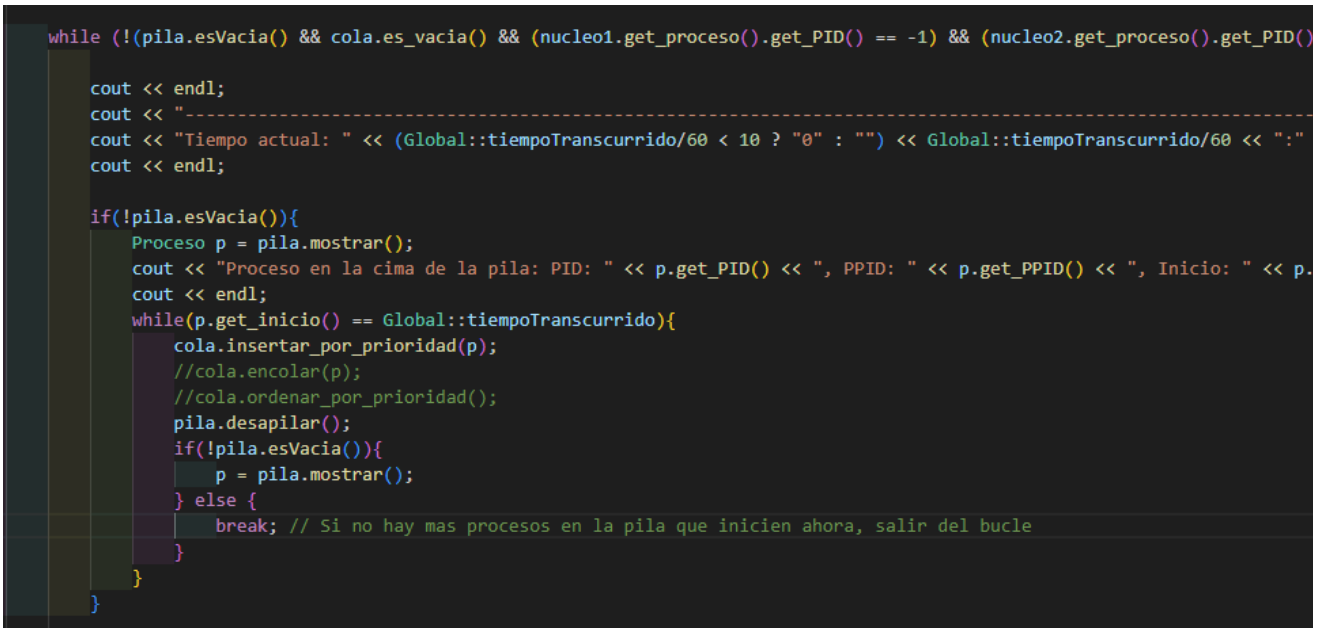
PLANIFICADOR DE PROCESOS DE SISTEMA OPERATIVO

Simulación del control de procesos en un sistema operativo



```
while (!(pila.esVacia() && cola.es_vacia() && (nucleo1.get_proceso().get_PID() == -1) && (nucleo2.get_proceso().get_PID()

    cout << endl;
    cout << "-----------------------------------------------------------------------------------------------------------
    cout << "Tiempo actual: " << (Global::tiempoTranscurrido/60 < 10 ? "0" : "") << Global::tiempoTranscurrido/60 << ":"
    cout << endl;

    if(!pila.esVacia()){
        Proceso p = pila.mostrar();
        cout << "Proceso en la cima de la pila: PID: " << p.get_PID() << ", PPID: " << p.get_PPID() << ", Inicio: " << p.
        cout << endl;
        while(p.get_inicio() == Global::tiempoTranscurrido){
            cola.insertar_por_prioridad(p);
            //cola.encolar(p);
            //cola.ordenar_por_prioridad();
            pila.desapilar();
            if(!pila.esVacia()){
                p = pila.mostrar();
            } else {
                break; // Si no hay mas procesos en la pila que inicien ahora, salir del bucle
            }
        }
    }
```


Matias Nicolas Vasquez Herbozo Y4507803R

Alejandro Fernández Ambrós 03248484-X

24/10/2026

G.I.I - UAH

ÍNDICE

ÍNDICE **1**
INTRODUCCIÓN **2**
CLASES Y FUNCIONES **2**
Clase Proceso 2
Clase Global 3
Clase Pila 4
Clase Nodo Pila 4
Clase Cola 5
Clase Nodo Cola 6
Clase Núcleo 6
Clase Lista 8
Clase Nodo Lista 9
PARTE 1 ..... **10**
Descripción de las Opciones en el Menú: 10
PARTE 2 ..... **16**
Descripción de las Opciones en el Menú: 16
RESULTADOS **22**
CONCLUSIÓN **22**

INTRODUCCIÓN

En esta práctica, se implementó un simulador de planificador de procesos en el que los procesos son asignados y gestionados en núcleos de procesamiento utilizando estructuras de datos como pilas, colas y listas enlazadas. La simulación está dividida en dos partes: la primera, con un enfoque más sencillo en la que los procesos se asignan a núcleos estáticos, y la segunda parte que permite agregar y eliminar núcleos de forma dinámica para equilibrar la carga de trabajo. A través de este simulador, se exploran principios de administración de procesos tales como la asignación de prioridades, la creación y eliminación de núcleos, y la estimación del tiempo medio de estancia de los procesos en el sistema.

CLASES Y FUNCIONES

Clase Proceso

La Clase Proceso representa una estructura donde se maneja la información relacionada con los procesos dentro del sistema planificador de procesos. Cada proceso se caracteriza por tener un identificador único (PID), el identificador de su proceso padre (PPID), su momento de inicio, duración, prioridad, y el núcleo al cual está asignado. La clase incluye métodos que permiten la creación del proceso y modificación de sus atributos.

Atributos de la Clase:

- ***static int contador_PID***: Contador estático utilizado para asignar un PID único a los procesos. Este se incrementa cada vez que se crea un nuevo proceso.
- ***int PID***: Identificador único del proceso.
- ***int PPID***: Identificador del proceso padre.
- ***int inicio***: Minuto en el que el proceso debe iniciar su ejecución, representado en minutos desde las 00:00 del sistema.
- ***int tiempoDeVida***: Duración del proceso indica cuánto tiempo en minutos se espera que esté en ejecución.
- ***int prioridad***: Nivel de prioridad del proceso de 0 a 9. Donde 0 representa la mayor prioridad, y 9 representa la menor.
- ***int nucleoAsignado***: Es el núcleo al cual se le asigna un proceso. Se inicia en -1,

indicando que no ha sido asignado a ningún núcleo hasta que este se encuentre en ejecución.

