Simulador – Gestión de memoria:

"Memoria Contigua y Paginada"

Manual de Usuario

Juan Pablo Cielo Coyotl
Jesús David Sahinos Gaona

Apartado de Memoria Contigua

I. Introducción a los Algoritmos de Gestión de Memoria

En sistemas operativos y gestión de memoria dinámica, los algoritmos de asignación de memoria son fundamentales para determinar cómo se asignan y se utilizan los bloques de memoria. Los algoritmos más comunes incluyen el **mejor ajuste**, **peor ajuste**, **siguiente ajuste** y **primer ajuste**. Estos algoritmos se utilizan para gestionar eficientemente la memoria disponible en un sistema, minimizando la fragmentación y optimizando el uso de los recursos.

a) Algoritmo de Mejor Ajuste (Best Fit)

El algoritmo de mejor ajuste busca el bloque de memoria libre más pequeño que sea suficiente para satisfacer la solicitud de memoria. Es decir, encuentra el espacio libre que más se aproxime al tamaño necesario, minimizando el espacio desperdiciado.

Ventajas:

- Reduce el desperdicio de memoria al minimizar los fragmentos sobrantes pequeños.

Desventajas:

- Puede ser más lento que otros algoritmos debido a la necesidad de buscar entre todos los bloques disponibles.
- Puede llevar a una mayor fragmentación externa a largo plazo.

b) Algoritmo de Peor Ajuste (Worst Fit)

El algoritmo de peor ajuste selecciona el bloque de memoria libre más grande disponible para satisfacer la solicitud. La idea es dejar los bloques más pequeños disponibles para solicitudes futuras que puedan encajar mejor en ellos.

Ventajas:

- Puede reducir la fragmentación externa al dejar bloques más pequeños disponibles para futuras solicitudes.

Desventajas:

- Generalmente resulta en un uso menos eficiente de la memoria porque puede dejar grandes fragmentos de memoria parcialmente usados.

c) Algoritmo de Primer Ajuste (First Fit)

El algoritmo de primer ajuste escanea la memoria desde el principio y asigna el primer bloque de memoria libre que sea lo suficientemente grande para satisfacer la solicitud.

Ventajas:

- Es rápido, ya que encuentra un bloque adecuado sin necesidad de escanear toda la lista.

- Suele ser más simple de implementar.

Desventajas:

- Puede dejar pequeños fragmentos de memoria al principio de la lista, lo que puede aumentar la fragmentación externa.

d) Algoritmo de Siguiente Ajuste (Next Fit)

El algoritmo de siguiente ajuste es una variación del primer ajuste. En lugar de comenzar siempre desde el principio de la lista de bloques libres, continúa la búsqueda desde el último punto donde se encontró un bloque libre adecuado.

Ventajas:

- Distribuye la búsqueda a lo largo de la memoria, evitando la acumulación de fragmentos en la parte inicial de la lista.

Desventajas:

- Similar al primer ajuste en términos de eficiencia y fragmentación, pero puede ser ligeramente mejor en términos de distribución de fragmentos.

Comparación y Uso

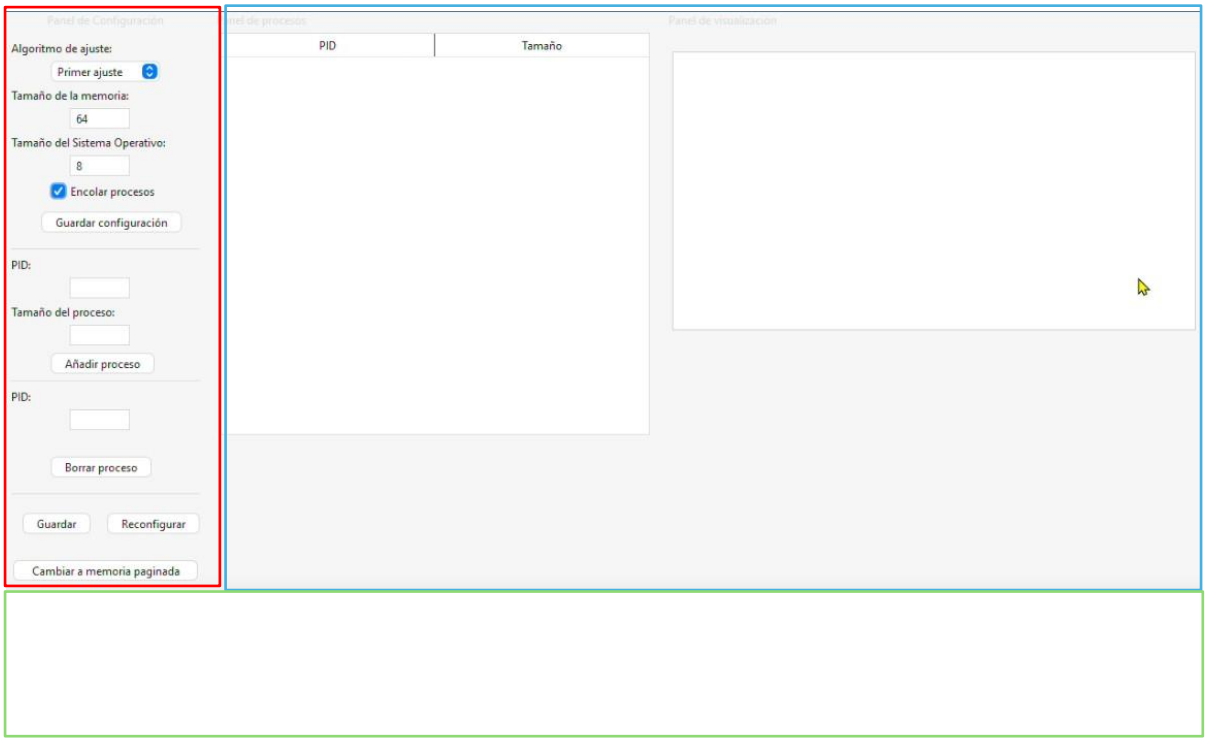
- **Mejor Ajuste** es ideal cuando se desea minimizar el desperdicio de memoria, pero puede ser computacionalmente costoso.
- **Peor Ajuste** puede ser útil para dejar grandes bloques de memoria libres, pero puede llevar a un uso ineficiente de la memoria.
- **Primer Ajuste** es rápido y simple, pero puede llevar a una mayor fragmentación.
- **Siguiente Ajuste** busca balancear entre la eficiencia del primer ajuste y la distribución de fragmentos, aunque sus beneficios pueden ser marginales en comparación.

