UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas

PROYECTO DE UNIDAD

Curso: Sistemas Operativos I

Docente: Ing. Hugo Manuel Barraza Vizcarra

Jahuira Pilco, Dayan Elvis (2022075749)

Tacna – Perú
2025

1. INTRODUCCIÓN

Comprender cómo se gestionan los procesos, la memoria y la planificación de la CPU resulta esencial. Estos aspectos constituyen la base del funcionamiento de cualquier sistema informático, ya que determinan el uso eficiente de los recursos disponibles y la correcta ejecución de las tareas. No obstante, su complejidad hace que muchas veces solo se comprendan de manera teórica, sin poder observar con claridad cómo interactúan los diferentes componentes en situaciones reales.

Con el fin de facilitar este aprendizaje, se plantea el diseño de un simulador de un sistema operativo simple que permita representar la ejecución de procesos y la gestión de memoria. A través de este simulador será posible experimentar con diferentes algoritmos de planificación de CPU y estrategias de asignación de memoria, lo cual proporcionará una visión práctica de los conceptos y permitirá analizar sus ventajas y limitaciones.

1.1. Problema

El diseño de un sistema operativo requiere seleccionar algoritmos de planificación y estrategias de asignación de memoria que equilibren la eficiencia, la equidad y el aprovechamiento de recursos. Sin embargo, el comportamiento real de estas técnicas puede variar según las características de los procesos y la disponibilidad de memoria, lo que genera la necesidad de contar con herramientas que permitan evaluar su desempeño en distintos escenarios.

La dificultad principal radica en identificar cómo influyen algoritmos como First-Come, First-Served (FCFS), Shortest Process Next (SPN) y Round Robin (RR) en las métricas de ejecución de procesos, tales como el tiempo de respuesta, de espera y de retorno. Asimismo, se requiere analizar los efectos de la fragmentación y la eficiencia de las estrategias de asignación de memoria First-Fit y Best-Fit.

Frente a este contexto, se plantea la construcción de un simulador de sistema operativo capaz de representar estos algoritmos y estrategias, generando resultados cuantitativos que permitan evaluar su impacto y realizar comparaciones objetivas.

1.2. Objetivos

1.2.1. Objetivo general

Diseñar e implementar un simulador de sistema operativo que permita analizar la planificación de procesos y la gestión de memoria utilizando algoritmos y estrategias reconocidas.

1.2.2. Objetivos específicos

- Implementar un simulador que gestione procesos con su respectivo PCB (Process Control Block).
- Calcular métricas de rendimiento como tiempo de respuesta, espera, retorno y throughput.
- Simular la planificación de CPU mediante los algoritmos FCFS, SPN y Round Robin.
- Implementar estrategias de asignación de memoria con First-Fit y Best-Fit.
- Analizar los resultados obtenidos y discutir los compromisos de cada técnica.

1.3. Alcance

El proyecto abarca la implementación de procesos con atributos básicos como identificador, tiempo de llegada, tiempo de servicio, inicio y fin de ejecución. Además, se calcularán métricas tanto individuales como globales para evaluar el rendimiento de los algoritmos de planificación.

La simulación incluirá la gestión de memoria como un espacio lineal, aplicando las estrategias de First-Fit y Best-Fit para la asignación de bloques. La entrada del simulador se configurará mediante archivos JSON, lo que permitirá definir procesos y solicitudes de memoria de manera flexible. Los resultados se mostrarán en tablas de fácil interpretación, reflejando los cálculos obtenidos en cada caso.

No se considerarán características avanzadas como prioridades de procesos, multiprogramación a gran escala ni la gestión de dispositivos de entrada/salida. Tampoco se implementarán técnicas de paginación o segmentación de memoria.

1.4. Supuestos

- El cambio de contexto no genera sobrecarga, salvo que se indique explícitamente.
- Todos los procesos llegan de manera conocida y sus tiempos de servicio son predefinidos.
- La memoria es un espacio lineal y no se contempla paginación ni segmentación.

