Tema 2:
Scheduling en Linux y Windows

Profesorado de Sistemas Operativos

Universidad de Granada

Sistemas Operativos

Grado de Ingeniería Informática
Grado en Ingeniería Informática y Matemáticas
Grado en Ingeniería Informática y Administración y Dirección de Empresas

2025-2026

Index

De tiempo real

- Los procesos bajo estas políticas tienen prioridades entre 1 y 99 y existe una cola para cada prioridad.
- SCHED_FIFO: Los procesos se ejecutan hasta que terminan, se bloquean, o un proceso de mayor prioridad les quita el uso del procesador. En este último caso el proceso permanece en la cabeza de la cola.
- SCHED_RR: Los procesos se ejecutan durante la duración de una rodaja de tiempo predefinida o hasta que un proceso de mayor prioridad les quita el uso del procesador. Los procesos que no agotan su rodaja porque se activa un proceso de más prioridad, lo harán cuando vuelvan a recibir la CPU.
- SCHED_DEADLINE: utilizado para procesos que requieren activarse periodicamente con un tiempo maximo de finalización en cada periodo. Los procesos se seleccionan según las restricciones de cada uno.

NO de tiempo real

- Los procesos bajo estas políticas tienen prioridad 0 y por lo tanto no se ejecutarán mientras haya algún proceso de tiempo real listo para ejecutarse.
- SCHED_OTHER/SCHED_NORMAL: se aplica un algoritmo de reparto de tiempo. Se establece otro nivel de prioridad (*nice value*) con valores entre -20 (más prioridad) y 19 (menos prioridad) para establecer prioridades en ese reparto.
- SCHED_BATCH: derivada de la anterior. El planificador asumirá que los procesos hacen un uso intensivo del procesador y no requieren de medidas para reducir la latencia. Útil para procesos no interactivos donde se desea reducir los cambios de contexto.
- SCHED_IDLE: adecuada para procesos que se desean ejecutar con la mínima prioridad (por debajo de los procesos con *nice value* igual a 19 de SCHED_OTHER o SCHED_BATCH)

Premisas

- Se desea realizar un reparto equitativo del tiempo de CPU entre los procesos que están listos para ejecutarse.
- Cada proceso tiene un peso y este debe tenerse en cuenta en el reparto de tiempo de procesador. Será por tanto equitativo pero teniendo en cuenta los pesos.
- Asumiremos inicialmente que se dispone de un procesador ideal que es capaz de repartir cualquier periodo de tiempo infinitesimal entre un conjunto de procesos en función de su peso.
  Nota: Este tipo de procesador no existe ya que una CPU solo es capaz de ejecutar una instrucción cada vez.

Tiempo virtual

- Para normalizar el paso del tiempo y conseguir independencia de la cantidad de procesos listos en cada momento, se define el tiempo virtual.
- Una unidad de tiempo virtual se establece como el tiempo requerido para ejecutar un número de unidades de tiempo real de cada proceso igual a su peso bajo las premisas anteriores. La cantidad de procesos es N.
- Esto quiere decir que el tiempo virtual transcurrido durante rt unidades de tiempo real ($VTRT$) es el indicado en la expresión 1.
- Puede observarse que la relación entre el tiempo virtual y el real depende de los procesos que están listos para ejecutarse (de sus pesos).

$$VTRT = \frac{rt}{\sum_{j=1}^{N} Peso_j} (\text{ rt unidades t. real}) \quad (1)$$

Tiempo virtual cont.

- La expresión muestra 2 el tiempo real que se tarda en ejecutar el proceso P_i durante un tiempo pt.
- La expresión muestra 3 el tiempo virtual transcurrido para ejecutar el proceso P_i durante un tiempo pt.
- Puede observarse como $VTPT$ solo depende del peso del proceso en cuestión y es invariante. Cambiar la relación entre el tiempo virtual y el real, pero no la relación entre el tiempo virtual y el tiempo de procesador.

$$RTPT = pt \times \frac{\sum_{j=1}^{N} Peso_j}{Peso_i} (\text{ pt unidades t. de procesador}) \quad (2)$$

$$VTPT = \frac{rt}{\sum_{j=1}^{N} Peso_j} = \frac{pt \times \frac{\sum_{j=1}^{N} Peso_j}{Peso_i}}{\sum_{j=1}^{N} Peso_j} = \frac{pt}{Peso_i} \quad (3)$$

Tiempo virtual cont.

