Sistemas Operativos 72.11

Scheduling

ITBA
Instituto Tecnológico de Buenos Aires

Scheduling

- Usualmente tenemos múltiples procesos peleando por el CPU
- ¿Qué pasa si tenemos 2 o más procesos en estado ready y al menos 1 CPU disponible?
- Proceso vs thread

available, a choice has to be made which process to run next. The part of the operating system that makes the choice is called the **scheduler**, and the algorithm it uses is called the **scheduling algorithm**. These topics form the subject matter of the following sections.

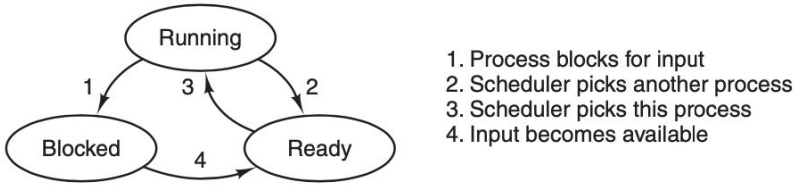


Figure 2-2. A process can be in running, blocked, or ready state. Transitions between these states are as shown.

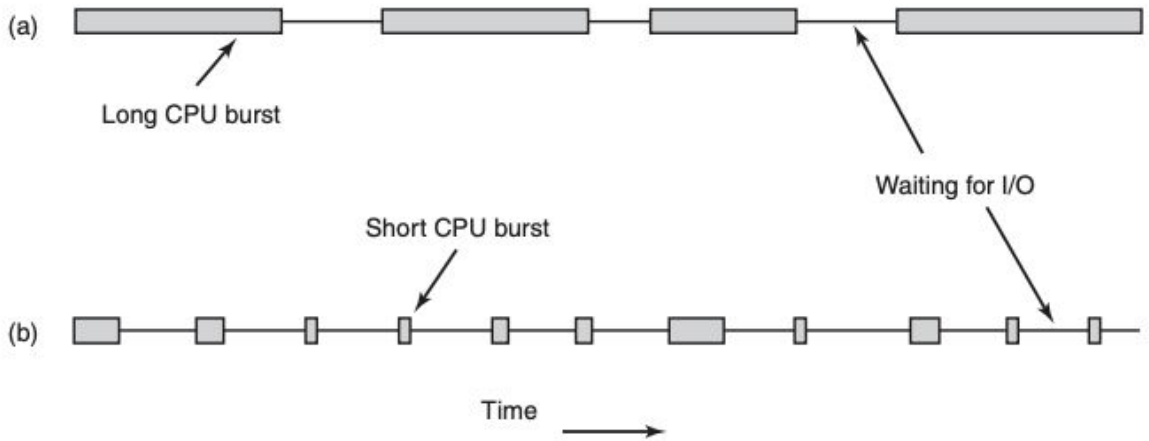
Scheduling
Observaciones

- Antiguamente se podía atender de a 1 tarea a la vez -> ¿cómo era el algoritmo de scheduling?
- Con el advenimiento de la multiprogramación se pueden tener muchos usuarios esperando su uso, con interfaces interactivas y poco poder de cómputo -> tiempo de CPU escaso
- Con las PCs
    - En general hay 1 tarea principal
    - El tiempo de CPU no es un recurso escaso
- ¿Recursos disponibles en sistemas portables?
- El switch de procesos es costoso (Estado del CPU, mapeo de memoria, caches)

Scheduling

Comportamiento de un proceso

- Usualmente los procesos alternan entre uso de CPU e I/O
- Nomenclatura: I/O => bloquearse esperando por un evento. Por ejemplo, escribir la memoria de video para refrescar la pantalla se considera uso de CPU.
- CPU-bound vs I/O-bound (la clave es la longitud de la ráfaga de CPU, no de I/O)
- Con procesadores más rápidos los procesos se vuelven más I/O-bound