2. MARCO CONCEPTUAL

Un sistema operativo cumple el rol fundamental de administrar los recursos de hardware y software de un computador, garantizando que múltiples programas puedan ejecutarse de manera eficiente y segura. Dentro de sus funciones principales se encuentran la planificación de procesos y la gestión de la memoria, elementos que constituyen la base de este proyecto.

2.1. Proceso y PCB (Process Control Block)

Un proceso es un programa en ejecución que requiere de recursos como CPU, memoria y dispositivos de entrada/salida. Cada proceso se representa mediante un PCB (Bloque de Control de Proceso), que almacena información esencial para su gestión: identificador (PID), estado, tiempo de llegada, tiempo de servicio, dirección de memoria asignada, entre otros.

2.2. Estados de un proceso

Un proceso evoluciona a través de distintos estados durante su ciclo de vida:

- **Nuevo:** el proceso ha sido creado, pero aún no se ejecuta.
- **Listo:** el proceso espera la asignación de CPU.
- **Ejecución:** el proceso se encuentra utilizando la CPU.
- **Bloqueado:** el proceso está a la espera de un evento externo (E/S).
- **Terminado:** el proceso ha finalizado su ejecución.

2.3. Algoritmos de planificación de CPU

First-Come, First-Served (FCFS): los procesos se atienden en el orden en que llegan. Es simple de implementar, pero puede provocar el efecto convoy, donde un proceso largo retrasa a los demás.

Shortest Process Next (SPN o SJF no expropiativo): selecciona al proceso con menor tiempo de servicio. Minimiza el tiempo promedio de espera, pero requiere conocer la duración de los procesos y puede perjudicar a los más largos.

Round Robin (RR): asigna un quantum fijo a cada proceso en la cola de listos. Favorece la equidad en sistemas interactivos, aunque su rendimiento depende del tamaño del quantum elegido

2.4. Métricas de rendimiento

Para evaluar los algoritmos de planificación se utilizan medidas cuantitativas:

- Tiempo de respuesta: diferencia entre el tiempo de inicio y el de llegada.
- Tiempo de espera: tiempo total en el que el proceso permanece en la cola de listos sin ejecutarse.

- Tiempo de retorno (turnaround): diferencia entre el tiempo de finalización y el de llegada.
- Throughput: cantidad de procesos completados por unidad de tiempo.

2.5. Gestión de memoria

La memoria principal es un recurso limitado y debe ser administrada eficientemente para permitir la ejecución concurrente de procesos. Una forma de hacerlo es mediante asignación contigua, donde cada proceso recibe un bloque continuo de memoria. En este esquema, la elección del bloque adecuado puede realizarse con distintas estrategias:

- **First-Fit:** asigna el primer bloque libre que sea lo suficientemente grande para el proceso. Es rápido, pero tiende a generar fragmentación externa.
- **Best-Fit:** selecciona el bloque libre más pequeño que pueda contener al proceso, reduciendo el desperdicio de espacio, aunque requiere mayor búsqueda y puede dejar bloques muy pequeños inutilizables.

3. DISEÑO DEL SIMULADOR

El simulador fue diseñado en C++ utilizando estructuras de datos básicas y orientadas a representar los elementos esenciales de un sistema operativo: procesos, planificación de CPU y gestión de memoria.

Para los procesos, se definió una estructura (struct Process) que actúa como un PCB (Process Control Block), almacenando atributos como identificador, tiempo de llegada, tiempo de servicio, inicio, fin y tiempo restante. Además, se incluyen funciones para calcular las métricas de respuesta, espera y retorno, de modo que puedan evaluarse automáticamente al finalizar la simulación.

La planificación de CPU se implementó a través de tres algoritmos:

- First-Come, First-Served (FCFS): los procesos se ejecutan en el orden de llegada, sin expropiación.
- Shortest Process Next (SPN/SJF no expropiativo): siempre se selecciona el proceso con menor tiempo de servicio, garantizando mejor tiempo promedio de espera.
- Round Robin (RR): se utiliza un quantum configurable, y los procesos se alternan en una cola circular implementada con deque.

