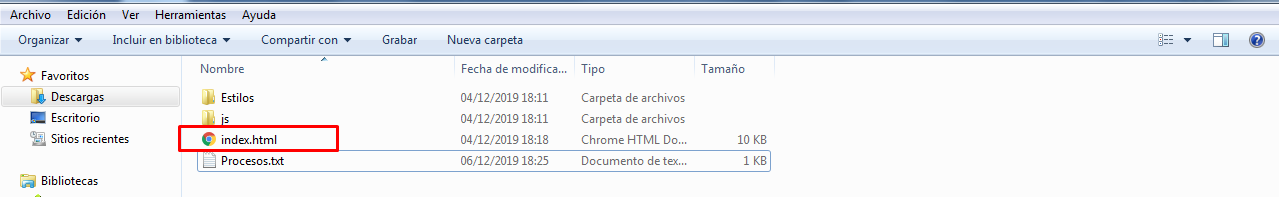
Además de la carga de procesos de forma manual, este simulador también admite la carga de procesos a través de un archivo de texto, para realizar la carga, proceda a abrir un documento de texto en blanco y complételo teniendo en cuenta la siguiente ilustración:



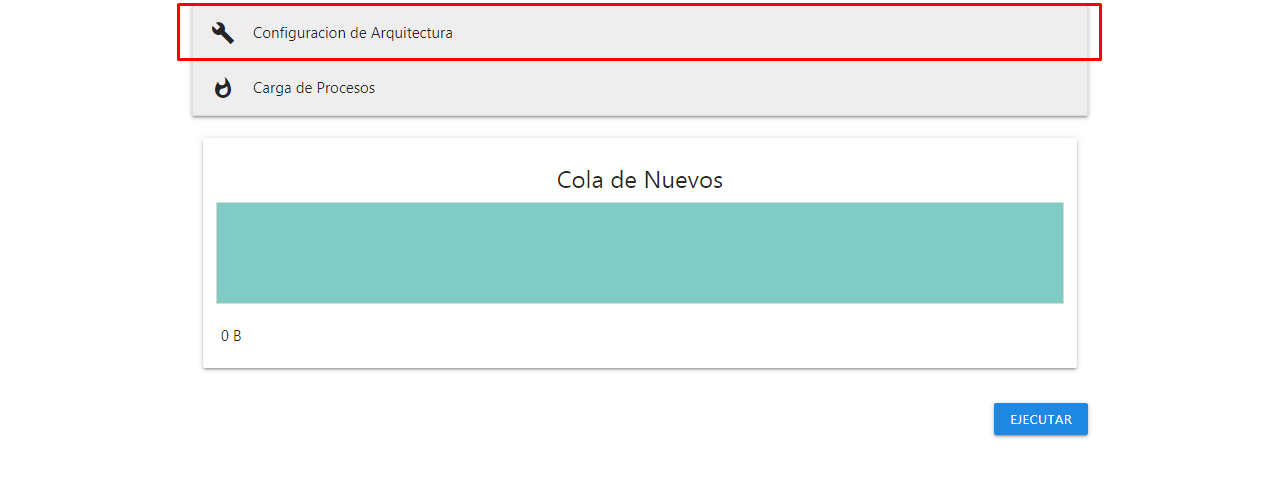
El archivo se va a componer de seis campos separados por coma, el primer campo representado por cuatro cifras es el tamaño del proceso, el segundo campo representado por tres cifras es el tiempo en el que arriba el proceso, el tercer campo representado por tres cifras es la primer ráfaga de CPU del proceso, el cuarto campo representado por tres cifras es la ráfaga de E/S del proceso, el quinto campo representado por tres cifras representa la segunda ráfaga de CPU del proceso, y el último campo representado por dos cifras representa la prioridad que posee el proceso.

