SISTEMAS OPERATIVOS

ACTIVIDAD DE APRENDIZAJE 7
TAREA 4

Profesor:
VIOLETA DEL ROCIO BECERRA VELAZQUEZ

VERDUZCO ROSALES LUIS ENRIQUE
223992388
Ingenieria en Computación

05/10/25

CUCEI DIVTIC D04

ÍNDICE

TABLA IMAGENES

Introducción a los Hilos POSIX (Pthreads)

Los Hilos POSIX, comúnmente conocidos como Pthreads, son un modelo de ejecución estandarizado por la IEEE (POSIX 1003.1c) que permite a un programa controlar múltiples hilos de ejecución concurrentes. A diferencia de la creación de un proceso completo, que es costoso en términos de recursos del sistema, los hilos son considerados "procesos ligeros" porque comparten el mismo espacio de memoria, descriptores de archivos y otros recursos del proceso padre.

Dentro de un proceso POSIX convencional:

- Existe un hilo inicial que ejecuta la función main().
- Este hilo puede crear más hilos para ejecutar otras funciones dentro del espacio de direcciones del proceso .
- Todos los hilos de un proceso se encuentran al mismo nivel.
    - Esto significa que son “hermanos”, a diferencia de los proceso cuya relación es “padre-hijo”
- Los hilos de un proceso comparten las variables y recursos globales (archivos, manejadores de señales, etc.) del proceso.
- Además, cada uno tiene una copia privada de sus parámetros iniciales y de las variables locales de la función que ejecuta (almacenados en su pila particular)

El estándar POSIX define, entre otras, funciones para:

- Creación de hilos.
- Creación/destrucción de atributos de creación de hilos .
- Terminación/espera a la terminación de un hilo.
- Identificación de hilos.

Beneficios clave de usar Pthreads:

- **Eficiencia:** La creación de hilos y el cambio de contexto entre ellos es mucho más rápido que con los procesos.
- **Comunicación simplificada:** Al compartir memoria, los hilos pueden intercambiar datos directamente sin necesidad de mecanismos complejos de comunicación entre procesos.
- **Paralelismo:** En sistemas con múltiples procesadores o núcleos, los hilos pueden ejecutarse simultáneamente en diferentes núcleos, mejorando significativamente el rendimiento.

La biblioteca de Pthreads proporciona una API (Interfaz de Programación de Aplicaciones) en C con funciones como **`pthread_create()`** para crear un nuevo hilo, **`pthread_join()`** para esperar a que un hilo termine y **`pthread_exit()`** para terminar el hilo actual.

1. Algoritmos de Planificación Apropiativos

Los algoritmos de planificación apropiativos (preemptive scheduling) se basan en la política de que el sistema operativo puede interrumpir forzosamente un proceso que se está ejecutando en la CPU y asignarla a otro. Esta interrupción, conocida como "apropiación" o "expulsión" (preemption), es decidida por el planificador de corto plazo.

Estos métodos nacieron por la necesidad de poder ordenar los procesos para ganar eficiencia a la hora de tratar con ellos, es decir, son los encargados de ordenar y dirigir los procesos para asegurar que ninguno de ellos monopolice el uso de la CPU.

La decisión de apropiarse de la CPU puede ocurrir en varios momentos:

- Cuando un proceso nuevo con mayor prioridad llega a la cola de listos.
- Cuando un proceso en ejecución excede su intervalo de tiempo asignado (quantum).
- Cuando un proceso que estaba bloqueado (esperando una operación de E/S, por ejemplo) se desbloquea y tiene una prioridad mayor que el proceso en ejecución.

Este tipo de planificación es fundamental para los sistemas operativos multitarea y de tiempo compartido, ya que garantiza que ningún proceso monopolice la CPU y mejora el tiempo de respuesta para los usuarios interactivos.

2. Algoritmo de Planificación por Prioridades

El **algoritmo de planificación por prioridades** asigna un valor de prioridad a cada proceso. El planificador seleccionará siempre el proceso con la **mayor prioridad** de la cola de procesos listos para asignarle la CPU.

Clasificación:

Este algoritmo se clasifica principalmente en dos variantes:

- **Apropiativo (Preemptive):** Si un nuevo proceso llega a la cola de listos con una prioridad más alta que la del proceso que se está ejecutando actualmente, el sistema operativo interrumpe al proceso en ejecución y le asigna la CPU al nuevo proceso de mayor prioridad.