En todos los casos, los procesos se ordenan inicialmente por tiempo de llegada y luego son gestionados en una cola de listos que cambia de acuerdo con el algoritmo seleccionado.

En cuanto a la memoria, se representó como un vector de bloques (struct Block), cada uno con atributos de tamaño, inicio, estado (libre/ocupado) y PID asignado. Se implementaron dos estrategias de asignación contigua:

- First-Fit: selecciona el primer bloque libre suficientemente grande.
- Best-Fit: selecciona el bloque más pequeño que pueda contener al proceso.

El simulador permite además fragmentar y fusionar bloques libres (coalescing), manteniendo actualizado el estado de la memoria. Las solicitudes de memoria se cargan desde un archivo JSON y se procesan según la estrategia definida.

La configuración general se gestiona a través de un archivo externo (config.json), que especifica:

- Algoritmo de CPU a utilizar.
- Quantum en caso de Round Robin.
- Estrategia y tamaño de memoria.
- Lista de procesos (con llegada y servicio).
- Solicitudes de memoria (PID y tamaño).

Finalmente, los resultados se imprimen en tablas: una con las métricas de los procesos y otra con el estado final de la memoria, lo que permite analizar comparativamente el comportamiento de cada algoritmo.

4. METODOLOGÍA DE EXPERIMENTOS

Para evaluar el funcionamiento del simulador y analizar el comportamiento de los algoritmos implementados, se definió una serie de experimentos controlados. Cada uno de ellos considera parámetros de entrada específicos, descritos en archivos de configuración en formato JSON, lo cual permite garantizar la repetibilidad de los resultados.

4.1. Variables consideradas

- Algoritmo de planificación de CPU: se seleccionaron tres algoritmos: FCFS, SPN y RR.
- Quantum de ejecución: aplicable únicamente al algoritmo Round Robin; se definieron valores de prueba de 2 y 4 unidades de tiempo.
- Estrategia de asignación de memoria: se evaluaron los esquemas First-Fit y Best-Fit.

- Conjunto de procesos: se definieron listas de procesos con tiempos de llegada y servicio distintos, con el fin de observar cómo se alteran las métricas bajo diferentes algoritmos.

4.2. Métricas de evaluación

Para cada experimento se calcularon las siguientes métricas:

- Tiempo de respuesta promedio.
- Tiempo de espera promedio.
- Tiempo de retorno (turnaround) promedio.
- Throughput (procesos completados por unidad de tiempo).
- Estado de la memoria después de las asignaciones solicitadas.

4.3. Escenarios de prueba

Se diseñaron tres escenarios principales:

- **Escenario A – Comparación de algoritmos de planificación**
  - Tres procesos con diferentes tiempos de llegada y servicio.
  - Evaluación con FCFS, SPN y RR (quantum = 4).
  - Objetivo: analizar diferencias en respuesta, espera y retorno.
- **Escenario B – Variación del quantum en Round Robin**
  - Mismos procesos que en el Escenario A.
  - Comparación de resultados con quantum = 2 y quantum = 4.
  - Objetivo: observar cómo influye el tamaño del quantum en la equidad y el rendimiento.
- **Escenario C – Gestión de memoria**
  - Definición de procesos con solicitudes de memoria de distintos tamaños.
  - Evaluación con First-Fit y Best-Fit sobre un espacio de memoria total de 1 MB.
  - Objetivo: identificar fragmentación, espacio libre remanente y eficiencia de asignación.

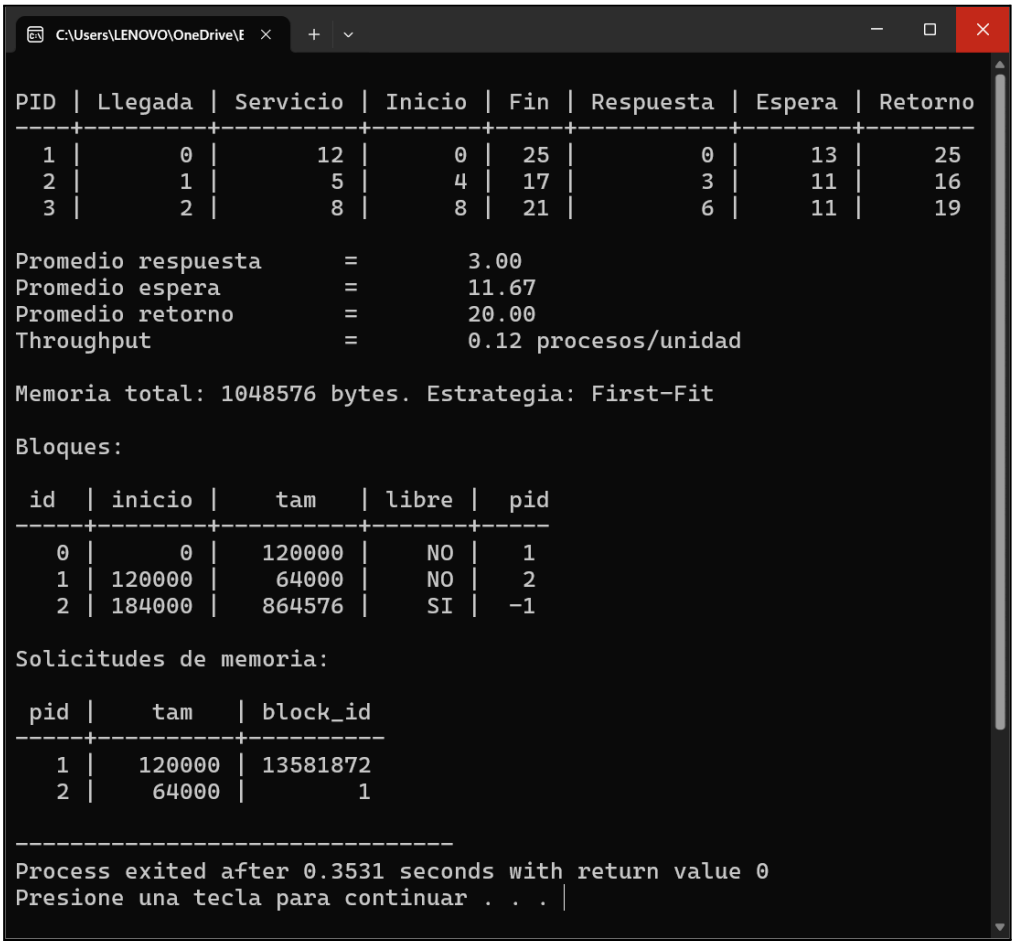
4.4. Procedimiento

a) Preparar el archivo config.json con los parámetros del escenario.
b) Ejecutar el simulador y registrar los resultados en tablas.
c) Calcular los promedios de las métricas y el throughput.
d) Analizar el estado de los bloques de memoria tras la asignación.
e) Repetir el procedimiento para cada escenario, variando únicamente el parámetro a evaluar.

5. RESULTADOS

El simulador produce como salida:

- Una tabla con los atributos y métricas de cada proceso.
- Los valores promedio de las métricas y el throughput.
- El estado final de la memoria, con los bloques libres y ocupados, además de las solicitudes atendidas.



6. DISCUSIÓN

- FCFS: sencillo pero genera convoy effect.
- SPN: mejora tiempos de espera, pero requiere conocer el servicio.
- RR: más justo, pero depende del quantum (muy pequeño → sobrecarga, muy grande → similar a FCFS).
- Memoria: First-Fit rápido pero fragmenta más; Best-Fit optimiza espacio pero es más lento.

7. CONCLUSIONES

- El simulador permitió observar diferencias claras entre algoritmos.
- No existe un algoritmo perfecto: depende del contexto.
- La estrategia de memoria influye en la fragmentación y en el aprovechamiento del espacio.

- El archivo JSON facilita la configuración de escenarios y la reproducibilidad de experimentos.