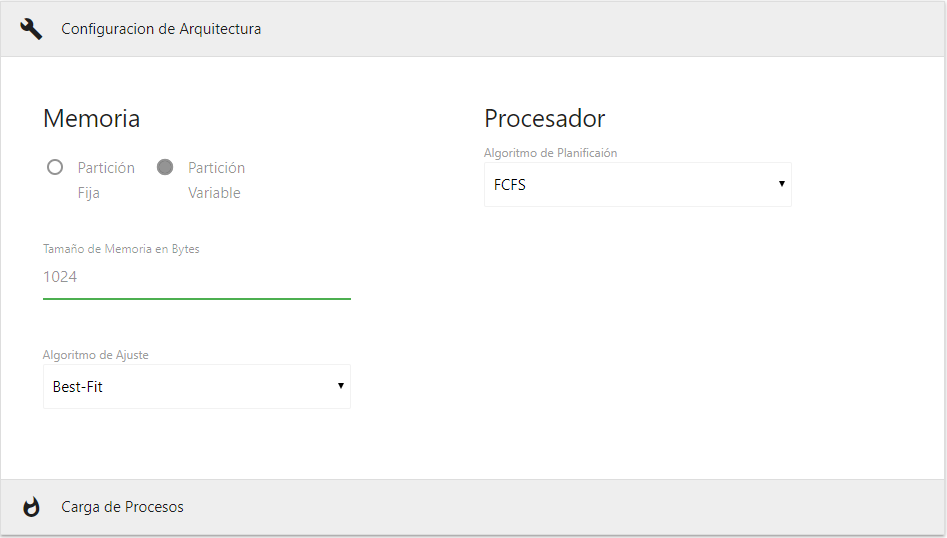
**Nota: Para el correcto funcionamiento del simulador con una carga de procesos desde un archivo, es importante que se respete la cantidad de cifras de cada campo, es decir, bajo ningún concepto puede disminuir o aumentar la cantidad de cifras que requiere cada campo en el archivo, de hacerlo, ocasionaría una respuesta inesperada de parte del programa.**

Finalizado el proceso, guarde su archivo con el nombre que desee, y luego diríjase al archivo index.html

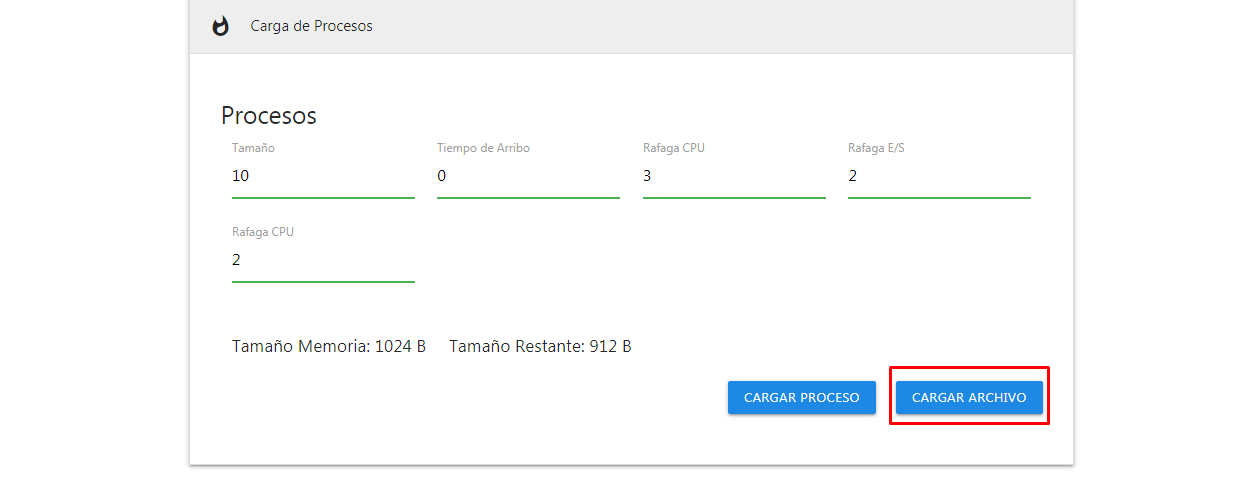


Diríjase a “Configuración de la Arquitectura” y repita la configuración con la que se explicó la carga de procesos de forma manual.