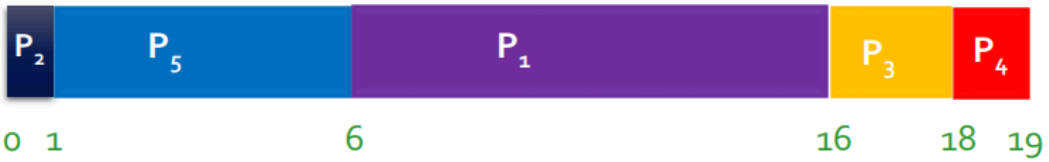
- **No Apropiativo (Non-preemptive) o Cooperativo:** Cuando un proceso obtiene la CPU, la mantiene hasta que la libera voluntariamente, ya sea porque ha terminado su ejecución o porque pasa a un estado de bloqueo (por ejemplo, para realizar una operación de E/S). Un proceso nuevo de mayor prioridad que llegue a la cola de listos deberá esperar a que el proceso actual libere la CPU.

Un problema común en este algoritmo es la **inanición** (starvation), donde los procesos de baja prioridad pueden quedarse esperando indefinidamente si siempre hay procesos de mayor prioridad listos para ejecutarse. Una solución a este problema es el **envejecimiento** (aging), una técnica que consiste en aumentar gradualmente la prioridad de los procesos que han estado esperando en la cola durante mucho tiempo.

Ejemplo

Proceso	Ráfaga	Prioridad
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

- La carta de Gantt



- Tiempo de espera promedio= 8.2 msec

3. Algoritmo de Planificación de Colas Múltiples

Este algoritmo, también conocido como **colas multinivel**, divide la cola de procesos listos en varias colas separadas. A cada cola se le puede asignar una prioridad diferente y, a menudo, cada una utiliza su propio algoritmo de planificación.

Por ejemplo, se podría tener una cola para **procesos de sistema** (con la más alta prioridad y usando un algoritmo como FCFS), otra para **procesos interactivos** (con prioridad media y usando Round Robin para un buen tiempo de respuesta) y una tercera para **procesos por lotes** (batch) (con la más baja prioridad y usando FCFS).

El planificador primero ejecuta todos los procesos de la cola de mayor prioridad. Solo si esta cola está vacía, pasará a ejecutar los procesos de la cola de prioridad inmediatamente inferior, y así sucesivamente. Esto puede ser **apropiativo**; si un proceso llega a una cola de mayor prioridad mientras se ejecuta uno de una cola inferior, el de la cola inferior será interrumpido.

Una variante más compleja es la de **colas multinivel retroalimentadas**, que permite a los procesos moverse entre las colas según su comportamiento (por ejemplo, un proceso que consume mucho tiempo de CPU puede ser degradado a una cola de menor prioridad).

4. Diferencia entre "Bloqueado y Suspendido" y "Listo y Suspendido"

Para entender esta diferencia, primero debemos definir los estados base:

- **Listo (Ready):** El proceso está en la memoria principal y tiene todo lo necesario para ejecutarse, solo está esperando que se le asigne la CPU.
- **Bloqueado (Blocked):** El proceso no puede ejecutarse porque está esperando que ocurra un evento externo, como la finalización de una operación de entrada/salida.
- **Suspendido (Suspended):** El proceso ha sido sacado de la memoria principal y almacenado temporalmente en el disco (swapping). Esto se hace para liberar memoria RAM para otros procesos.

La diferencia radica en la razón por la que el proceso no puede ejecutarse cuando está suspendido:

- **Listo y Suspendido (Ready, Suspended):** El proceso fue suspendido (movido a disco) mientras estaba en estado "Listo". Está preparado para ejecutarse tan pronto como sea cargado de nuevo en la memoria principal y el planificador le asigne la CPU. No está esperando ningún evento externo.
- **Bloqueado y Suspendido (Blocked, Suspended):** El proceso fue suspendido mientras estaba en estado "Bloqueado". Por lo tanto, el proceso tiene dos condiciones que le impiden ejecutarse: está esperando que ocurra un evento (como una E/S) **y además** no está en la memoria principal. Para poder ejecutarse, primero debe completarse el evento por el que esperaba y luego debe ser cargado de nuevo en la memoria.