II. Como usar el simulador (parte 1)

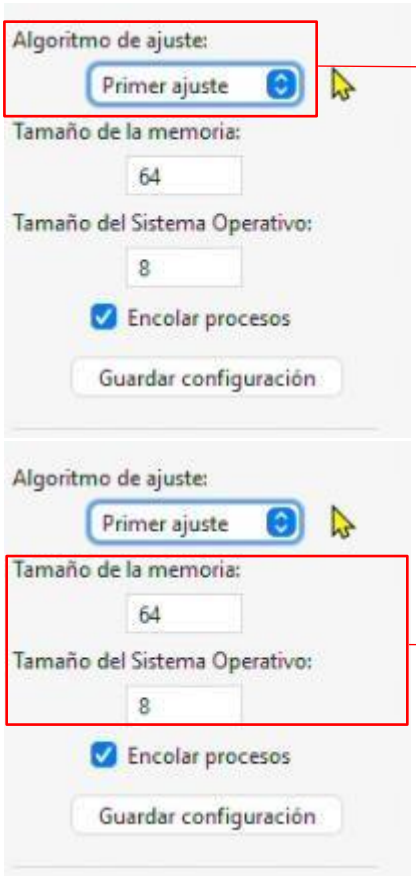
Primero que nada, es importante conocer la interfaz, y como tal lo que manejaras como usuario se encuentra enmarcado en rojo, donde podrás ingresar los datos para manejar el simulador.

Enmarcado en azul se encuentra la parte de los resultados del simulador y en verde el panel para la memoria gráfica.

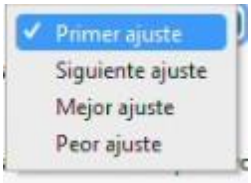
Mas adelante se explicará como ingresar datos, así como lo que representa de manera más especifica cada apartado.



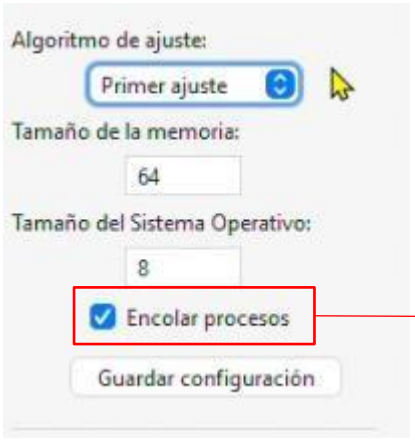
Al momento de ingresar datos debes tener en cuenta:



Aquí es donde se puede elegir el algoritmo con el que se trabajara

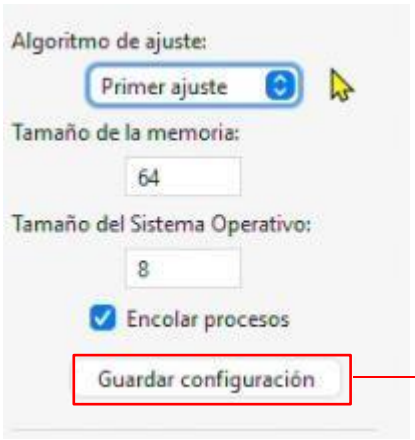


Aquí es donde podrá ingresar tanto el tamaño de la memoria como el del sistema operativo



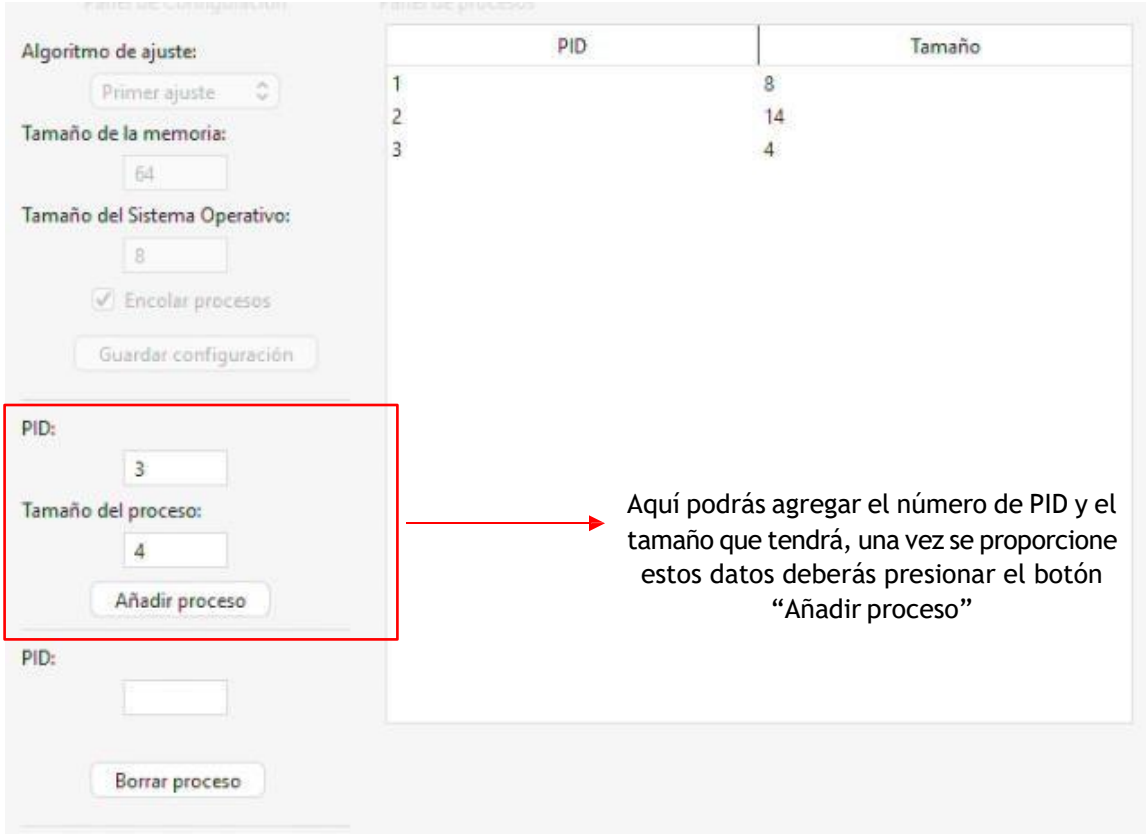
Aquí puedes elegir si quieres encolar el proceso.