Funciones de la Clase:

- ***Proceso()***: Constructor que inicializa todos los atributos con valores predeterminados. Este constructor es útil para crear procesos sin valores definidos, con PID, PPID, inicio, tiempoDeVida, prioridad, y nucleoAsignado en -1.
- ***Proceso(int ppid, int inicio, int tiempoDeVida, int prioridad)***: Es un constructor que asigna valores específicos a los atributos principales del proceso (PPID, inicio, tiempoDeVida, y prioridad). Recordar que el PID se genera automáticamente utilizando contador_PID y el nucleoAsignado se establece en -1, indicando que el proceso aún no ha sido asignado a un núcleo.
- ***Getters y Setter***: Estos métodos permiten acceder y modificar los atributos clave de cada proceso de forma controlada. Los getters ***(get_PPID, get_inicio, get_tiempo_de_vida, get_prioridad, y get_nucleo_asignado)*** devuelven valores que identifican el proceso ya previamente asignados. Y los setters ***(set_PPID, set_inicio, set_tiempo_de_vida, set_prioridad, y set_nucleo_asignado)*** permiten modificar estos atributos, ajustando la información de cada proceso según las necesidades del sistema.

Clase Global

La Clase Global contiene variables globales esenciales para el funcionamiento del sistema de planificación de procesos. Esto permite llevar el control del tiempo de los procesos y proporciona tiempo global para organizar los eventos en el sistema, de esta manera se puede administrar los tiempos y eventos a lo largo de la ejecución.

Atributos de la Clase:

- ***time_t tiempoReferencia***: Es una variable que almacena el tiempo de referencia inicial del sistema (es decir, el momento en que se inicia la simulación). Esta variable es clave para calcular el tiempo transcurrido desde que se inició el sistema de planificación.
- ***int tiempoTranscurrido***: Variable que almacena el tiempo en minutos desde que comenzó la simulación. Este valor se incrementa conforme avanzan las operaciones en el sistema como un conteo acumulativo del tiempo de la ejecución.

Clase Pila

La clase Pila gestiona una estructura de datos de tipo pila (LIFO - Last In, First Out) que contiene elementos de tipo Proceso, ordenados y manipulados para facilitar el control de ejecución en un sistema de planificación de procesos. Su funcionalidad incluye apilar y desapilar elementos, ordenar y mostrar procesos, así como modificar la estructura de la pila según las necesidades del planificador.

Atributos de la Clase:

- ***pnodo cima***: Apunta al nodo superior de la pila (NodoPila), accediendo a los procesos en el orden en que fueron apilados.

Funciones de la Clase:

- ***Pila()***: Constructor por defecto. Inicializa cima a NULL, indicando una pila vacía al comienzo.
- ***bool esVacia()***: Verifica si la pila está vacía, devolviendo true si no hay procesos en ella.
- ***void apilar(Proceso p)***: Añade un proceso p a la cima de la pila, creando un nuevo NodoPila y ajustando el puntero cima al nuevo nodo.
- ***void desapilar()***: Elimina el proceso que se encuentra en la cima de la pila. Si la pila no está vacía, la cima se ajusta al siguiente nodo y el nodo anterior se elimina de la memoria.
- ***Proceso mostrar()***: Muestra el proceso que está en la cima de la pila. Si la pila está vacía, devuelve un proceso vacío.
- ***void ordenarPorTiempoInicio()***: Ordena la pila de menor a mayor en base al tiempo de inicio de cada proceso. Utiliza una pila auxiliar para organizar los procesos en el orden deseado.
- ***void mostrarTodos()***: Muestra todos los procesos de la pila, mostrando los atributos de cada proceso en orden desde la cima al fondo.

Clase Nodo Pila

La clase NodoPila es un nodo que se utiliza como componente básico de la estructura Pila. Cada NodoPila almacena un Proceso y un puntero al siguiente nodo, la estructura enlazada permite implementar la lógica de una pila.

Atributos de la Clase:

- ***Proceso proceso***: Contiene la instancia de Proceso almacenada en este nodo, que representa un proceso en el sistema.
- ***NodoPila siguiente****: Apunta al siguiente nodo en la pila. Permite enlazar nodos para mantener el orden y estructura de la pila.

Funciones de la Clase:

- ***NodoPila()***: Constructor del nodo por defecto que inicializa el nodo sin parámetros, nodo vacío.
- ****NodoPila(Proceso p, NodoPila sig = NULL)***: Constructor que inicializa el nodo con un proceso p y un puntero siguiente, que apunta al siguiente nodo en la pila (nodo siguiente = NULL si no se especifica).

Clase Cola

La Clase Cola permite gestionar los procesos en una estructura de cola, donde los procesos se ordenan en base a su prioridad y su orden de llegada. Esta estructura permite almacenar, organizar y manejar procesos para ser ejecutados en un sistema que sigue el orden FIFO (primero en entrar, primero en salir).

Atributos de la Clase

- ***NodoCola* primero***: Apunta al primer nodo de la cola, el cual es el siguiente en salir.
- ***NodoCola* ultimo***: Apunta al último nodo de la cola, al cual se añade cualquier nuevo proceso.
- ***int longitud***: Almacena la cantidad de nodos (procesos) actualmente en la cola.

Funciones de la Clase

- ***Cola()***: Constructor que inicializa la cola como vacía, con primero y último en NULL y longitud en 0.
- ***void encolar(Proceso proceso)***: Añade un proceso al final de la cola. Si la cola está vacía, el nuevo nodo se convierte tanto en el primero como en el último. Aumenta la longitud de la cola.
- ***void insertar_por_prioridad(Proceso proceso)***: Inserta un proceso en la cola y luego la ordena en función de la prioridad del proceso.

- ***Proceso desencolar()***: Retira y devuelve el proceso al frente de la cola, si no está vacía. Si la cola está vacía, devuelve un proceso vacío.
- ***bool es_vacia() const***: Verifica si la cola está vacía, devolviendo true si no contiene nodos y false en caso contrario.
- ***int get_longitud()***: Retorna el número de nodos (procesos) en la cola.
- ***void mostrarCola()***: Muestra en pantalla todos los procesos en la cola. Para cada proceso, muestra sus atributos en formato de tabla. Crea una copia temporal de la cola para restaurar los elementos a su estado original después de imprimir.
- ***void ordenar_por_prioridad()***: Ordena los procesos de la cola de menor a mayor prioridad, utilizando un algoritmo de burbuja para realizar el ordenamiento.

Clase Nodo Cola

La Clase NodoCola representa un nodo individual en la estructura de la clase Cola. Cada nodo contiene un proceso y un puntero al siguiente nodo en la cola. Además, cada nodo tiene un atributo de prioridad, lo que permite a la clase Cola organizar los procesos según este valor cuando se requiere

Atributos de la Clase

- ***NodoCola* siguiente***: Puntero que conecta el nodo actual con el siguiente en la cola, permitiendo el encadenamiento de nodos.
- ***Proceso proceso***: Almacena el proceso asociado a este nodo, incluyendo su PID, PPID, inicio, duración, y prioridad.
- ***int prioridad***: Representa el nivel de prioridad del proceso en el nodo, donde un valor más bajo indica mayor prioridad.

Funciones de la Clase

- ***NodoCola()***: Constructor por defecto que inicializa un nodo con un proceso vacío, siguiente apuntando a NULL y prioridad en 0. Este constructor es útil para crear nodos sin valores definidos inicialmente.
- ***NodoCola(Proceso proceso, int prioridad, NodoCola* sig = NULL)***: Constructor que permite establecer un proceso específico, su prioridad y un puntero al siguiente nodo en la cola.

