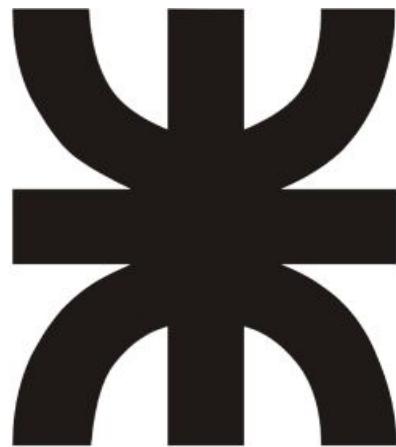


Universidad Tecnológica Nacional

Facultad Regional Resistencia



Trabajo práctico: SIMULADOR C1G7

Integrantes:

- PALLARES, Ulises.
- SCHUSTER, Exequiel Andres.
- TRONCOSO, Mariano Adrian.
- VALENZANO, Maximiliano

Carrera: Ingeniería en Sistemas de Información

Asignatura: Sistemas Operativos

Profesores:

- Mag. Ing. Liliana CUENCA PLETSCH.
- Dr. Ing. Sergio GRAMAJO
- Ing. RISTOFF Alberto
- Ing. ROA Jorge Alejandro

Introducción

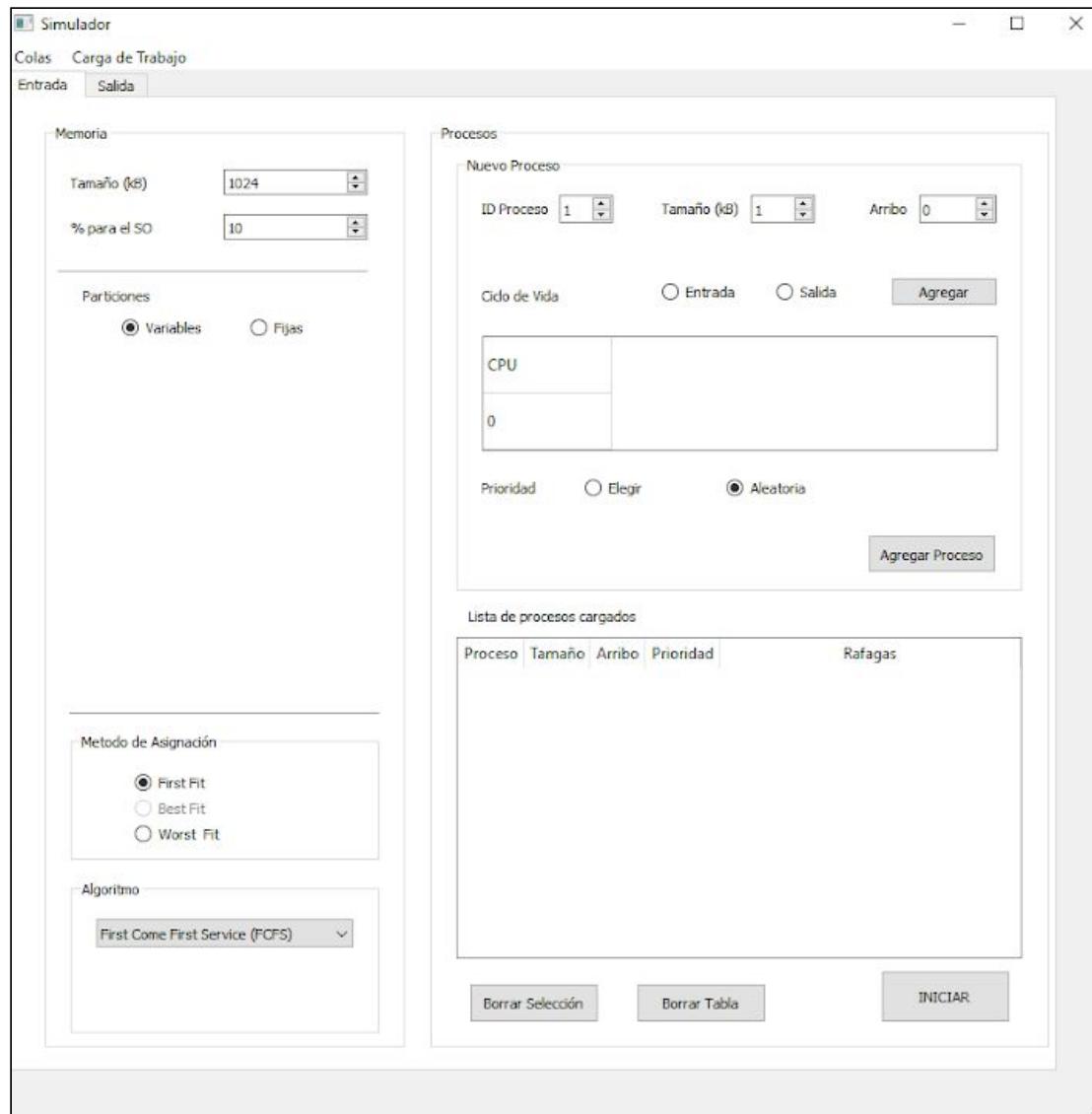
A continuación se presenta la documentación del trabajo de Simulador de planificación de procesos y administración de memoria. Se agregan varias funcionalidades a las entregas anteriores, por lo cual algunas interfaces son diferentes, debido a que se hicieron correcciones de errores o agregaron funcionalidades.

Para el desarrollo e implementación del programa utilizamos las siguientes herramientas: lenguaje de programación Python, junto con la librería gráfica PyQt, la cual cuenta con el programa de diseño Qt Designer, que facilita mucho el diseño de las interfaces, ya que uno puede diseñar de forma “gráfica” la interfaz, y luego el programa genera el código necesario. También utilizamos SQLite para almacenar los distintos procesos y sus configuraciones. Elegimos esta herramienta dada a su buena integración con Python, debido a que viene como parte de su biblioteca estándar.

A lo largo del documento presentaremos una guía de uso del programa, donde se especifican qué datos se deben cargar, cuales son las restricciones para los mismos (valores minimos y maximos sobre todo), el orden en el cual deben ser cargados, cómo seleccionar opciones diferentes de las predeterminadas, los posibles errores que se pueden presentar, cómo ver los resultados de la simulación, entre otros aspectos.

Inicio del Simulador

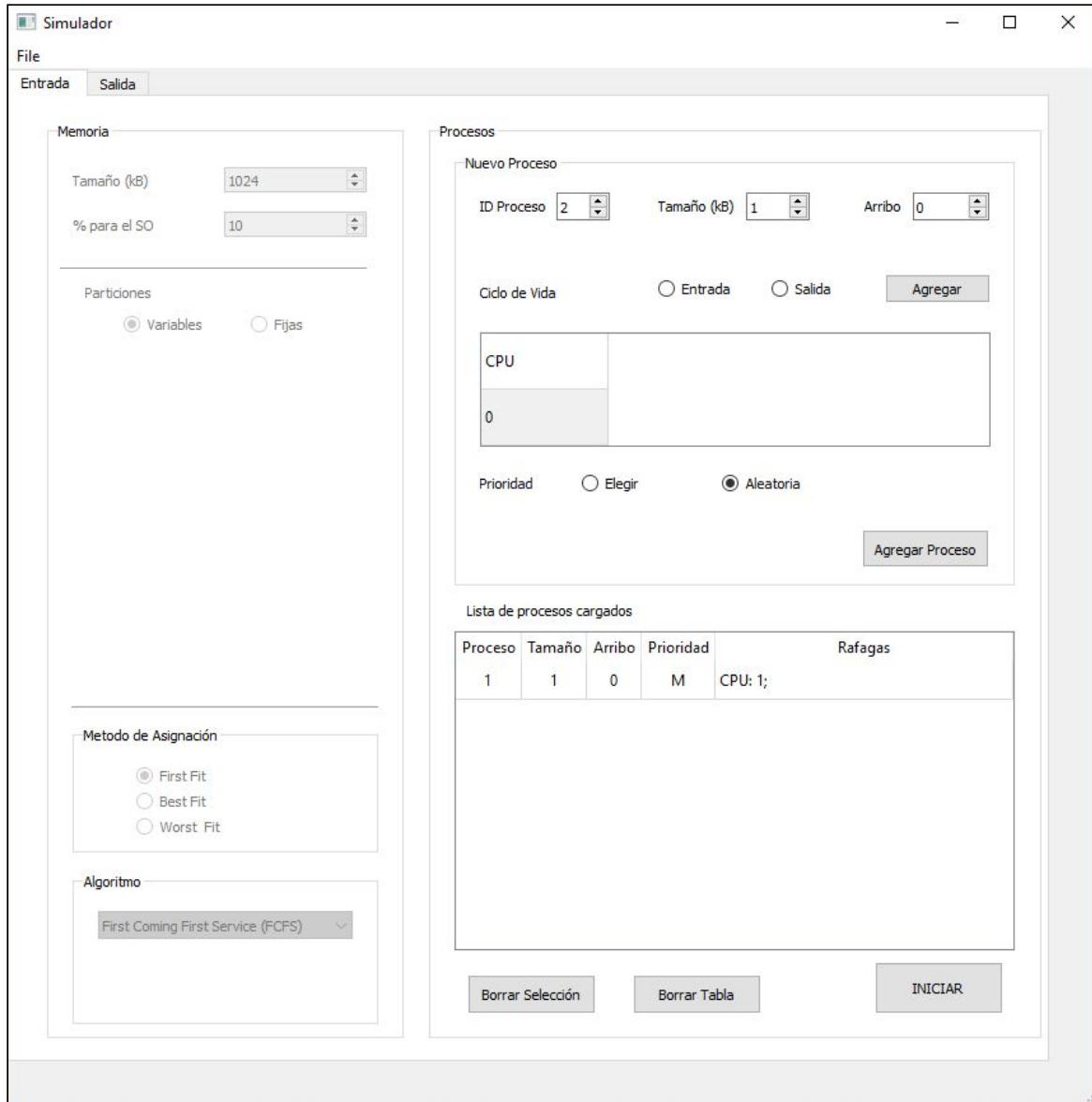
Para ejecutar el programa recurra al documento “Tutoriales C1G7”. Al iniciar el programa, aparece la siguiente pantalla:



Estamos posicionados en la pestaña de entrada del Simulador, como se puede apreciar hemos sectorizado los elementos de la simulación (memoria, procesos y algoritmos de planificación) junto con sus características. Como vemos existen opciones preestablecidas para la carga de trabajo, las cuales pueden cambiarse si se desea.