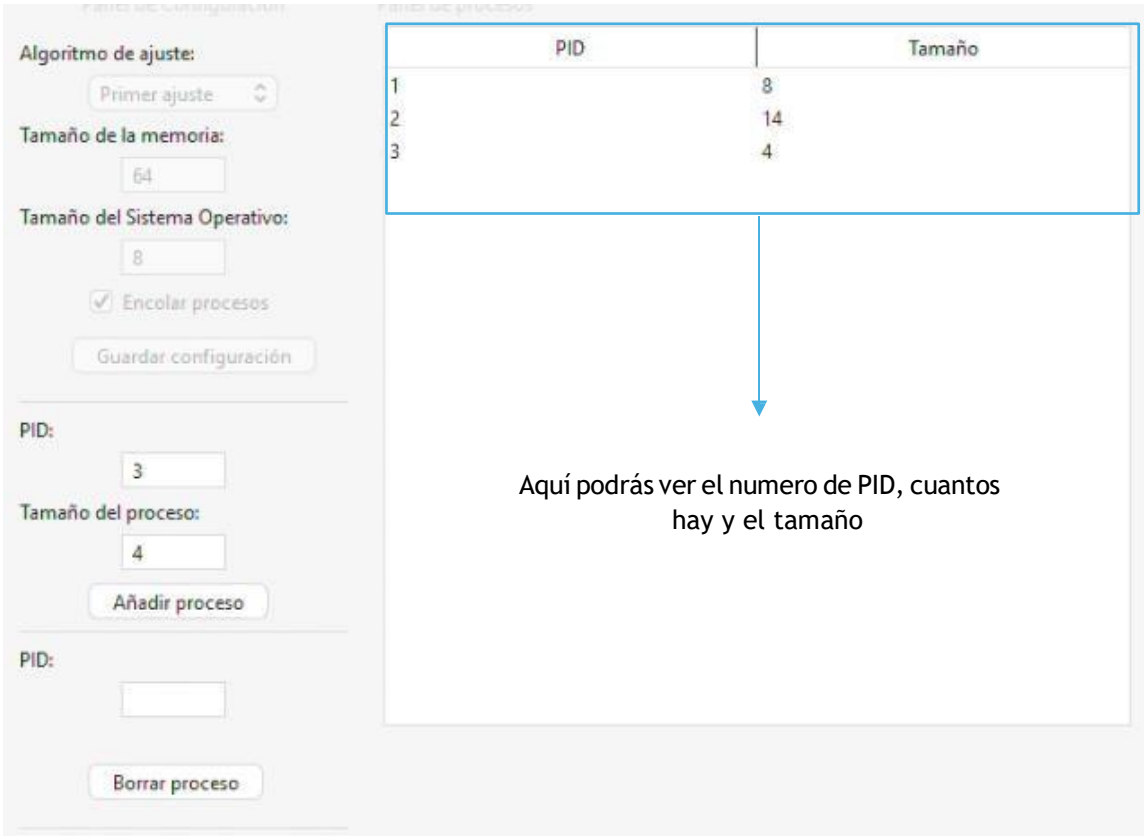
Encolar un proceso significa agregar un proceso a una cola de espera específica. Esto es parte fundamental de la administración y planificación de procesos.



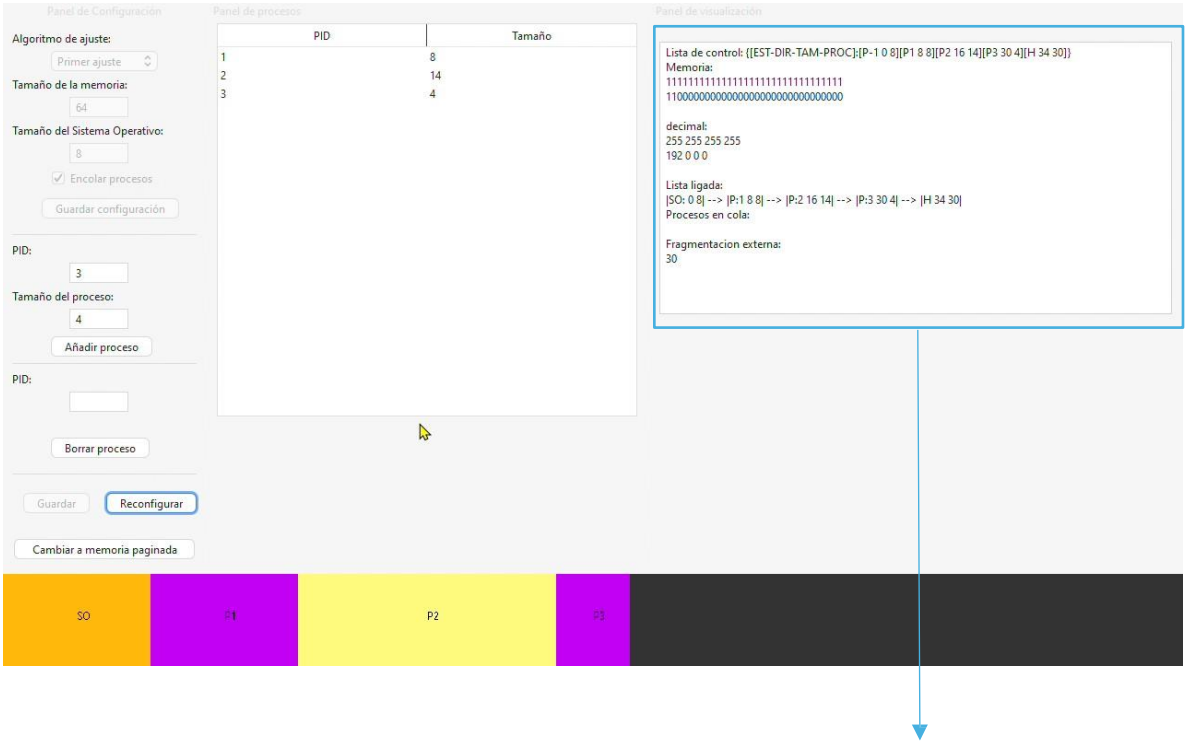
Después podrás guardar la configuración, una vez hecho esto, no podrás modificarlo a menos que presiones el botón reconfigurar.