Clase Núcleo

La clase Núcleo representa el núcleo de procesamiento del sistema planificador de procesos, encargado de manejar la ejecución de procesos y su cola de espera. Un núcleo mantiene un proceso en ejecución, una cola de procesos a la espera, y el tiempo de inicio y finalización del proceso actual. Su implementación permite administrar la asignación y ejecución de procesos dentro del sistema de manera organizada.

Atributos de la Clase

- ***static int Contador_ID***: Contador estático utilizado para asignar un identificador único a cada núcleo a medida que se crean.
- int id: Identificador único del núcleo.
- ***Proceso proceso_en_ejecucion***: El proceso actualmente en ejecución en el núcleo.
- ***Cola cola_procesos***: La cola de procesos en espera que serán ejecutados en este núcleo cuando esté disponible.
- ***time_t tiempo_inicio***: Tiempo en el que el proceso en ejecución comenzó, basado en el tiempo del sistema.
- ***time_t tiempo_fin***: Tiempo calculado en el que se espera que el proceso en ejecución termine, determinado por el tiempo de inicio y el tiempo de vida del proceso.

Funciones de la Clase

- ***Nucleo()***: Constructor por defecto que inicializa el núcleo con un ID único generado por Contador_ID, y configura los tiempos de inicio y fin en -1 para indicar que el núcleo está libre.
- ***Nucleo(int id, Proceso proceso)***: Constructor que permite crear un núcleo con un identificador específico y un proceso inicial en ejecución. Los tiempos de inicio y fin se calculan automáticamente.
- ***void add_proceso(Proceso proceso)***: Añade un proceso al núcleo. Si el núcleo está libre, lo asigna a proceso_en_ejecucion; si no, el proceso es agregado a la cola de espera con base en su prioridad.
- ***void eliminar_proceso()***: Elimina el proceso en ejecución si ha terminado, desencola el siguiente proceso de la cola y lo asigna al núcleo. Si no hay procesos en la cola, se mantiene el núcleo libre.
- ***void detalles_proceso() / detalles_proceso(bool i)***: Muestra los detalles del proceso en ejecución en el núcleo. La función con parámetro bool i permite indicar si el proceso ha iniciado (true) o ha finalizado (false), proporcionando

información del estado del proceso en el núcleo.
- ***void detalles_nucleo()***: Muestra los detalles completos del núcleo, incluyendo el proceso en ejecución y la cola de espera. Si no hay proceso en ejecución o la cola está vacía, se muestra un mensaje indicándolo.

Clase Lista

La Clase Lista representa una lista doblemente enlazada de núcleos. Cada núcleo en la lista se representa a través de un nodo (NodoLista), permitiendo la gestión de los núcleos en el sistema planificador de procesos. Esta estructura permite la inserción, eliminación, y búsqueda de núcleos, así como la administración de los procesos en cada núcleo y la gestión de la carga de trabajo.

Atributos de la Clase

- ***NodoLista primero****: Puntero al primer nodo de la lista, donde se almacena el núcleo inicial.
- ***NodoLista ultimo****: Puntero al último nodo de la lista, donde se almacena el núcleo final.
- i***nt longitud***: Almacena el número total de núcleos en la lista, lo que también representa el número de núcleos operativos en el sistema.

Funciones de la Clase

- ***Lista()***: Constructor que inicializa la lista, asignando nullptr a primero y ultimo y estableciendo la longitud en 0.
- ***bool es_vacia()***: Verifica si la lista está vacía. Retorna true si no hay nodos en la lista.
- ***get_longitud()***: Retorna la longitud de la lista, que equivale al número de núcleos operativos en el sistema.
- ***void ordenar_menor_mayor()***: Ordena los núcleos en la lista de menor a mayor utilizando un algoritmo de burbuja modificado. Cambia el orden de los nodos y asegura que los punteros se mantengan en la estructura correcta.
- ***void insertar_nucleo()***: Crea un núcleo y lo añade al final de la lista. Si la lista está vacía, inicializa el primer y último nodo al nuevo núcleo.
- ***void eliminar(int posicion)***: Elimina el núcleo en la posición indicada. Ajusta los punteros de los nodos adyacentes para preservar la integridad de la lista.
- ***void insertar_proceso(Proceso proceso, int posicion = 0)***: Inserta un proceso

en el núcleo de la posición especificada, usando el método add_proceso del núcleo.

- ***void eliminar_proceso(int posicion)***: Elimina el proceso en ejecución en el núcleo en la posición dada.
- **int nucleo_menos_carga(bool imprimir=false)**: Encuentra el núcleo con menos carga de procesos en su cola de espera. Si imprimir es true, muestra información sobre el núcleo con menor carga. Si todos los núcleos tienen más de dos procesos en espera, crea un nuevo núcleo.
- ***int nucleo_mas_carga(bool imprimir=false)***: Encuentra el núcleo con mayor carga de procesos en su cola de espera. Si imprimir es true, muestra información sobre el núcleo con mayor carga.
- ***void estado_nucleo(int posicion)***: Muestra la información del núcleo en la posición indicada. Utiliza el método detalles_nucleo de Nucleo.
- ***void mostrar_estado_nucleos()***: Muestra el estado de todos los núcleos en la lista, iterando desde el primer nodo hasta el último.
- ***Nucleo coger(int n)***: Retorna el núcleo en la posición especificada. Si la posición es inválida, muestra un mensaje y retorna un núcleo vacío.

Clase Nodo Lista

La clase NodoLista representa un nodo individual en una lista doblemente enlazada de núcleos. Cada NodoLista contiene un núcleo y punteros a los nodos anterior y siguiente en la lista, lo que permite la navegación en ambas direcciones.

Atributos de la Clase

- ***Nucleo nucleo***: Contiene el núcleo asociado a este nodo.
- ***NodoLista siguiente****: Puntero al siguiente nodo en la lista.
- ***NodoLista anterior****: Puntero al nodo anterior en la lista.

Funciones de la Clase

- ***NodoLista()***: Constructor por defecto que inicializa los punteros siguiente y anterior en nullptr, y crea un núcleo vacío.
- ***NodoLista(Nucleo nucleo, NodoLista siguiente = nullptr, NodoLista anterior = nullptr)****: Constructor que permite inicializar el nodo con un núcleo específico y opcionalmente enlazarlo a otros nodos en posiciones siguiente y anterior.

PARTE 1

En la Parte 1 de esta práctica, el objetivo es simular el funcionamiento de un planificador de procesos dentro de un sistema operativo que utiliza colas de prioridad para la asignación de tareas a tres núcleos de procesamiento. La simulación se centra en calcular el tiempo medio de estancia de los procesos en el sistema operativo, lo que significa el tiempo medio que permanece un proceso activo dentro del sistema desde que es creado hasta que finaliza su ejecución.

El sistema gestiona los procesos a través de una pila y una cola de espera organizadas por orden de prioridad. La pila contiene los procesos en orden de inicio y se mueve a la cola de espera al momento adecuado. Cuando un núcleo queda libre, se asigna el proceso de mayor prioridad en espera. La simulación del sistema se plasma en el menú que permite al usuario crear procesos, visualizarlos, eliminarlos, y/o simular la ejecución de los mismos.

