Simulación de Planificación de Procesos

Sistemas Operativos (SI2004)

Juan Esteban Trujillo

Camilo Alvarez

Samuel Calderon

Profesor
José Luis Montoya Pareja

Universidad: EAFIT

8 de agosto de 2025

1. Introducción

En el contexto de los sistemas operativos modernos, múltiples procesos compiten por el uso del procesador. Debido a que el CPU solo puede ejecutar una instrucción a la vez, el sistema operativo debe implementar mecanismos de planificación para repartir el tiempo de forma justa entre los procesos. Esta planificación requiere que se realice un cambio de contexto, donde el estado del proceso actual se guarda y se restaura el del siguiente proceso.

El propósito de este proyecto es simular este comportamiento mediante la creación de un programa en C++ que imita la ejecución de varios procesos, gestionados por un planificador round-robin. A través de esta simulación, se visualiza claramente cómo se da un cambio de contexto, qué elementos del estado del proceso se conservan, y cómo se administra el quantum de CPU.

2. Objetivos

- Comprender cómo se representa internamente un proceso dentro de un sistema operativo.
- Simular un planificador round-robin que distribuya el uso del CPU de manera equitativa.
- Observar, paso a paso, el comportamiento de los procesos y los cambios de contexto.
- Aplicar estructuras en C++ para modelar componentes del sistema operativo.
- Usar mecanismos clave-valor para organizar instrucciones simuladas.

3. Descripción del funcionamiento

El programa Procplanner simula la planificación de procesos en un sistema operativo utilizando el algoritmo Round Robin. Cada proceso tiene registros (AX, BX, CX), un contador de programa (PC), un quantum y un estado (Listo, Ejecutando, Bloqueado, Terminado). Los procesos y sus instrucciones se leen desde archivos de texto. El planificador ejecuta los procesos por turnos, mostrando el estado de cada uno en tablas y registrando la ejecución en un archivo de log.

3.1. Estructura del Proceso

Cada proceso se representa mediante una estructura en C++ llamada Proceso, la cual contiene la siguiente información:

```
typedef struct {
   int pid;       // Identificador del proceso
   int pc;        // Contador de programa (posición de la instrucción)
   int ax, bx, cx; // Registros simulados
   int quantum;   // Tiempo de CPU asignado por turno
   char estado[10]; // Estado del proceso (Listo, Ejecutando, Terminado)
} Proceso;
```

3.2. Carga de procesos

La carga de procesos se realiza exclusivamente desde un archivo de texto (.txt). El programa lee un archivo donde cada línea describe un proceso con el siguiente formato:

PID: 1 AX=2 BX=3 CX=1 Q=3

o bien:

PID: 2, AX=5, BX=2, CX=4, Quantum=5

El sistema interpreta cada línea, inicializa los registros y el quantum, y almacena cada proceso en una estructura de tipo clave-valor (unordered_map<int, Proceso>), usando el PID como clave.

3.3. Planificador round-robin

Se implementó un planificador simple basado en el algoritmo Round Robin, que selecciona los procesos en orden circular y asigna a cada uno su quantum de CPU. Si un proceso termina antes de agotar su quantum, el planificador pasa inmediatamente al siguiente proceso disponible.

En cada cambio de contexto, el programa imprime información relevante, por ejemplo:

[Cambio de contexto]
Guardando estado de Proceso 1: PC=2, AX=5, BX=3
Cargando estado de Proceso 2: PC=0, AX=1, BX=2

Esto permite visualizar claramente el mecanismo de guardado y restauración de estados entre procesos.

3.4. Instrucciones simuladas

Las instrucciones que puede ejecutar cada proceso están orientadas a modificar los registros simulados (AX, BX, CX), realizar saltos condicionales o incondicionales, y simular

operaciones típicas de un procesador sencillo. El sistema interpreta cada instrucción como una cadena de texto y la ejecuta sobre los registros del proceso correspondiente. Las instrucciones soportadas incluyen:

- ADD REG1, REG2: Suma el valor de REG2 al registro REG1. Ejemplo: ADD AX, BX suma BX a AX.

- ADD REG, VAL: Suma un valor inmediato al registro. Ejemplo: ADD CX, 1 suma 1 a CX.

- SUB REG, REG2: Resta el valor de REG2 al registro REG1. Ejemplo: SUB BX, AX resta AX a BX.

- SUB REG, VAL: Resta un valor inmediato al registro. Ejemplo: SUB CX, 2 resta 2 a CX.

- MUL REG, REG2: Multiplica el registro REG1 por el valor de REG2. Ejemplo: MUL AX, BX.

- MUL REG, VAL: Multiplica el registro por un valor inmediato. Ejemplo: MUL BX, 3.

- INC REG: Incrementa en 1 el registro indicado. Ejemplo: INC AX.

- NOP: No realiza ninguna operación (instrucción de espera o relleno).

- JMP N: Salta a la instrucción número N (el contador de programa PC se actualiza a N-1).

3.5. Ejecución paso a paso

El simulador Procplanner ejecuta cada proceso siguiendo una secuencia detallada que permite observar el comportamiento del planificador y el avance de los procesos en cada ciclo. El flujo de ejecución es el siguiente:

1. Selección del proceso: El planificador Round Robin selecciona el siguiente proceso en estado “Listo” según el orden de los PIDs.
2. Cambio de contexto: Si corresponde, se realiza un cambio de contexto, mostrando en pantalla y en el log el guardado del estado del proceso saliente y la carga del estado del proceso entrante.

3. Lectura de instrucciones: Se cargan las instrucciones asociadas al proceso desde su archivo correspondiente.

4. Ciclo de ejecución: Mientras el proceso tenga quantum disponible y queden instrucciones por ejecutar:
   4.1. Se lee la instrucción actual (según el PC).
   4.2. Se interpreta y ejecuta la instrucción, modificando los registros o el PC según corresponda.
   4.3. Se muestra en consola el ciclo actual, el PC, el quantum restante y la instrucción ejecutada.
   4.4. Se imprime la tabla con el estado actualizado de todos los procesos.
   4.5. Se descuenta un ciclo de quantum y se incrementa el PC (o se salta si la instrucción es un JMP).
   4.6. Se simula una posible interrupción aleatoria (bloqueo), cambiando el estado del proceso a "Bloqueado" si ocurre.

5. Finalización del turno: Al terminar el quantum, agotar las instrucciones o ser interrumpido, el proceso actualiza su estado a "Terminado" o "Bloqueado" según corresponda. El planificador pasa al siguiente proceso.

6. Repetición: El ciclo se repite hasta que todos los procesos hayan terminado o estén bloqueados.

4. Entorno de Desarrollo

- Sistema operativo: Windows 11 usando WSL (Windows Subsystem for Linux)
- Distribución de Linux en WSL: Ubuntu 24.04.2 LTS
- Compilador utilizado: g++ (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0

5. Resultados

Al realizar diferentes simulaciones con procesos distintos se pudieron obtener los siguientes resultados:

- Durante la simulación, se generan tablas que muestran el estado de los procesos en cada ciclo, incluyendo los valores de los registros, el quantum restante y el estado actual. Ejemplo de tabla inicial:

PID	PC	AX	BX	CX	Quantum	Estado
1	0	2	3	1	8	Listo
2	0	5	2	4	5	Listo
3	0	8	1	6	6	Listo
4	0	3	7	2	4	Listo
5	0	0	0	0	3	Listo

- A lo largo de la ejecución, se observa el avance de cada proceso, los cambios de contexto y la evolución de los registros. Por ejemplo, tras ejecutar varias instrucciones y cambios de contexto, el estado final es:

PID	PC	AX	BX	CX	Quantum	Estado
---	--	--	--	--	-------	------
1	4	5	1	2	4	Terminado
2	3	15	10	4	0	Terminado
3	3	7	2	7	3	Bloqueado
4	3	4	9	2	0	Terminado
5	2	0	1	0	1	Bloqueado

- Evolución paso a paso: El log y la salida muestran cómo cada proceso ejecuta sus instrucciones, cómo cambian los valores de los registros y cómo se producen interrupciones y bloqueos. Ejemplo de ciclo:

Ciclo 2 | PC = 2 | Quantum restante = 6 | Inst: SUB BX 2

PID	PC	AX	BX	CX	Quantum	Estado
---	--	--	--	--	-------	------
1	2	5	1	1	6	Ejecutando
2	0	5	2	4	5	Listo
3	0	8	1	6	6	Listo
4	0	3	7	2	4	Listo
5	0	0	0	0	3	Listo

- Tipo de procesador y memoria RAM utilizada:
    - Procesador utilizado:
        - 13th Gen Intel(R) Core(TM) i7-13620H

    - Memoria RAM utilizada:
        - Maximum resident set size (kbytes): 3712

6. Conclusiones

- El simulador Procplanner permitió observar de manera clara el funcionamiento del algoritmo Round Robin, mostrando cómo se reparte el tiempo de CPU entre los procesos y cómo se gestionan los cambios de contexto y los estados de cada proceso.

- Se evidenció que los procesos pueden ser interrumpidos y quedar bloqueados antes de agotar su quantum, lo que refleja situaciones reales de sistemas operativos donde ocurren interrupciones inesperadas.

- El registro detallado de cada ciclo, con la evolución de los registros y el quantum, facilita el análisis del comportamiento de cada proceso.

- El uso de archivos de texto para la definición de procesos e instrucciones hace que el simulador sea flexible y fácil de adaptar a diferentes escenarios de prueba.

- La simulación demostró que la eficiencia del planificador depende tanto del quantum asignado como de la naturaleza de las instrucciones y la ocurrencia de interrupciones.