1 Objetivo

El objetivo de esta tarea es que los estudiantes comprendan y apliquen los conceptos fundamentales de **estructuras de datos** en el contexto de la simulación de un sistema operativo. A través de la implementación de este simulador, los estudiantes aprenderán cómo utilizar diversas estructuras de datos, como **arreglos**, **listas enlazadas**, **pilas**, **colas** y **listas circulares**, para gestionar eficientemente la memoria de los procesos y la planificación de su ejecución. Se espera que logren integrar estas estructuras de manera adecuada para resolver problemas de asignación de memoria, manejo de procesos y optimización del uso de recursos en el sistema operativo.

2 Descripción

El simulador debe gestionar de manera eficiente la memoria y la planificación de procesos, emulando los principios de un sistema operativo real. Deberá asignar y liberar memoria dinámicamente, gestionar el stack de cada proceso y manejar las colas de procesos en espera o suspendidos. Además, los estudiantes deberán implementar un planificador de procesos que determine el orden de ejecución. Se pueden implementar distintos algoritmos de planificación, como **FIFO**, **Round Robin** y **SJF (Shortest Job First)**.

3 Estructuras de Datos Sugeridas

1. **Arreglos**: Para representar el espacio de memoria dividido en bloques.
2. **Listas Enlazadas Simples**: Para manejar la lista de procesos en memoria.
3. **Pilas**: Para gestionar el stack de cada proceso.
4. **Colas**: Para implementar la lista de procesos en espera o suspendidos.
5. **Listas Circulares**: Para implementar la rotación de procesos en el planificador tipo Round Robin.
6. **Arreglos** o **Listas**: Para almacenar la información de los procesos y su estado (nuevo, listo, ejecutando, bloqueado, finalizado).

4 Funcionalidad

1. **Simulación de Asignación Dinámica de Memoria y Liberación**: Usando listas enlazadas, el simulador debe asignar y liberar memoria de acuerdo con la solicitud y finalización de los procesos.
    - **First Fit**: Asigna el primer bloque libre que sea lo suficientemente grande.
    - **Best Fit**: Encuentra el bloque más pequeño que sea lo suficientemente grande para minimizar la fragmentación.
    - **Worst Fit**: Encuentra el bloque más grande disponible.

2. **Gestión del Stack de Procesos**: Cada proceso debe tener un stack gestionado mediante pilas, reflejando el uso de memoria para las llamadas a funciones.

3. **Planificación de Procesos**: Implementar un planificador que utilice al menos dos algoritmos, como:

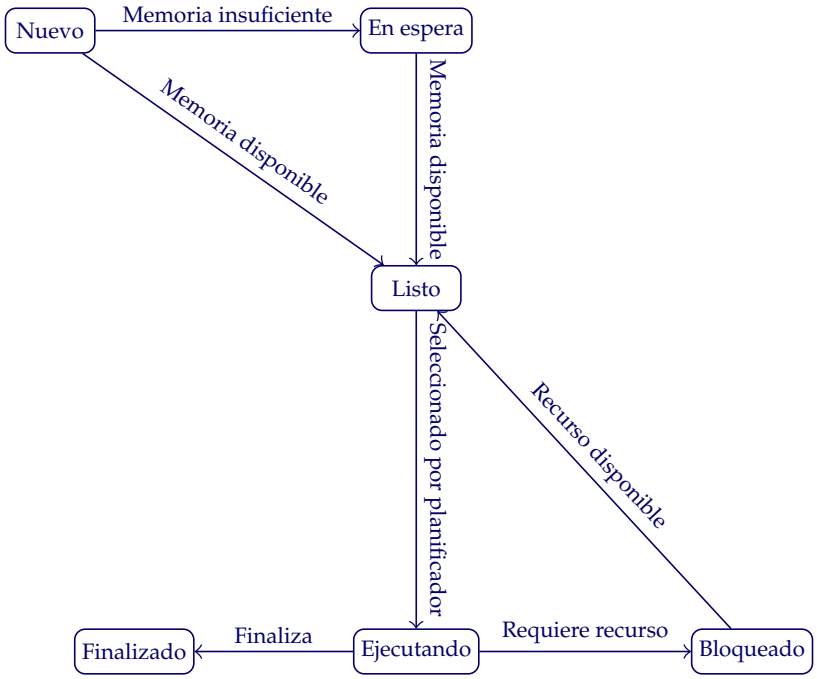
   - **FIFO**: Los procesos se ejecutan en el orden en que llegan.
   - **Round Robin**: Los procesos se ejecutan en ciclos, alternando en turnos de tiempo fijos (quantum).
   - **SJF**: El proceso con menor tiempo de ejecución es el primero en ser ejecutado.

4. **Manejo de Múltiples Procesos**: Los procesos deben poder entrar y salir de la ejecución, mientras el planificador decide cuál ejecutar.

5. **Simulación de Fragmentación de Memoria**: Simular la fragmentación interna y externa durante la asignación y liberación de memoria.

6. **Cambio de Contexto**: Implementar el cambio de contexto entre procesos gestionado por el planificador, manteniendo el estado de los procesos (memoria asignada, stack, etc.).

Este diagrama modela los cambios de estado de los procesos en un sistema de gestión de memoria, capturando las transiciones clave entre asignación de memoria, ejecución y finalización.



5 Ejemplo de Uso

5.1 Estados y Asignación de Memoria (First Fit)

A continuación, se muestra un ejemplo de archivo de entrada en formato texto (‘entrada.txt‘):

```
# Algoritmo de asignación de memoria
algoritmo: firstfit

# Bloques de memoria disponibles (en KB)
memoria:
  - 120
  - 60
```

```
  - 80
  - 40

# Algoritmo de planificación de procesos
planificacion: roundrobin
quantum: 3

# Lista de procesos
procesos:
  - nombre: P1
    tiempo_ejecucion: 8
    memoria_solicitada: 100
  - nombre: P2
    tiempo_ejecucion: 4
    memoria_solicitada: 50
  - nombre: P3
    tiempo_ejecucion: 9
    memoria_solicitada: 200
  - nombre: P4
    tiempo_ejecucion: 5
    memoria_solicitada: 75
  - nombre: P5
    tiempo_ejecucion: 2
    memoria_solicitada: 30
```

Para ejecutar el simulador con el archivo de entrada, este debe ser pasado como argumento al simulador utilizando el siguiente comando en la terminal:

```
./simulador --file entrada.txt
```

El simulador leerá el archivo 'entrada.txt' y asignará memoria a los procesos utilizando el algoritmo de **First Fit**. Si algún proceso no puede ser asignado debido a la falta de bloques de memoria adecuados, será puesto en estado de **Espera** hasta que haya suficiente memoria disponible.

5.2 Salida

Puede ser carta Gantt, orden de procesos y estados.

6 Entrega y Evaluación

- Código fuente completo que incluya la implementación del simulador de gestión de memoria y el planificador.
- Documentación explicando el diseño, los algoritmos implementados y los resultados obtenidos.

La tarea es en grupos de tres personas.