INSTITUTO POLITÉCNICO NACIONAL ESCUELA SUPERIOR DE CÓMPUTO

INGENIERÍA EN SISTEMAS COMPUTACIONALES

SISTEMAS OPERATIVOS

David Díaz Araujo

Práctica 5:
Gestión de la planificación de procesos/hilos

Alumnos:

Arcos Hermida Aldo Alejandro (N° lista 5)

Chavez Becerra Bianca Cristina (N° lista 9)

Islas Osorio Enrique (N° lista 20)

Juárez Cabrera Jessica (N° lista 21)

Palmerin García Diego (N° lista 28)

Grupo: 4CM1

Objetivo

Comprender la forma en que se planifican los procesos dentro de los sistemas LINUX/UNIX.

Descripción

Como ya hemos visto previamente, un proceso es un programa en ejecución. Por su parte, un hilo es una secuencia de código en ejecución dentro del contexto de un proceso.

Tipos de procesos

Para poder empezar a hablar de la gestión de la planificación de procesos en Linux, debemos tener en cuenta los tipos de procesos que existen y cómo se clasifican.

El primer tipo son los **procesos de tiempo real**, cuya característica es que su tiempo de respuesta tiene que estar acotado y así no exceder un umbral o un punto máximo.

Pasemos con el segundo tipo, a esto se les conoce como **procesos de segundo plano**. dentro de esta categoría tenemos a los compiladores, procesos de cálculo, entre otros. dichos procesos no tienen interacción con el usuario, y su tiempo de respuesta no suele ser crítico.

Finalmente, tenemos a los procesos interactivos. regularmente no suelen usar mucho tiempo de cómputo, sin embargo, en dado caso de que lo requieran su respuesta debe ser rápida, debido a que hay una interacción directa con el usuario, y se le debe dar una experiencia o sensación de rapidez.

Políticas para la planificación

Dentro de Linux, existen 3 políticas de planificación para los dos tipos de procesos existentes: los procesos normales de UNIX y los procesos de tiempo real.

Empezando por los presos normales de UNIX y los interactivos, tenso planificación con respecto a la política SCHED_NORMAL, basada en Round Robin, con la diferencia de que cada proceso tiene un quantum diferente, que en la que se intentará potenciar el tiempo de respuesta de los procesos interactivos.

En cuanto a los procesos en tiempo real, son los que tienen mayor prioridad en el sistema. dentro de estos existen dos variantes: la primera es SCHED_FIFO, en la que los procesos no pierden el procesador hasta ser bloqueados por sí mismos o hasta activar un proceso de tiempo real con mayor prioridad. La segunda variante es SCHED_RR, Se comportan igual que en el caso anterior, pero al momento de terminar su Quantum ser en el procesador a otros procesos con la misma prioridad.

Es importante destacar que los procesos de tiempo real tienen una prioridad fija.

Algoritmos de planificación

Los algoritmos de planificación coordinan diferentes formas de realizar la ejecución de procesos, de una manera organizada y eficiente para el procesador, y que este no sea monopolizado por ningún proceso. Para ello el sistema operativo decide qué proceso ejecutar primero y luego sigue un algoritmo de planificación.

Los algoritmos de planificación de procesos son utilizados para maximizar la utilización del procesador, así como la productividad. Pero también se encargan de minimizar el tiempo de espera, el tiempo de retorno y de respuesta.

- FCFS
  Se conoce también como. El procesador ejecuta cada proceso hasta que termina, en función del orden de llegada. La cantidad de tiempo de espera de cada proceso depende del número de procesos que se encuentren en la cola en el momento de su petición de ejecución y del tiempo que cada uno de ellos tenga en uso al procesador, y es independiente de las necesidades del propio proceso.

