**SCHEDULING STRATEGIES IN OPERATING SYSTEMS: IMPLEMENTATION AND BENCHMARKING OF KEY ALGORITHMS IN JAVA**

**ESCUELA DE INGENIERÍA CIENCIA Y TECNOLOGÍA**

**UNIVERSIDAD DEL ROSARIO**

Sebastian Dario Perez Pantoja, David Hernando Sánchez Lombana

e-mail: [sebastiand.perez@urosario.edu.co](mailto:sebastiand.perez@urosario.edu.co) , [davidh.sanchez@urosario.edu.co](mailto:davidh.sanchez@urosario.edu.co)

**RESUMEN**

El reporte se centra en la implementación de métodos fundamentales de scheduling dentro del simulador UR-OS una application basada en Java, que intenta incluir los elementos más importantes de un sistema computacional.

PALABRAS CLAVE: Process Scheduling, Java, Operating Systems, Computational Complexity.

1. **INTRODUCCIÓN**

La planificación de procesos, o process scheduling, es un componente esencial en los sistemas operativos modernos, ya que gestiona la asignación de la CPU entre múltiples procesos en ejecución. Su correcto funcionamiento impacta directamente en el rendimiento del sistema, optimizando el uso de recursos y mejorando la experiencia del usuario.

En este reporte, se presentan e implementan cinco algoritmos de planificación: First-Come, First-Served (FCFS), Shortest Job First No Apropiativo (SJF NP), Shortest Job First Apropiativo (SJF P), Round Robin (RR) y Multilevel Feedback Queue (MFQ). La implementación se lleva a cabo en la versión 0.0.3.8 del sistema UR OS, desarrollado por Pedro Wightman para el curso de Sistemas Operativos.

El objetivo de este trabajo es comprender cómo interactúan la CPU y el sistema operativo en la gestión de múltiples procesos, analizando la eficiencia y el impacto de cada algoritmo en el rendimiento del sistema.

1. **FUNDAMENTOS DEL SCHEDULING DE PROCESOS: DEFINICION, TIPOS DE SCHEDULING, METRICAS CLAVES.**

Los métodos de ordenación de procesos, se basan en algoritmos diseñados para optimizar el uso de la CPU y mejorar el rendimiento del sistema, medible por parte de una serie de métricas que mencionaremos proximamente.

Referente a los algoritmos que se probarán en para este reporte, tenemos:

* + First-Come, First-Served (FCFS): Asignación de procesos en orden de llegada sobre la cola de ejecución o Ready Queue.
    - Shortest Job First Non Preemptive (SJF NP): Asignación de procesos con prioridad sobre aquellos con menor tiempo de ejecución estimado. No permite interrupciones una vez el proceso haya llegado al CPU.
    - Shortest Job First Preemptive (SJF P): Asignación de procesos con prioridad sobre aquellos con menor tiempo de ejecución estimado. En caso de que aparezca en la ReadyQueue un proceso con menor tiempo de ejecución, se interrumpe la cpu y se re-evalua en busqueda del proceso con menor tiempo de ejecución estimado.
    - Round Robin (RR): Se define un tiempo fijo en terminos de ciclos bajo el nombre de quantum. Cada vez que termine el ciclo del quantum, se interrumpe la CPU y el proceso es enviado al final de la cola.
    - Multilevel Feedback Queue (MFQ): Se basa en la utilización de multiples colas, cada una con un nivel de prioridad diferente. Se priorizan los procesos con menor valor de este parametro.

Referente a las métricas clave para evaluar el rendimiento de estos procesos tenemos:

* + Waiting Time: Tiempo total que un proceso pasa en la cola de espera antes y entre sus ejecuciones.
  + Turnaround Time: Tiempo total desde que un proceso llega al sistema hasta que finaliza su ejecución.
  + Response Time: Tiempo desde que un proceso llega al sistema hasta que se ejecuta por primera vez.
  + Throughtput: Número de procesos completados por unidad de tiempo
  + CPU Utilización: Porcentaje de tiempo de trabajo de la CPU a lo largo del periodo de procesamiento.

1. **IMPLEMENTACIÓN: \*FCFS, SJN NP \*, SJN P, RR, MFQ, ETC.**

**A continuación se presenta la implementación en pseudo-codigo de los diferentes métodos de scheduling trabajados.**

**Casos Apropiativos:**

**FCFS:**

**SJN NP:**

Definir min\_BTR = Max\_sim\_cycles // Valor máximo inicial para comparar

Definir min\_BTR\_process = NULL // Variable para almacenar el proceso con menor Burst Time restante

Para cada proceso p en la lista de procesos:

temp ← BurstRemainingTime(p) // Obtener tiempo de ráfaga restante del proceso

Si temp < min\_BTR Entonces

min\_BTR ← temp

min\_BTR\_process ← p

Sino Si temp = min\_BTR Entonces

Aplicar criterio de desempate (Tie-Breaker)

Si min\_BTR\_process ≠ NULL Entonces

Eliminar min\_BTR\_process de la lista de procesos

Llamar a InterruptHandler(RQ\_TO\_CPU, min\_BTR\_process) // Asignar proceso a la CPU

Round Robin:

No Apropiativos:

SJF P:

MFQ:

1. **PRUEBAS Y BENCHMARKING.**
2. **DESAFÍOS Y CONSIDERACIONES.**
3. **CONCLUSIONES**
4. **REFERENCES¿?**