Descripción de las Opciones en el Menú:

```
--------------------------------------------------------------------
00:00              MENU PRINCIPAL

1. Crear pila de procesos. (datos creados manualmente en el codigo)
2. Mostrar todos los procesos en la pila de procesos.
3. Borrar pila de procesos del sistema.
4. Mostrar cola de espera de procesos.
5. Mostrar procesos en nucleos.
6. Aumentar tiempo del sistema (n minutos).
7. Simular ejecucion de procesos.
8. Salir.
Seleccione una opcion:
```

1. **Crear pila de procesos**

Esta opción inicializa una pila de procesos y se crean Procesos con datos creados manualmente en el código y se almacenan en ella. Los procesos se agregan usando ***pila.apilar(Proceso p)***, y luego la pila se ordena por el tiempo de inicio de los procesos con ***pila.ordenarPorTiempoInicio()***. Dado que la pila se utiliza en toda instancia de la simulación, es importante ejecutar esta opción para tener una pila creada y ejecutar posteriormente las opciones 6 y/o 7.

Una vez el usuario haya seleccionado la opción 1, se ejecuta y muestra un mensaje de confirmación indicando que la pila de procesos se ha creado y ordenado correctamente.

```
Seleccione una opcion: 1
--------------------------------------------------------------------------------

Pila de procesos creada correctamente.
```

2. **Mostrar todos los procesos en la pila**

Esta opción permite ver todos los procesos cargados en la pila, mostrando sus atributos clave. Muestra datos como el PID, PPID, tiempo de inicio, duración, y prioridad de cada proceso, utilizando ***pila.mostrarTodos().*** Ofrecer al usuario una visión clara de los procesos que están disponibles y listos para entrar en ejecución, este listado previo puede ayudar a anticipar la ejecución y verificar el orden y detalles de los procesos antes de iniciar la simulación.

```
Seleccione una opcion: 2
--------------------------------------------------------------------------------

PID: 1, PPID: 1, Minutos de inicio: 0, Tiempo de vida: 4 minutos, Prioridad: 0
PID: 2, PPID: 1, Minutos de inicio: 0, Tiempo de vida: 5 minutos, Prioridad: 1
PID: 3, PPID: 1, Minutos de inicio: 1, Tiempo de vida: 4 minutos, Prioridad: 3
PID: 4, PPID: 2, Minutos de inicio: 4, Tiempo de vida: 3 minutos, Prioridad: 7
PID: 5, PPID: 2, Minutos de inicio: 6, Tiempo de vida: 2 minutos, Prioridad: 2
PID: 6, PPID: 3, Minutos de inicio: 9, Tiempo de vida: 5 minutos, Prioridad: 1
PID: 7, PPID: 3, Minutos de inicio: 10, Tiempo de vida: 3 minutos, Prioridad: 4
PID: 8, PPID: 4, Minutos de inicio: 12, Tiempo de vida: 2 minutos, Prioridad: 6
PID: 9, PPID: 7, Minutos de inicio: 13, Tiempo de vida: 1 minutos, Prioridad: 8
PID: 10, PPID: 5, Minutos de inicio: 15, Tiempo de vida: 4 minutos, Prioridad: 2
PID: 11, PPID: 4, Minutos de inicio: 16, Tiempo de vida: 3 minutos, Prioridad: 3
PID: 14, PPID: 6, Minutos de inicio: 18, Tiempo de vida: 2 minutos, Prioridad: 4
PID: 12, PPID: 5, Minutos de inicio: 18, Tiempo de vida: 2 minutos, Prioridad: 5
PID: 13, PPID: 6, Minutos de inicio: 18, Tiempo de vida: 1 minutos, Prioridad: 1
PID: 16, PPID: 8, Minutos de inicio: 19, Tiempo de vida: 3 minutos, Prioridad: 6
PID: 15, PPID: 7, Minutos de inicio: 19, Tiempo de vida: 5 minutos, Prioridad: 0
PID: 17, PPID: 10, Minutos de inicio: 20, Tiempo de vida: 2 minutos, Prioridad: 7
PID: 19, PPID: 9, Minutos de inicio: 21, Tiempo de vida: 4 minutos, Prioridad: 2
PID: 18, PPID: 8, Minutos de inicio: 21, Tiempo de vida: 1 minutos, Prioridad: 8
```

En caso no se haya creado previamente los procesos y almacenados en la pila con la opción 1. Se mostrará en pantalla que la pila se encuentra vacía.

```
Seleccione una opcion: 2
--------------------------------------------------------------------------------

La pila est├í vac├¡a.
```

3. **Borrar pila de procesos**

Esta opción elimina todos los procesos de la pila, dejando el sistema sin procesos en ella. Esta operación reinicia la pila y se asegura de que el usuario pueda empezar desde cero si desea recargar una nueva pila con los procesos con la opción 1 pero con diferentes PID ya que estos son identificadores únicos para cada proceso. Para ello, se verifica si la pila está vacía usando ***pila.esVacia()***. Si contiene procesos, se elimina cada uno usando ***pila.desapilar()*** hasta que la pila esté vacía.

Se muestra un mensaje indicando que la pila ha sido vaciada correctamente, o si en su defecto si ya se encontraba vacía.

```
Seleccione una opcion: 3
--------------------------------------------------------------------------------

Pila de procesos eliminada correctamente.
```

En caso de que la pila esté vacía y se intente ejecutar la opción 3 de vaciar la pila. Este mostrará el siguiente mensaje de que la pila ya se encuentra vacía.

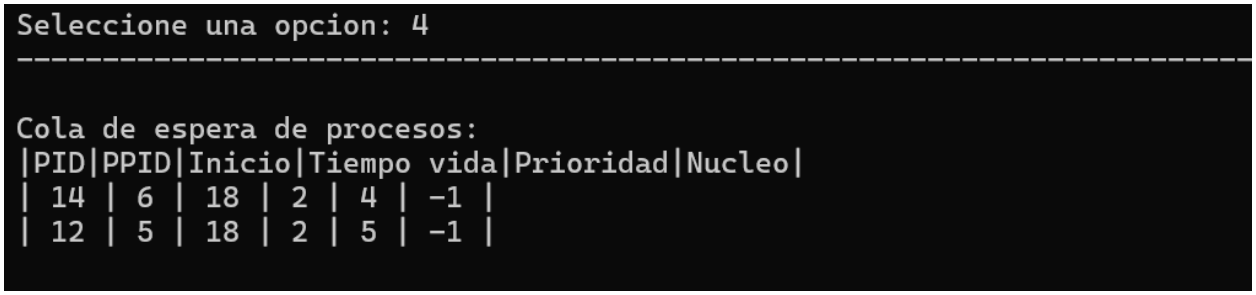
```
Seleccione una opcion: 3
--------------------------------------------------------------------------------

La pila de procesos ya est├í vac├¡a.
```