Importante: **el orden en el cual están descritos los campos de entrada en este documento es el orden en el cual se debe realizar la carga de dichos datos**, debido a que al cargar algunas configuraciones, se deshabilitará la modificación de datos cargados previamente. Por ejemplo, una vez que se ingresó el tamaño de la memoria y luego se agregaron particiones, el tamaño de la memoria ya no podrá ser modificado, para esto tendremos que reiniciar la ejecución del programa. O por ejemplo, cuando se carga el primer proceso en la tabla de procesos, ya no podrán hacerse modificaciones sobre los

siguientes aspectos: memoria, espacio ocupado por el sistema operativo, particiones, algoritmo de planificación, algoritmo de administración de memoria. El principal fin de esta restricción es eludir posibles errores al momento de ejecutar. Esto podemos verlo en la siguiente imagen:



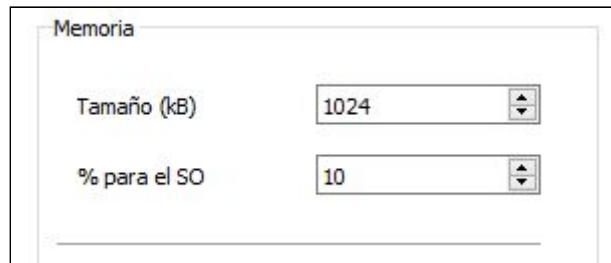
Como vemos en la imagen, al cargar un proceso en la “Lista de procesos cargados”, las opciones ubicadas a la izquierda de la pantalla, se encuentran bloqueadas, esto es, que no se pueden modificar.

Tamaño de memoria y porcentaje de Sistema Operativo

El primer aspecto que se tiene que seleccionar es el tamaño de memoria, el cual puede variar entre **32 y 4096 kb**. Por defecto, 1024 kb se encuentra como valor predeterminado. Al igual que todos los spinbox, es posible modificar el valor de esté de dos maneras: con las

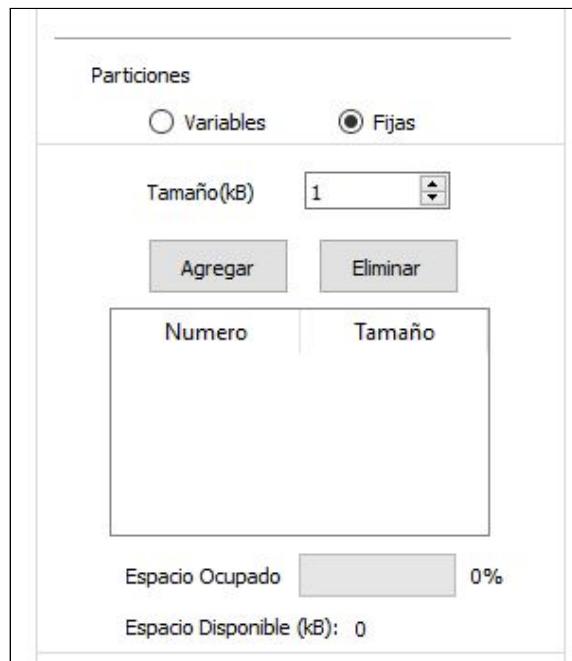
flechas, las cuales incrementan o decrementan el valor en uno, o directamente seleccionar con el ratón dicho campo e ingresar un valor numérico permitido.

A continuación se encuentra un *spinbox*, en el cual se especifica el porcentaje de la memoria ocupado por el sistema operativo, el cual puede variar entre el **10 y 30 %** del tamaño de la memoria.



Particiones

Como primera medida se debe elegir el tipo de particionado que se desea utilizar, o simplemente usa la opción que se encuentra por defecto se encuentra seleccionada la opción de particiones variables. En el caso de escoger la opción de particiones fijas aparece lo siguiente:



En la imagen vemos qué podemos agregar particiones de forma manual, con el botón “Agregar”, indicando el tamaño de cada una, el cual puede variar **entre 1 kb y el tamaño de memoria restante** una vez que se asigne el espacio al sistema operativo. El número de partición es un campo autoincremental que se calcula automáticamente.

En el caso de querer cargar una partición cuyo tamaño sea superior al tamaño disponible veremos lo siguiente:

Memoria

Tamaño (kB)	1024
% para el SO	10

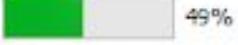
Particiones

Variables Fijas

Tamaño(kB)	500
------------	-----

Agregar **Eliminar**

Numero	Tamaño
1	500

Espacio Ocupado  49%

Espacio Disponible (kB): 422

Al agregar la partición, los datos aparecerán en dicha tabla. En el caso de haber cargado mal la partición, por ejemplo un tamaño no deseado, es posible seleccionar la misma en la tabla y eliminarla con el botón “Eliminar”.

Método de asignación

Luego se debe seleccionar qué algoritmo de administración de memoria se quiere. Se cuenta con las siguientes opciones. First-Fit (que se encuentra seleccionada por defecto), Best-Fit, Worst-Fit.

Importante: en el caso de que se elija **particiones fijas**, sólo podremos optar por las opciones de First-Fit y Best-Fit, debido a que la opción de **Worst-Fit** se encontrará deshabilitada.

Particiones

Variables Fijas

Tamaño(kB)

Número	Tamaño

Espacio Ocupado 0%

Espacio Disponible (kB): 0

Método de Asignación

First Fit
 Best Fit
 Worst Fit

Por el contrario, en el caso de elegir **particiones variables**, sólo podremos optar por las opciones de First-Fit y Worst-Fit, debido a que la opción de **Best-Fit se encontrará deshabilitada**, al igual que la sección para la carga de particiones, dado que no tiene utilidad en este modo.

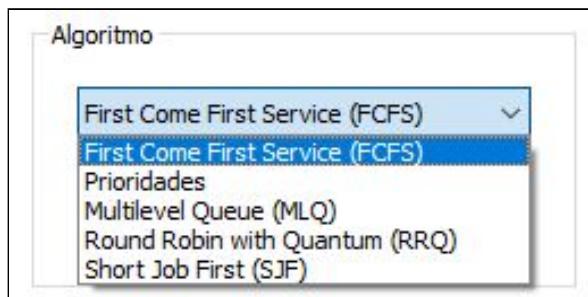
Particiones

Variables Fijas

Método de Asignación

First Fit
 Best Fit
 Worst Fit

Algoritmo



Posteriormente debemos realizar la selección del algoritmo de planificación de procesos. Se cuenta con las siguientes opciones: FCFS (la cual se encuentra seleccionada por defecto), RRQ, de los algoritmos basados en prioridades implementamos dos: Prioridades y SJF. También implementamos MLQ. No implementamos STRF.

La forma en la cual implementamos los algoritmos es bastante parecida para todos, en cuanto a variables, ciclos, condicionales, estructuras de datos, etc. El más “distinto” es el algoritmo de colas multinivel.

A continuación se explican las particularidades de cada algoritmo:

FCFS: se ordenan los procesos en la cola de listos por tiempo de arribo en cada tiempo de ejecución.

Prioridades: se ordenan los procesos en la cola de listos por prioridad en cada tiempo de ejecución. Existen 3 prioridades: Alta (A), Media (M) y Baja (B).

RRQ: el primer proceso en llegar es el que ingresa a la CPU, y el tiempo que en ella se aloje será el del quantum, saliendo de la misma cuando este llegue a su fin. En el caso que no haya otro proceso en cola, se volverá a ejecutar el proceso que acaba de salir.

SJF: se ordenan los procesos de la cola de listos de acuerdo al tiempo de culminación de cada uno, en cada tiempo de ejecución.

MLQ: La implementación del algoritmo de colas multinivel la realizamos de la siguiente manera: utilizamos 3 colas y en cada una se ejecuta un algoritmo distinto.

La selección de las colas, cuándo se realiza la carga de procesos, es la misma que usamos para asignar prioridades en algoritmo de prioridades, es decir:

- si queremos que el proceso pertenezca a la cola 1 (prioridad alta) debemos asignarle la prioridad “A”.
- si queremos que el proceso pertenezca a la cola 2 (prioridad media), debemos asignarle la prioridad “M”.

- si queremos que el proceso pertenezca a la cola 3 (prioridad baja), debemos asignarle la prioridad “B”.

La elección del **número de colas** se hace de forma implícita, con esto queremos decir que si sólo queremos utilizar una sola cola, debemos asignarle la misma prioridad a todos los procesos, si deseamos usar dos colas, tenemos que usar sólo dos prioridades.

El usuario no elige **qué algoritmo implementar en cada cola**, debido a que consideramos que esto podría desembocar en la utilización de algoritmos que no se corresponden a la prioridad de la cola. Por ejemplo: no deberíamos utilizar un algoritmo FCFS en la cola 1.

En la ejecución de este algoritmo los procesos son ordenados de acuerdo a su número de cola. Siempre que se encuentre un proceso de número de cola superior, se ejecutará ese por sobre los demás. En la cola de mayor prioridad se ejecuta el algoritmo RRQ, debido a que un proceso de alta prioridad no puede monopolizar la CPU.

En la cola de prioridad media se ejecuta el algoritmo SJF, dado que consideramos que debería priorizarse los procesos que están cercanos a su culminación por sobre los demás en esa misma cola.

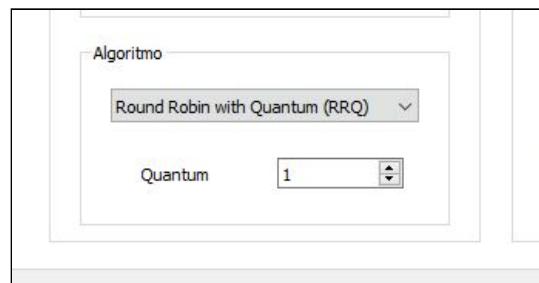
En la cola de menor prioridad se ejecuta el algoritmo FCFS, debido a que como son procesos de baja prioridad se deberían concluir aquel que primero llegue a la cola, dado que al no ser procesos críticos, no requieren de una ejecución inmediata.

En el caso que llegue un proceso de mayor prioridad al que se está ejecutando actualmente en la CPU, por cuestiones de implementación, se decidió que el método sea **no apropiativo**.

Tampoco se implementaron colas retroalimentadas debido a que la complejidad de las mismas no hacían al propósito de estudio del simulador.

En el caso de elegir la opción RRQ, o MLQ, se deberá especificar el tamaño del quantum o el tiempo de quantum para la cola de mayor prioridad respectivamente. Por lo cual una vez que seleccionamos una de esas opciones, aparece lo siguiente:

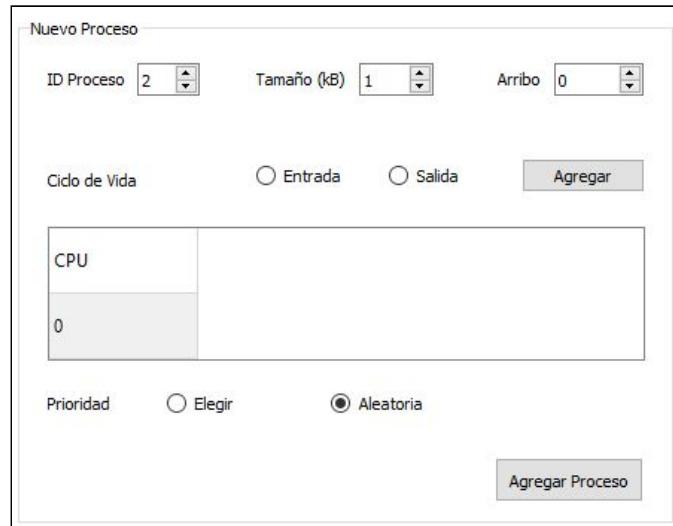
En el caso de que seleccionemos los algoritmos RRQ o MLQ aparecerá lo siguiente:



Donde se debe especificar el tamaño del **Quantum** a utilizar, el cual varía entre **1 y 5**.