Scheduling
Cuándo

- Creación de un proceso: ¿padre o hijo?
- Finalización de un proceso: ¿cuál de todos los listos?
  - ¿Y si no hay ninguno listo? -> les va a pasar -> idle process
- Bloqueo de proceso por I/O: ¿cuál de todos los listos?
  - Motivo de bloqueo: P1 alta prioridad bloqueado hasta que P2 libera un semáforo
- Interrupción:
  - Dispositivo que completó su trabajo (disco): ¿el que se bloqueó por este evento?
  - Timer: 50/60Hz -> en cada interrupción o cada n interrupciones

Scheduling
Clasificación

- Dependiendo de qué se haga con las interrupciones del timer

- Non-preemptive: Elige un proceso y corre hasta que se bloquea o hasta que libera el CPU voluntariamente
- Preemptive: Elige un proceso y corre hasta que se vence un plazo establecido (time slice / quantum), incluso si está en estado ready. Es necesaria la interrupción del timer para darle el control al SO (scheduler)

preempt

verb [T]

US /priˈempt/

to prevent something from happening by taking action first:

- *State laws preempted local governments from restricting newspaper displays.*

If a broadcast is preempted, it is replaced by another, usually more important broadcast:

- *One station preempted its Friday night schedule to televise the high school playoffs.*

https://dictionary.cambridge.org/

Scheduling
Categorías

- Las decisiones de scheduling para diferentes tipos de sistemas operativos pueden variar significativamente

- Batch
  - No hay interacción con el usuario
  - Non-preemptive o preemptive con un quantum largo
- Interactivo
  - Preemptive ¿por qué?
  - Uso general, programas arbitrarios
- Real time
  - Sorpresivamente, a veces no es necesario un scheduler preemptive
  - Uso específico, programas específicos.

Scheduling
Objetivos

All systems
- Fairness - giving each process a fair share of the CPU
- Policy enforcement - seeing that stated policy is carried out
- Balance - keeping all parts of the system busy

Batch systems
- Throughput - maximize jobs per hour
- Turnaround time - minimize time between submission and termination
- CPU utilization - keep the CPU busy all the time

Interactive systems
- Response time - respond to requests quickly
- Proportionality - meet users' expectations

Real-time systems
- Meeting deadlines - avoid losing data
- Predictability - avoid quality degradation in multimedia systems

Figure 2-40. Some goals of the scheduling algorithm under different circumstances.

- Procesos similares -> similar servicio
- Política: procesos de seguridad corren tan pronto como están disponibles
- Métricas
- ¿Maximizar throughput minimiza el turnaround time?
- CPU cerca del 100%
- Subir un video de 500MB a Drive en 60 segundos vs cerrar el navegador en el mismo tiempo

Scheduling

Batch: First-Come First-Served

- Cola tradicional
- Non-preemptive
- Bloqueados van al final
- Simple y fácil de programar

- Caso patológico: Tengo un proceso CPU-bound que corre por 1s y se bloquea en disco y varios procesos I/O-bound que realizan 1000 accesos a disco cada uno.

{CPU I/O I/O I/O I/O} x 1000

¿Alternativa con preemption?

Scheduling
Batch: Shortest Job First

- Asume que los tiempos de ejecución son conocidos
- Non-preemptive

Figure 2-41. An example of shortest-job-first scheduling. (a) Running four jobs in the original order. (b) Running them in shortest job first order.

- ¿Turnaround time?
  - (a): (A + A+B + A+B+C + A+B+C+D) / 4 = (4A + 3B + 2C +D) / 4 = 14s
  - (b): (B + B+C + B+C+D + B+C+D+A) / 4 = (4B + 3C + 2D +A) / 4 = 11s

- Óptimo si todos los procesos están disponibles simultáneamente -> Tiempo de llegada diferente

Scheduling

Batch: Shortest Remaining Time Next

- Versión preemptive de shortest job first
- Asume que los tiempos de ejecución son conocidos
- Al llegar un nuevo trabajo se compara su tiempo restante (total) con el restante del trabajo actual
- Beneficia

Scheduling
Interactivo: Round-Robin

- Preemptive
- A cada proceso se le asigna un intervalo de tiempo (quantum) para correr
  - Si al finalizar sigue ready, se le quita el CPU de todos modos
  - Si se bloquea antes de que venza el quantum, se pasa al siguiente
- Fácil de implementar
  - Lista de procesos ready
- El tamaño del quantum es importante ¿por qué?