4. **Mostrar cola de espera de procesos**

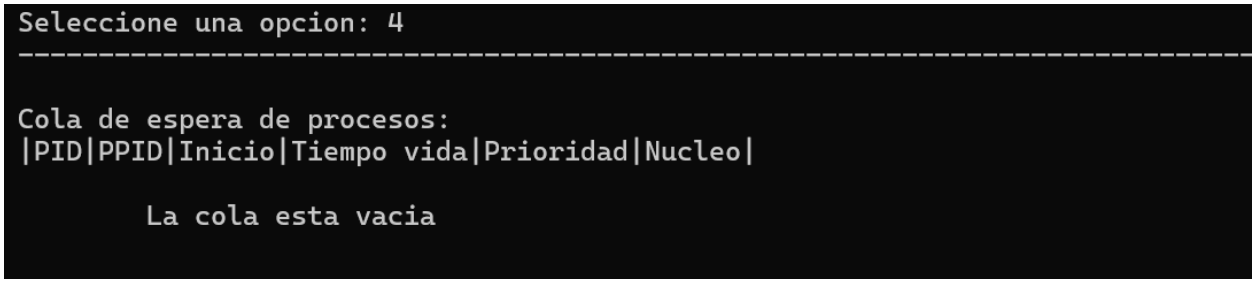
Muestra el contenido actual de la cola de espera, que es la estructura donde se almacenan los procesos a medida que están listos para ser ejecutados pero aún no han sido asignados a un núcleo. Esta cola organiza los procesos con base en sus prioridades, lo cual es esencial para la ejecución eficiente en los núcleos de procesamiento. Para ello; primero, la cola se ordena por prioridad con cola.ordenar_por_prioridad(), y luego se visualiza con cola.mostrarCola().

Si se da el caso que la cola de espera se encuentra almacenando los procesos. Se mostrará

en pantalla la información de cada proceso almacenado en la cola y ordenado por orden de prioridad.



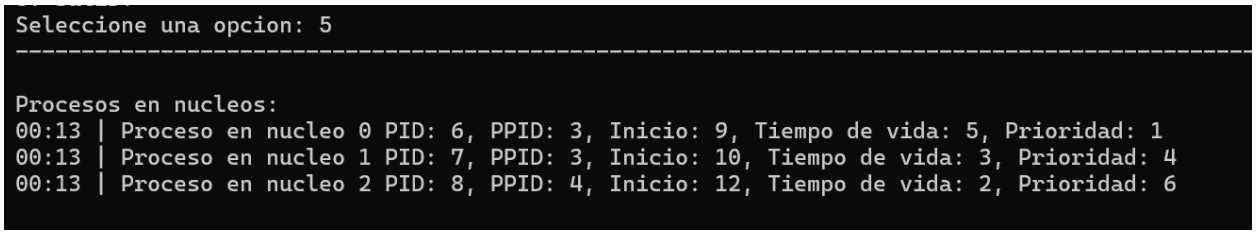
Si la cola se encuentra ovación se mostrará un mensaje indicando que la cola se encuentra vacía.

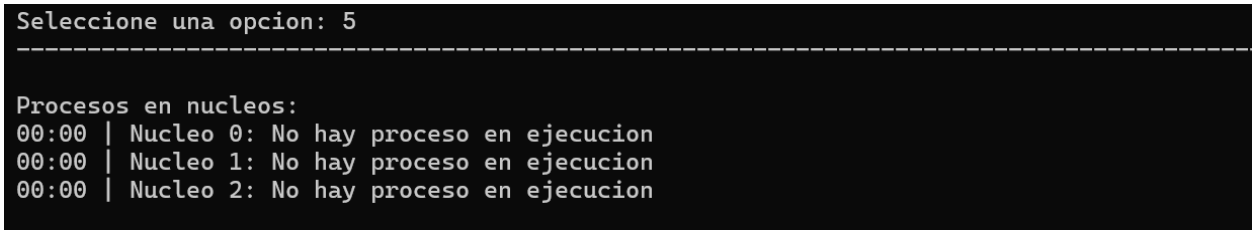


5. **Mostrar procesos en núcleos**

Muestra los detalles de los procesos que actualmente están en ejecución en los tres núcleos (nucleo1, nucleo2, nucleo3) mediante ***detalles_proceso()*** en cada núcleo.

Resultado: Se muestra una lista con el estado actual de los procesos en cada núcleo.





6. **Aumentar tiempo del sistema (n minutos)**

Incrementa el tiempo del sistema en n minutos y simula la ejecución de procesos durante ese intervalo. Para cada minuto:

Los procesos con tiempo de inicio igual al tiempo actual son trasladados de la pila a la cola. Los procesos de la cola se asignan a los núcleos disponibles, y los procesos que terminan su ejecución se eliminan de los núcleos. El tiempo medio de estancia en el sistema se calcula y se actualiza a medida que finalizan los procesos.

El resultado muestra el estado actual de la cola de procesos y de cada núcleo después de cada minuto de simulación.

```
Seleccione una opcion: 6
--------------------------------------------------------------------------------

Ingrese el numero de minutos a aumentar: 15



No hay procesos en la pila ni en la cola de espera, y ambos nucleos estan libres.

Ejecucion de procesos finalizada.

Tiempo medio de estancia en el sistema operativo: 0 minutos.
```

```
Seleccione una opcion: 6
--------------------------------------------------------------------------------------------------

Ingrese el numero de minutos a aumentar: 15


--------------------------------------------------------------------------------------------------
Tiempo actual: 00:00

Proceso en la cima de la pila: PID: 1, PPID: 1, Inicio: 0, Tiempo de vida: 4, Prioridad: 0

|PID|PPID|Inicio|Tiempo vida|Prioridad|Nucleo|
| 1 | 1 | 0 | 4 | 0 | -1 |
| 2 | 1 | 0 | 5 | 1 | -1 |



00:00 | Proceso iniciado en nucleo 0: PID: 1, PPID: 1, Inicio: 0, Tiempo de vida: 4, Prioridad: 0


00:00 | Proceso iniciado en nucleo 1: PID: 2, PPID: 1, Inicio: 0, Tiempo de vida: 5, Prioridad: 1


00:00 | Proceso en nucleo 0 PID: 1, PPID: 1, Inicio: 0, Tiempo de vida: 4, Prioridad: 0
00:00 | Proceso en nucleo 1 PID: 2, PPID: 1, Inicio: 0, Tiempo de vida: 5, Prioridad: 1
00:00 | Nucleo 2: No hay proceso en ejecucion
```

7. **Simular ejecución de procesos hasta finalización**

Similar a la opción 6, pero esta opción simula el tiempo hasta que todos los procesos han finalizado. La simulación se detiene cuando no hay más procesos en la pila, la cola de espera, ni en los núcleos.