Procesos

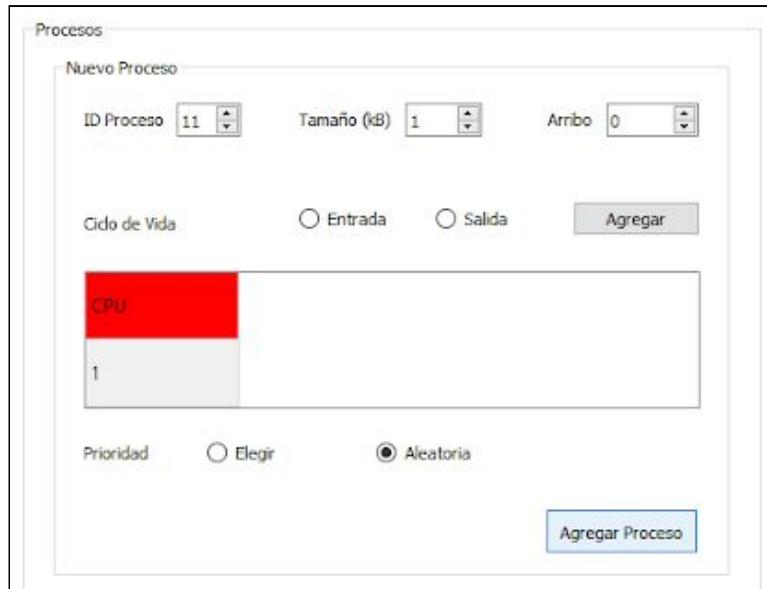
En la parte derecha de la ventana de Entrada, se encuentra la sección de carga de procesos.



Para realizar la carga de un nuevo proceso se deben ingresar los siguiente datos del mismo:

ID Proceso: El identificador que cada proceso puede tener es un número entero que debe ser como **mínimo 1**, y tiene un **máximo de 100**. Sin embargo, sólo podemos cargar hasta un de **10 procesos**.

En el caso que intentemos cargar más de 10 procesos, se mostrará lo siguiente, haciendo referencia a un error en la carga:



Obviamente, cada identificador de procesos debe ser **único**, por lo cual, al intentar cargar un proceso cuyo ID ya existe en la lista, de mostrar lo siguiente:

The screenshot shows a window titled "Procesos" with a sub-section "Nuevo Proceso". It contains fields for "ID Proceso" (set to 1), "Tamaño (kB)" (set to 1), and "Arribo" (set to 0). Below these are radio buttons for "Ciclo de Vida" (Entrada or Salida) and an "Agregar" button. A table labeled "CPU" shows row 1. Under "Prioridad", there are radio buttons for "Elegir" and "Aleatoria", with "Aleatoria" selected. A blue "Agregar Proceso" button is at the bottom. At the bottom of the window, a table titled "Lista de procesos cargados" shows one row with ID 1, size 1 kB, arrival 0, priority B, and rafagas CPU: 1.

Proceso	Tamaño	Arribo	Prioridad	Rafagas
1	1	0	B	CPU: 1;

En la imagen anterior se ve que no podemos tener dos “procesos 1”.

Tamaño: El tamaño del proceso va a depender de **como se quiso particionar la memoria**. Si el particionamiento es fijo, el tamaño máximo de los procesos es igual al tamaño de la partición más grande existente en la tabla de particiones. En cambio, para particiones variables, el tamaño máximo de un proceso será el máximo espacio de memoria sin contar lo ocupado por el sistema operativo.

En la siguiente imagen vemos qué sucede si queremos cargar un proceso cuyo tamaño (201 kb) supere el tamaño de la partición más grande (200 kb):

<p>Memoria</p> <p>Tamaño (kB) <input type="text" value="1024"/></p> <p>% para el SO <input type="text" value="10"/></p> <p>Particiones</p> <p><input type="radio"/> Variables <input checked="" type="radio"/> Fijas</p> <p>Tamaño(kB) <input type="text" value="122"/></p> <p>Agregar Eliminar</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Número</th> <th>Tamaño</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>200</td> </tr> <tr> <td>2</td> <td>200</td> </tr> <tr> <td>3</td> <td>200</td> </tr> </tbody> </table> <p>Espacio Ocupado <div style="background-color: green; width: 100px; height: 10px; display: inline-block;"></div> 100% Espacio Disponible (kB): 0</p>	Número	Tamaño	1	200	2	200	3	200	<p>Procesos</p> <p>Nuevo Proceso</p> <p>ID Proceso <input type="text" value="1"/> Tamaño (kB) <input type="text" value="200"/> Arribo <input type="text" value="0"/></p> <p>Ciclo de Vida <input type="radio"/> Entrada <input type="radio"/> Salida Agregar</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>CPU</td> <td></td> </tr> <tr> <td>1</td> <td></td> </tr> </table> <p>Prioridad <input type="radio"/> Elejir <input checked="" type="radio"/> Aleatoria Agregar Proceso</p> <p>Lista de procesos cargados</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Proceso</th> <th>Tamaño</th> <th>Arribo</th> <th>Prioridad</th> <th>Rafagas</th> </tr> </thead> </table>	CPU		1		Proceso	Tamaño	Arribo	Prioridad	Rafagas
Número	Tamaño																	
1	200																	
2	200																	
3	200																	
CPU																		
1																		
Proceso	Tamaño	Arribo	Prioridad	Rafagas														

Tiempo de Arribo: o simplemente “Arribo”, debe ser como mínimo 0, y no puede ser mayor a 100. El objetivo de esta restricción es simplemente que exista un límite, debido a que con tiempos mayores, el simulador también funcionará.

Ciclo de Vida: cada proceso tiene necesariamente un cierto tiempo de ejecución inicial en CPU, el cual necesariamente tiene que ser un valor mayor a 0, al cual también se le pueden agregar rafagas de entrada y salida. Para ello se tiene que seleccionar que tipo de rafaga de desea, y posteriormente indicar los tiempos tanto para la entrada o salida y de CPU.

El tiempo de cada rafaga no puede ser vacío, cero, o mayor a 30. En caso de querer cargar un proceso con una rafaga que posea alguna de dichas características, no será posible.

Ejemplo de ciclo de vida de un proceso:

Ciclo de Vida		<input checked="" type="radio"/> Entrada	<input type="radio"/> Salida	Agregar
CPU	E	CPU		
8	7	6		

Al agregar una nueva rafaga de Entrada / Salida, automáticamente aparece la opción de agregar una nueva rafaga de procesador, junto con el tiempo de la misma, debido a qué debemos respetar la estructura básica del ciclo de vida de un proceso:

CPU - E / S - CPU

La cantidad máxima de rafagas de un proceso es **5**, es decir, el ciclo de vida más largo que podemos cargar es el siguiente:

CPU - E / S - CPU - E / S - CPU

En el caso qué queramos cargar más rafagas, la forma en la qué se muestra el error es idéntica a la utilizada en el caso de querer cargar una cantidad de procesos que superen el límite máximo.

Prioridad

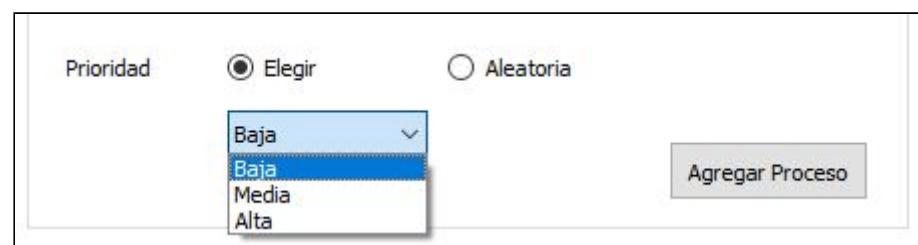
Esta sección solo es útil en el caso que se seleccione el algoritmo de Prioridades o el MLQ, debido a que sí se elige otro algoritmo de planificación, esté campo es ignorado por el programa. Implementamos **tres niveles** de prioridad para los procesos, esta puede ser **Alta, Media o Baja**, esta puede seleccionarse manualmente o en el caso de que esta característica no interese existe la opción de que le sea asignado al proceso una prioridad aleatoria. Esta última opción se encuentra seleccionada por defecto.

En el caso del algoritmo MLQ, la prioridad representa la cola en la cual se encontrará el proceso, ejecutándose en la cola de mayor prioridad el algoritmo RR, el la cola de prioridad media el algoritmo SJF, mientras que en la cola de prioridad más baja se ejecutará el algoritmo FCFS.

Prioridad aleatoria:



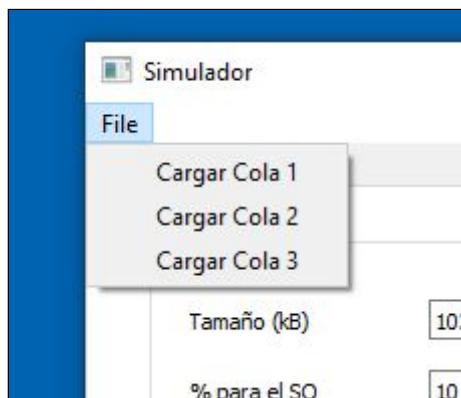
Prioridad elegida por el usuario:



Una vez especificados todos estos datos con respecto a un proceso, se lo agrega a la lista de procesos con el botón “Agregar”. Vemos un ejemplo de una lista de procesos cargados a continuación:

Lista de procesos cargados					
Proceso	Tamaño	Arribo	Prioridad	Rafagas	
1	16	7	A	CPU: 8; E: 4; CPU: 9;	
2	2	6	M	CPU: 3; S: 2; CPU: 8;	
3	17	0	A	CPU: 10; E: 4; CPU: 7;	
4	8	9	B	CPU: 5; S: 3; CPU: 2;	
5	9	10	M	CPU: 9; E: 1; CPU: 9;	

El simulador también cuenta con una forma rápida y sencilla de realizar la **carga de colas de procesos**, la cual se realiza mediante la selección de una cola de los mismos desde la base de datos, para esto seleccionamos “File”, opción ubicada en la parte superior izquierda de la ventana, y se nos aparecerá lo siguiente:



Donde podemos elegir una de las colas de procesos almacenadas ahorrandonos el trabajo de cargar manualmente uno a uno. Al seleccionar una cola, se cargan los datos de los procesos de la misma en la “Tabla de procesos cargados”. No se pueden cargar más de una cola en la tabla.

También se cuenta con las opciones de **borrado de uno o varios procesos de la lista** (Borrar Selección) y con la borrar toda la lista de procesos cargada (Borrar tabla).