- El tiempo virtual fluye más despacio para los procesos de mayor prioridad.
- También puede interpretarse como que a los procesos de mayor prioridad les da tiempo a ejecutar más instrucciones que a los de menos en el mismo tiempo.

Volviendo a la realidad: no existe la CPU hipotética. Opciones:

- Establecer tiempos de CPU de una longitud prestablecida fija que se van asignando a los procesos y ordenar la ejecución de los mismos de forma que completen sus rodajas de tiempo en el mismo orden que en el caso del procesador hipotético. Este es el enfoque empleado en el método EEVDF.
- Planificar la ejecución de forma que dentro de un horizonte temporal concreto todos los procesos se ejecutan una misma cantidad de tiempo virtual. Este es el principio utilizado en la técnica CFS.

EEVDF(*Earliest Eligible Virtual Deadline First*)

- Es el algoritmo de planificación prestablecido desde la versión 6.6 del kernel de Linux (2023) para la política SCHED_OTHER/SCHED_NORMAL.
- No es un método nuevo, se basa en un artículo científico publicado en 1995.
- Selecciona para ejecutar el proceso cuyo plazo de finalización de su próxima rodaja de CPU esté más cercano (*deadline*).
- Añade que un proceso solo será seleccionado si es *elegible*. Los procesos elegibles son aquellos cuyo tiempo de elegibilidad no sea mayor que el tiempo virtual del sistema.

EEVDF cont.

- El tiempo virtual de fin de la próxima rodaja del proceso P_i (VTENS(P_i)) se calcula como se indica en la expresión 4 (recordar expresión 3). Se elige para ser ejecutado el proceso, de entre los elegibles, cuyo valor asociado sea menor.
- En la expresión 4, si el proceso no llega a consumir la radaja de tiempo completa, se utilizaría la duración de la ráfaga de CPU.
- El punto en el tiempo en el que se hace elegible cada proceso se va actualizando con el tiempo de fin de la siguiente rodaja de CPU. Si un proceso es favorecido y recibe más tiempo de CPU del ideal, no será elegible hasta el momento en que estaba prevista la finalización de su rodaja de tiempo de procesador.

$$VTENS(P_i) = \text{tiempo elegible}P_i + \frac{\text{tamaño rodaja } P_i}{Peso_i} \quad (4)$$

EEVDF cont.

- Cuando un proceso es creado se inicializa el valor de tiempo al que es elegible con el tiempo virtual del sistema y el tiempo de finalización de su rodaja de procesador se obtiene aplicando la expresión 4.
- Cuando un proceso termina de ejecutar su rodaja (o se bloquea) se actualiza el tiempo en que volverá a ser elegible con su valor de *VTENS* y se calcula el nuevo valor para *VTENS*.
- Es importante resaltar como *VTENS*, si se agota la rodaja, se va incrementando con un valor constante para cada proceso.
- Los procesos con más peso incrementan *VTENS* en pasos más pequeños y por tanto serán elegibles antes.

Ejemplo EEVDF

Asumamos que la duración de una rodaja de procesador tendrá una duración de 6ms (tiempo real).

Proceso	t. real creación	Prioridad	Rodaja virtual
P1	0	5	6/5=1.2
P2	0	3	6/3=2.0
P3	0	2	6/2=3.0

En tiempo virtual, con los procesos que están listos para ejecutarse del ejemplo y cuyos pesos suman 10, 6ms se corresponden con 6/10=0.6ms.
La rodaja en tiempo virtual es el tiempo virtual que se va sumando a cada proceso para calcular el momento en el que vuelve a ser elegible.

Ejemplo EEVDF cont.

Las cifras que acompañan a cada proceso son el tiempo virtual a partir del que serán elegibles y el tiempo de finalización de la próxima rodaja.

vt=0.0	vt=0.6	vt=1.2	vt=1.8
P1(0.0,1.2)*	P1(1.2,2.4)X	P1(1.2,2.4)*	P1(2.4,3.6)X
P2(0.0,2.0)	P2(0.0,2.0)*	P2(2.0,4.0)X	P2(2.0,4.0)X
P3(0.0,3.0)	P3(0.0,3.0)	P3(0.0,3.0)	P3(0.0,3.0)*

vt=2.4	vt=3.0	vt=3.6	vt=4.2
P1(2.4,3.6)*	P1(3.6,4.8)X	P1(3.6,4,8)*	P1(4.8,6.0)X
P2(2.0,4.0)	P2(2.0,4.0)*	P2(4.0,6.0)X	P2(4.0,6.0)
P3(3.0,6.0)X	P3(3.0,6.0)	P3(3.0,6.0)	P3(3.0,6.0)