Ahora, después de elegir la configuración que usaras es momento de agregar los PID's y sus tamaños, a su vez si necesitas borrar cualquiera de estos podrás hacerlo con las opciones de abajo.


Algoritmo de ajuste:
Primer ajuste
Tamaño de la memoria:
64
Tamaño del Sistema Operativo:
8
Encolar procesos
Guardar configuración
PID:
3
Tamaño del proceso:
4
Añadir proceso
PID:
Borrar proceso
PID
Tamaño
1
8
2
14
3
4
Aquí podrás agregar el número de PID y el tamaño que tendrá, una vez se proporcione estos datos deberás presionar el botón "Añadir proceso"

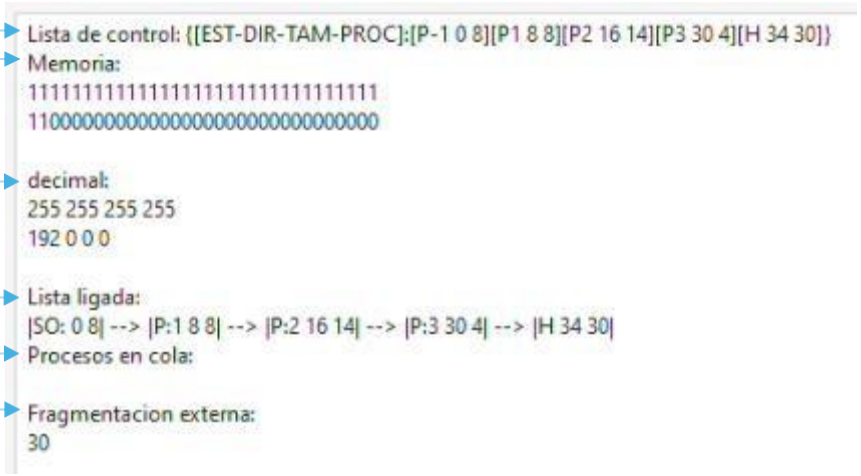

Algoritmo de ajuste:
Primer ajuste
Tamaño de la memoria:
64
Tamaño del Sistema Operativo:
8
Encolar procesos
Guardar configuración
PID:
3
Tamaño del proceso:
4
Añadir proceso
PID:
Borrar proceso
PID
Tamaño
1
8
2
14
3
4
Aquí podrás ver el numero de PID, cuantos hay y el tamaño

Para ir mostrando sus características y la función del simulador mismo, al momento de guardar por ejemplo los datos mostrados, esto mostrara lo siguiente:



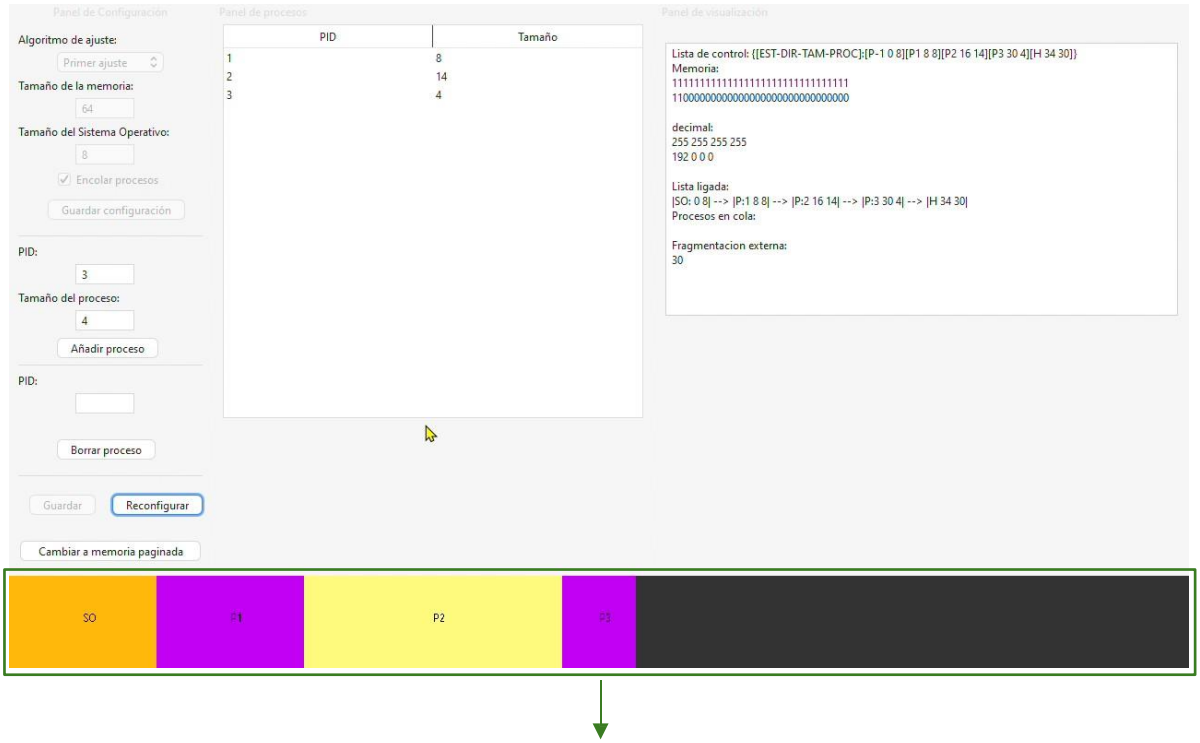
Aquí se muestra información importante sobre los procesos