Es posible borrar un conjunto de procesos de la siguiente manera: solamente se pueden borrar procesos que se encuentren **consecutivos** en la tabla de procesos, es decir, si por

ejemplo tenemos 10 procesos, y deseamos borrar los procesos con identificador 0,1 y 2 debemos seleccionar con el ratón dichos procesos, y luego “Borrar selección”, de la siguiente manera:

Lista de procesos cargados				
Proceso	Tamaño	Arribo	Prioridad	Rafagas
0	26	10	M	CPU: 5; S: 7; CPU: 2;
1	18	0	B	CPU: 6; E: 8; CPU: 5;
2	16	9	B	CPU: 3; S: 8; CPU: 9;
3	19	1	B	CPU: 2; E: 9; CPU: 7;
4	14	10	B	CPU: 3; S: 7; CPU: 9;
5	24	1	A	CPU: 3; E: 10; CPU: 3;
6	5	3	A	CPU: 7; S: 10; CPU: 10;
7	25	0	P	CPU: 2; E: 6; CPU: 1;

Borrar Selección
Borrar Tabla
INICIAR

El resultado es el siguiente:

Lista de procesos cargados				
Proceso	Tamaño	Arribo	Prioridad	Rafagas
0	18	0	B	CPU: 6; E: 8; CPU: 5;
1	19	1	B	CPU: 2; E: 9; CPU: 7;
2	24	1	A	CPU: 3; E: 10; CPU: 3;
3	5	3	A	CPU: 7; S: 10; CPU: 10;
4	25	0	B	CPU: 3; E: 6; CPU: 1;
5	4	7	M	CPU: 4; S: 8; CPU: 6;
6	14	8	M	CPU: 3; E: 2; CPU: 5;

Cómo vemos, eliminamos los procesos 0, 1 y 2 pero tampoco se encuentran los identificadores de los procesos 7, 8 y 9. Estos últimos no se han borrado, sino que sus identificadores se han modificado para seguir con el orden secuencial de dicho campo de la tabla. Dicho de otra manera:

Al inicio tenemos los identificadores:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Borramos los procesos 0,1 y 2, con lo cual deberíamos tener:

3, 4, 5, 6, 7, 8, 9

Pero, en nuestra implementación quedaría de la siguiente manera:

0, 1, 2, 3, 4, 5, 6

Como mencionamos antes, sólo es posible eliminar procesos que se encuentren consecutivos en la tabla, por lo tanto, en el ejemplo anterior no podríamos borrar los procesos 3, 6 y 9 de forma simultánea, sino que tendríamos que borrar los procesos de a uno.

Ejecución

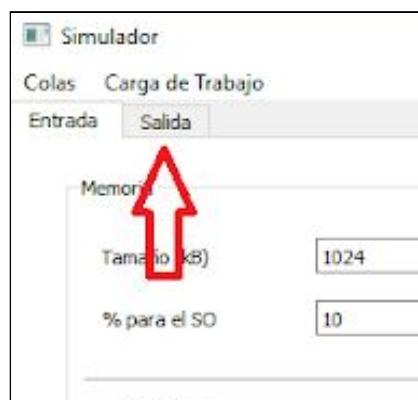
Una vez que tengamos todas las configuraciones establecidas, con respecto a los algoritmos de planificación y administración de memoria, particiones y procesos, podemos iniciar la ejecución del simulador seleccionando INICIAR.

Luego veremos que la opción de iniciar una nueva ejecución se encontrará inhabilitada. Por lo cual, si queremos realizar una nueva “corrida”, necesariamente debemos reiniciar el programa.



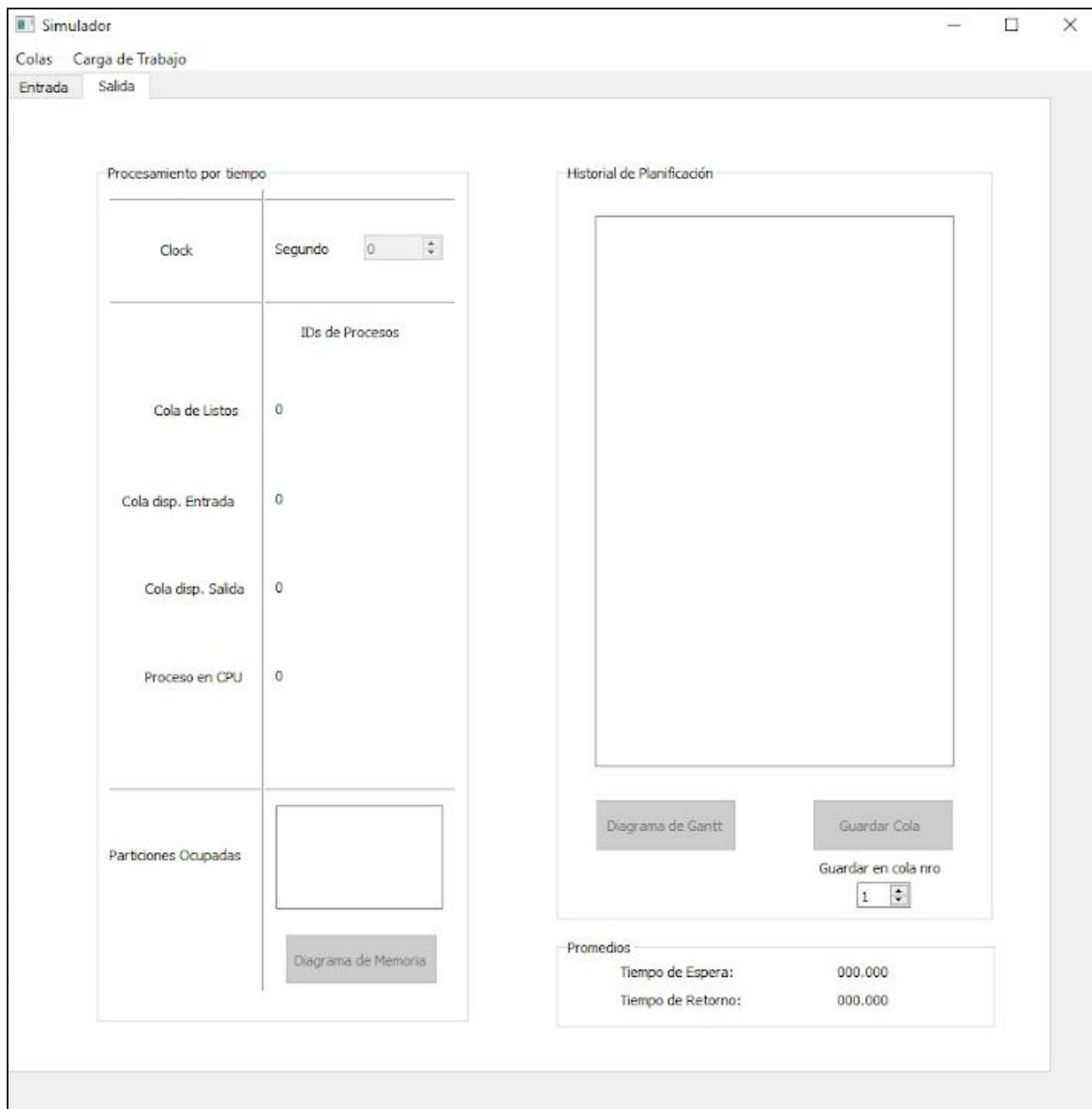
Esta opción de diseño, nos permite evitar eventuales controles en el caso que queramos realizar varias ejecuciones seguidas, simplificando mucho el código.

Para ver los resultados de la planificación, debemos dirigirnos a la pestaña de salida.



Resultados de salida

En la pestaña de salida o de resultados, antes de ejecutar vemos lo siguiente:

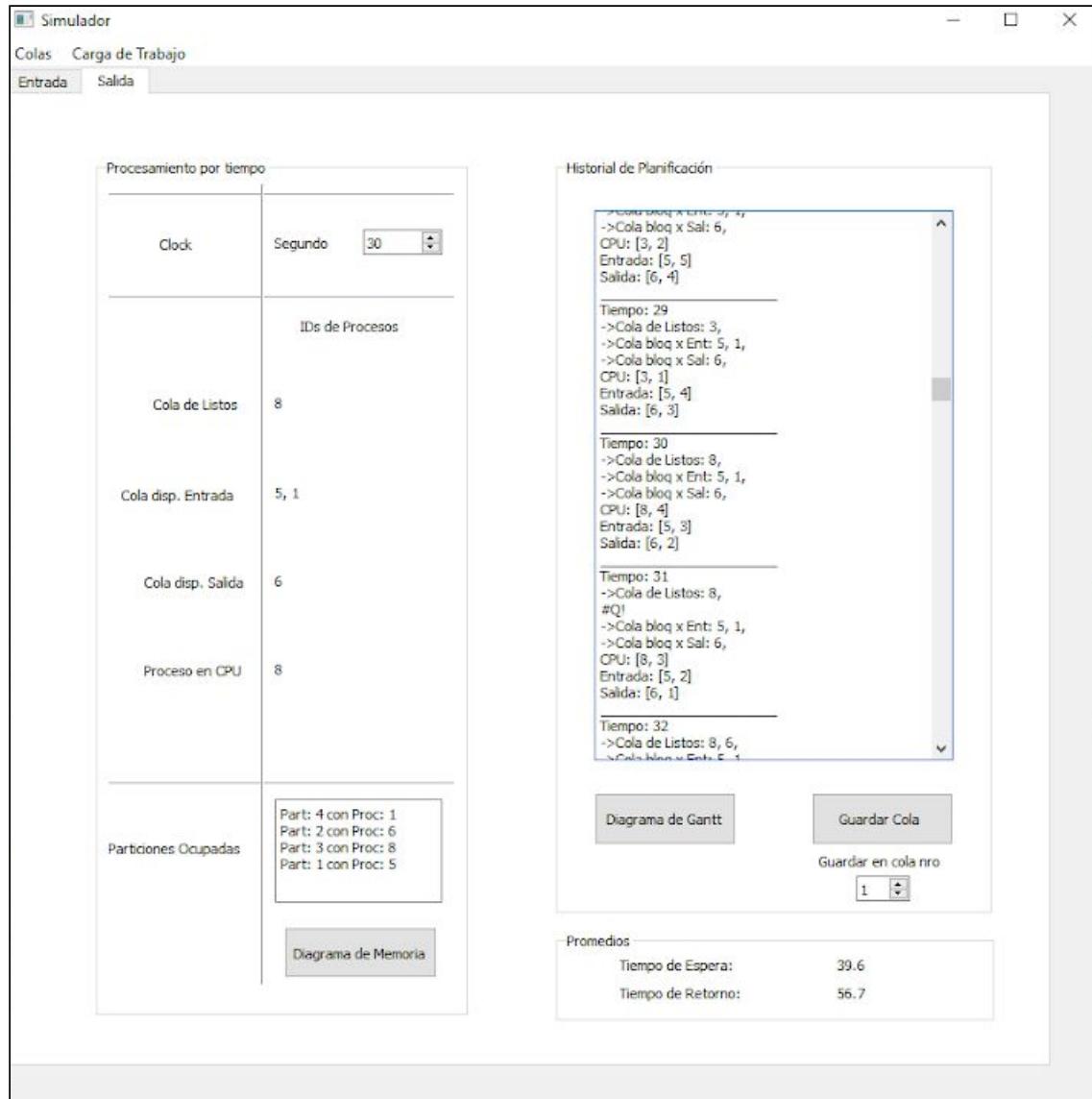


Antes de iniciar una ejecución del simulador, las siguientes opciones se encuentran deshabilitadas, debido a que hasta ese momento no se cuenta con los datos necesarios para realizar ciertas acciones:

- Clock
- Diagrama de Memoria
- Diagrama de Gantt
- Guardar Cola

Dichas opciones se habilitan luego realizar una ejecución del simulador, es decir, luego de seleccionar la opción de “INICIAR”.