Figure 2-42. Round-robin scheduling. (a) The list of runnable processes. (b) The list of runnable processes after *B* uses up its quantum.

Scheduling
Interactivo: Round-Robin

- Costo del process switch - context switch
  - Context switch: 1ms
  - Quantum: 4ms
  - ¿Costo?

  - Quantum: 100ms
  - ¿Costo?

- ¿Entonces? -> entre 20 y 50 ms es razonable

- ¿Quantum mayor a la ráfaga media de CPU?

Scheduling

Interactivo: Priority scheduling

- Round-Robin asume que todos los procesos son igualmente importantes
- Se le asignan prioridades a los procesos y aquellos con la máxima prioridad son elegidos
- Preemptive
- La prioridad necesita ser ajustada periódicamente para evitar: _____
- La prioridad se puede asignar estáticamente o dinámicamente
  - Dependiendo del usuario
  - Dependiendo del comportamiento del proceso
    - Para I/O-bound puede ser 1/f, donde f es la proporción del último quantum usado

an hour. The UNIX system has a command, *nice*, which allows a user to voluntarily reduce the priority of his process, in order to be nice to the other users. Nobody ever uses it.

Scheduling

Interactivo: Priority scheduling - Multiple queues

- Es conveniente agrupar los procesos por clases usando:
  - Priority scheduling entre clases
  - Round-Robin dentro de cada clase
- Con cierta frecuencia se le asigna un quantum grande a los procesos CPU-bound
- Cuando un proceso usa todo el quantum, se le baja la prioridad

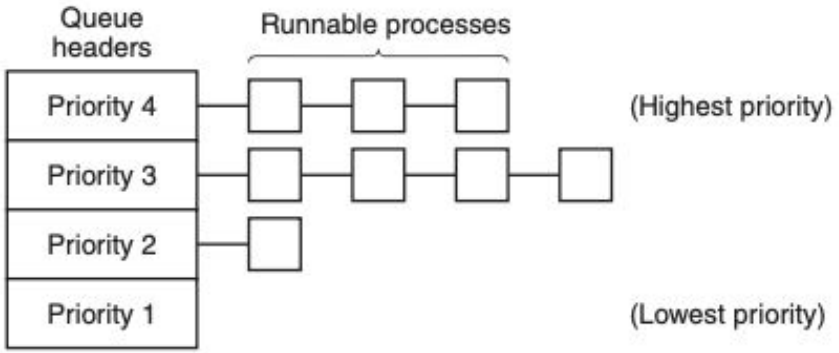


Figure 2-43. A scheduling algorithm with four priority classes.

Ejemplo: Proceso CPU-bound que necesita 100 quantums (q) para terminar.
Inicialmente se le asigna 1q, luego 2q, 4q, 8q, 16q, 32q, 64q (usa 37)
Solo 7 context switches en lugar de 100 con Round-Robin

¿Qué pasa con los I/O-bound?