El resultado se muestra el estado de la cola y cada núcleo hasta la finalización de todos los procesos, y luego se imprime el tiempo medio de estancia de los procesos.

```
Seleccione una opcion: 7
-----------------------------------------------------------------------------------------------------------------

A continuacion se simulara el paso del tiempo en el sistema operativo hasta que finalicen todos los procesos

-----------------------------------------------------------------------------------------------------------------
Tiempo actual: 00:15

Proceso en la cima de la pila: PID: 10, PPID: 5, Inicio: 15, Tiempo de vida: 4, Prioridad: 2

|PID|PPID|Inicio|Tiempo vida|Prioridad|Nucleo|
| 10 | 5 | 15 | 4 | 2 | -1 |



00:15 | Proceso iniciado en nucleo 0: PID: 10, PPID: 5, Inicio: 15, Tiempo de vida: 4, Prioridad: 2


00:15 | Proceso en nucleo 0 PID: 10, PPID: 5, Inicio: 15, Tiempo de vida: 4, Prioridad: 2
00:15 | Nucleo 1: No hay proceso en ejecucion
00:15 | Nucleo 2: No hay proceso en ejecucion
```

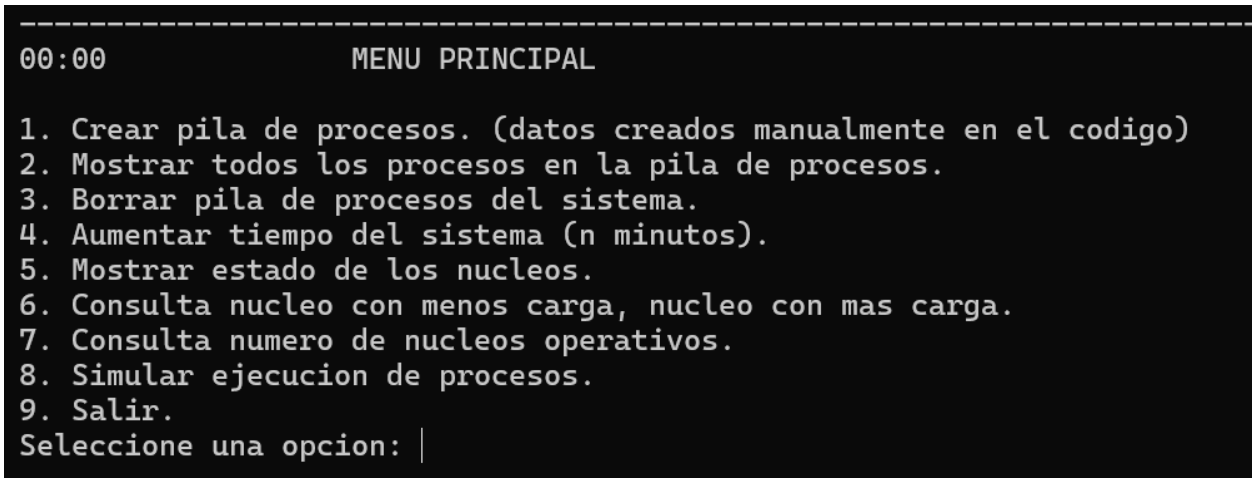
```
-----------------------------------------------------------------------------------------------------------------
Tiempo actual: 00:26

|PID|PPID|Inicio|Tiempo vida|Prioridad|Nucleo|

        La cola esta vacia


00:26 | Proceso terminado en nucleo 0: PID: 17, PPID: 10, Inicio: 20, Tiempo de vida: 2, Prioridad: 7

00:26 | Nucleo 0: No hay proceso en ejecucion
00:26 | Nucleo 1: No hay proceso en ejecucion
00:26 | Nucleo 2: No hay proceso en ejecucion







Ejecucion de procesos finalizada.

Tiempo medio de estancia en el sistema operativo: 3.52632 minutos.
```

8. **Salir**

Finaliza el programa y cierra la ejecución de la simulación de procesos.

PARTE 2

En la Parte 2 de esta práctica, el enfoque principal es la gestión de procesos dentro de un sistema operativo utilizando una lista de núcleos que permite una asignación más dinámica de los procesos. En esta parte, se implementan y simulan operaciones complejas de planificación y balanceo de carga de procesos, considerando múltiples núcleos y gestionando los procesos con colas de prioridad.

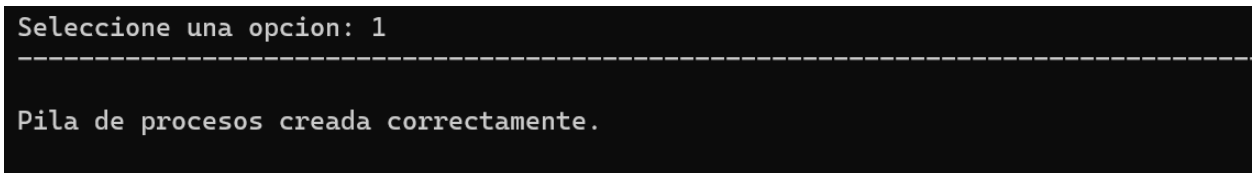


Descripción de las Opciones en el Menú:

1. **Crear pila de procesos**

Esta opción inicializa una pila de procesos y se crean Procesos con datos creados manualmente en el código y se almacenan en ella. Los procesos se agregan usando ***pila.apilar(Proceso p)***, y luego la pila se ordena por el tiempo de inicio de los procesos con ***pila.ordenarPorTiempoInicio()***. Dado que la pila se utiliza en toda instancia de la simulación, es importante ejecutar esta opción para tener una pila creada y ejecutar posteriormente las opciones 4 y/o 8.

Una vez el usuario haya seleccionado la opción 1, se ejecuta y muestra un mensaje de confirmación indicando que la pila de procesos se ha creado y ordenado correctamente.



2. **Mostrar todos los procesos en la pila**

Esta opción permite ver todos los procesos cargados en la pila, mostrando sus atributos clave. Muestra datos como el PID, PPID, tiempo de inicio, duración, y prioridad de cada proceso, utilizando ***pila.mostrarTodos().*** Ofrecer al usuario una visión clara de los procesos que están disponibles y listos para entrar en ejecución, este listado previo puede ayudar a anticipar la ejecución y verificar el orden y detalles de los procesos antes de iniciar la simulación.

```
Seleccione una opcion: 2
--------------------------------------------------------------------------------

PID: 1, PPID: 1, Minutos de inicio: 0, Tiempo de vida: 4 minutos, Prioridad: 0
PID: 2, PPID: 1, Minutos de inicio: 0, Tiempo de vida: 5 minutos, Prioridad: 1
PID: 3, PPID: 1, Minutos de inicio: 1, Tiempo de vida: 4 minutos, Prioridad: 3
PID: 4, PPID: 2, Minutos de inicio: 4, Tiempo de vida: 3 minutos, Prioridad: 7
PID: 5, PPID: 2, Minutos de inicio: 6, Tiempo de vida: 2 minutos, Prioridad: 2
PID: 6, PPID: 3, Minutos de inicio: 9, Tiempo de vida: 5 minutos, Prioridad: 1
PID: 7, PPID: 3, Minutos de inicio: 10, Tiempo de vida: 3 minutos, Prioridad: 4
PID: 8, PPID: 4, Minutos de inicio: 12, Tiempo de vida: 2 minutos, Prioridad: 6
PID: 9, PPID: 7, Minutos de inicio: 13, Tiempo de vida: 1 minutos, Prioridad: 8
PID: 10, PPID: 5, Minutos de inicio: 15, Tiempo de vida: 4 minutos, Prioridad: 2
PID: 11, PPID: 4, Minutos de inicio: 16, Tiempo de vida: 3 minutos, Prioridad: 3
PID: 14, PPID: 6, Minutos de inicio: 18, Tiempo de vida: 2 minutos, Prioridad: 4
PID: 12, PPID: 5, Minutos de inicio: 18, Tiempo de vida: 2 minutos, Prioridad: 5
PID: 13, PPID: 6, Minutos de inicio: 18, Tiempo de vida: 1 minutos, Prioridad: 1
PID: 16, PPID: 8, Minutos de inicio: 19, Tiempo de vida: 3 minutos, Prioridad: 6
PID: 15, PPID: 7, Minutos de inicio: 19, Tiempo de vida: 5 minutos, Prioridad: 0
PID: 17, PPID: 10, Minutos de inicio: 20, Tiempo de vida: 2 minutos, Prioridad: 7
PID: 19, PPID: 9, Minutos de inicio: 21, Tiempo de vida: 4 minutos, Prioridad: 2
PID: 18, PPID: 8, Minutos de inicio: 21, Tiempo de vida: 1 minutos, Prioridad: 8
```