En la siguiente imagen se ve un ejemplo de la ventana de Salida después de realizar una ejecución del simulador:



Procesamiento por Tiempo

Dentro de esta sección, encontraremos el espacio dedicado para la selección del instante de la simulación que queramos observar. Mediante el Clock podemos especificar el tiempo deseado. El valor del clock puede variar entre **0 y lo que dure la ejecución** total de los procesos, es decir, hasta el segundo en el cual el último proceso sale del procesador.

A continuación de esto, se mostrarán las colas de procesos listos, suspendidos y en CPU correspondientes a cada tiempo de simulación.

- Cola de Listos: procesos que están cargados en memoria y preparados para ser ejecutados una vez que sea su turno.

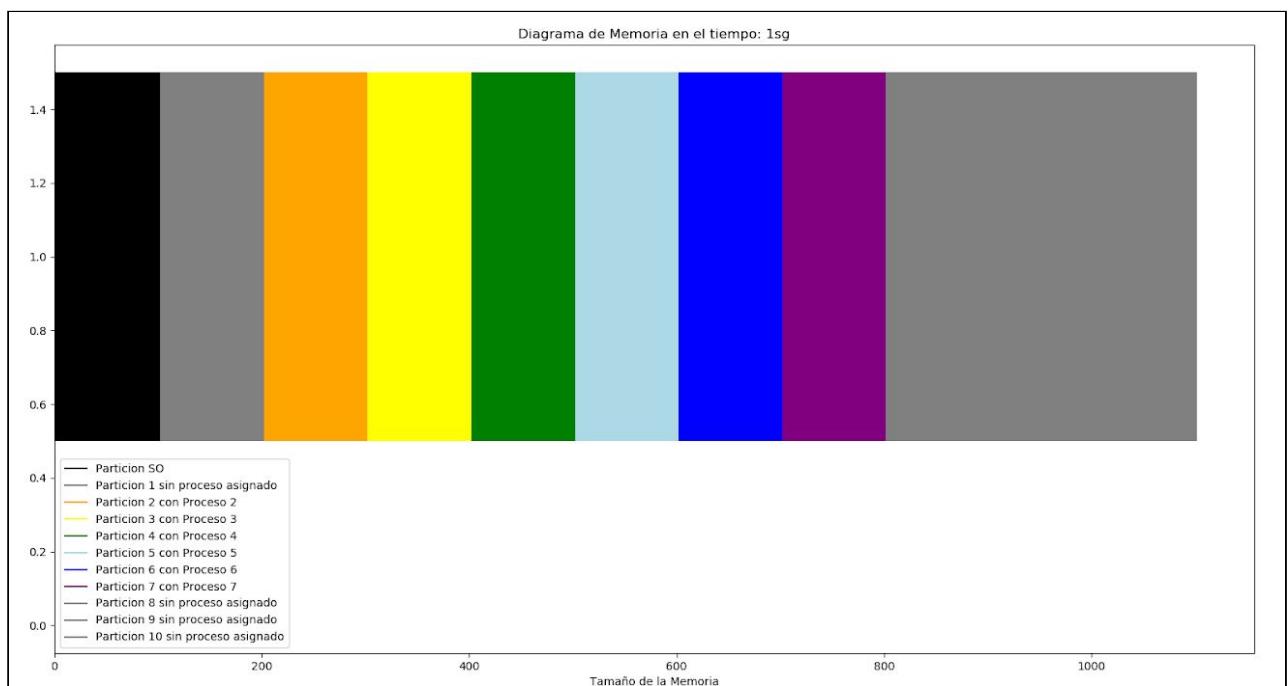
- Cola de Suspendidos: procesos que están en espera de realizar interacciones de entrada/salida con algún periférico.
- Proceso CPU: proceso que está actualmente siendo ejecutado en el procesador.

A medida que se modifica el valor del clock, las colas de los procesos y la lista de particiones se actualizan de acuerdo a los valores que tomaron en dicho instante durante la simulación. El valor del clock, como todo *spinbox*, se puede incrementar o decrementar sucesivamente, o bien, se puede ingresar directamente un valor de un instante de tiempo por teclado, para ver cómo se encuentran las colas en dicho instante.

Diagrama de Memoria

Un diagrama de memoria se corresponde a un instante de tiempo determinado. Para ver uno debemos seleccionar “Diagrama de Memoria” una vez finalizada la ejecución.

De esta manera podemos visualizar un gráfico que aparecerá en otra ventana, el cual muestra la distribución de los procesos en la memoria (ya sea si trabajamos con particiones fijas o variables), indicando con los diferentes colores los procesos que utilizan espacio en la memoria. La partición asignada al sistema operativo siempre aparecerá en color negro, y conjunto de particiones que no tengan asignados ningún proceso, en color gris.



Cada diagrama de memoria está compuesto por: un título, el cuál indica a qué instante de ejecución hace referencia, el eje horizontal que representa el tamaño total de la memoria, el eje vertical que no representa nada, y además contamos con una sección de referencias (ubicada en la esquina inferior izquierda), donde aparecen las particiones, si las mismas tienen asignados procesos o no, y eventualmente el proceso asignado.

Historial de Planificación

Se encuentra ubicado en la parte izquierda de la ventana Salida, en este se mostrarán, ordenados cronológicamente, los datos de los contenidos en la simulación.

Entre estos datos tenemos para cada tiempo de ejecución: los procesos que conforman las colas de listos, bloqueados por entrada, bloqueados por salida, también podemos ver qué proceso se está ejecutando en CPU, cuál en el dispositivo de entrada y cual en el dispositivo de salida. Vemos un ejemplo:

Historial de Planificación	
Entrada: [5, 10]	Salida: [6, 9]
Tiempo: 24	
->Cola de Listos: 3,	
->Cola bloq x Ent: 5, 1,	
->Cola bloq x Sal: 6,	
CPU: [3, 6]	
Entrada: [5, 9]	
Salida: [6, 8]	
Tiempo: 25	
->Cola de Listos: 3,	
->Cola bloq x Ent: 5, 1,	
->Cola bloq x Sal: 6,	
CPU: [3, 5]	
Entrada: [5, 8]	
Salida: [6, 7]	

Para el caso de cualquiera de las colas, solamente se muestran los identificadores de los procesos, en cambio, en las secciones de CPU, Entrada y Salida, además de los identificadores, se puede ver cuánto tiempo le falta al proceso para abandonar el dispositivo, es decir, el tiempo de remanencia. Por ejemplo, en la imagen anterior: en el instante 24, el proceso que se encuentra ejecutándose en la CPU es el 3, y le quedan 6 unidades de tiempo para abandonarla.

En resumen, podemos analizar cómo fue la ejecución de dos maneras: mediante la sección “Procesamiento por tiempo”, donde de acuerdo al instante que indiquemos en el clock, se verán varios aspectos de la simulación en ese momento, o bien, mediante el “Historial de planificación”, donde se podemos ver un registro de la misma.

Procesamiento por tiempo	
Clock	Segundo <input type="button" value="28"/>
	IDs de Procesos
Cola de Listos	3
Cola disp. Entrada	5, 1
Cola disp. Salida	6
Proceso en CPU	3
Particiones Ocupadas	Part: 4 con Proc: 1 Part: 2 con Proc: 6 Part: 3 con Proc: 3 Part: 1 con Proc: 5
	<input type="button" value="Diagrama de Memoria"/>

Guardar en cola nro

Historial de Planificación

```

->Cola bloq x Ent: 5, 1,
->Cola bloq x Sal: 6,
CPU: [3, 4]
Entrada: [5, 7]
Salida: [6, 6]

Tiempo: 27
->Cola de Listos: 3,
->Cola bloq x Ent: 5, 1,
->Cola bloq x Sal: 6,
CPU: [3, 3]
Entrada: [5, 6]
Salida: [6, 5]

Tiempo: 28
->Cola de Listos: 3,
->Cola bloq x Ent: 5, 1,
->Cola bloq x Sal: 6,
CPU: [3, 2]
Entrada: [5, 5]
Salida: [6, 4]

Tiempo: 29
->Cola de Listos: 3,
->Cola bloq x Ent: 5, 1,
->Cola bloq x Sal: 6,
CPU: [3, 1]
Entrada: [5, 4]
Salida: [6, 3]

Tiempo: 30
->Cola de Listos: 8,
->Cola bloq x Ent: 5, 1,

```

Promedios

Tiempo de Espera:	39.6
Tiempo de Retorno:	56.7

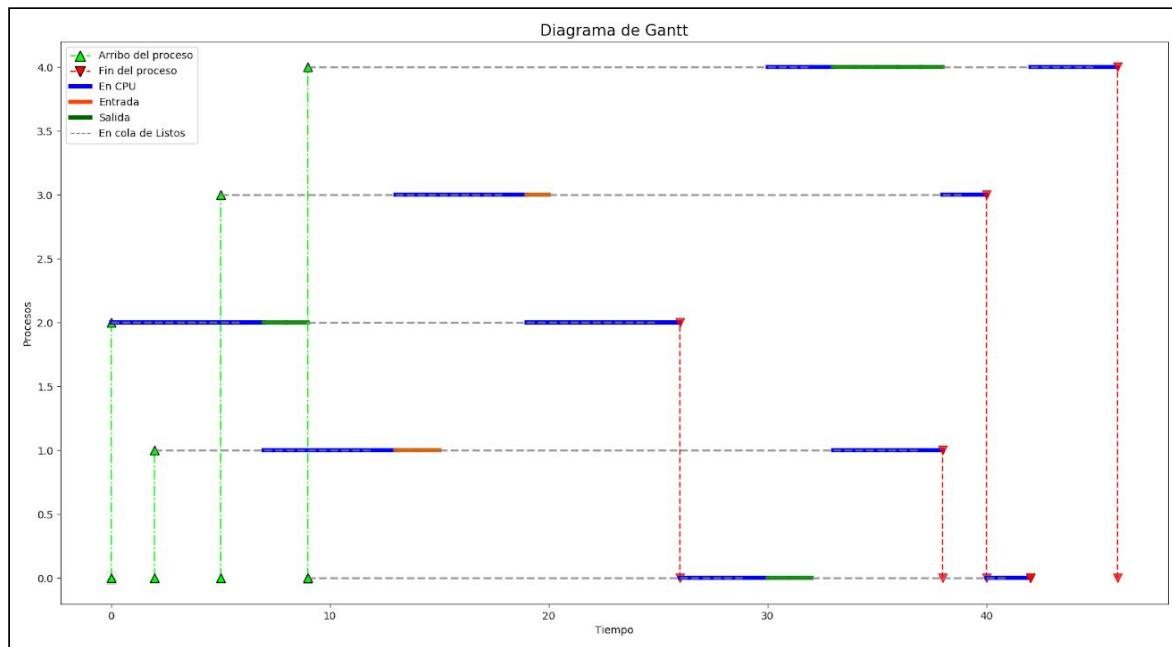
Obviamente existe una correspondencia entre ambas secciones de acuerdo a los resultados mostrados para el mismo tiempo de ejecución, la principal diferencia entre ambas es qué en “Procesamiento por tiempo” podemos ver las particiones utilizadas y en el “Historial de Planificación” no, sin embargo, en el “Historial de Planificación” podemos ver el tiempo de remanencia del proceso en el dispositivo y “Procesamiento por tiempo” no.