Ejemplo práctico:

PROCESO	LLEGADA	DURACIÓN
P1	0	5
P2	4	2
P3	1	7
P4	2	1
P5	3	8

Tiempos de espera:		Tiempos de retorno:
P1	(0-0)=0	P1 = 5
P2	(21-4)=17	P2 = 23
P3	(5-1)=4	P3 = 12
P4	(12-2)=10	P4 = 13
P5	(13-3)=10	P5 = 21

Tiempo medio de espera: (0 + 17 + 4 + 10 + 10) / 5 = 8,2
Tiempo medio de retorno: (5 + 23 + 12 + 13 + 21)/5 = 14,8

- SJF
  En este algoritmo , da bastante prioridad a los procesos más cortos a la hora de ejecución y los coloca en la cola. Selecciona al proceso con el próximo tiempo ejecución más corto y lo ejecuta hasta que finaliza el proceso. Si hay dos procesos cuyas ráfagas de la CPU tiene la misma duración, se emplea el algoritmo FCFS o FIFO para romper el empate.

Ejemplo práctico:

PROCESO	LLEGADA	DURACIÓN
P1	0	5
P2	4	2
P3	1	7
P4	2	1
P5	3	8

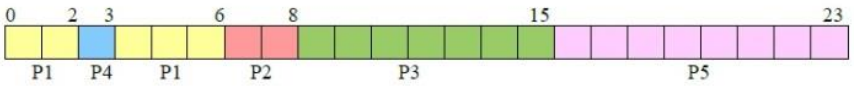
Tiempos de espera:		Tiempos de retorno:
P1	(0-0)=0	P1 = 5
P2	(6-4)=2	P2 = 8
P3	(8-1)=7	P3 = 15
P4	(5-2)=3	P4 = 6
P5	(15-3)=12	P5 = 23

Tiempo medio de espera: (0 + 2 + 7 + 3 + 12) / 5 = 4,8
Tiempo medio de retorno: (5 + 8 + 15 + 6 + 23) / 5 = 11,4

- SRTF
  Es similar al SJF con la diferencia de que si un nuevo proceso pasa a listo se activa el dispatcher para ver si es más corto que lo que queda por ejecutar del proceso en ejecución. Si es así, el proceso en ejecución pasa a listo y su tiempo de estimación se decremento con el tiempo que ha estado ejecutándose. Los procesos llegan a la cola y solicitan un intervalo de CPU, si dicho intervalo es inferior al que le falta al proceso en ejecución para abandonar la CPU, el nuevo proceso pasa a la CPU y el que se ejecutaba a la cola de preparados.

Ejemplo práctico:

PROCESO	LLEGADA	DURACIÓN
P1	0	5
P2	4	2
P3	1	7
P4	2	1
P5	3	8



Tiempos de espera:		Tiempos de retorno:
P1	(0-0)=0	P1 = 6
P2	(6-4)=2	P2 = 8
P3	(8-1)=7	P3 = 15
P4	(2-2)=0	P4 = 3
P5	(15-3)=12	P5 = 23

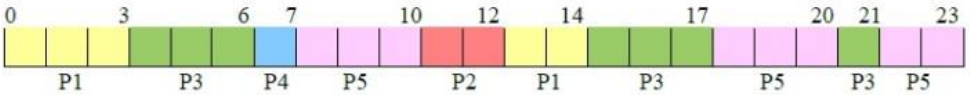
Tiempo medio de espera: (0 + 2 + 7 + 0 + 12) / 5 = 4,2
Tiempo medio de retorno: (6 + 8 + 15 + 3 + 23) / 5 = 11

- ROUND ROBIN
  Cada proceso tiene asignado un intervalo de tiempo de ejecución, llamado cuantum o cuanto. Si el proceso agota su cuantum de tiempo, se elige a otro proceso para ocupar la CPU. Si el proceso se bloquea o termina antes de agotar su cuantum también se alterna el uso de la CPU. La cola de procesos se estructura como una cola circular. La organización de la cola es FIFO.

Ejemplo práctico:

PROCESO	LLEGADA	DURACIÓN
P1	0	5
P2	4	2
P3	1	7
P4	2	1
P5	3	8

Q=3



Tiempos de espera:		Tiempos de retorno:
P1	(0-0)=0	P1 = 14
P2	(10-4)=6	P2 = 12
P3	(3-1)=2	P3 = 21
P4	(6-2)=4	P4 = 7
P5	(7-3)=4	P5 = 23

Tiempo medio de espera: (0 + 6 + 2 + 4 + 4) / 5 = 3,2
Tiempo medio de retorno: (14 + 12 + 21 + 7 + 23) / 5 = 15,4

El algoritmo de planificación en Linux

Hasta la versión 2.6, el algoritmo de planificación era relativamente sencillo y se basaba en elegir el proceso al que le quedaba mayor cantidad de quantum por consumir, premiándose así los procesos interactivos.