En caso no se haya creado previamente los procesos y almacenados en la pila con la opción 1. Se mostrará en pantalla que la pila se encuentra vacía.

```
Seleccione una opcion: 2
--------------------------------------------------------------------------------

La pila est├í vac├¡a.
```

3. **Borrar pila de procesos**

Esta opción elimina todos los procesos de la pila, dejando el sistema sin procesos en ella. Esta operación reinicia la pila y se asegura de que el usuario pueda empezar desde cero si desea recargar una nueva pila con los procesos con la opción 1 pero con diferentes PID ya que estos son identificadores únicos para cada proceso. Para ello, se verifica si la pila

está vacía usando ***pila.esVacia()***. Si contiene procesos, se elimina cada uno usando ***pila.desapilar()*** hasta que la pila esté vacía.

Se muestra un mensaje indicando que la pila ha sido vaciada correctamente, o si en su defecto si ya se encontraba vacía.

```
Seleccione una opcion: 3
--------------------------------------------------------------------------------

Pila de procesos eliminada correctamente.
```

En caso de que la pila esté vacía y se intente ejecutar la opción 3 de vaciar la pila. Este mostrará el siguiente mensaje de que la pila ya se encuentra vacía.

```
Seleccione una opcion: 3
--------------------------------------------------------------------------------

La pila de procesos ya est├í vac├¡a.
```

4. **Aumentar tiempo del sistema (n minutos)**

Aumenta el tiempo del sistema en una cantidad especificada por el usuario y simula la ejecución de procesos. Durante esta simulación, se realiza una serie de pasos para gestionar los procesos:

Inserción de procesos en núcleos y sus colas si su tiempo de inicio coincide con el tiempo inicial 0.

Se verifica si los procesos en ejecución han completado su tiempo de vida, en cuyo caso son eliminados del núcleo y se eliminan los núcleos que están inactivos.

Se actualizan los tiempos de inicio y fin para calcular la estancia en el sistema.

Se insertan en los núcleos o sus colas de espera los procesos que inicien su ejecución en el tiempo actual.

```
Seleccione una opcion: 4
-----------------------------------------------------------------------------------------------------

Ingrese el numero de minutos a aumentar: 1


-----------------------------------------------------------------------------------------------------
Tiempo actual: 00:00

Proceso en la cima de la pila: PID: 1, PPID: 1, Inicio: 0, Tiempo de vida: 4, Prioridad: 0

00:00 | Proceso iniciado en nucleo 0: PID: 1, PPID: 1, Inicio: 0, Tiempo de vida: 4, Prioridad: 0

00:00 | Proceso iniciado en nucleo 2: PID: 2, PPID: 1, Inicio: 0, Tiempo de vida: 5, Prioridad: 1

Proceso en la cima de la pila: PID: 3, PPID: 1, Inicio: 1, Tiempo de vida: 4, Prioridad: 3


Nucleo en posicion 0:
00:00 | Proceso en nucleo 0 PID: 1, PPID: 1, Inicio: 0, Tiempo de vida: 4, Prioridad: 0
00:00 | Tiempo de inicio ejecucion: 0, Tiempo de finalizacion: 4
00:00 | Tiempo de ejecucion restante: 4 minutos
00:00 | Cola de procesos vacia

Nucleo en posicion 1:
00:00 | Proceso en nucleo 2 PID: 2, PPID: 1, Inicio: 0, Tiempo de vida: 5, Prioridad: 1
00:00 | Tiempo de inicio ejecucion: 0, Tiempo de finalizacion: 5
00:00 | Tiempo de ejecucion restante: 5 minutos
00:00 | Cola de procesos vacia
```

5. **Mostrar estado de los núcleos**

Muestra el estado actual de todos los núcleos en la lista. Se detalla cada núcleo, incluyendo el proceso en ejecución (si hay alguno), la cola de espera, el tiempo de inicio y fin del proceso en ejecución, y el tiempo de espera restante para completar su tarea.

```
Seleccione una opcion: 5
--------------------------------------------------------------------------------

Estado de los nucleos:

Nucleo en posicion 0:
00:01 | Proceso en nucleo 0 PID: 1, PPID: 1, Inicio: 0, Tiempo de vida: 4, Prioridad: 0
00:01 | Tiempo de inicio ejecucion: 0, Tiempo de finalizacion: 4
00:01 | Tiempo de ejecucion restante: 3 minutos
00:01 | Cola de procesos vacia

Nucleo en posicion 1:
00:01 | Proceso en nucleo 2 PID: 2, PPID: 1, Inicio: 0, Tiempo de vida: 5, Prioridad: 1
00:01 | Tiempo de inicio ejecucion: 0, Tiempo de finalizacion: 5
00:01 | Tiempo de ejecucion restante: 4 minutos
00:01 | Cola de procesos vacia
```

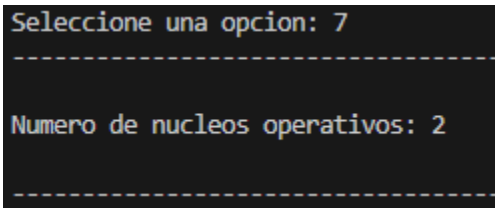
6. **Consulta núcleo con menos carga y núcleo con más carga**

Identifica y muestra los núcleos que tienen la menor y mayor carga de procesos en sus colas de espera. Esta consulta es útil para balancear la carga entre los núcleos, permitiendo la asignación dinámica de procesos a los núcleos menos saturados.

```
Seleccione una opcion: 6
--------------------------------------------------------------------------------

Nucleo con menos carga de procesos:
00:01 | Proceso en nucleo 0 PID: 1, PPID: 1, Inicio: 0, Tiempo de vida: 4, Prioridad: 0
00:01 | Tiempo de inicio ejecucion: 0, Tiempo de finalizacion: 4
00:01 | Tiempo de ejecucion restante: 3 minutos
00:01 | Cola de procesos vacia

Nucleo con mas carga de procesos:
00:01 | Proceso en nucleo 0 PID: 1, PPID: 1, Inicio: 0, Tiempo de vida: 4, Prioridad: 0
00:01 | Tiempo de inicio ejecucion: 0, Tiempo de finalizacion: 4
00:01 | Tiempo de ejecucion restante: 3 minutos
00:01 | Cola de procesos vacia
```

7. **Consulta número de núcleos operativos**

Muestra el número total de núcleos actualmente activos en la lista de núcleos. Esto permite al usuario conocer cuántos núcleos están siendo utilizados en el sistema para ejecutar los procesos.