5. Tiempo de Respuesta

El **tiempo de respuesta** (Response Time) es el intervalo de tiempo que transcurre desde que una solicitud o un proceso es ingresado al sistema hasta que se produce la **primera respuesta**. No se refiere al tiempo que tarda en completarse toda la tarea, sino al tiempo que tarda en comenzar a ser atendido. Es una métrica de

rendimiento crucial en sistemas interactivos, donde es importante que el usuario perciba que el sistema está reaccionando rápidamente a sus comandos.\

6. Algoritmo de Planificación Apropiativo RR (Round Robin)

El algoritmo **Round Robin (RR)** es un algoritmo de planificación apropiativo diseñado especialmente para sistemas de tiempo compartido. Su funcionamiento es el siguiente:

1. Los procesos listos se mantienen en una cola FIFO (First-In, First-Out).
2. A cada proceso se le asigna un pequeño intervalo de tiempo de CPU, llamado **quantum** o **time slice**.
3. El planificador toma el primer proceso de la cola y le permite ejecutarse durante un quantum.
4. Si el proceso termina antes de que su quantum expire, libera la CPU voluntariamente.
5. Si el proceso sigue en ejecución cuando el quantum se agota, el temporizador del sistema genera una interrupción, el sistema operativo interviene, se apropia de la CPU, mueve al proceso al final de la cola de listos y asigna la CPU al siguiente proceso en la cola.

La eficacia del RR depende en gran medida del tamaño del quantum. Un quantum muy grande hace que el algoritmo se comporte como FCFS, mientras que un quantum muy pequeño aumenta la sobrecarga por los constantes cambios de contexto.

7. ¿Qué es el Quantum?

El **quantum**, también conocido como **rodaja de tiempo** (time slice), es la **unidad de tiempo máxima** que se le asigna a un proceso para que utilice la CPU de forma ininterrumpida dentro de un algoritmo de planificación apropiativo como Round Robin.

Es un parámetro fundamental del sistema operativo. Una vez que el quantum de un proceso expira, este es expulsado de la CPU para dar paso a otro proceso, garantizando así que todos los procesos reciban una porción equitativa del tiempo del procesador y evitando que uno solo monopolice el sistema.

Conclusión

Al realizar esta investigación sobre los hilos POSIX y los algoritmos de planificación, me di cuenta de que hay mucha información sobre la organización y reglas dentro de la computadora que normalmente no vemos. Ahora entiendo que el sistema operativo actúa como un gestor estricto y eficiente.

El algoritmo que más me llamó la atención fue el Round Robin (RR), porque es el más justo. Usa un "quantum" para darle a cada proceso un cierto tiempo, asegurándose de que todos avancen y que el sistema se sienta más rápido y responsivo para nosotros. Entender que los procesos pueden estar en diferentes estados, como listo, bloqueado o incluso suspendido, me ayudó a visualizar cómo se gestiona todo este trabajo.

Bibliografía

Atlantic International University. (s.f.). Sistemas Operativos Avanzados. Sesión 2. Recuperado de: https://cursos.aiu.edu/SISTEMAS%20OPERATIVOS%20AVANZADOS/Sesi%C3%B3n%202/PDF/SESION%202%20SISTEMAS%20OPERATIVOS%20AVANZADOS.pdf

Cátedra, R. (2023). M2 Resumen Sistemas Operativos. [Apuntes de curso]. Studocu. Recuperado de: https://www.studocu.com/es-ar/document/universidad-siglo-21/sistemas-operativos/m2-resumen-sistemas-operativos/64408007

Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur. (2020). Sistemas Operativos - Módulo 7: Planificación de CPU. Recuperado de: https://cs.uns.edu.ar/~so/data/apuntes/SO-2020-mod%2007.pdf

Departamento de Informática, Universidad de Jaén. (s.f.). 6.- Algoritmos de Planificación II. Recuperado de: https://lsi.vc.ehu.eus/pablogn/docencia/manuales/SO/TemasSOuJaen/PLANIFICACIONDEPROCESOS/6AlgoritmosdePlanificacionII.htm

Departamento de Informática, Universidad de Jaén. (s.f.). *Algoritmo de planificación SJF*. Recuperado de: https://lsi.vc.ehu.eus/pablogn/docencia/manuales/SO/TemasSOuJaen/PLANIFICACIONDEPROCESOS/PlanificadorProcesos/alg_planif_SJF.html

Díaz, F. (2004). Administración y Sistemas Operativos Práctica #1. Universidad de Valladolid. Recuperado de: https://www.infor.uva.es/~fdiaz/aso/2004_05/doc/ASO_PR01_20041122.pdf

Díaz, F. (2006). Sistemas Operativos Tema 5: Planificación de procesos. Universidad de Valladolid. Recuperado de: https://www.infor.uva.es/~fdiaz/so/doc/SO_TE05_20061210.pdf

Moral, J. (2017, 14 de febrero). Planificación de procesos en los sistemas operativos. Blog de José Moral. Recuperado de: http://jmoral.es/blog/planificacion-procesos