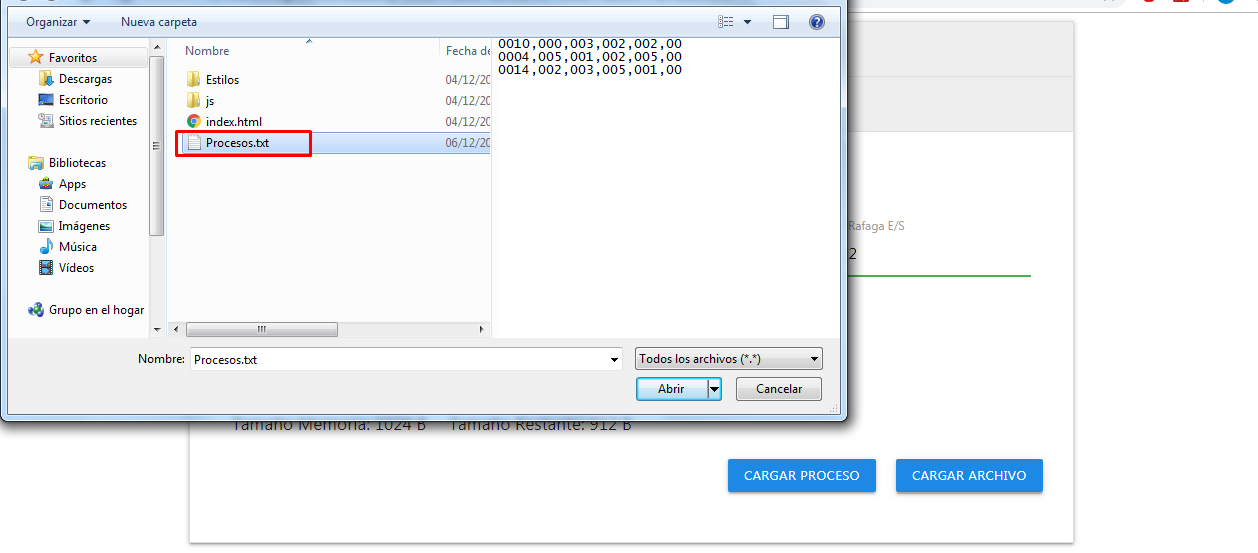


A continuación, diríjase al apartado de “Carga de Procesos”, debería de poder ver en su pantalla lo siguiente:



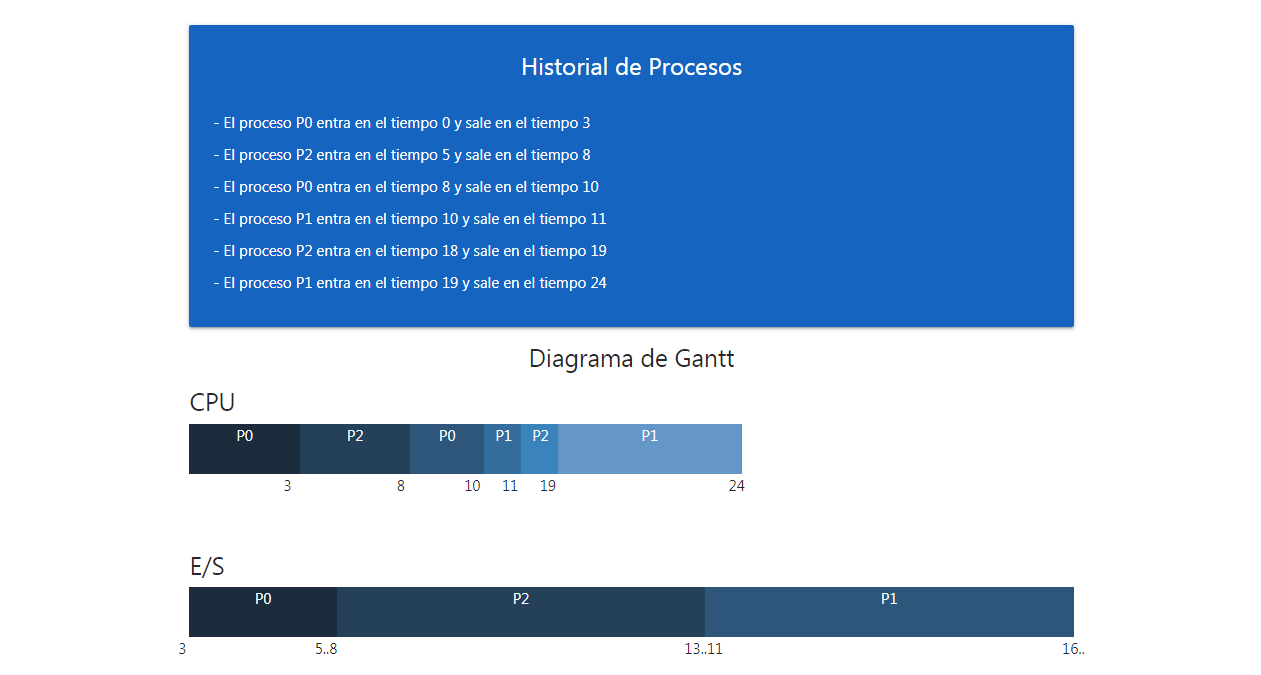
Ahora, en lugar de completar los campos que se aprecian en pantalla y finalmente pulsar en “Cargar Proceso”, vamos a cargar los procesos que habíamos definido en nuestro archivo de texto anteriormente, para ello pulse “Cargar Archivo”

Se desplegará una ventana para que localice la ubicación de su archivo de texto, una vez que lo localice y presione en “Abrir”, debería de poder ver de manera representativa los procesos cargados en la Cola de Nuevos, y también en forma de tabla de la siguiente manera





Para finalizar y correr el simulador con los procesos que ha cargado en el archivo de texto, presione ejecutar. Ahora debería de poder ver los resultados en el Historial de Procesos como en la siguiente ilustración con su diagrama de Gantt respectivo.



Y más abajo nos mostrará el Historial de memoria junto al porcentaje de utilización:

