Práctica 03: Algoritmos de Planificación 1

Maestro:

Javier Rosales Martinez

Materia:

Seminario de Solución de Problemas de Sistemas Operativos

Sección:

D06

Alumno:

Alejandro Covarrubias Sánchez

Código:

221350192

Antecedentes

Los algoritmos de planificación son métodos utilizados en sistemas operativos para gestionar cómo y cuándo se ejecutan los procesos, o tareas, en un procesador. Su principal objetivo es distribuir eficientemente el tiempo de CPU entre los procesos para maximizar el rendimiento del sistema.

Existen varios tipos de algoritmos de planificación, siendo FIFO (First-In, First-Out) el más simple. En este, los procesos se ejecutan en el orden en que llegan, pero puede ser ineficiente cuando un proceso largo bloquea a los demás. Otro ejemplo es SJF (Shortest Job First), que prioriza los procesos más cortos para reducir el tiempo de espera promedio, aunque también puede ser ineficiente para los procesos largos y requiere estimar con precisión el tiempo de ejecución de cada proceso.

Metodología

El programa está escrito en Python e implementa y simula dos algoritmos de planificación de procesos: FIFO (First-In, First-Out) y SJF (Shortest Job First). La solución está diseñada para leer un archivo que contiene la información de los procesos y luego “ejecutarlos” utilizando cada uno de los algoritmos. En este caso, la “ejecución” es la impresión en pantalla del proceso, mostrando los tiempos de inicio y finalización de cada proceso. No se utilizaron librerías externas en este programa, solo funciones básicas del lenguaje Python, como la función `open()` para manejar archivos.

La solución está diseñada con una función independiente para cada algoritmo de planificación. La función principal `main()` gestiona la lectura de los datos de procesos desde un archivo de texto, añadiendo cada línea a una lista de procesos donde cada uno es una tupla con tres valores: el nombre del proceso, su duración y su prioridad y luego ejecuta cada algoritmo para observar y comparar los diferentes tiempos de inicio y finalización dependiendo del algoritmo utilizado.

`fifo_scheduling()` ejecuta los procesos en el orden en que aparecen en la lista sin ninguna reordenación, manteniendo un contador del tiempo actual `current_time` que se va actualizando según el tiempo de ejecución de cada proceso.

```
def fifo_scheduling(processes):
  """
  Ejecuta cada proceso sin ordenar la lista, mostrando el tiempo de inicio y finalización
  """
  current_time = 0

  for process in processes:
    start_time = current_time
    end_time = current_time + process[1]
    print(f"{process[0]}\t| Inicio: {start_time}s | Fin: {end_time}s")
    current_time = end_time
```

`sjf_scheduling()` recibe la misma lista de procesos, pero antes de ejecutarlos, los ordena en función de la duración del proceso. Luego, ejecuta los procesos en este orden, actualizando los tiempos de inicio y fin.

```
def sjf_scheduling(processes):
  """
  Ordena la lista de procesos según su duración, y los ejecuta mostrando el tiempo de inicio y finalización
  """
  current_time = 0
  # Ordena la lista de acuerdo a la duración del proceso
  processes_sorted = sorted(processes, key=lambda x: x[1])

  for process in processes_sorted:
    start_time = current_time
    end_time = current_time + process[1]
    print(f"{process[0]}\t| Inicio: {start_time}s | Fin: {end_time}s")
    current_time = end_time
```

Conclusión

El programa implementa una simulación básica de dos de los principales algoritmos de planificación de procesos en sistemas operativos. Está diseñado para leer un archivo de texto que contiene información sobre procesos (nombre, duración y prioridad) y simular su ejecución utilizando los algoritmos implementados. También imprime los tiempos de inicio y fin de cada proceso bajo cada algoritmo, lo que permite comparar los resultados de los diferentes enfoques.

Actualmente, las funciones de simulación y la lectura de datos desde el archivo funcionan correctamente, lo que permite comparar cómo se comportan los diferentes algoritmos. En futuras versiones sería útil permitir una entrada de archivo más flexible y validar mejor los datos. Además, se podrían añadir visualizaciones más detalladas, como gráficos de Gantt. Igual sería interesante calcular métricas de rendimiento, simular entornos multiprocesador y manejar tiempos de llegada distintos. Por último, agregar una interfaz de usuario y una simulación más detallada acercaría el programa a una aplicación más completa y útil.

Referencias

Wolf, G. (2016). *Planificación de procesos: Algoritmos de planificación*. Sistemas Operativos - Gunnar Wolf. https://sistop.gwolf.org/html/03_planificacion_de_procesos.html