Los datos mostrados son la **lista de control**, la memoria en binario y decimal, la lista ligada, los procesos en cola (que en este caso no hay) y la fragmentación externa:



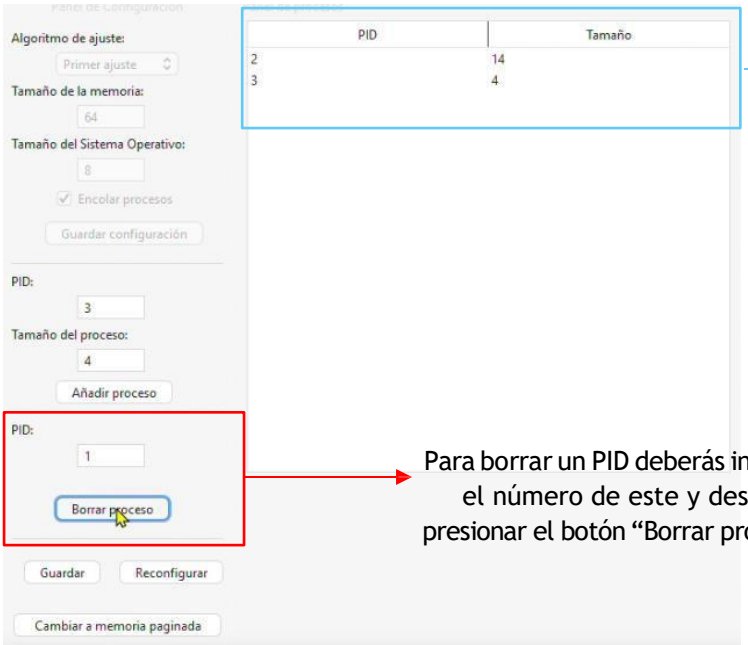
Lista de control (también conocida como lista de control de procesos o PCB list por sus siglas en inglés "Process Control Block list") es una estructura de datos utilizada para gestionar y

organizar información sobre los procesos que están siendo manejados por el sistema operativo.



En este lugar podrás ver el tamaño ocupado por el sistema operativo, el espacio que ocupa cada PID en memoria y en negro es espacio sobrante.

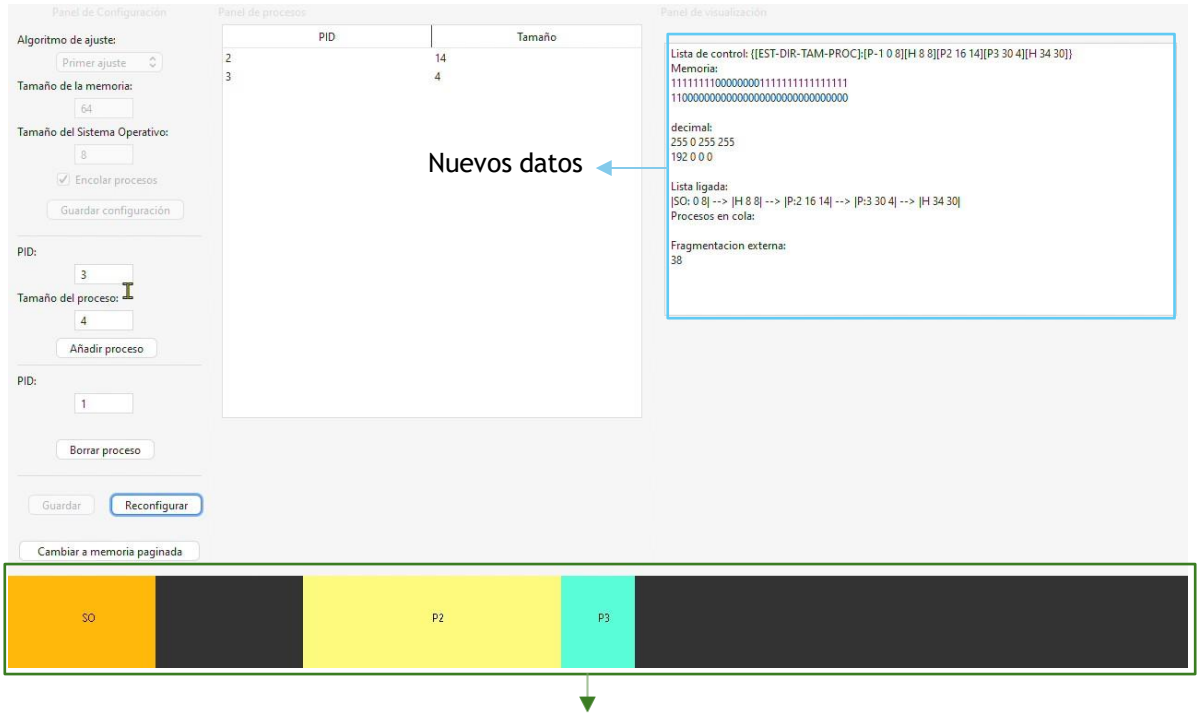
Ilustrando como borrar un PID se debe hacer lo siguiente:



Una vez borrado el PID deseado, se podrá ver aquí

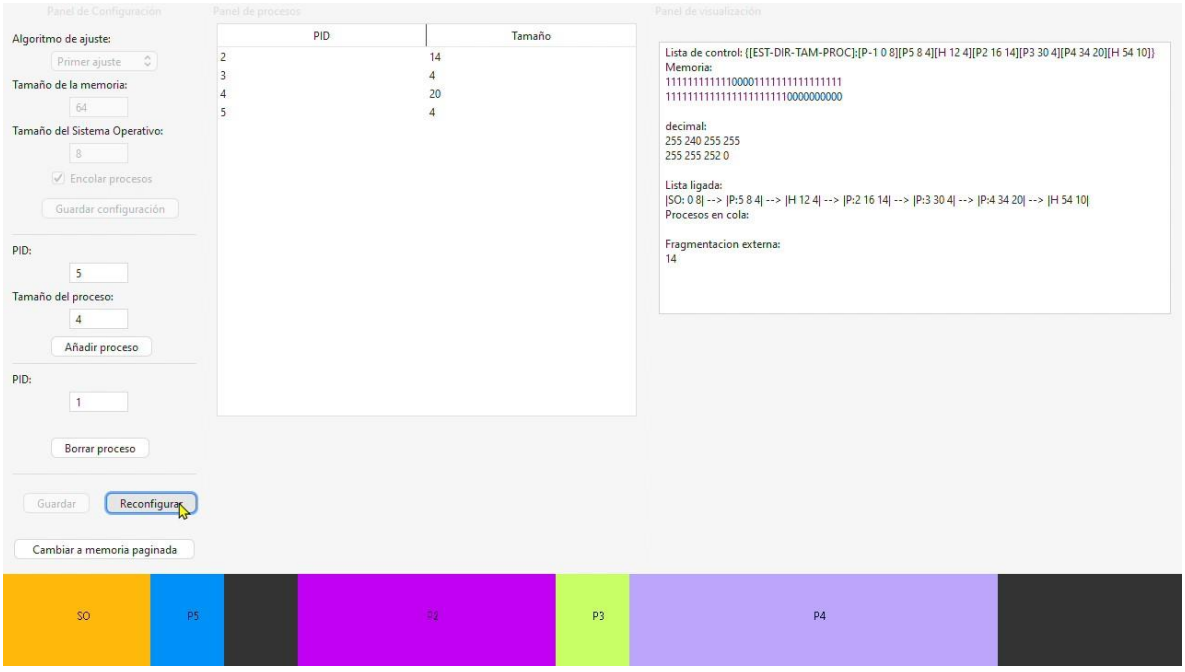
Para borrar un PID deberás ingresar el número de este y después presionar el botón “Borrar proceso”

Teniendo en cuenta el ejemplo de borrado anterior y teniendo en cuenta que guardaste, estos serían los nuevos datos (los cuales serán resaltados con sus respectivos colores):



Como puedes observar, al quitar el PID 1 ahora su espacio se muestra en negro, dando a entender que esta vacío

Por último, si deseas agregar más procesos, estos se irán acomodando con respecto a su tamaño y al algoritmo, como puedes ver en la imagen de abajo



Apartado de Memoria Paginada

III. **Introducción a los algoritmos de reemplazo de páginas LRU (Least Recently Used) y FIFO (First In, First Out)**

a) Algoritmo LRU (Least Recently Used)

Concepto:

El algoritmo LRU reemplaza la página que no ha sido utilizada durante el mayor tiempo. Se basa en la premisa de que las páginas usadas más recientemente seguirán siendo utilizadas, mientras que las que no han sido usadas recientemente probablemente no se usarán en un futuro cercano.

Funcionamiento:

1. Cada vez que una página es referenciada, se actualiza su tiempo de uso.
2. Cuando se necesita reemplazar una página, se elige la página que tiene el tiempo de uso más antiguo.

Ventajas:

- Generalmente ofrece un buen rendimiento porque se basa en el uso pasado para predecir el uso futuro.

Desventajas:

- Puede ser costoso en términos de tiempo y espacio si se implementa con precisión, ya que necesita mantener un registro exacto del orden de uso de todas las páginas.

Implementación Común:

- Usar una lista enlazada doble o una estructura de datos similar para mantener el orden de uso.
- Usar una pila o matrices asociativas para seguimiento rápido del uso de páginas.

b) Algoritmo FIFO (First In, First Out)

Concepto:

El algoritmo FIFO reemplaza la página que ha estado en la memoria por más tiempo. Se basa en la idea de que la página que llegó primero debe ser la primera en salir cuando se necesita espacio para una nueva página.

Funcionamiento:

1. Las páginas se colocan en una cola cuando son cargadas en memoria.
2. Cuando se necesita reemplazar una página, se elige la página que está al frente de la cola (la que ha estado más tiempo en la memoria).

Ventajas:

- Simple y fácil de implementar.
- No requiere mantener registros complejos de uso.

Desventajas:

- Puede llevar a un rendimiento subóptimo, conocido como "anomalía FIFO" o "anomalía de Belady", donde aumentar el número de marcos de página puede resultar en un aumento del número de fallos de página.

Implementación Común:

- Usar una estructura de cola simple.
- Mantener un puntero o índice para rastrear la página más antigua.

c) Comparación y Uso

LRU:

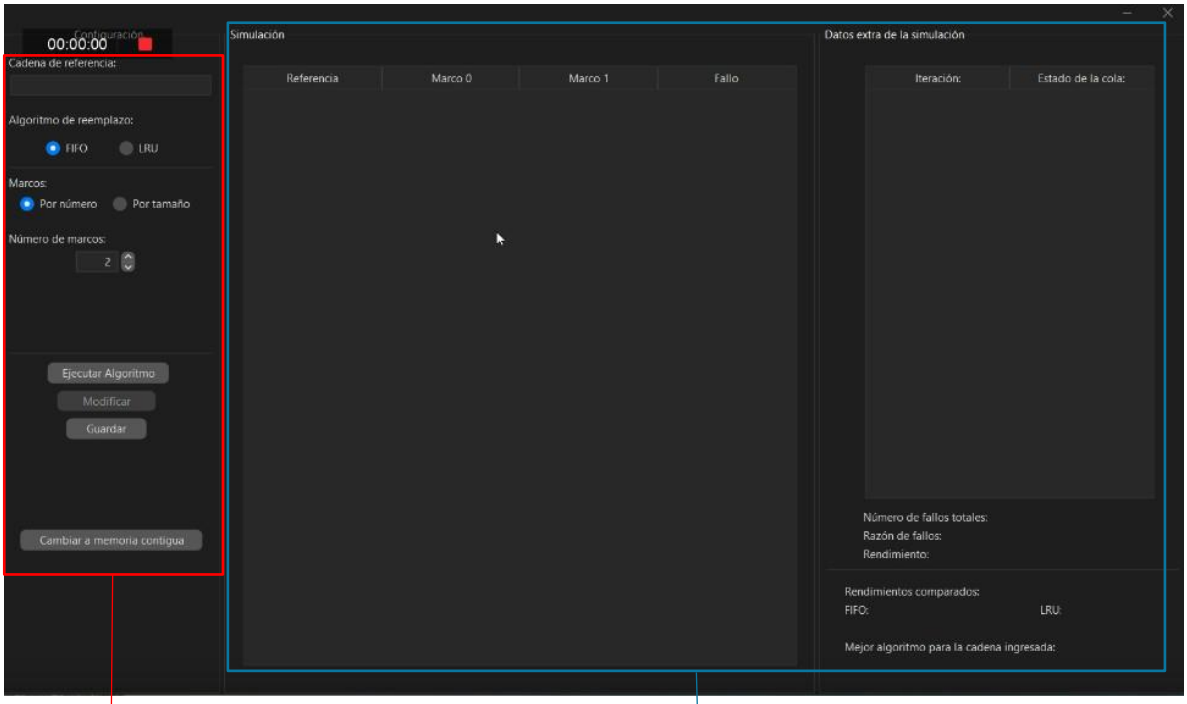
- Más complejo de implementar pero generalmente ofrece mejor rendimiento en términos de fallos de página.
- Adecuado para sistemas donde el patrón de acceso a las páginas es variable y se beneficia del seguimiento de uso reciente.