En la sección inferior del Historial de Planificación tenemos las siguientes opciones: “Diagrama de Gantt” y “Guardar Cola”.



Diagrama de Gantt

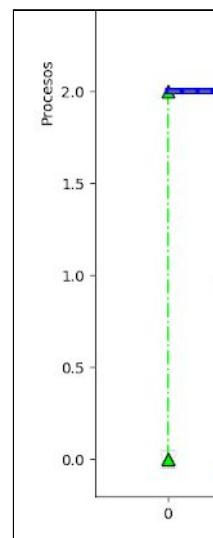
Al seleccionar Diagrama de Gantt se abre una ventana que muestra justamente el diagrama de Gantt resultante de la simulación, a continuación vemos un ejemplo:



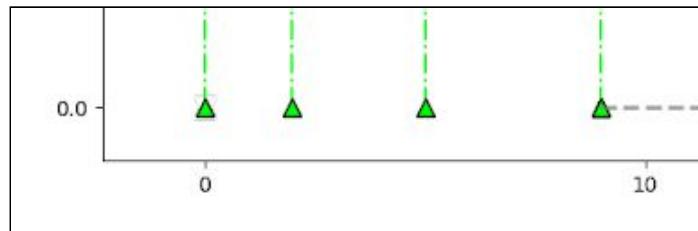
Para interpretar correctamente el diagrama se debe tener en cuenta lo siguiente:

El eje horizontal representa el tiempo transcurrido durante la ejecución (el cual toma los valores desde 0 hasta el tiempo de salida del último proceso), y el eje vertical hace referencia a los identificadores de los procesos.

El arribo de un proceso se representa con la línea de segmentos vertical de color verde claro, la cual va siempre desde la ordenada 0, hasta la ordenada correspondiente al proceso. Por ejemplo, en la imagen anterior, vemos que el proceso 2 arribo en el instante 0.

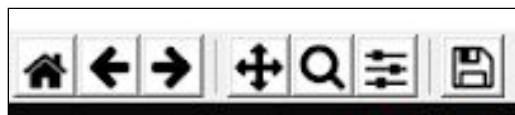


Existe una clara dificultad a la hora de querer saber determinados valores de tiempo en el eje horizontal, que no coinciden con los valores mostrados. Por ejemplo, en la siguiente imagen no se puede apreciar en específicamente el tiempo de arriba algunos procesos, vemos un ejemplo:



Claramente vemos que dichos arribos están entre 0 y 10, pero necesitamos más precisión.

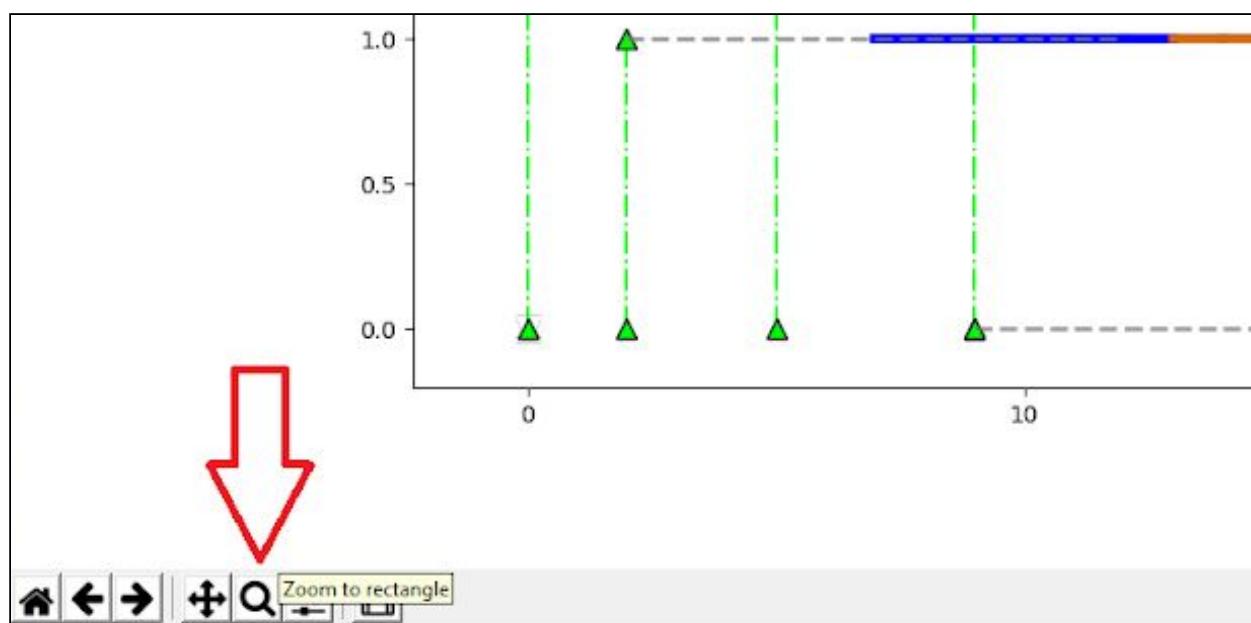
Para solucionar esté inconveniente, la librería gráfica utilizada para la realización de los diagramas nos provee la siguiente barra de herramientas, ubicada en la esquina inferior izquierda de la ventana:



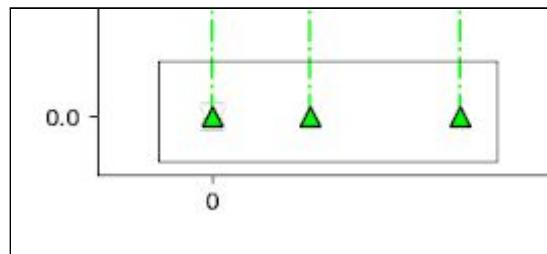
A nuestros fines, sólo nos interesan las opciones de la “casa” y de la “lupa”.



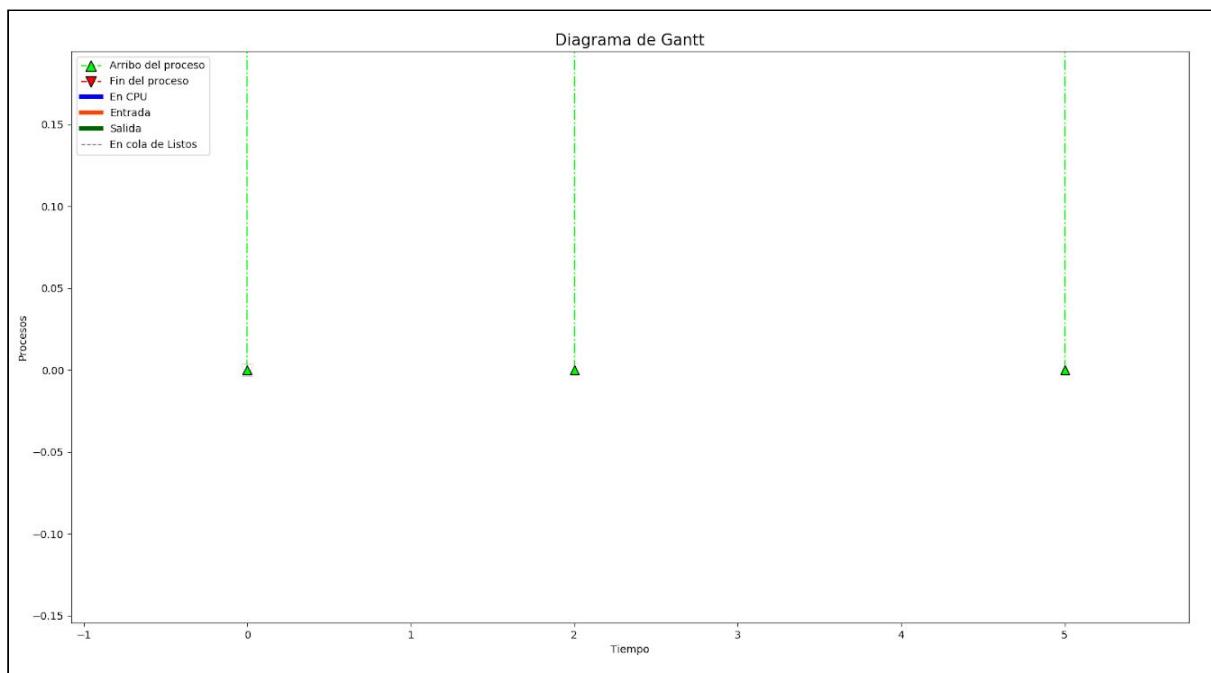
La lupa nos sirve para hacer un *zoom* en el gráfico, entonces podemos apreciar mejor algunas cosas. Para utilizarla debemos primero seleccionarla:



Luego, con el ratón, hay que escoger la zona rectangular que queramos ampliar, de la siguiente manera:



Una vez seleccionada, veremos lo siguiente:



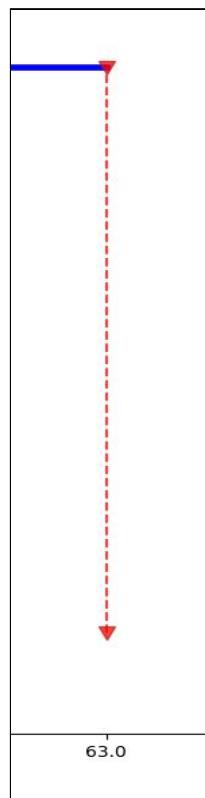
La parte del diagrama que hemos seleccionado ahora ocupa toda la pantalla, con lo cual es posible ver, con una mayor facilidad, los valores que antes no podíamos apreciar.

Una vez que hayamos visto los datos que necesitábamos, seleccionamos la opción de la "casa", la cual restaura el gráfico a su forma original, es decir, sin ampliaciones.

Aclarado esto, podemos continuar con la descripción del diagrama.

Una vez que el proceso arribó, su ciclo de vida es representado de forma horizontal.

