UNIDAD DE INGENIERÍA EN COMPUTACIÓN
Sede Central Cartago
Principios de Sistemas Operativos
Profesor: Kenneth Obando Rodríguez

TEC | Tecnológico de Costa Rica

Trabajo Corto 2 (%) Fecha de Entrega: ver calendario

Sincronización y Algoritmos de Scheduling

Descripción General

En este trabajo, cada estudiante debe implementar un **simulador de una línea de ensamblaje de productos**, usando **procesos o hilos** que se comuniquen entre sí mediante **mecanismos de comunicación entre procesos (IPC)**, como pipes o colas compartidas. Cada estación realizará una operación sobre el producto, y el orden en que los productos son procesados se regirá por un **algoritmo de planificación** implementado por el estudiante. El trabajo es en los grupos ya definidos.

Objetivos de Aprendizaje

- Aplicar mecanismos de **comunicación entre procesos o hilos** (IPC).
- Implementar técnicas de **sincronización** (semáforos, mutexes, exclusión mutua).
- Comprender y simular **algoritmos de planificación de procesos**.
- Analizar y reportar **métricas de rendimiento** como tiempo de espera, turnaround, etc.

Requisitos Técnicos

1. Estructura de la Línea de Ensamblaje

- La línea debe tener **tres estaciones de trabajo** (ejemplo: Corte → Ensamblaje → Empaque).
- Cada estación debe ser un **proceso o hilo independiente**.
- Los productos se deben pasar entre estaciones mediante **pipes o colas compartidas**.
- Las estaciones pueden tener **tiempos de procesamiento distintos**, definidos al inicio.

2. Generación de Productos

- Generar al menos **10 productos**, cada uno con un identificador único.
- Los productos deben tener un **tiempo de llegada simulado**.
- Deben almacenarse inicialmente en una **cola de entrada**.

3. Algoritmos de Scheduling

- Implementar al menos dos algoritmos:
  - **FCFS (First Come First Serve)**
  - **Round Robin**, con *quantum* de tiempo configurable
- El algoritmo debe determinar el **orden de procesamiento en cada estación**.

4. Sincronización

- Asegurar que solo **un producto se procese a la vez por estación**.
- Usar **semáforos o mutexes** para manejar la exclusión mutua.
- No debe haber condiciones de carrera ni interbloqueos.

5. Simulación de Tiempo

- Simular el tiempo de procesamiento con `sleep()` según la estación.
- El *quantum* debe forzar interrupciones en Round Robin.

6. Salida Esperada

- Para cada producto registrar:
  - Tiempo de llegada
  - Tiempo de entrada y salida por cada estación
  - Tiempo total (turnaround)
  - Tiempo de espera total
- Al final, mostrar un resumen con:
  - Promedio de espera
  - Promedio de turnaround
  - Orden final de procesamiento

Requerimientos de Entrega

- Código fuente comentado.
- Archivo `README.md` con instrucciones de ejecución.
- Captura o log de ejecución completa.
- Informe en PDF o texto con:
  - Descripción de la solución
  - Justificación técnica
  - Comparación entre algoritmos
- Video explicativo de no más de 10 minutos

Lenguajes Permitidos

- **C**: uso de `pipe`, `fork`, `pthread`, semáforos POSIX.
- **Rust**: uso de `std::sync::mpsc`, `crossbeam`, `Arc<Mutex<»`.
- **Go**: uso de `goroutines`, `channels`, `sync.Mutex`, `sync.WaitGroup`.

Rúbrica de Evaluación (100 puntos)

Criterio	Puntos
Implementación correcta de estaciones con IPC	25
Uso adecuado de semáforos/mutexes para sincronización	20
Implementación de ambos algoritmos de scheduling	20
Registro correcto de métricas y reporte de resultados	15
Calidad del código (estructura, comentarios, claridad)	10
Informe con análisis comparativo	10

Entrega

Debe presentar un archivo comprimido en el TecDigital antes de las 10:00pm del día de entrega. La actividad se puede realizar en parejas.

Si la entrega se realiza después de la hora de entrega, se le penalizará con 5 puntos porcentuales que se acumulan cada 24 horas. Por ejemplo si entrega a las 10:05pm su evaluación tendrá una nota base de 95%, si entrega después de las 10:05 p.m. del siguiente día, su nota base será 90%, y así sucesivamente.