Práctica 4 - Planificación de procesos

Page • 1 enlace entrante • Tag

Proceso

- **Programa** → Porción de código, almacenado en memoria secundaria, que al ser ejecutado en la CPU provee una funcionalidad específica. Su ciclo de vida comprende desde que se lo edita hasta que se borra (del disco)
- **Proceso** → *Programa* en ejecución. Cuenta con Program Counter, y su ciclo de vida comprende desde que se lo ejecuta (se crea la PCB) hasta que termina
    - **CPU Bound**
    - **I/O Bound** → Idealmente, deberían ser procesos con prioridad alta y quantum chico
- **PCB** → Estructura de datos asociada al *proceso*. Provee información sobre el estado actual de este y su localización en memoria. Es lo primero que se crea cuando se crea un proceso, y lo último que se borra cuando termina. Entre sus campos se puede encontrar:
    - **Identificadores** → PID, PPID, ...
    - **Contador de Programa (PC)**
    - **Estado del proceso (parte de planificación)** → Nuevo, listo, en ejecución, bloqueado, suspendido, terminado
    - **Información de su ejecución (prioridad)**

Planificación

La responsabilidad principal del Sistema Operativo es controlar los procesos, y ejecutar la mayor cantidad posible

Tiempos de los Procesos

- **Tiempo de Retorno** → Intervalo de tiempo que comprende desde que se ejecuta un proceso hasta su termino
- **Tiempo de Espera** → Tiempo en que el proceso se encuentra en el sistema sin contar el tiempo de utilización de la CPU

- **Quantum** → Intervalo de tiempo que le es asignado a cada proceso para ejecutarse (y evitar acaparar la CPU). Terminado este intervalo regresa a la Cola de Listos
- **Inanición** → Situación de imposibilidad de ejecución de un proceso

Planificadores

Módulos del Kernel que realizan distintas tareas asociadas a la planificación de procesos

- **Long-term scheduler** → Admite nuevos procesos a memoria (pasa de Nuevo a Listo)
- **Medium-term scheduler** → Realiza el swapping entre el disco y la memoria cuando el SO lo determina (saca a los procesos del estado Suspendido)
- **Short-term scheduler** → Elige entre los procesos que estan listos en memoria para darles CPU
- **Dispatcher** → Realiza la tarea de *context-switch*

Algoritmos de Planificación

Deben maximizar el uso de la CPU. Pueden ser **apropiativos o no**, lo que significa que el algoritmo puede expulsar a un proceso para apropiarse de un recurso y asignárselo a otro que tenga mayor prioridad

- **First Come First Served (FCFS)** → Cuando hay que elegir un proceso para ejecutar, se selecciona el mas viejo, no apropiativo
- **Shortest Job First (SJB)** → Selecciona el proceso con la ráfaga mas corta, puede provocar inanición, no apropiativo
    - **Shortest Remaining Time First (SRTF)** → Versión apropiativa
- **Round Robin (RR)** → Cada proceso se ejecuta un *quantum*, al terminar se lo coloca al final de la Cola de Listos
    - **Con Timer Variable (TV)** → El proceso puede terminar antes su quantum
    - **Con Timer Fijo (TF)** → El proceso si o si se debe ejecutar las unidades asignadas. Su implementación depende del timer
- **Prioridades** → Se selecciona el proceso con mayor prioridad, existen varias Colas de Listos por prioridad
    - **Aging** → Mecanismo que le permite a un proceso cambiar su prioridad durante su ciclo de vida
- **Virtual Round Robin (VRR)** → Los procesos que regresan de una E/S se colocan en una cola auxiliar, esta tiene prioridad sobre la Cola de Listos.

Cuando se elije un proceso de la cola auxiliar se le otorgan tnatas unidades de tiempo como le hayan faltado en su ráfaga de CPU anterior

Criterios de desempate:

- **Orden de llegada**
- **PID de los procesos**

Colas Multinivel

Se divide a la Cola de Listos en varias colas, y los procesos se colocan en las colas según una clasificación que realice el sistema operativo. Cada cola posee su propio algoritmo de planificación (**planificador horizontal**) y, a su vez, existe un algoritmo que planifica las colas (**planificador vertical**)

- **Retroalimentación** → Un proceso puede cambiar de cola
- Beneficia a procesos *CPU Bound*
- Puede ocurrir inanición si con los procesos ligados a E/S si siempre llegan procesos ligados a CPU

Planificación con múltiples procesadores

- **Planificación temporal** → Qué proceso y durante cuanto
- **Planificación espacial** → En qué procesador ejecutar
    - **Huella** → Estado que el proceso va dejando en la cache de un procesador
    - **Afinidad** → Preferencia de un proceso para ejecutar en un procesador
- Asignación de procesos estática o dinámica (balanceo de carga)
- Política de tiempo compartido o espacio compartido (grupo, particiones)
    - **Tiempo Compartido** → Se puede cosiderar una cola global o una cola local a cada procesador
- Por homogeneidad
    - **Procesadores homogéneos** → Todas las CPUs son iguales. No existen ventajas físicas sobre el resto
    - **Procesadores heterogéneos** → Cada procesador tiene su propia cola, su propio clock y su propio algoritmo de planificación
- Por acoplamiento
    - **Procesadores débilmente acoplados** → Cada CPU tiene su propia memoria principal y canales

- **Procesadores fuertemente acoplados** → Comparten memoria y canales
- P**rocesadores especializados** → Uno o más procesadores principales de uso general y uno o más procesadores de uso específico

Cosas que aparecen en la práctica

Programas de Linux

- top, htop → Muestran procesos de forma dinámica y en tiempo real
- pgrep/pkill → Buscar o avisar procesos en base a un nombre y otros atributos
- renice → Altera la prioridad de los procesos que están ejecutándose
- xkill → Permite cerrar ventanas X
- atop → Registra el uso histórico de los recursos para su análisis posterior

Pipes

Redireccionan la sálida estandar de un proceso para utilizarla como la entrada estándar de otro. Para crear un Pipe en C se emplea la función Pipe(), que abre dos descriptores de fichero y almacena su valor en los dos enteros que contiene un arreglo de descriptores de fichero. Para crearlo es necesario un vector de dos enteros, que se utiliza para devolver, si todo salió bien, dos nuevos descriptores de ficheros para ser utilizados en el Pipe y comenzar la comunicación.

Así, por ejemplo, podemos tener la siguiente formula:

$$S_{n+1} = \frac{1}{n} T_n + \frac{n-1}{n} S_n$$

Donde:

T_i = duración de la ráfaga de CPU i-ésima del proceso.

S_i = valor estimado para el i-ésimo caso

S_i = valor estimado para la primer ráfaga de CPU. No es calculado.

Es posible reescribir la formula permitiendo darle un peso mayor a los casos mas recientes y menor a casos viejos (o viceversa). Se plantea la siguiente formula:

$$S_{n+1} = \alpha T_n + (1-\alpha) S_n \qquad (2)$$

Con $0 \langle \alpha \langle 1$.