X→No elegible; *→Proceso elegido (de los elegibles el tiempo más cercano de finalización de rodaja)

VT	**0**			**0.6**			**1.2**			**1.8**	2.0		**2.4**			**3.0**			**3.6**		4.0	**4.2**			**4.8**			**5.4**			**6.0**
P1 interval.	X						X						X						X						X					X
P1 ejec.																															
P2 interval.	X										X										X									X
P2 ejec.																															
P3 interval.	X															X														X
P3 ejec.																															

Ejemplo EEVDF cont.

La figura muestra:

- Los momentos de tiempo virtual en que se toman decisiones de planificación (cifras VT en negrita).
- Los intervalos en los que es elegible cada proceso. Están delimitados por el simbolo *X*. En cada uno de esos periodos solo pueden ejecutarse una vez.
- Los momentos en que se está ejecutando cada proceso.
- Puede verse como P1 es seleccionado 5 veces, P2 3 veces y P3 dos veces. Estas cifras coinciden con sus pesos.

Otro ejemplo EEVDF

La rodaja de procesador tendrá una duración de 6ms (tiempo real). Como P2 y P3 son creados después alteran la relación entre el tiempo real y el virtual

	Creación (real)	Prioridad	Rodaja virt	Creación (virt)
P1	0	5	6/5=1.2	0.0
P2	4	3	6/3=2.0	0.0+4/5=0.8
P3	8 (P2+4)	2	6/2=3.0	0.8+4/8=1.3

El divisor del tiempo real para obtener el virtual es 5 cuando solo existe P1, 8 con P1 y P2 y 10 cuando están listos los tres procesos.

Otro ejemplo EEVDF cont.

vt=0.0	vt=0.8(!)	vt=1.05	vt=1.3(!)
P1(0.0,1.2)*	P1 cont.	P1(1.2,2.4)X	P1(1.2,2.4)
	P2(0.8,2.8)	P2(0.8,2.8)*	P2 cont.
			P3(1.3,4.3)

vt=1.7	vt=2.3	vt=2.9
P1(1.2,2.4)*	P1(2.4,3.6)X	P1(2.4,3.6)*
P2(2.8,4.8)X	P2(2.8,4.8)X	P2(2.8,4.8)
P3(1.3,4.3)	P3(1.3,4.3)*	P3(4.3,7.3)X

(!) $\rightarrow$ Se produce un cambio en la escala del tiempo virtual y afecta al momento de finalización de la rodaja del proceso en ejecución.

Otro ejemplo EEVDF cont.

- La primera rodaja de P1 estaba previsto que finalizara en el punto 0+6/5=1.2 de tiempo virtual. Sin embargo, la aparición de P2 modifica la escala del tiempo virtual.
- Cuando aparece P2 faltaban por ejecutar 0.4 unidades de tiempo virtual. Como se corresponde con un tercio de la rodaja, calculamos la duración en el nuevo tiempo virtual de ese tercio. 6/3=2ms de tiempo real son 2/8 unidades virtuales. 0.8+0.25=1.05.
- Ocurre lo mismo con la aparición de P3. La rodaja de P2 estaba prevista que finalizara en vt=1.05+6/8=1.8. En vt=1.3 faltaban por ejecutar 0.5 unidades virtuales (2/3 de la rodaja). Si pasamos al nuevo tiempo virtual 2/3 de 6 ms obtenemos: 6*2/3=4 y 4/10=0.4 unidades virtuales. Finalmente 1.3+0.4=1.7.

CFS(*Completely Fair Scheculer*)

- Ha sido el planificador prederminado de Linux desde la versión 2.6.23 (2007) hasta la 6.6 (2023).
- Ha sido sustituido por EEVDF porque CFS requiere considerar unos serie de casos particulares que EEVDF trata de forma sencilla. Se trata de casos con procesos muy sensibles a la latencia.
- También utiliza el concepto de tiempo virtual, pero en este caso de ejecución (*vruntime*).
- El peso se calcula según la expresión 5
- La prioridad se refiere al *nice value* mencionado anteriormente.

$$Peso(\text{prioridad}) \approx \frac{1024}{1{,}25^{\text{prioridad}}}, -20 \leq \text{prioridad} \leq 19$$

$$88761 \geq Peso(\text{prioridad}) \geq 15 \tag{5}$$

CFS cont.