El fin de un proceso también se representa con una línea vertical, pero en este caso, de color rojo, la cual va desde la ordenada correspondiente al número de proceso hasta 0. Vemos un ejemplo:

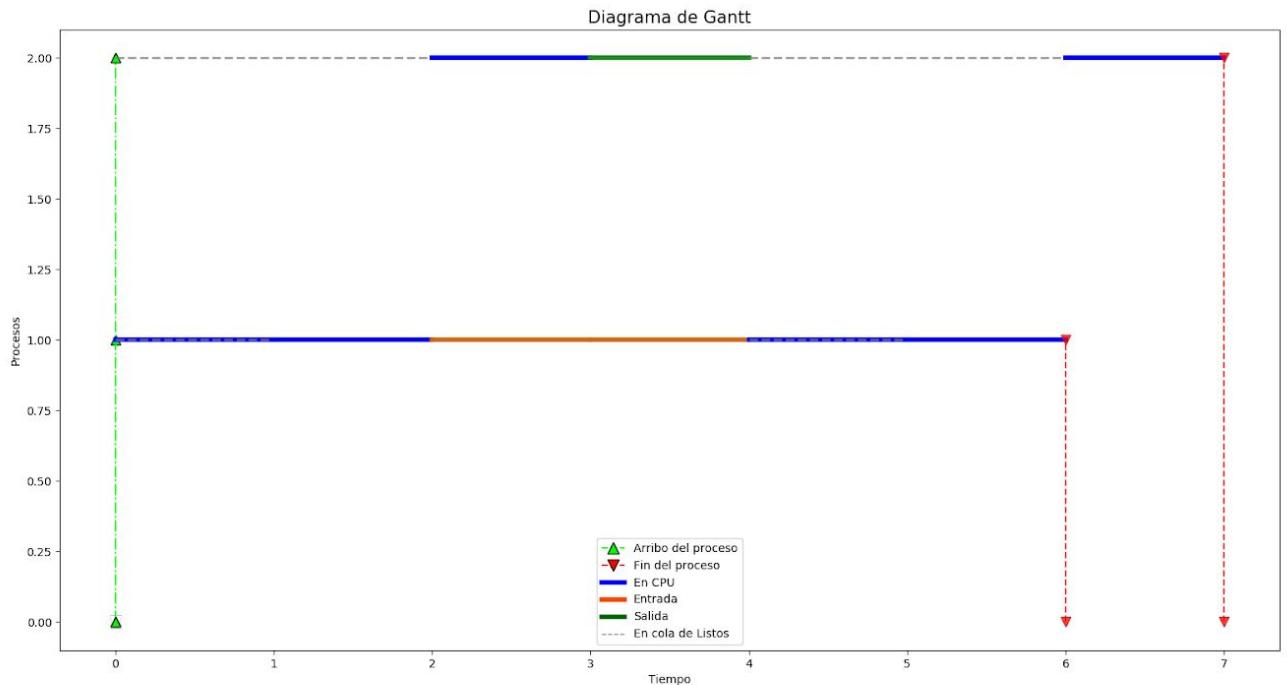


Usamos las siguientes referencias para representar el ciclo de vida de los procesos indicando la interpretación de cada símbolo y color del diagrama:



La ubicación del cuadro anterior depende de cómo sea la disposición del diagrama, y siempre se ubica en algún espacio libre, con el fin de no causar una molestia al momento de apreciar el diagrama.

Para comprender mejor nuestro Diagrama de Gantt, veremos el siguiente ejemplo:

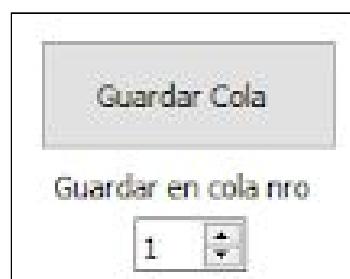


El diagrama anterior cuenta simplemente con dos procesos:

- El proceso 1 arriba en el tiempo 0, tiene 3 rafagas, (CPU - Entrada - CPU) y finaliza en el tiempo 6.
- El proceso 2 también arriba en el tiempo 0, con lo cual las líneas que representan el arribo de ambos procesos son coincidentes. También tiene 3 rafagas (CPU - Salida - CPU). Vemos qué al finalizar su rafaga de Salida, el procesador se encuentra ocupado ejecutando el proceso 1, con lo cual se ubica en la cola de listos hasta que el dicho proceso termine de usar el recurso. Una vez que el proceso 1 abandona la CPU, el proceso 2 se ejecuta y posteriormente finaliza en el tiempo 7.

Guardar cola

La última opción de dicha sección es “Guardar Cola”, la cual justamente permite guardar la cola de procesos que se acaba de utilizar en la simulación.



Existen 3 colas disponibles en la base de datos, como veremos más adelante en la sección “Base de datos”, estas colas se generan automáticamente al iniciar la ejecución y están cargadas con procesos con valores “aleatorios”.

Si uno desea guardar una cola, debe elegir una de las existentes para reemplazarla. Con esto queremos decir que siempre en el programa existirán 3 colas, al inicio son generadas automáticamente por el mismo simulador, pero uno puede reemplazar alguna de esas 3 por la que desea guardar. Por lo tanto, sólo tenemos la posibilidad de guardar 3 colas diferentes.

Veamos el siguiente ejemplo: Al iniciar el programa, cargamos la cola 2 de la base de datos con la opción de cargar colas vista anteriormente, tenemos entonces:

Lista de procesos cargados				
Proceso	Tamaño	Arribo	Prioridad	Rafagas
0	10	7	B	CPU: 10; S: 10; CPU: 6;
1	16	7	B	CPU: 1; E: 10; CPU: 10;
2	28	1	B	CPU: 9; S: 7; CPU: 2;
3	13	3	A	CPU: 7; E: 8; CPU: 10;
4	11	2	A	CPU: 4; S: 9; CPU: 3;

También somos capaces de agregar, modificar y eliminar procesos de dicha cola.

Para el ejemplo:

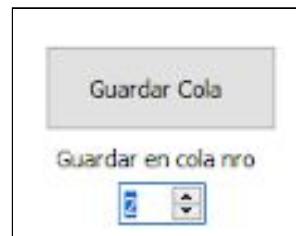
- Eliminaremos el proceso 2
- Modificaremos el tiempo arribo del proceso 0, sustituyendo dicho valor por 0
- Agregaremos 2 procesos con rafagas simples

La tabla de procesos queda de la siguiente manera:

Lista de procesos cargados				
Proceso	Tamaño	Arribo	Prioridad	Rafagas
0	10	0	B	CPU: 10; S: 10; CPU: 6;
1	16	7	B	CPU: 1; E: 10; CPU: 10;
2	13	3	A	CPU: 7; E: 8; CPU: 10;
3	11	2	A	CPU: 4; S: 9; CPU: 3;
6	10	10	M	CPU: 10;
7	10	10	A	CPU: 10;

Después de realizar una corrida del simulador, podemos guardar dicha cola de procesos en la base de datos. Cómo dijimos antes, se puede elegir entre cualquiera de las 3 colas, en

esté caso reemplazamos la número 2, entonces indicamos esté valor en el *spinbox* y seleccionamos “Guardar Cola”:

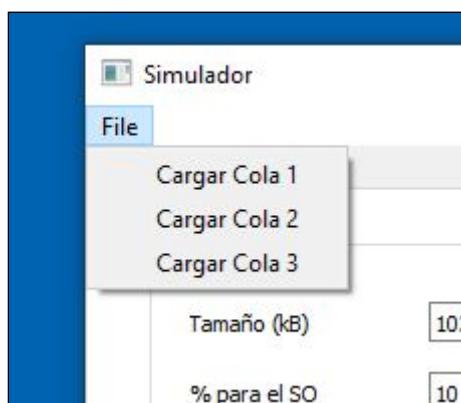


Vemos que sí queremos utilizar la cola 2 de procesos nuevamente en otra ejecución, podemos hacerlo cargando nuevamente la misma.

Base de datos:

Para la implementación de la base de datos utilizamos SQL, mediante la herramienta “SQLite”, debido a que varios integrantes del grupo tienen conocimientos avanzados con el lenguaje y que dicho programa es sencillo de descargar, conectar, cargar datos, consultar.

Antes de querer cargar una cola desde o hacia la base de datos, debemos tener instalado el programa SQLite. La descripción del proceso de instalación se encuentra en la misma carpeta que el simulador. (ver Tutorial)



Cuando iniciamos el simulador y queremos cargar una cola existente de procesos desde la base de datos, se generar 3 colas diferentes, con 5 procesos cada una.

Las características de cada proceso de eligen de forma aleatoria bajo ciertas reglas.

- El ID del proceso es un campo autoincremental, no tiene mayores complicaciones
- El tamaño del mismo, para estos casos, varía entre 1 y 30
- La prioridad del proceso obviamente puede ser Alta, Media o Baja
- El tiempo de arribo varía entre 1 y 10

Todos los procesos comienzan con una rafaga de CPU, y luego dependiendo del campo autoincremental ID de proceso, un proceso tiene una rafaga de Entrada o de Salida, cuando el identificador es un número par: el proceso tiene las siguientes rafagas:

CPU - S - CPU

En cambio, cuando el identificador es impar, el proceso tiene las siguientes rafagas:

CPU - E - CPU

Cada uno de los tiempos de cada rafaga también es aleatorio, y varía entre 1 y 10.

A continuación vemos un ejemplo de una Lista de procesos cargados mediante la elección de una cola de la base de datos.

Lista de procesos cargados				
Proceso	Tamaño	Arribo	Prioridad	Rafagas
0	12	9	A	CPU: 4; S: 2; CPU: 2;
1	1	2	B	CPU: 6; E: 2; CPU: 5;
2	3	0	B	CPU: 7; S: 2; CPU: 7;
3	2	5	B	CPU: 6; E: 1; CPU: 2;
4	8	9	M	CPU: 3; S: 5; CPU: 4;

Cabe aclarar que cada uno de los campos de la lista anterior puede ser modificado desde dicha lista si se desea, siempre y cuando se cumpla con las restricciones mencionadas.