Política carriage return - anécdota

Scheduling

Interactivo: Priority scheduling - Shortest Process Next

- Parecido a su contraparte de sistemas batch
- Es más difícil determinar el más corto
- Se estima con comportamiento previo (bash) -> shortest (estimated) running time

- La estimación es T_0 y su próxima ejecución es T_1
- Se puede actualizar la estimación con la siguiente fórmula $\mathbf{aT_0 + (1 - a)T_1}$
- La elección de a puede olvidar ejecuciones previas rápidamente (a pequeño) o recordarlas (a grande)

- Si a = 1/2 $T_0, \quad T_0/2 + T_1/2, \quad T_0/4 + T_1/4 + T_2/2, \quad T_0/8 + T_1/8 + T_2/4 + T_3/2$

- Esta técnica se conoce como envejecimiento (aging)

Scheduling

Interactivo: Guaranteed scheduling

- Prometamos que si hay n usuarios conectados, cada uno recibirá 1/n del tiempo de CPU
- Alternativamente, si 1 usuario tiene n procesos, cada proceso recibirá 1/n del tiempo de CPU
- Registrar el tiempo de uso de cada usuario / proceso en términos de la proporción usada
- Elegir aquel proceso con el la menor proporción

Scheduling
Interactivo: Lottery scheduling

- Cada proceso tiene tickets de “lotería” para diferentes recursos.
- ¡Vamos a sortear 20ms de CPU 50 veces por segundo (50Hz)!
- ¿Y si asignamos más tickets a algunos procesos? -> poseer una fracción f de los tickets implica obtener una fracción f del recurso
- Procesos que cooperan entre sí pueden compartir tickets
- Resolver problemas complejos con otro algoritmo: servidor de video que provee de stremming a sus clientes con diferentes frame rates -> 10, 20 y 50 FPS

Scheduling

Interactivo: Fair-Share Scheduling

- ¿Si tenemos 2 usuarios, pero uno ejecuta 9 procesos y otro ejecuta solo 1 proceso? (Asumir Round-Robin)
- El usuario asociado a un proceso es relevante
- Se le asigna una fracción del tiempo a cada usuario:

Usuario 1, procesos A B C D
Usuario 2, proceso E

Si le asignamos 50% de CPU a cada usuario:

A **E** B **E** C **E** D **E** A **E** B **E** C **E** D **E** …

Si le asignamos el doble de CPU al usuario 1 que al usuario 2:

A B **E** C D **E** A B **E** C D **E** …

Ejercicio 1

5 trabajos (A-E) llegan al centro de cómputo casi al mismo tiempo. Los mismos tienen tiempos estimados de ejecución de 10, 6, 2, 4 y 8 minutos respectivamente. Su prioridades (determinadas estáticamente) son 3, 5, 2, 1 y 4 respectivamente, siendo 5 la máxima prioridad. Para cada uno de los siguientes algoritmos de scheduling, determine el turnaround time medio.

1. Round Robin
2. Priority scheduling
3. First-come, First-served (considerar el orden A B C D E)
4. Shortest job first

Para 1. asuma que el sistema es multiprogramado y que cada trabajo obtiene la misma proporción de CPU. Para el resto asuma que solo 1 trabajo ejecuta hasta que termina sin interrupciones.

Todos los trabajos son CPU-bound
Ignore el tiempo necesario para el context switch

Ejercicio 2

Los procesos de un sistema particular tienen un tiempo de ejecución promedio **T** antes de bloquearse por I/O. El tiempo necesario para un context switch es **S**, el cual se considera desperdiciado. Para el algoritmo de scheduling Round-Robin con un quantum **Q**, provea una fórmula de la eficiencia del CPU para cada una de las siguiente condiciones.

1. $Q = \infty$
2. $Q > T$
3. $S < Q < T$
4. $Q = S$
5. Q cercano a 0

Ejercicio 3

5 trabajos están esperando para ser ejecutados. Sus tiempos de ejecución esperados son 9, 6, 3, 5 y X. ¿En qué orden deben ejecutarse para minimizar el turnaround time?

Ejercicio 3

La técnica de envejecimiento (aging) con a = 1/2 es usada para estimar los tiempos de ejecución de un proceso. Las últimas 4 ejecuciones registraron los siguientes tiempos (de más antiguo a más reciente): 40, 20, 40 y 15 ms. ¿Cuál es la estimación para la siguiente ejecución?

Glosario

-