El planificador actual es mucho más sofisticado en aras a la escalabilidad con el número de procesos -se elige el proceso a ejecutar en tiempo constante, independiente del número de procesos en la cola de preparados- y de procesadores -cada CPU tiene su propia cola de procesos preparados. Sus principales características, para los procesos normales, son:

- Cada proceso tiene una prioridad estática entre 100, la máxima, y 139, la mínima. Esta prioridad se cambia con las llamadas al sistema nice() y setpriority(). El "valor de nice" máximo es +19 y corresponde a la prioridad estática 139, y el mínimo es -20 y corresponde a la prioridad estática 100.

- La prioridad estática P determina el quantum del proceso. Si P<120, entonces el quantum es (140-P)x20, sino es (140-P)x5. El cambio de tratamiento está en 120, correspondiente a un "valor de nice" 0.
- Cada proceso tiene también asociado un bonus de interactividad de 0 a 10 que modifica su prioridad estática para obtener una prioridad dinámica que es la que finalmente usa el planificador para elegir el proceso a ejecutar. El bonus se resta a la prioridad y al resultado se le suma 5, de forma que un bonus de cero hace 5 puntos menos prioritario el proceso y un bonus de 10 lo hace 5 puntos más prioritario.

Procesos activos y expirados

Para promover los procesos interactivos evitando al mismo tiempo la posibilidad de que los procesos menos prioritarios sufran de inanición, el planificador gestiona dos colas distintas de procesos preparados:

- La cola de procesos activos, que son aquéllos cuyo quantum no ha expirado y por tanto serían elegibles por el planificador.
- La cola de procesos cuyo quantum ha expirado. Mientras un proceso está en esta cola, el planificador no lo elige. Cuando se vacía la cola de procesos activos, todos los procesos saldrán de la cola de procesos expirados y se "reactivarán", con su quantum relleno de nuevo.
- Cuando un proceso interactivo acaba su quantum se "reactiva" automáticamente sin pasar por la segunda cola, a no ser que en ella haya algún proceso esperando un tiempo muy largo o que tenga más prioridad que él. De esta forma se evita que los procesos en la cola de expirados puedan permanecer ahí indefinidamente.

Conclusiones

Después de la realización de esta práctica, pudimos comprender a fondo los tipos de procesos que existen, su clasificación y cómo todo esto influye al momento de hacer uso de los diferentes algoritmos para la planificación y gestión de los procesos.

También conocimos la variedad de algoritmos que son utilizados para esta tarea, que junto a los ejemplos se pudo conocer y entender mejor la manera en que operan.

Es importante destacar la manera en la que la planificación de procesos ha evolucionado y se ha hecho cada vez más sofisticada, con el objetivo de brindar una mayor optimización de recursos y de tiempo, así como una mejor experiencia al usuario.

Referencias

- Carreres, J. (2015, 22 noviembre). *Los algoritmos de planificación de procesos*. Los algoritmos de planificación de procesos. Recuperado 25 de abril de 2022, de https://josecarreres.wordpress.com/2015/10/05/planificacion-y-procesos-en-windows-y-linux/

- F. (2016, 1 octubre). *Planificación de Procesos e Hilos*. Sistemas Operativos #1. Recuperado 25 de abril de 2022, de https://sisoperativoutp1995.wordpress.com/2016/10/01/planificacion-de-procesos-e-hilos/
- Saez, S. P. C. J. Y. (s. f.). *DSO. Tema 7. GestiÃ³n de procesos. PlanificaciÃ³n*. Tema 7. Gestión de Procesos. Planificación. Recuperado 25 de abril de 2022, de http://sop.upv.es/gii-dso/es/t7-1-gestion-de-procesos/gen-t7-1-gestion-de-procesos.html#:%7E:text=En%20Linux%2C%20la%20planificaci%C3%B3n%20de,de%20ejecuci%C3%B3n%2C%20como%20ya%20veremos.
- Silberschatz, A. G. (2005). Fundamentos de sistemas operativos. Séptima edición.Madrid: McGraw-Hill.