- La expresión de *vruntime* puede verse en la expresión 6.
- Se estabece un horizonte de planificación (periodo) y se reparte ese tiempo equitativamente en función del peso (expresión 7).
- La expresión 8 muestra como el incremento de *vruntime* es igual para todos los procesos que consumen su rodaja. Se obtiene sustituyendo el tiempo de procesador de una rodaja en 6.

$$\text{vruntime}_{P_i} += \frac{pt \times Peso(0)}{Peso_{P_i}} \tag{6}$$

$$\text{rodaja}_{P_i} = \frac{\text{periodo} \times Peso_{P_i}}{\sum_{j=1}^{N} Peso_{P_j}} \tag{7}$$

$$\text{vruntime}_{P_i} += \frac{\text{rodaja}_{P_i} \times Peso(0)}{Peso_{P_i}} = \frac{\text{periodo} \times \cancel{Peso_{P_i}} \times Peso(0)}{\sum_{j=1}^{N} Peso_{P_j} \times \cancel{Peso_{P_i}}}$$

$$= \frac{\text{periodo} \times Peso(0)}{\sum_{j=1}^{N} Peso_{P_j}}$$

CFS cont.

- Durante un periodo se desea ejecutar todos los procesos listos, aunque en una proporción fijada por el peso (ver expresión 7). Para evitar demasiados cambios de contexto se fija un tiempo de rodaja de CPU mínimo **medio**. Si el periodo es menor que *N* veces ese tiempo mínimo, el periodo se aumenta hasta ese valor (siendo *N* el número de procesos listos).
- Cuando se bloquea un proceso su valor de *vruntime* se queda congelado y se verá beneficiado al volver a estar listo. Este beneficio se limita al tiempo de una ronda de planificación para evitar que se pueda acaparar el procesador.
- Cuando se incorpoda un nuevo proceso se le asigna como *vruntime* el valor mínimo de los procesos listos por la misma razón.
- Se utiliza un árbol para mantener ordenados de forma eficiente por *vruntime* los procesos listos. El que tiene el menor valor está en siempre la hoja situada más a la izquierda.

Planificación en sistemas multiprocesador

Planificación particionada con migración

- Se utiliza una cola por cada procesador. Eso reduce la competencia por el acceso a esa cola y reduce los problemas de acceso simultáneo.
- Se utilizan criterios de migración para mantener la carga entre procesadores equilibrada.
- La gestión multiprocesador y el algoritmo de planificación son independientes entre sí.
- Se mantiene de forma natural la afinidad para maximizar el aprovechamiento de las memorias caché de los procesadores.

Windows

- Utiliza una planificación multi-cola de tipo apropiativo.
- Cada cola tiene una prioridad y dentro de las mismas se utiliza un esquema *round robin*.
- Las prioridades son valores entre 0 y 31. Los procesos con prioridades más altas [16,31] se califican como de tiempo real. La prioridad 0 está reservada para el sistema. Las prioridades [1,15] de califican como *Dynamic (normal).*
- Si se crea/desbloquea un proceso de mayor prioridad que el proceso al que se tiene asignado el procesador, aunque no haya finalizado la rodaja de tiempo, se le asignará la CPU al recién llegado.
- El *quantum* de CPU usado es más largo en las versiones de Windows para servidores que en el resto.

Windows cont.

- La planificación se hace al nivel de hebras.
- Se puede incrementar temporalmente la prioridad de una hebra por cuestiones como, entre otras:
  - reducir la latencia/reducir el tiempo de respuesta de un programa interactivo.
  - favorecer a hebras que vuelven a estar listas después de una operación de E/S.
  - sacar a una hebra de un estado de inanición.
- Después de un incremento de prioridad se puede producir una reducción de prioridad.
- En entornos multiprocesador las hebras listas para ejecutarse se dividen (desde Windows 8/2012) en colas por grupos de procesadores. Los procesadores se agrupan para favorecer el mantenimiento de la afinidad.

Tema 2:
Scheduling en Linux y Windows

Profesorado de Sistemas Operativos

Universidad de Granada

Sistemas Operativos

Grado de Ingeniería Informática
Grado en Ingeniería Informática y Matemáticas
Grado en Ingeniería Informática y Administración y Dirección de Empresas

2025-2026