FIFO:

- Muy sencillo de implementar pero puede tener un peor rendimiento debido a la falta de consideración del patrón de uso reciente.
- Adecuado para sistemas donde la simplicidad y la rapidez de implementación son cruciales, y los patrones de acceso son más predecibles.

IV. Como usar el simulador (parte 1)

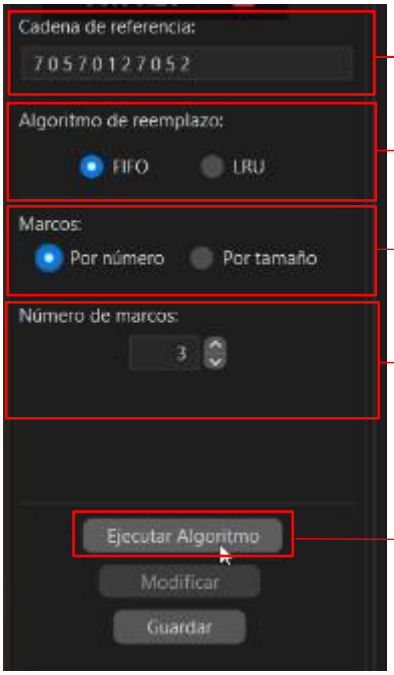
Para esta parte de igual forma se enmarcará de rojo las partes en las que se solicite información que tiene que dar el usuario y en azul las partes dadas por el simulador, en sí, los resultados.



Datos que se le solicita al usuario

Resultados

Comenzando, ingresando los datos encontraremos:



Aquí debes ingresar una cadena de números separados por espacio

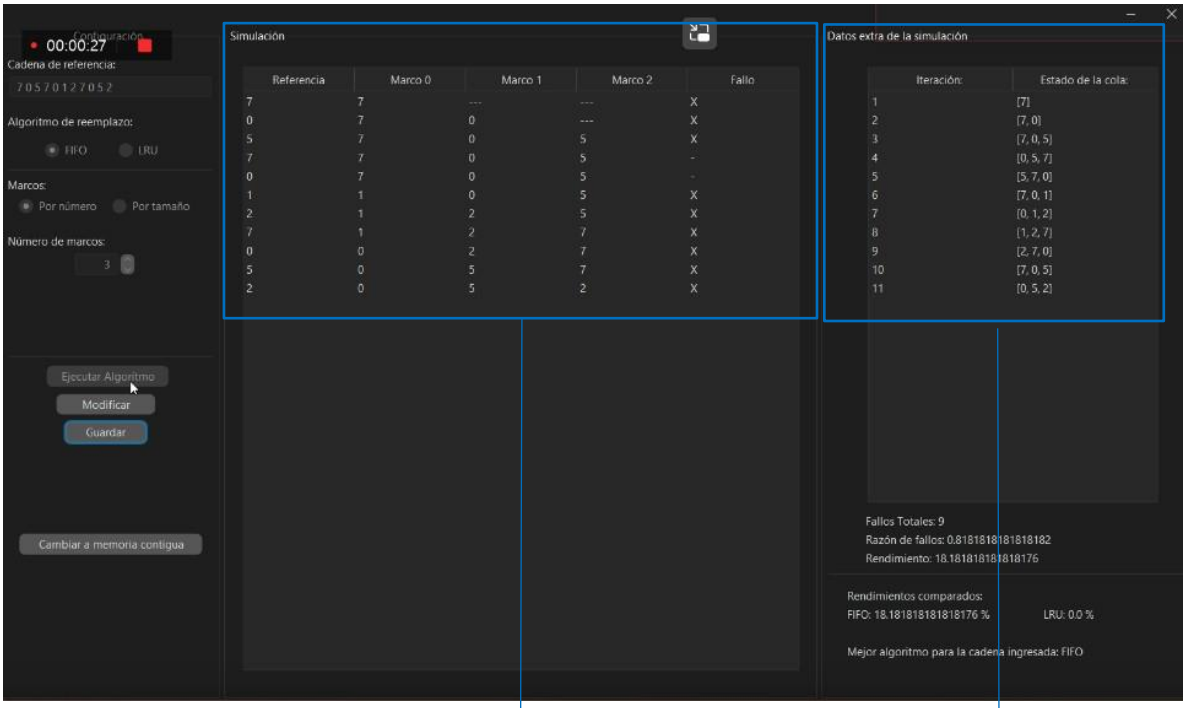
Aquí puedes elegir cual de los 2 algoritmos van a utilizar con la cadena anterior

Aquí debes elegir como vas a ocupar los marcos, ya sea por número o por tamaño

Aquí puedes elegir el número de marcos a ocupar, esto afecta la eficacia de los algoritmos, ya sea lo empeore o lo mejore

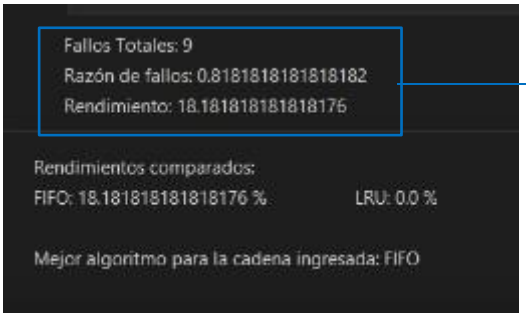
Una vez ingresados los datos, hay que presionar el botón “Ejecutar algoritmo”

Utilizando como ejemplo los datos anteriores, el resultado sería el siguiente, donde:

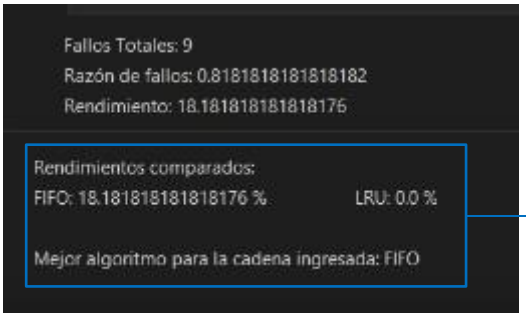


Aquí se muestran los resultados principales de la simulación

En este lugar se muestran datos extra del simulador



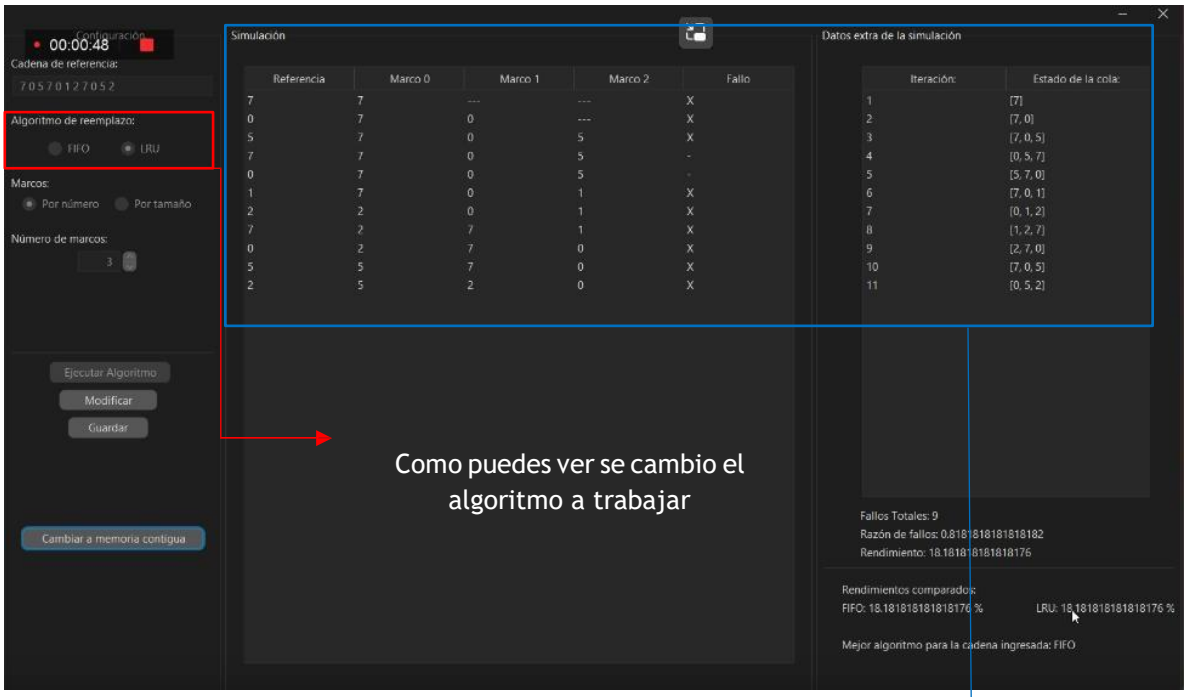
En esta parte se muestra un resumen del algoritmo actual, cuantos fallo tuvo, la razón y el rendimiento.



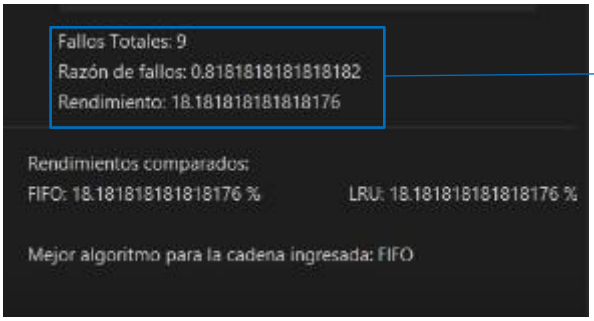
En esta parte se muestra la comparación de ambos algoritmos, si en su dado caso se comparan, por ende, muestra que algoritmo es mejor

Antes de utilizar otro algoritmo o modificar algún dato, primero se debe presionar el botón “Modificar”, de otro modo no se le permitirá

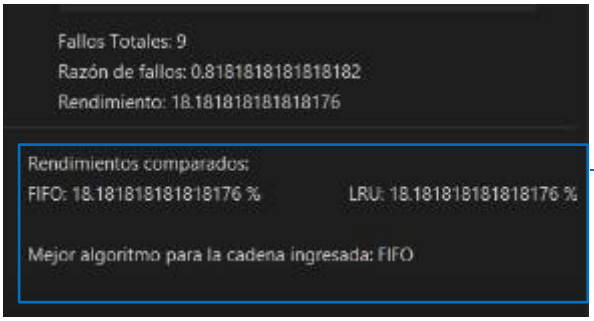
Luego después de cambiar el algoritmo a LRU, podemos observar los nuevos datos:



Estos son los nuevos datos del algoritmo LRU en este caso



Como puedes observar muestra los datos del algoritmo actual con los datos actuales



Aquí puedes ver después de ejecutar 2 algoritmos la comparación del rendimiento y cual es la mejor opción

V. **Conclusión y agradecimientos:**

Esperamos que este simulador te ayude a entender mejor como se gestiona la memoria en un sistema operativo internamente , gracias por usar nuestro simulador.