8. **Simular ejecución de procesos**

Simula el flujo completo de tiempo en el sistema operativo, ejecutando todos los procesos hasta su finalización. En cada minuto, se realizan las siguientes acciones:

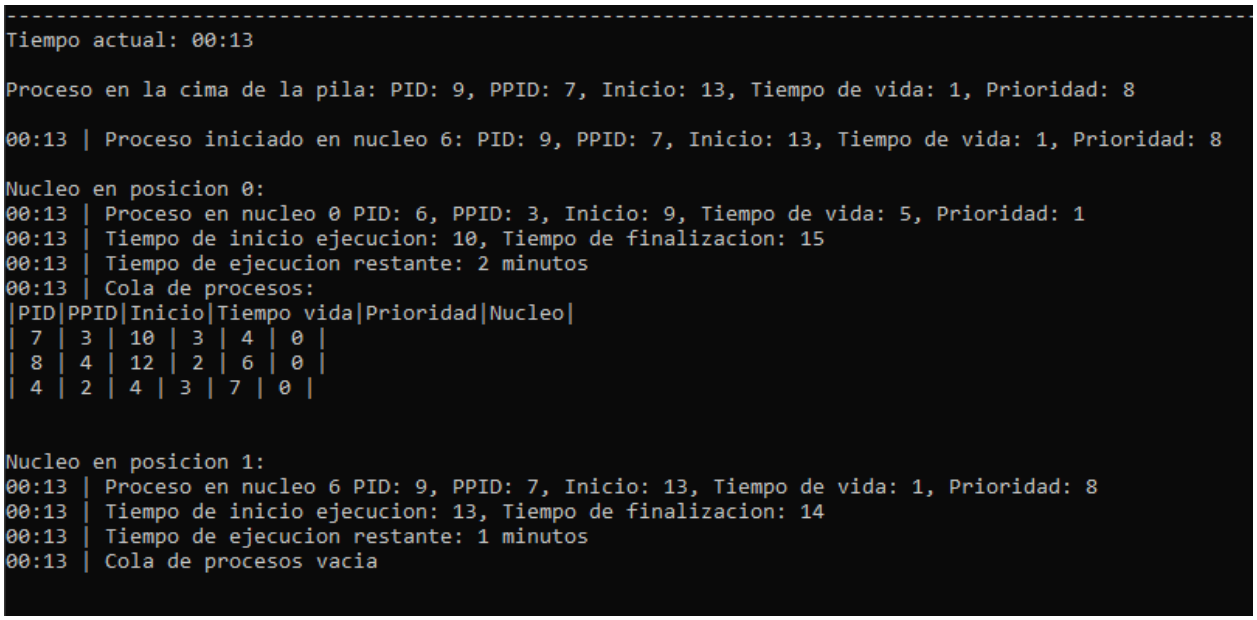
Inserción de procesos en núcleos y sus colas si su tiempo de inicio coincide con el tiempo inicial 0.

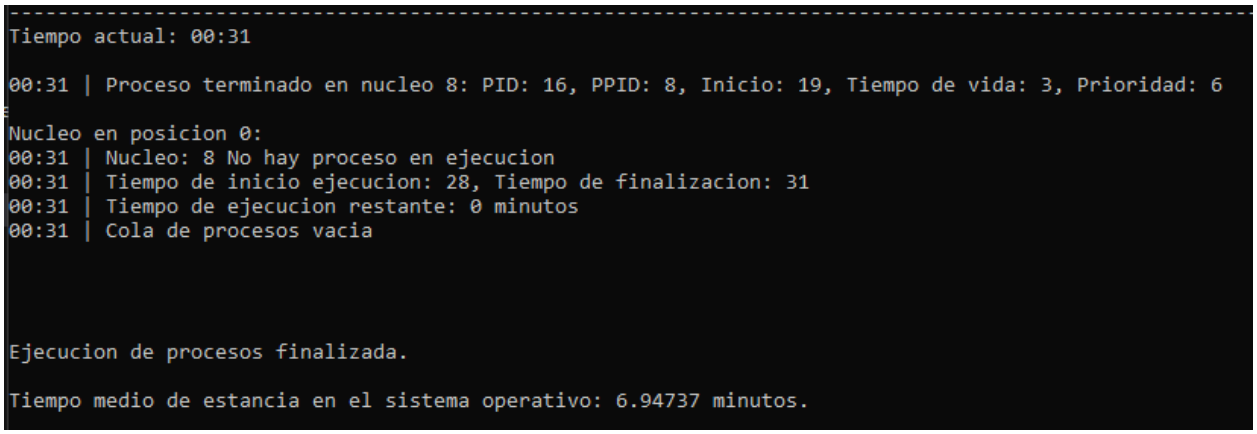
Eliminación de procesos completados y liberación de núcleos que están inactivos.

Se actualizan los tiempos de inicio y fin para calcular la estancia en el sistema.

Se insertan en los núcleos o sus colas de espera los procesos que inicien su ejecución en el tiempo actual.

La simulación continúa hasta que todos los procesos en la pila, en las colas, y en ejecución han sido finalizados, y al finalizar muestra el tiempo medio de estancia en el sistema.





9. **Salir**

Finaliza el programa y cierra la ejecución de la simulación de procesos.

RESULTADOS

La simulación del planificador de procesos desarrollada en esta práctica muestra una organización eficiente para administrar procesos en función de sus prioridades y tiempos de ejecución.

En la primera parte de la práctica, el sistema logra procesar y asignar correctamente procesos a tres núcleos fijos, obteniendo un tiempo medio de estancia en el sistema.

En la segunda parte, el sistema gestiona la carga de trabajo en núcleos dinámicos, eliminando núcleos vacíos o sobrecargados y creando nuevos cuando se requiere, optimizando así el uso de recursos en tiempo real y alcanzando un tiempo medio de estancia.

Los resultados indican que el sistema es capaz de adaptar dinámicamente su estructura para mantener una distribución equitativa de la carga de trabajo, adaptando según el enunciado la eficiencia del procesamiento y los tiempos de espera para los procesos en cola.

CONCLUSIÓN

- Este trabajo destacó la importancia del uso de la pila, cola y lista enlazada para gestionar eficientemente los procesos y núcleos en un sistema de planificación. La pila nos permitió organizar los procesos en orden de ingreso, la cola facilitó la gestión de los procesos en espera por prioridad y tiempo de inicio, mientras que la lista enlazada hizo posible una asignación dinámica de los núcleos.
- La implementación de la lógica para cada estructura, como el apilamiento de procesos, y encolar según prioridad y la asignación dinámica de núcleos, proporcionó una base sólida para el desarrollo de un sistema de planificación realista.
- A lo largo del desarrollo de esta práctica, hemos gestionado la planificación del curso de ejecución de los procesos, asignando correctamente los recursos y manejando la carga de trabajo. Permitiendo que el sistema pudiera adaptarse a variaciones en tiempo indicados por el usuario, optimizando el uso de los núcleos y minimizando el tiempo de espera en cola.