En la siguiente imagen vemos la estructura de tabla que usamos:

Database Structure Browse Data Edit Pragmas Execute SQL

Table: procesos

New Record Delete Record

	id_conjunto	idp	tamaño	arribo	prioridad	rafagas
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	1	28	6	M	CPU: 3; E: 9; CPU: 10;
2	1	2	6	10	A	CPU: 2; S: 3; CPU: 9;
3	1	3	22	5	B	CPU: 3; E: 5; CPU: 9;
4	1	4	10	10	A	CPU: 7; S: 1; CPU: 7;
5	1	5	6	9	B	CPU: 5; E: 10; CPU: 5;
6	1	6	13	3	B	CPU: 7; S: 9; CPU: 7;
7	1	7	28	6	A	CPU: 6; E: 4; CPU: 7;
8	1	8	30	9	M	CPU: 2; S: 5; CPU: 9;
9	1	9	3	5	A	CPU: 4; E: 5; CPU: 1;
10	1	10	9	10	A	CPU: 6; S: 4; CPU: 1;
11	2	1	29	4	A	CPU: 3; E: 10; CPU: 4;
12	2	2	4	6	A	CPU: 1; S: 8; CPU: 8;
13	2	3	25	4	M	CPU: 2; E: 3; CPU: 3;
14	2	4	26	7	M	CPU: 2; S: 8; CPU: 2;
15	2	5	13	9	M	CPU: 6; E: 5; CPU: 4;
16	2	6	7	4	A	CPU: 7; S: 5; CPU: 5;
17	2	7	22	6	M	CPU: 4; E: 6; CPU: 8;
18	2	8	25	2	B	CPU: 2; S: 5; CPU: 5;
19	2	9	9	10	M	CPU: 7; E: 8; CPU: 7;
20	2	10	22	7	M	CPU: 10; S: 2; CPU: 8;
21	3	1	12	7	B	CPU: 7; E: 5; CPU: 9;
22	3	2	30	6	A	CPU: 8; S: 2; CPU: 7;
23	3	3	9	0	A	CPU: 3; E: 10; CPU: 1;
24	3	4	20	10	A	CPU: 8; S: 8; CPU: 4;
25	3	5	6	10	A	CPU: 1; E: 4; CPU: 2;
26	3	6	2	8	B	CPU: 3; S: 1; CPU: 7;
27	3	7	27	1	A	CPU: 4; E: 6; CPU: 10;
28	3	8	29	11	M	CPU: 9; S: 9; CPU: 1;
29	3	9	6	10	M	CPU: 10; E: 1; CPU: 7;

La tabla con los siguiente campos: id_conjunto, idp, tamaño, arribo, prioridad y rafagas.

El “id_conjunto” hace referencia a la cola a la que pertenece el proceso, por lo cual cada registro tiene una clave compuesta, formada por la id_conjunto y su propio idp (identificador de proceso). Con esto logramos tener idénticos idp en diferentes colas.

Ejercicio de prueba:

Utilizamos los datos (tabla de procesos y particiones) del ejercicio número 3 de la guía de Trabajos Prácticos: Administración de Memoria Contigua para realizar una corrida del simulador que permita verificar si los resultados obtenidos al hacerlo en forma manual son los mismos.

Tabla de procesos:

TR	TA	TI	TAM(KB)
1	0	5	20
2	0	4	5
3	0	10	14
4	0	3	25
5	0	2	3
6	0	10	30
7	0	5	25
8	0	5	70

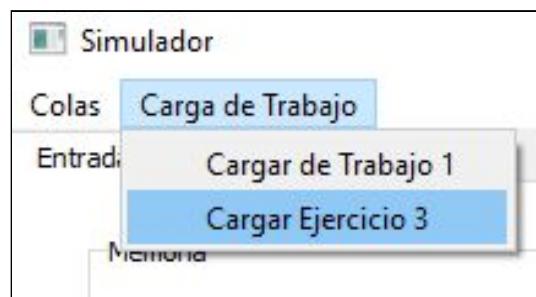
Los siguientes son los tamaños establecidos para los trabajos que se ejecutan normalmente en el centro de cómputos:

SISTEMA OPERATIVO 32K
TRABAJOS MUY PEQUEÑOS 10K
TRABAJOS PROMEDIOS 25K
TRABAJOS MUY GRANDES 70K

- Indique como se realizaría la asignación de memoria según un método de asignación Particiones Fijas que usa un algoritmo Best-Fit. El algoritmo de planificación de CPU es SRTF
- Indique como se realizaría la asignación de memoria según un método de asignación Particiones variables que usa un algoritmo Worst-Fit. El algoritmo de planificación de CPU es SRTF

Debido a que no implementamos el algoritmo SRTF, para la resolución utilizamos Round-Robin con un quantum de 5.

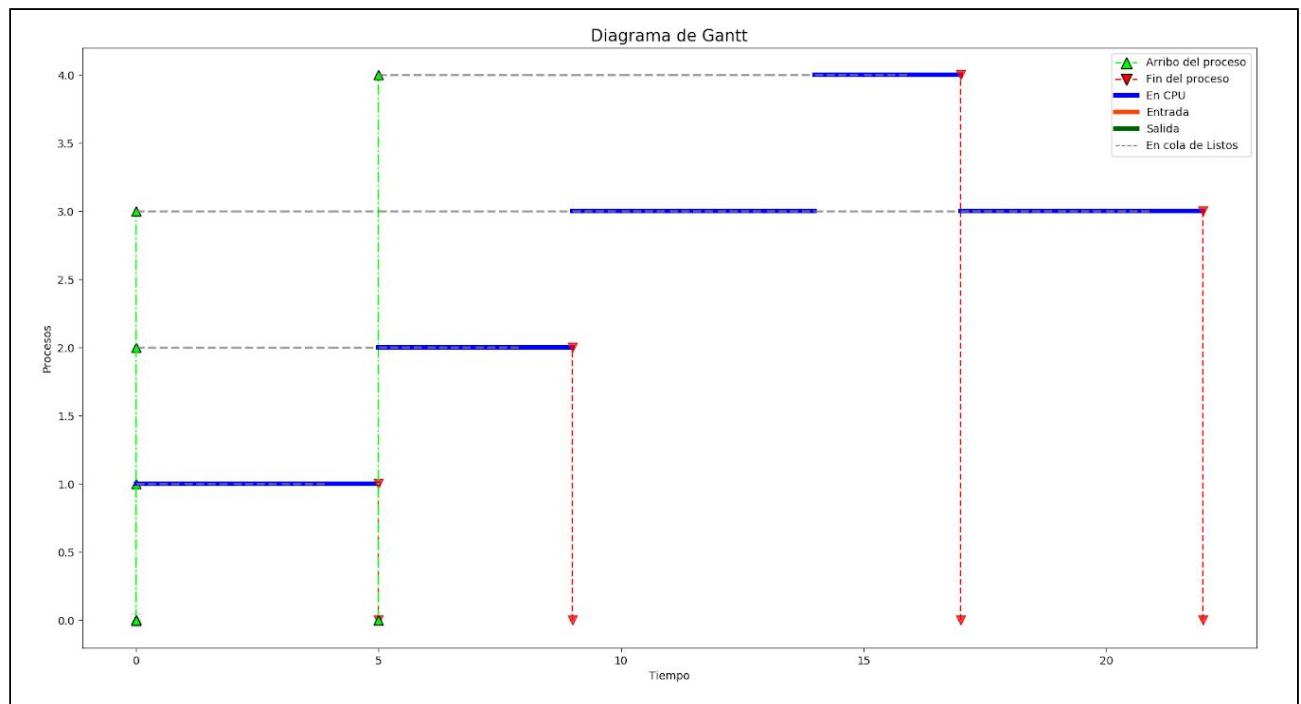
Para realizar la carga de trabajo de forma rápida, creamos el siguiente “acceso directo”:



Al seleccionar esta opción, veremos lo siguiente en la pestaña de entrada del simulador:

<p>Memoria</p> <p>Tamaño (kB) <input type="text" value="137"/></p> <p>% para el SO <input type="text" value="23"/></p> <hr/> <p>Particiones</p> <p><input type="radio"/> Variables <input checked="" type="radio"/> Fijas</p> <p>Tamaño (kB) <input type="text" value="1"/></p> <p>Agregar Eliminar</p> <table border="1" style="width: 100px; border-collapse: collapse;"> <tr><td>Número</td><td>Tamaño</td></tr> <tr><td>1</td><td>10</td></tr> <tr><td>2</td><td>25</td></tr> <tr><td>3</td><td>70</td></tr> </table> <p>Espacio Ocupado <div style="background-color: green; width: 100px; height: 10px; display: inline-block;"></div> 100%</p> <p>Espacio Disponible (kB): 0</p> <hr/> <p>Método de Asignación</p> <p><input type="radio"/> First Fit <input checked="" type="radio"/> Best Fit <input type="radio"/> Worst Fit</p> <hr/> <p>Algoritmo</p> <p>Round Robin with Quantum (RRQ) <input type="button" value="▼"/></p> <p>Quantum <input type="text" value="5"/></p>	Número	Tamaño	1	10	2	25	3	70	<p>Procesos</p> <p>Nuevo Proceso</p> <table border="1" style="width: 100px; border-collapse: collapse;"> <tr><td>ID Proceso</td><td><input type="text" value="1"/></td><td>Tamaño (kB)</td><td><input type="text" value="1"/></td><td>Arribo</td><td><input type="text" value="0"/></td></tr> <tr><td>Ciclo de Vida</td><td><input type="radio"/> Entrada</td><td><input type="radio"/> Salida</td><td>Agregar</td></tr> <tr><td colspan="6"><table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>CPU</td><td></td></tr> <tr><td>0</td><td></td></tr> </table></td></tr> </table> <p>Prioridad <input type="radio"/> Elegir <input checked="" type="radio"/> Aleatoria</p> <p>Agregar Proceso</p> <p>Lista de procesos cargados</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Proceso</th><th>Tamaño</th><th>Arribo</th><th>Prioridad</th><th>Rafagas</th></tr> </thead> <tbody> <tr><td>1</td><td>20</td><td>0</td><td>M</td><td>CPU: 5;</td></tr> <tr><td>2</td><td>5</td><td>0</td><td>M</td><td>CPU: 4;</td></tr> <tr><td>3</td><td>14</td><td>0</td><td>M</td><td>CPU: 10;</td></tr> <tr><td>4</td><td>25</td><td>0</td><td>M</td><td>CPU: 3;</td></tr> </tbody> </table> <p>Borrar Selección Borrar Tabla INICIAR</p>	ID Proceso	<input type="text" value="1"/>	Tamaño (kB)	<input type="text" value="1"/>	Arribo	<input type="text" value="0"/>	Ciclo de Vida	<input type="radio"/> Entrada	<input type="radio"/> Salida	Agregar	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>CPU</td><td></td></tr> <tr><td>0</td><td></td></tr> </table>						CPU		0		Proceso	Tamaño	Arribo	Prioridad	Rafagas	1	20	0	M	CPU: 5;	2	5	0	M	CPU: 4;	3	14	0	M	CPU: 10;	4	25	0	M	CPU: 3;
Número	Tamaño																																																					
1	10																																																					
2	25																																																					
3	70																																																					
ID Proceso	<input type="text" value="1"/>	Tamaño (kB)	<input type="text" value="1"/>	Arribo	<input type="text" value="0"/>																																																	
Ciclo de Vida	<input type="radio"/> Entrada	<input type="radio"/> Salida	Agregar																																																			
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>CPU</td><td></td></tr> <tr><td>0</td><td></td></tr> </table>						CPU		0																																														
CPU																																																						
0																																																						
Proceso	Tamaño	Arribo	Prioridad	Rafagas																																																		
1	20	0	M	CPU: 5;																																																		
2	5	0	M	CPU: 4;																																																		
3	14	0	M	CPU: 10;																																																		
4	25	0	M	CPU: 3;																																																		

Seleccionamos “INICIAR” y obtenemos el siguiente Diagrama de Gantt:



Al realizar el ejercicio a mano obtuvimos el mismo resultado:

