PLANIFICACIÓN DE PROCESOS

DEPARTAMENTO DE CIENCIAS E INGENIERÍA DE LA COMPUTACIÓN

UNIVERSIDAD NACIONAL DEL SUR

AGENDA

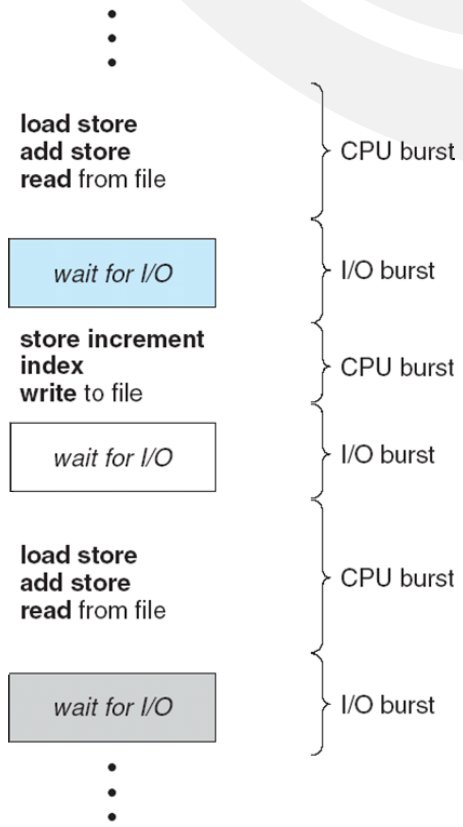
1. Conceptos Básicos
2. Criterios de Planificación
3. Algoritmos de Planificación
4. Planificación de hilos
5. Planificación de Múltiples Procesadores
6. Planificación en Tiempo Real
7. Ejemplos de Sistemas Operativos

AGENDA

1. **Conceptos Básicos**
2. Criterios de Planificación
3. Algoritmos de Planificación
4. Planificación de hilos
5. Planificación de Múltiples Procesadores
6. Planificación en Tiempo Real
7. Ejemplos de Sistemas Operativos

CONCEPTOS BÁSICOS

- Máxima utilización de CPU obtenida con multiprogramación
- La ejecución de procesos consiste de ***ciclos*** de ejecución de CPU y esperas en E/S.
- Distribución de ráfagas de CPU



PLANIFICADOR DE CPU

- Selecciona entre los procesos en memoria que están listos para ejecutar, y aloca la CPU a uno de ellos.
- La decisión de planificar la CPU puede tener lugar cuando un proceso:
  1. Conmuta de ejecutando a estado de espera.
  2. Conmuta de ejecutando a estado de listo.
  3. Conmuta de espera a listo.
  4. Termina.
- La planificación de 1 y 4 es ***no apropiativa***.
- Las otras planificaciones son ***apropiativas***.

DESPACHADOR

- El módulo despachador pasa el control de la CPU al proceso seleccionado por el planificador de corto término; esto implica:
  - cambio de contexto
  - conmutación a modo usuario
  - salta a la dirección apropiada en el programa de usuario para reiniciarlo

- *Latencia de despacho* – tiempo que toma al despachador para detener un proceso e iniciar otro.

AGENDA

1. Conceptos Básicos
2. **Criterios de Planificación**
3. Algoritmos de Planificación
4. Planificación de hilos
5. Planificación de Múltiples Procesadores
6. Planificación en Tiempo Real
7. Ejemplos de Sistemas Operativos

CRITERIOS DE PLANIFICACIÓN

- **Utilización de CPU** – mantener la CPU tan ocupada como sea posible
- **Procesamiento total (Throughput)** – número de procesos que completan sus ejecución por unidad de tiempo.
- **Tiempo de retorno** – cantidad de tiempo para ejecutar un determinado proceso.
- **Tiempo de Espera** – cantidad de tiempo que un proceso ha estado esperando en las colas.
- **Tiempo de respuesta** – cantidad de tiempo que transcurre desde que fue hecho un requerimiento hasta que se produce la primer respuesta, **no salida.**

AGENDA

1. Conceptos Básicos
2. Criterios de Planificación
3. **Algoritmos de Planificación**
4. Planificación de hilos
5. Planificación de Múltiples Procesadores
6. Planificación en Tiempo Real
7. Ejemplos de Sistemas Operativos

PLANIFICACIÓN PRIMERO-ENTRAR, PRIMERO-SERVIDO (FCFS)

- Ejemplo:

Proceso	Tiempo de Ráfaga
P_1	24
P_2	3
P_3	3

- Suponer que los procesos llegan en el orden: P_1 , P_2 , P_3

  La carta de Gantt para la planificación es:

- Tiempo de espera para P_1 = 0; P_2 = 24; P_3 = 27
- Tiempo medio de espera: (0 + 24 + 27)/3 = 17

PLANIFICACIÓN FCFS

Suponer que los procesos llegan en el orden

$$P_2 , P_3 , P_1 .$$

- La carta de Gantt para la planificación es:

- Tiempo de espera para $P_1 = 6; P_2 = 0, P_3 = 3$
- Tiempo medio de espera: (6 + 0 + 3)/3 = 3
- Mucho mejor que el caso anterior.
- *Efecto Convoy* los procesos cortos delante de los procesos largos

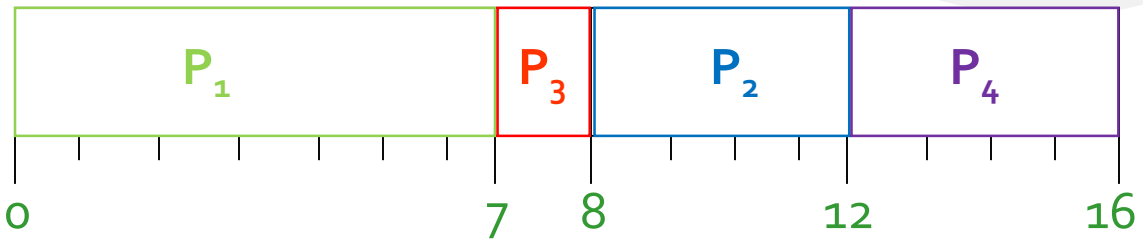
PLANIFICACIÓN JOB-MÁS CORTO PRIMERO (SJF)

- Se asocia con cada proceso la longitud de su *próxima* ráfaga de CPU. Se usa estas longitudes para planificar los procesos con el tiempo más corto.
- Dos esquemas:
  - **No apropiativo**
  - **Apropiativo**
- SJF es óptimo – da el mínimo tiempo de espera promedio para un dado conjunto de procesos.

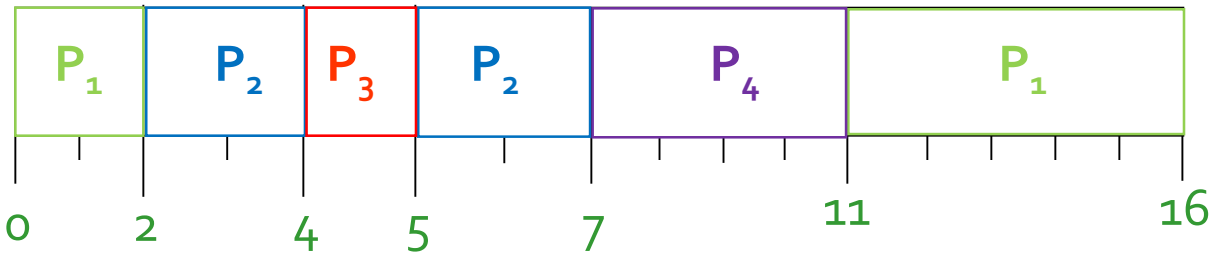
EJEMPLO DE SJF

Proceso	Tiempo de llegada	Ráfaga
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- **SJF (no apropiativo)**



- Tiempo medio de espera = (0 + 6 + 3 + 7)/4 = 4
- **SJF (apropiativo)**



- Tiempo medio de espera = (9 + 1 + 0 +2)/4 = 3

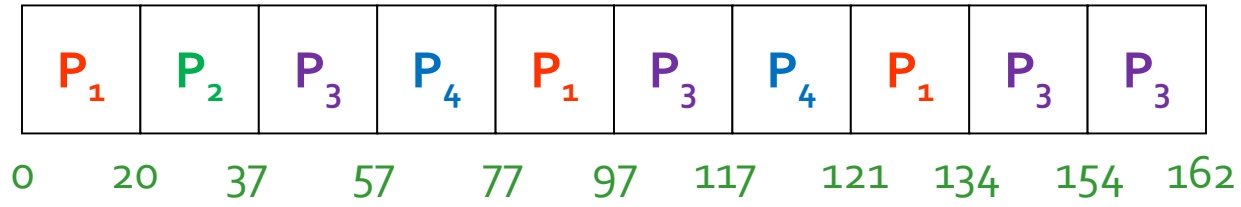
ROUND ROBIN (RR)

- Cada proceso toma una pequeña unidad de tiempo de CPU (***quantum***). Luego de este tiempo el proceso es quitado de la CPU y agregado a la cola de listos.
- Si hay *n* procesos en la cola de listos y el tiempo del quantum es *q*, entonces cada proceso toma 1/*n* del tiempo de CPU en rebanadas de a lo sumo *q* unidades de tiempo a la vez. Los procesos no esperan mas que (*n*-1)*q* unidades de tiempo.
- Rendimiento
  - *q* largo $\Rightarrow$ Primero-Entrar, Primero-Salir
  - *q chico* $\Rightarrow$ *q* debe ser grande con respecto al cambio de contexto, sino la sobrecarga es demasiado grande.
  - Con un Quantum PEQUEÑO se incrementan los Cambios de Contexto

EJEMPLO: RR CON QUANTUM = 20

Proceso	Ráfaga
P_1	53
P_2	17
P_3	68
P_4	24

- La carta de Gantt:



- Típicamente, mayor tiempo de retorno promedio que SJF, pero mejor *respuesta*.

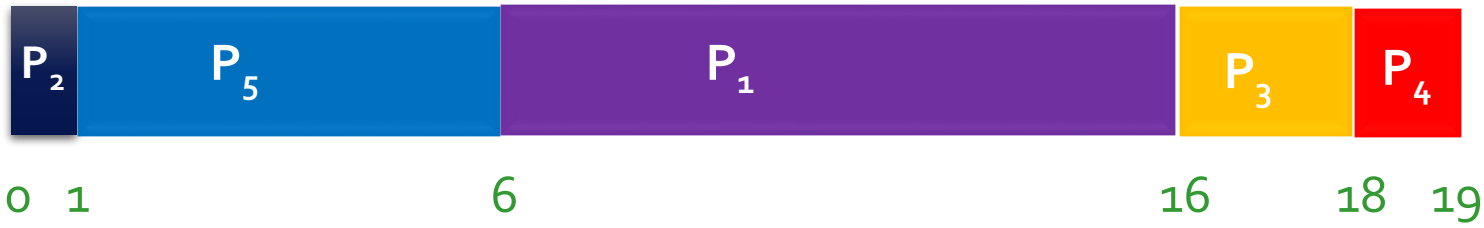
PLANIFICACIÓN POR PRIORIDAD

- Con cada proceso se asocia un número
- La CPU es alocada al proceso con prioridad más alta según la convención elegida.
  - Apropiativo
  - No apropiativo
- SJF es un algoritmo de planificación con prioridad.
- **Problema** $\Rightarrow$ **Inanición** – los procesos de baja prioridad pueden no llegar a ejecutarse nunca.
- **Solución** $\equiv$ **Envejecimiento** – se incrementa en el tiempo la prioridad de los procesos en espera.

EJEMPLO: PLANIFICACIÓN CON PRIORIDAD

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



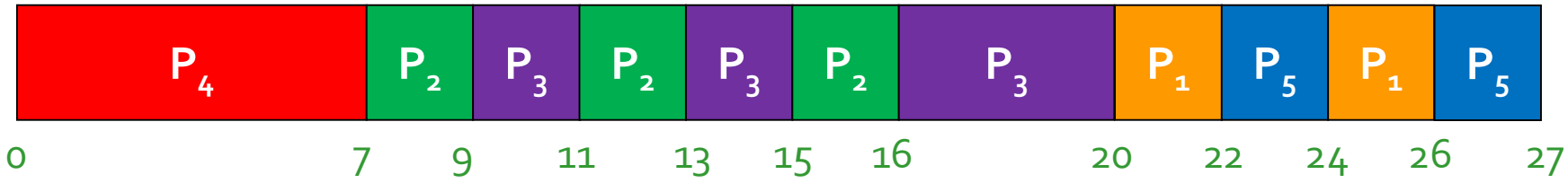
- Tiempo de espera promedio= 8.2 msec

EJEMPLO: PLANIFICACIÓN CON PRIORIDAD CON ROUND ROBIN

Proceso	Ráfaga	Prioridad
P_1	4	3
P_2	5	2
P_3	8	2
P_4	7	1
P_5	3	3

Ejecuta el proceso con la mayor prioridad. Procesos con la misma prioridad ejecutan con round-robin.

- La carta de Gantt con tiempo quantum 2 ms

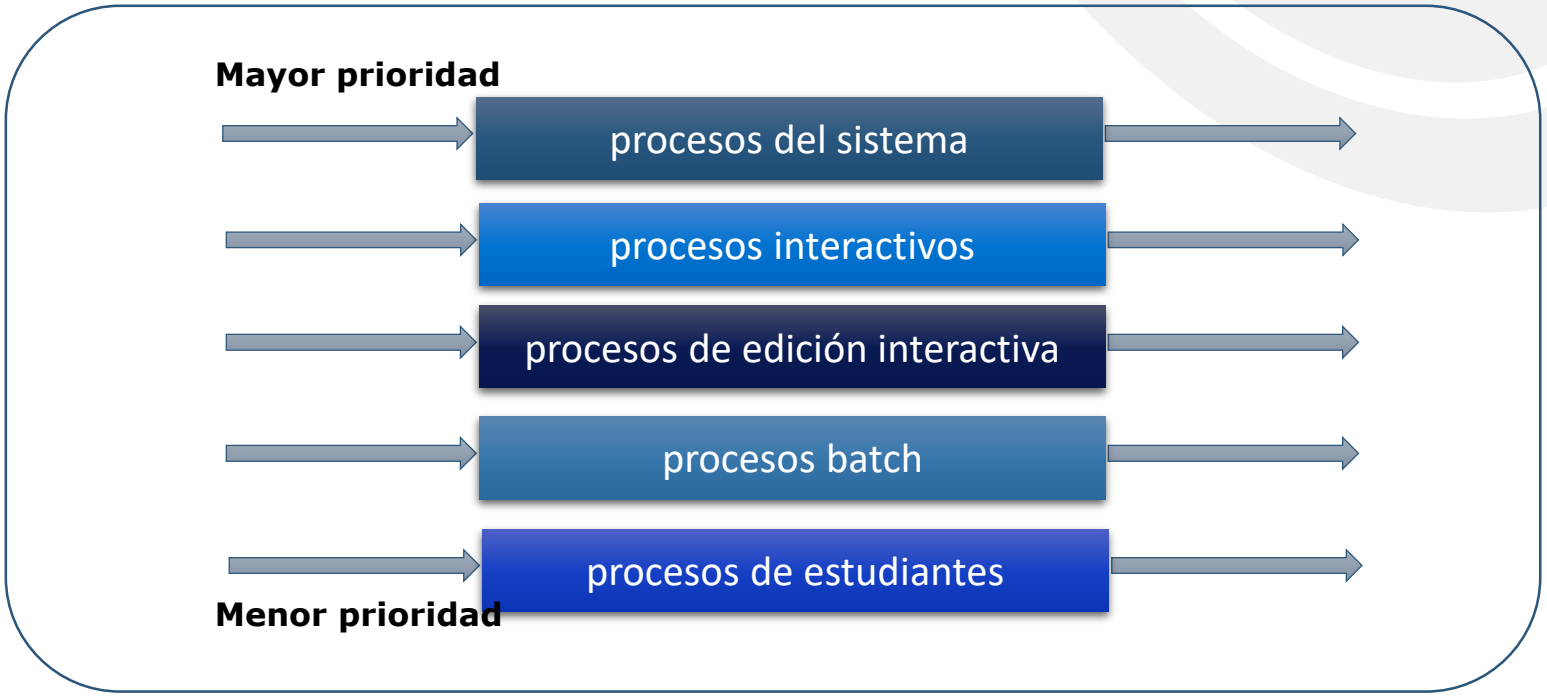


COLAS MULTINIVEL

- La cola de listos esta particionada en varias colas separadas, cada una tiene diferente prioridad.
- Algoritmo de planificación para *cada una de las colas*.
- Algoritmo de planificación *entre las colas*. Se elige el proceso que está en la cola de mayor prioridad.
  - Planificación con prioridad fija. Posibilidad de inanición.
  - Tajada de tiempo – cada cola tiene una cierta cantidad de tiempo de CPU que puede planificar entre sus procesos.

COLAS MULTINIVEL

Por ejemplo: colas con prioridad basada en el tipo de proceso

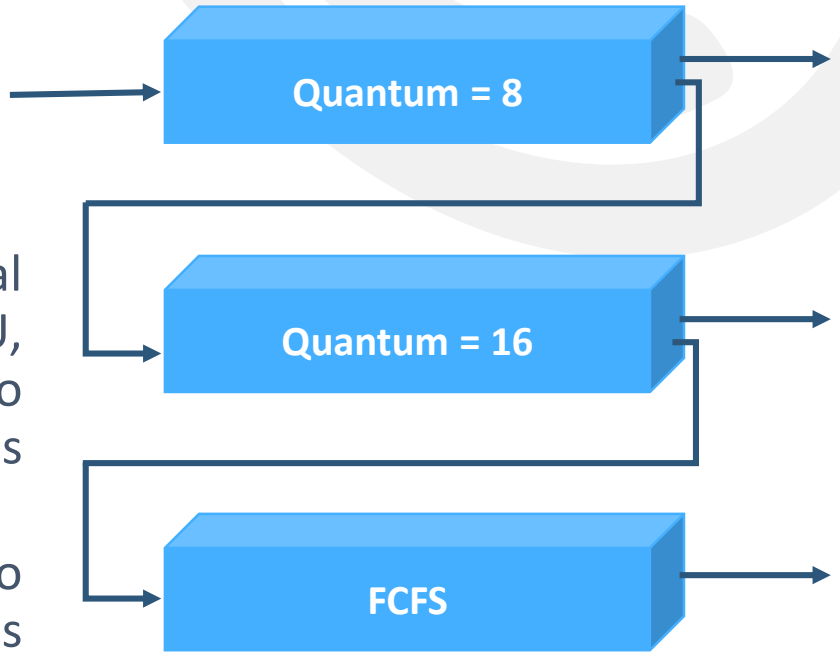


COLAS MULTINIVEL REALIMENTADAS

- Un proceso puede moverse entre varias colas.
- El planificador de colas multinivel realimentadas está definido por los siguientes parámetros:
  - Número de colas
  - Algoritmos de planificación para cada cola
  - Método usado para determinar cuando mejorar un proceso
  - Método usado para determinar cuando degradar un proceso
  - Método usado para determinar en que cola entra un proceso cuando necesita servicio.

EJEMPLO DE COLAS MULTINIVEL REALIMENTADAS

- Tres colas:
  - Q_0 – quantum de 8 milisegundos
  - Q_1 – quantum de 16 milisegundos
  - Q_2 – FCFS
- Planificación
  - Un nuevo job entra a la cola Q_0 el cual es servido FCFS. Cuando gana la CPU, el job recibe 8 milisegundos. Si no finaliza en 8 milisegundos, el job es movido a la cola Q_1.
  - En Q_1 el job es nuevamente servido FCFS y recibe 16 milisegundos adicionales. Si aún no completa, es movido a la cola Q_2.

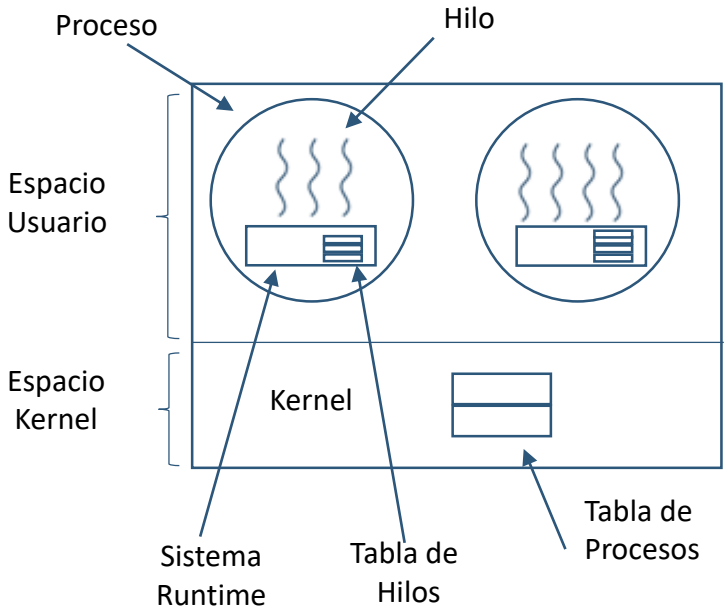


AGENDA

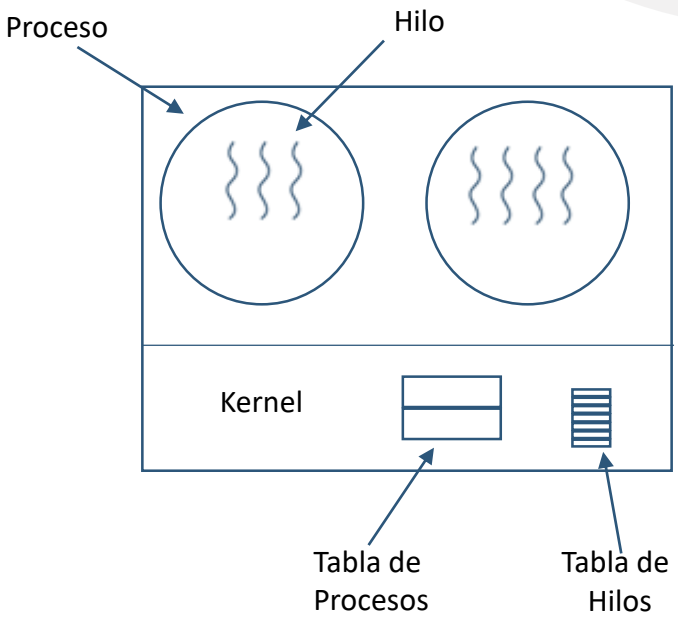
1. Conceptos Básicos
2. Criterios de Planificación
3. Algoritmos de Planificación
4. **Planificación de hilos**
5. Planificación de Múltiples Procesadores
6. Planificación en Tiempo Real
7. Ejemplos de Sistemas Operativos

PLANIFICACIÓN DE HILOS

- Planificación Local – Como deciden las librerías de hilos poner el hilo en un LWP (Light-Weight Process). ***PROCESS-CONTENTION SCOPE (PCS)***
- Planificación Global – Como el kernel decide que hilo del kernel es el siguiente que corre. ***SYSTEM-CONTENTION SCOPE (SCS)***



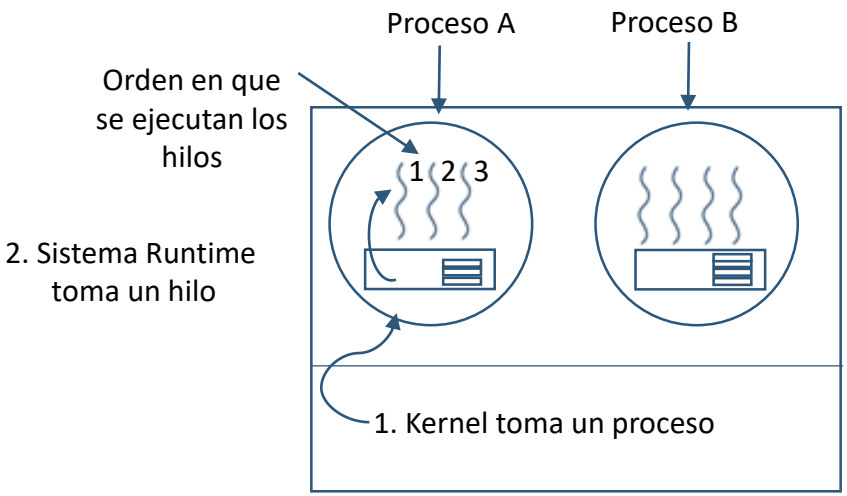
Implementación de librería a nivel de usuario



Implementación a nivel de kernel

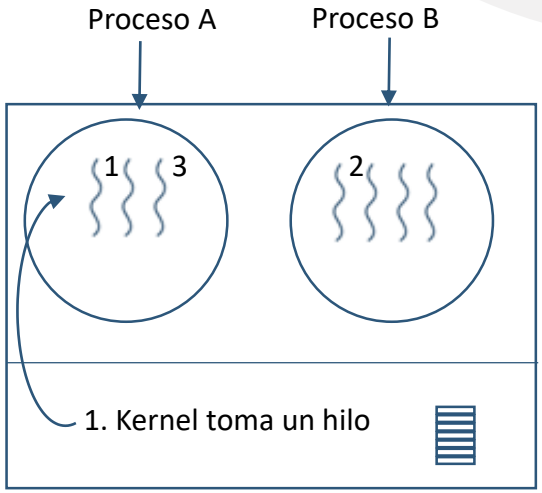
PLANIFICACIÓN DE HILOS

Hilos a NIVEL DE USUARIO. Quantum de proceso de 50 msec e hilos ejecutándose 5 msec



Posible: A1, A2, A3, A1, A2, A3
No es posible: A1, B1, A2, B2, A3, B3

Hilos a NIVEL DE KERNEL. Hilos ejecutándose 5 msec



Posible: A1, A2, A3, A1, A2, A3
También es posible: A1, B1, A2, B2, A3, B3

AGENDA

1. Conceptos Básicos
2. Criterios de Planificación
3. Algoritmos de Planificación
4. Planificación de hilos
5. **Planificación de Múltiples Procesadores**
6. Planificación en Tiempo Real
7. Ejemplos de Sistemas Operativos

PLANIFICACIÓN MÚLTIPLE-PROCESADOR

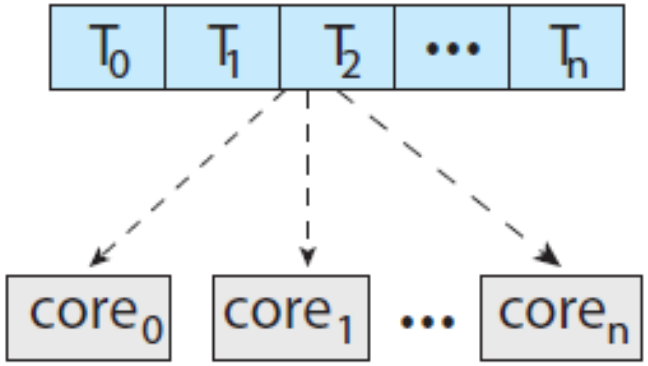
La planificación de CPU es más compleja cuando hay disponibles múltiples CPUs. *Procesadores homogéneos* en un multiprocesador.

- *Cómo se planifica*
- *Carga compartida*
- *Dónde se ejecuta*

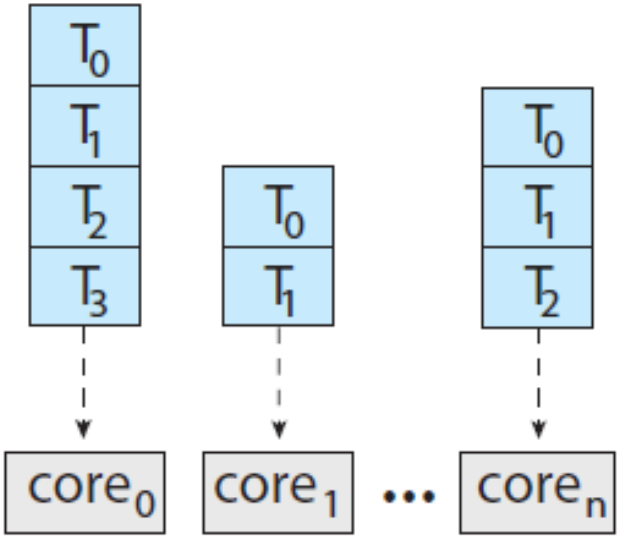
PLANIFICACIÓN MÚLTIPLE-PROCESADOR

FORMAS DE PLANIFICACIÓN

- *Multiprocesamiento Asimétrico* – solo un procesador accede a las estructuras de datos del sistema, simplificando el manejo de datos compartidos.
- *Multiprocesamiento Simétrico* – cada uno de los procesadores puede realizar la planificación.



Un cola compartida



Un cola por core

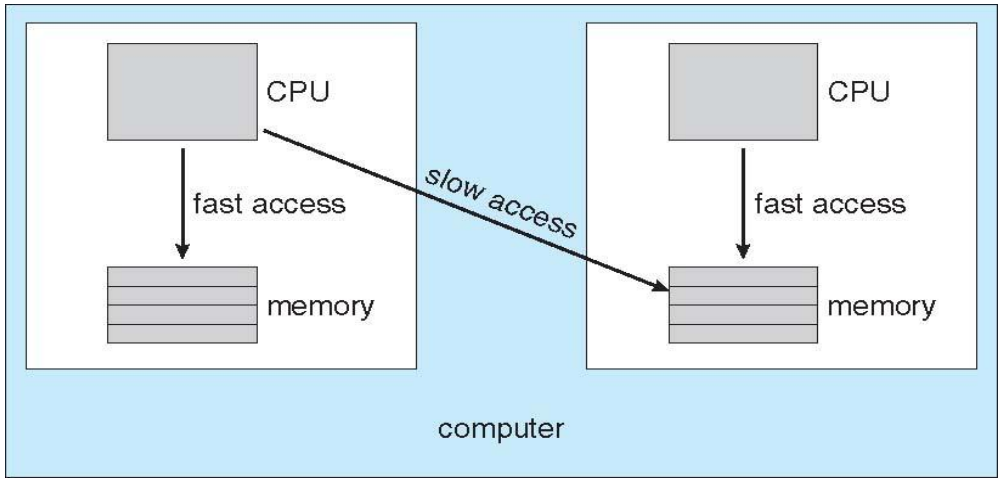
PLANIFICACIÓN MÚLTIPLE-PROCESADOR

Carga compartida

- Mantener a todos los procesadores ocupados y distribuida la carga. Es necesario en el caso que cada procesador tiene su propia cola.
- Formas de alcanzarlo
  - Push Migration
  - Pull Migration

PLANIFICACIÓN MÚLTIPLE-PROCESADOR

- *Dónde se ejecuta.* *PROCESSOR AFFINITY* – el proceso tiene afinidad con el procesador en el cual se está ejecutando. Puede ser **SOFT** o **HARD AFFINITY**



AGENDA

1. Conceptos Básicos
2. Criterios de Planificación
3. Algoritmos de Planificación
4. Planificación de hilos
5. Planificación de Múltiples Procesadores
6. **Planificación en Tiempo Real**
7. Ejemplos de Sistemas Operativos

PLANIFICACIÓN TIEMPO REAL

- *Sistemas de Tiempo Real Duro (**HARD REAL-TIME**)* – requiere completar tareas críticas en una cantidad de tiempo garantizado.
- *Computación de Tiempo Real Blando (**SOFT REAL-TIME**)* – requiere que los procesos críticos reciban prioridad sobre otros.

CONSIDERAR LA LATENCIA

PLANIFICACIÓN TIEMPO REAL

Tipo de tareas

- Aperiódicas
- Periódicas

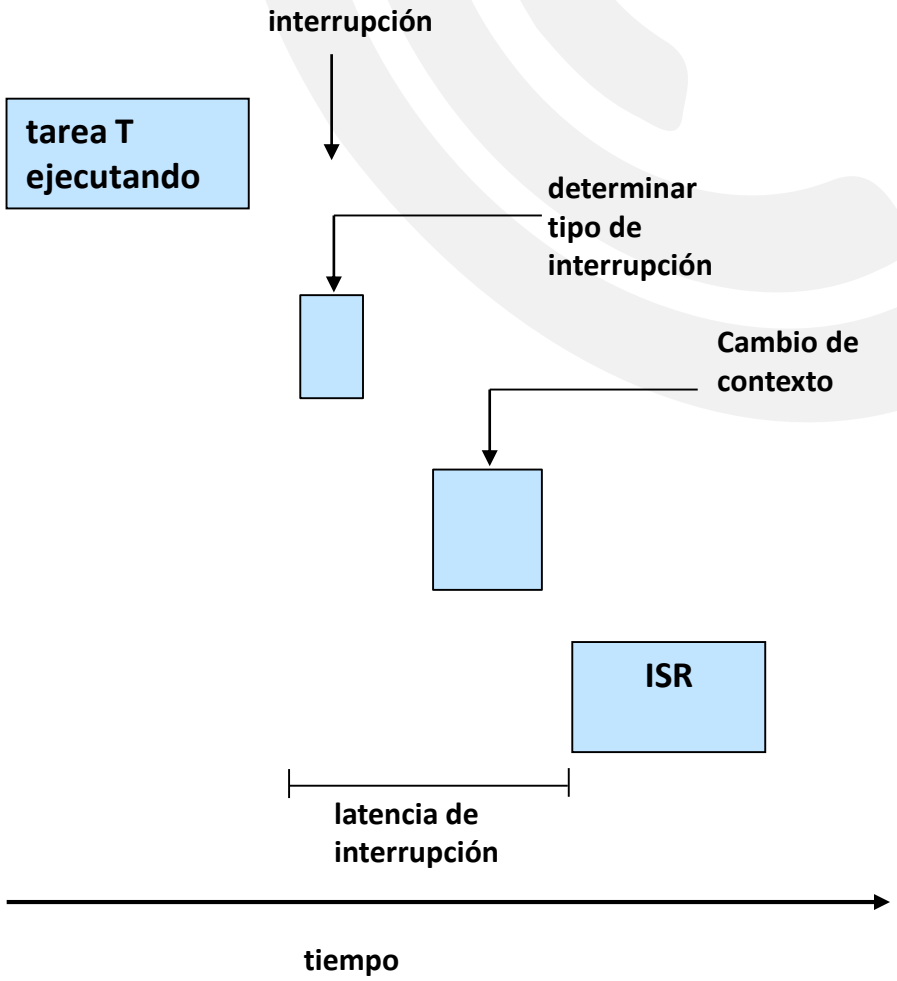
Características de un S.O. en tiempo real

- Determinismo
- Capacidad de respuesta
- Control de usuario
- Fiabilidad
- Operación a prueba de fallas (fail-soft)

PLANIFICACIÓN TIEMPO REAL

Dos tipos de latencias afectan el rendimiento

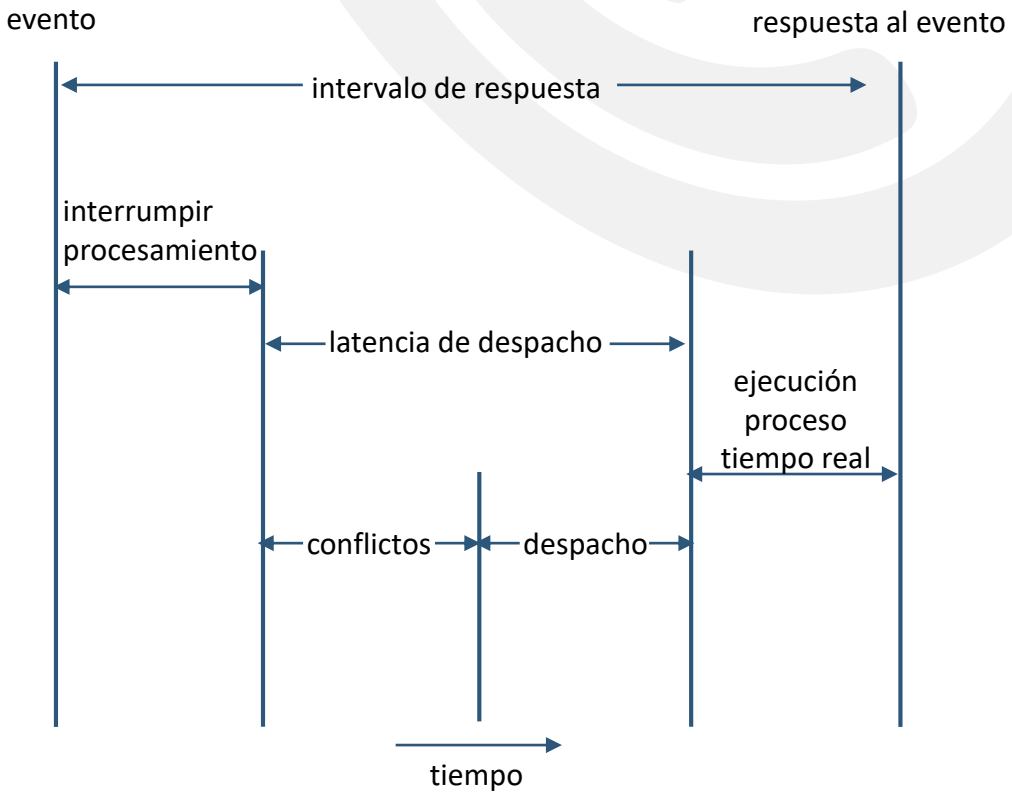
1. **LATENCIA DE INTERRUPCIÓN** – tiempo desde que arriba la interrupción hasta que comienza la rutina de atención de la interrupción.
2. **LATENCIA DE DESPACHO (DISPATCH LATENCY)** – tiempo del despachador de parar un proceso e iniciar otro.



PLANIFICACIÓN TIEMPO REAL

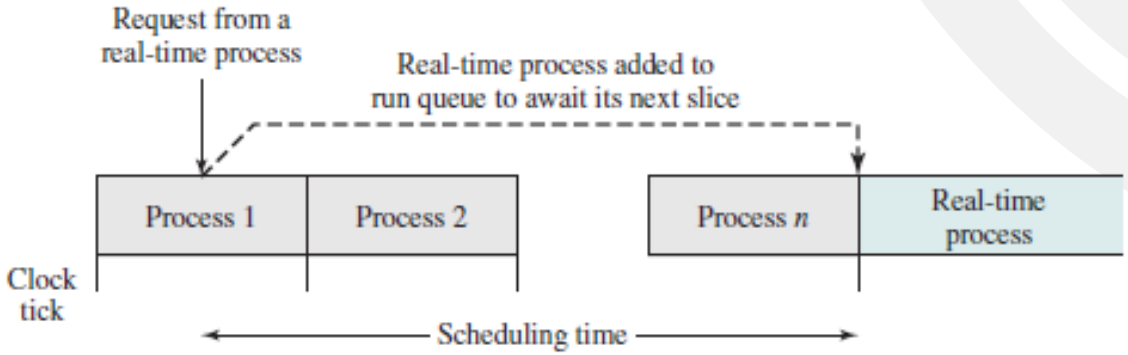
Fase conflicto de la **LATENCIA DE DESPACHO**.

1. Apropiación de cualquier proceso que se está ejecutando en modo kernel.
2. Liberar recurso de proceso con baja prioridad necesitado por proceso con alta prioridad.

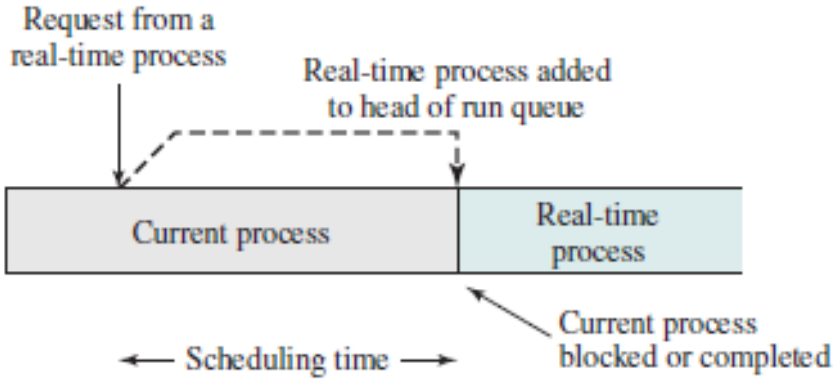


PLANIFICACIÓN TIEMPO REAL

- Planificación round-robin

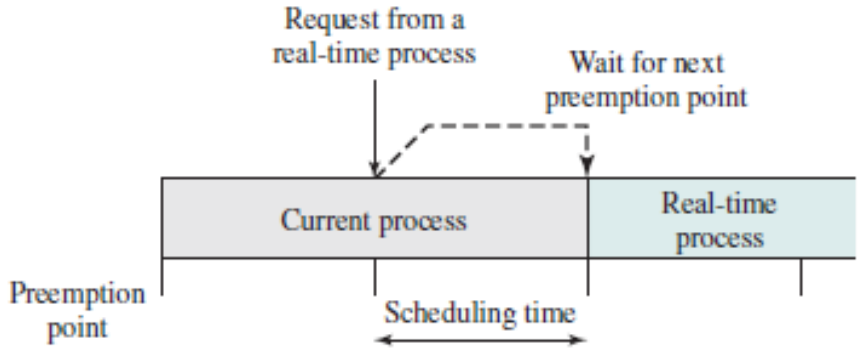


- Planificación basada en prioridades no apropiativa

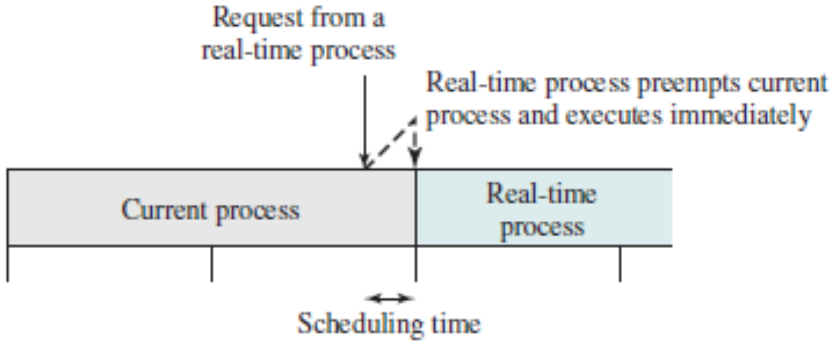


PLANIFICACIÓN TIEMPO REAL

- Planificación basada en prioridades apropiativa en punto de apropiación

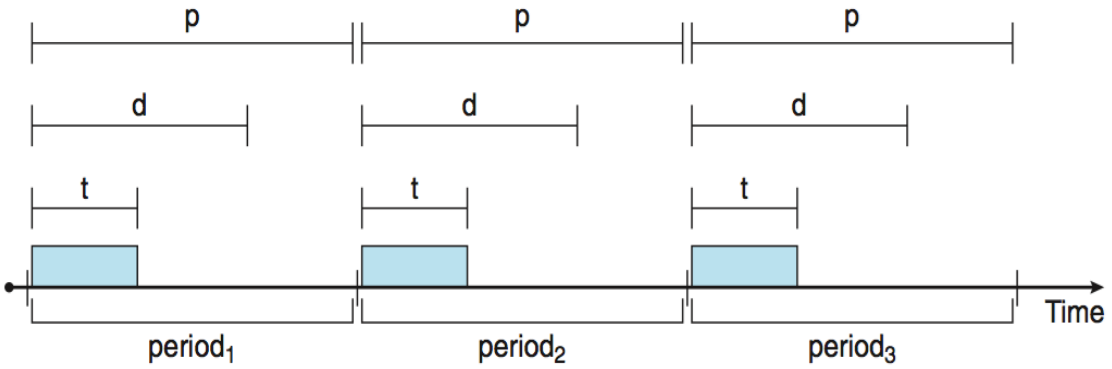


- Planificación apropiativa inmediata



PLANIFICACIÓN TIEMPO REAL

- Para la planificación en tiempo real blando *SOFT REAL-TIME*, el planificador debe ser apropiativo y basado en prioridades.
- Para tiempo real duro (*HARD REAL-TIME*) debe soportar vencimientos (deadlines).
  - Procesos tienen nuevas características:
    - **PERIÓDICOS** requieren la CPU a intervalos constante. **APERIÓDICOS**.
    - tiempo de procesamiento ***t***, deadline ***d***, período ***p***
    - $0 \leq t \leq d \leq p$
    - **Rate** de la tarea periódica es $1/p$



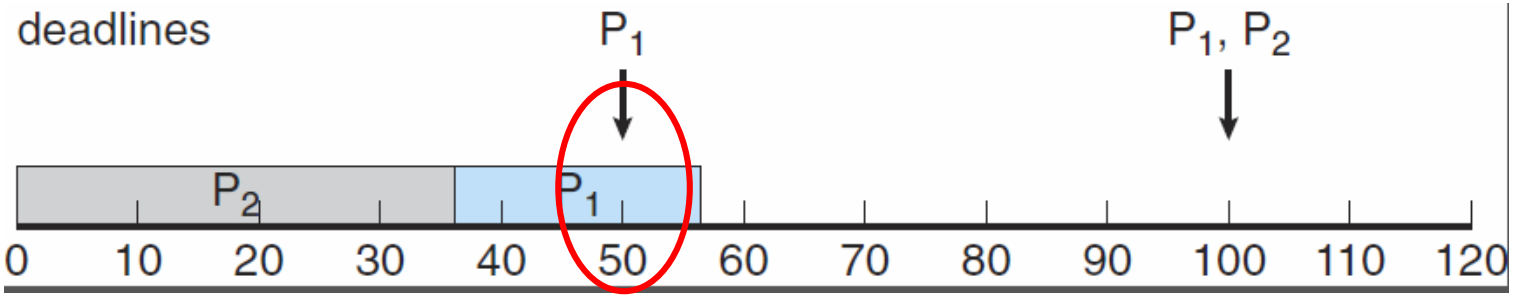
PLANIFICACIÓN TIEMPO REAL

- Planifica las tareas periódica utilizando prioridades estáticas con apropiación.
- Medida de utilización de CPU t_i/p_i
- Para la aceptación de un nuevo proceso se debe cumplir la siguiente condición: $\sum \frac{t_i}{p_i} \leq 1$
- Por ejemplo:

P_1: $p_1 = 50$, $t_1 = 20$, $t_1/p_1 = 0{,}40$

P_2: $p_2 = 100$, $t_2 = 35$, $t_2/p_2 = 0{,}35$

prior(P_2) > prior(P_1)



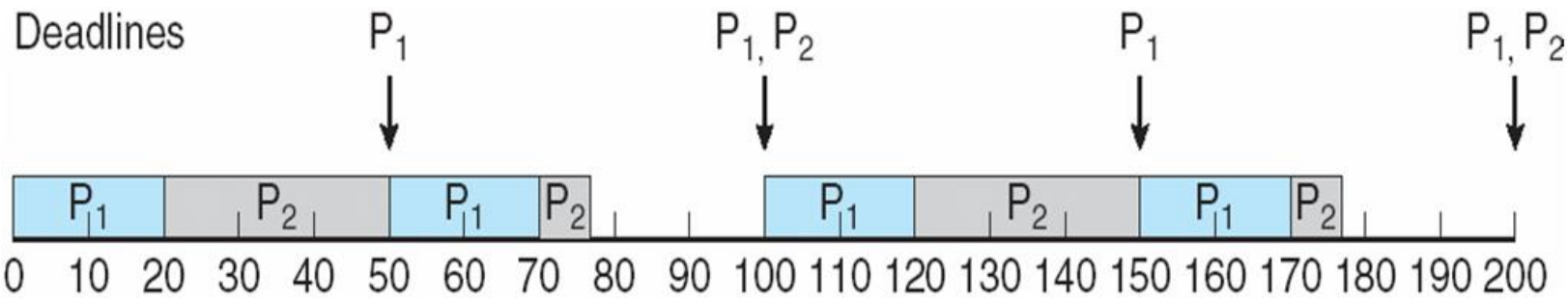
PLANIFICACIÓN TIEMPO REAL – RATE MONOTONIC

- La prioridad se asigna en función de la inversa de su período. Prioridad ESTÁTICA
- Períodos cortos = prioridad alta; Períodos largos= prioridad baja
- Por ejemplo:

P_1: $p_1 = 50$, $t_1 = 20$ – $t_1/p_1 = 0{,}40$

P_2: $p_2 = 100$, $t_2 = 35$ – $t_2/p_2 = 0{,}35$

prior(P_1) > prior(P_2)



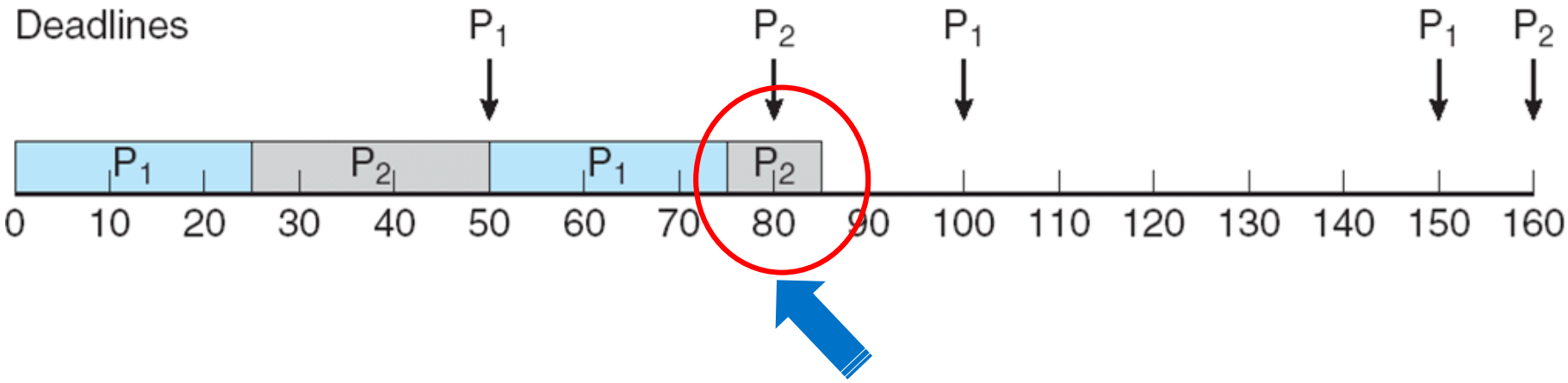
PLANIFICACIÓN TIEMPO REAL – RATE MONOTONIC

- Por ejemplo:

  P_1: $p_1 = 50$, $t_1 = 25$ – $t_1/p_1 = 0{,}50$

  P_2: $p_2 = 80$, $t_2 = 35$ – $t_2/p_2 = 0{,}44$

  prior(P_1) > prior(P_2)



No cumple el deadline

PLANIFICACIÓN TIEMPO REAL – EDF

Earliest-Deadline-First (EDF)

- Las prioridades son asignadas de acuerdo al deadline.

  cuanto más cercano el deadline, mayor la prioridad;
  cuanto más tardío el deadline, menor la prioridad.

Las prioridades son DINÁMICAS.

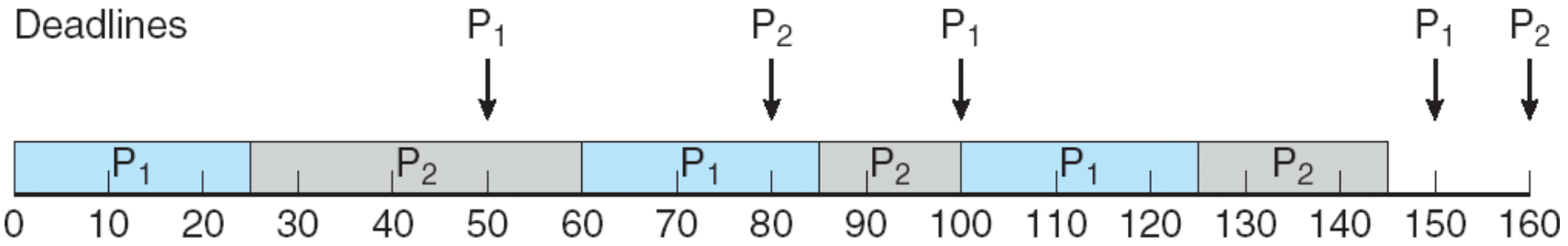
PLANIFICACIÓN TIEMPO REAL – EDF

- Por ejemplo:

  P_1: $p_1 = 50$, $t_1 = 25 – t_1/p_1 = 0{,}50$

  P_2: $p_2 = 80$, $t_2 = 35 – t_2/p_2 = 0{,}44$

  inicialmente, prior(P_1) > prior(P_2)



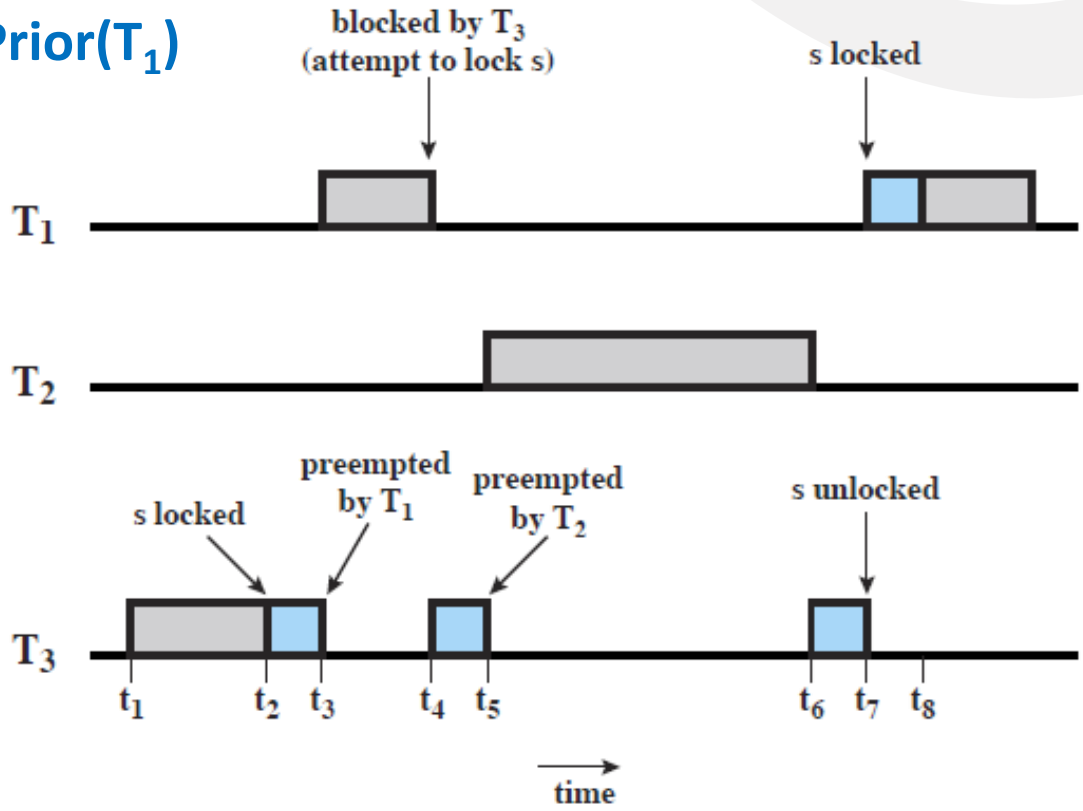
PLANIFICACIÓN TIEMPO REAL

Inversión de Prioridades

Esta situación ocurre cuando un proceso de mayor prioridad espera por un proceso de menor prioridad.

Prior(T_3) < Prior(T_2) < Prior(T_1)

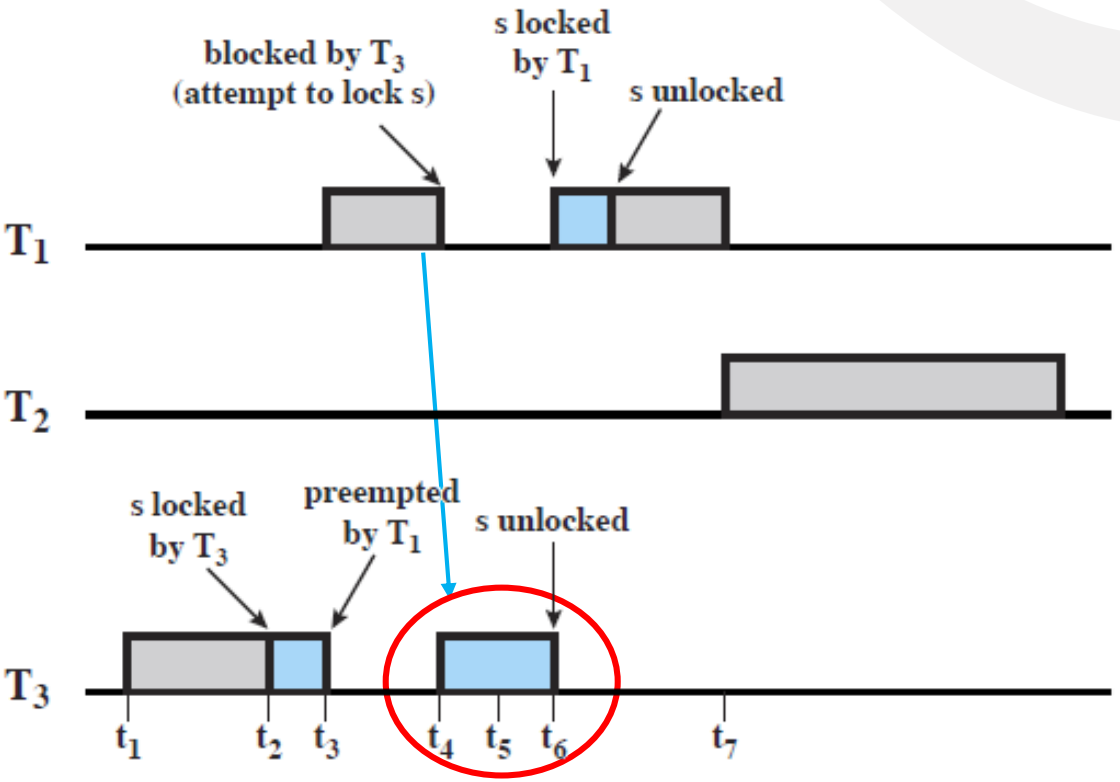
Problema: T_1 espera que termine T_3 y T_2



PLANIFICACIÓN TIEMPO REAL

Inversión de Prioridades

Una solución es **herencia de prioridades**



EJEMPLOS DE SISTEMAS OPERATIVOS

- Unix Tradicional
- Linux
- Solaris
- Windows

PLANIFICACIÓN TRADICIONAL DE UNIX

- La planificación tradicional en UNIX emplea colas multinivel (los niveles se definen en bandas de prioridades) usando Round Robin en cada una de ellas.

$$P_j(i) = \text{Base}_j + \frac{CPU_j(i)}{2} + \text{nice}_j \quad (1)$$

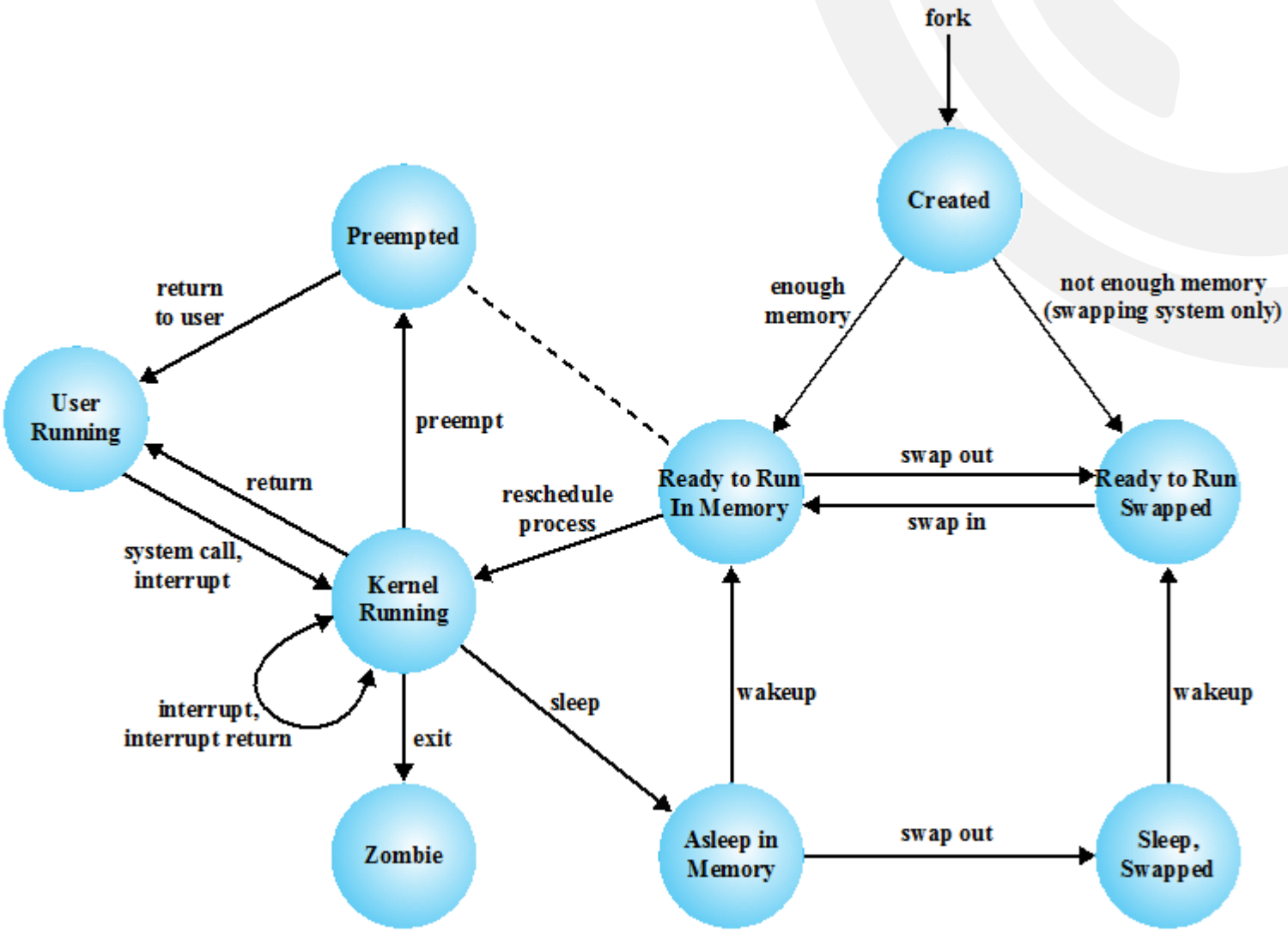
$$CPU_j(i) = \frac{CPU_j(i-1)}{2} \quad (2)$$

(1)Es utilizada para ajustar dinámicamente la prioridad (producto del uso de CPU).

(2) Es usada para implementar el "envejecimiento" cuando el proceso espera. Así evita la inanición.

- $CPU_j(i)$ = Mide la utilización del procesador por el proceso *j* en el intervalo *i*.
- $P_j(i)$ = Prioridad del proceso *j* en el comienzo del intervalo *i*; valores bajos implican prioridades altas.
- Base_j = Prioridad base del proceso *j*.
- nice_j = Factor de ajuste controlable por el usuario

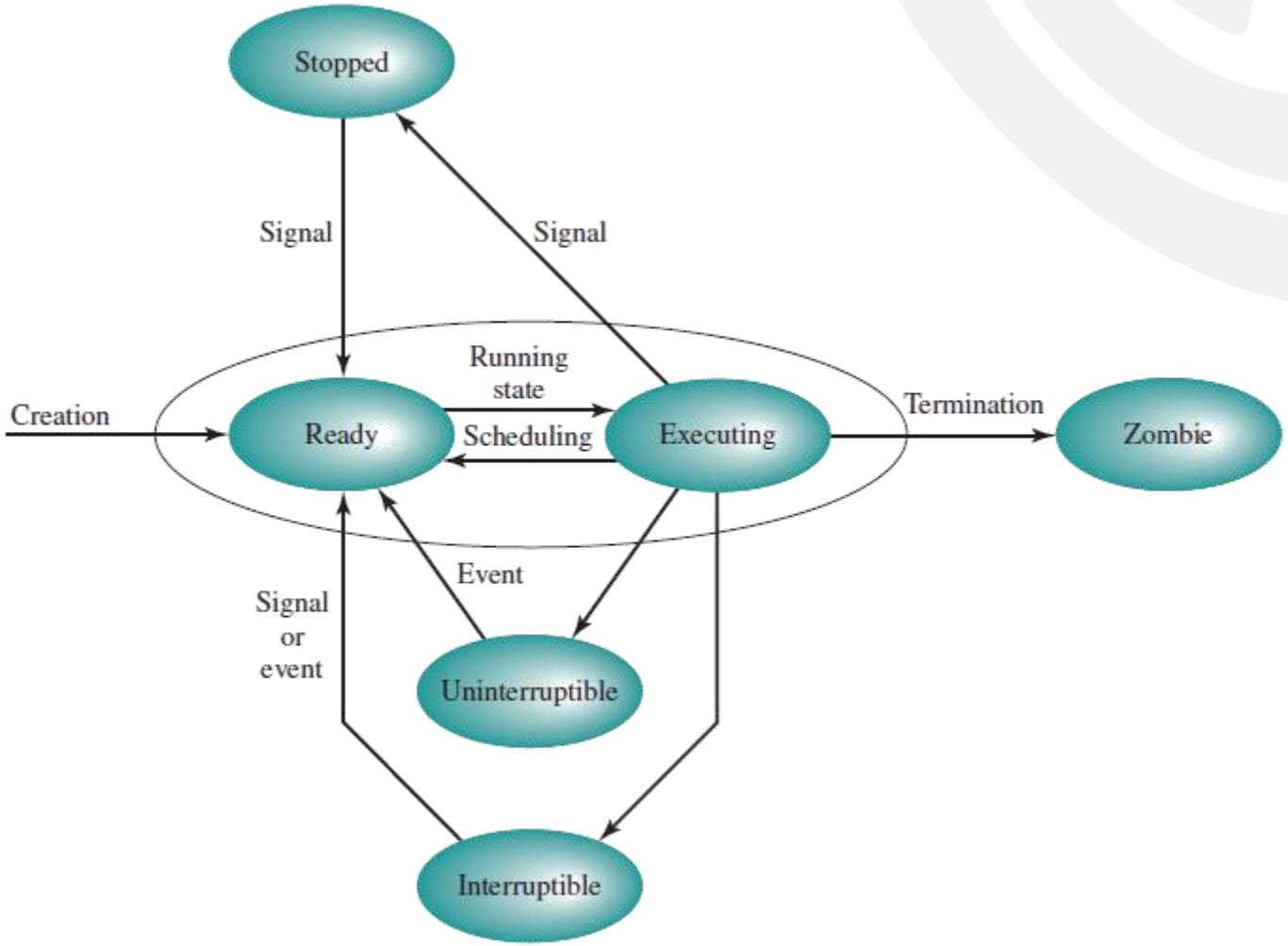
UNIX – DIAGRAMA ESTADO PROCESOS



PLANIFICACIÓN EN LINUX

- Anterior a la versión del kernel 2.5, corre una variación del algoritmo de planificación standard de UNIX
- Version 2.5 propone un tiempo de planificación constante de orden *O*(1)
  - Apropiativo, basado en prioridades. Dos rangos de prioridades: tiempo compartido y tiempo real.
  - **Tiempo Real** rango entre 0 a 99 y valor de **nice** entre 100 y 140
  - Altas prioridades obtiene un mayor q
  - Una tarea está disponible para *running* mientras le quede tiempo en su intervalo de tiempo (**active**)
  - Si no le queda tiempo (**expired**), no está disponible para *running* hasta que todas las otras tareas utilicen sus intervalos de tiempo
  - Todas las tareas en runable se mantiene la información en la estructura de datos **runqueue** por CPU
    - Dos arreglos de prioridad (active, expired)
    - Tareas indexadas por prioridad
    - Cuando no hay más tareas en active, los arreglos se intercambian
  - Funcionó bien, pero los tiempos de respuesta deficientes para los procesos interactivos

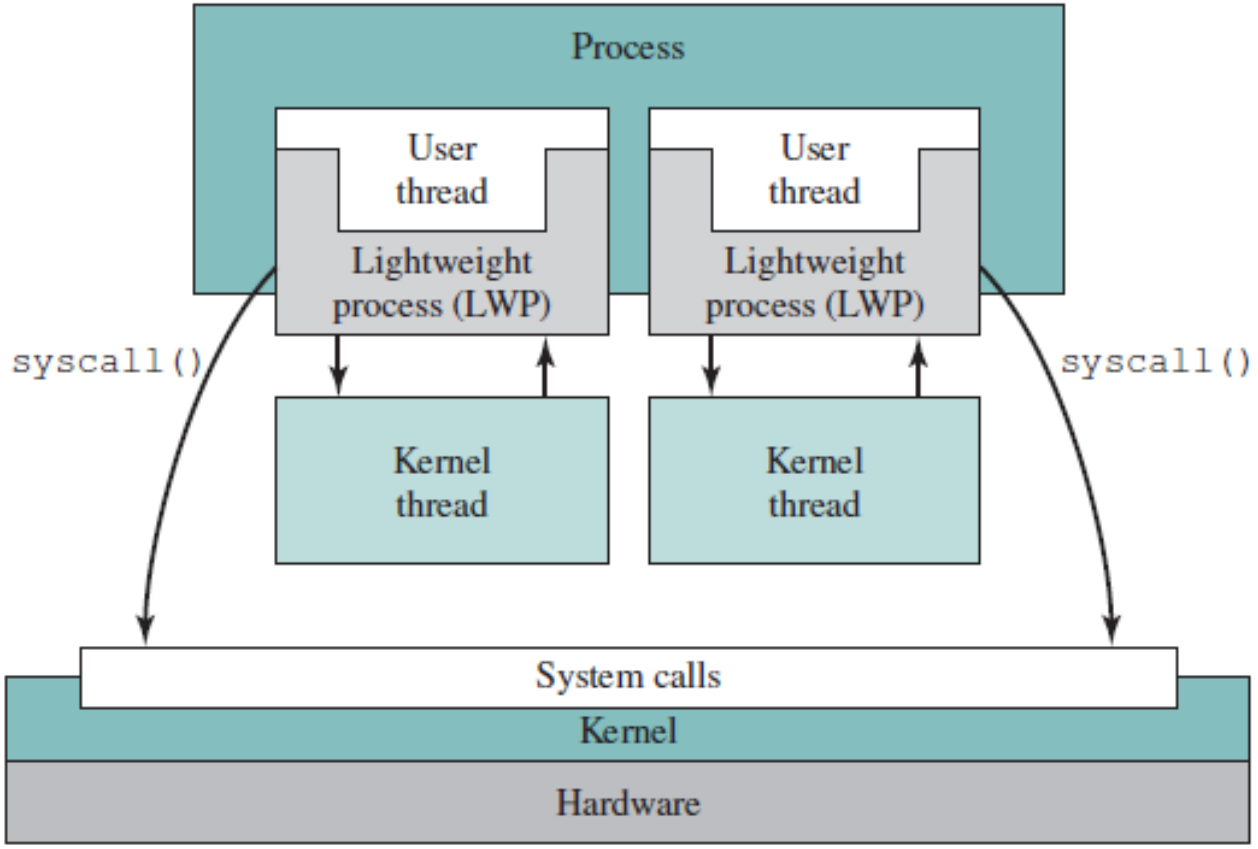
LINUX - DIAGRAMA ESTADO PROCESOS



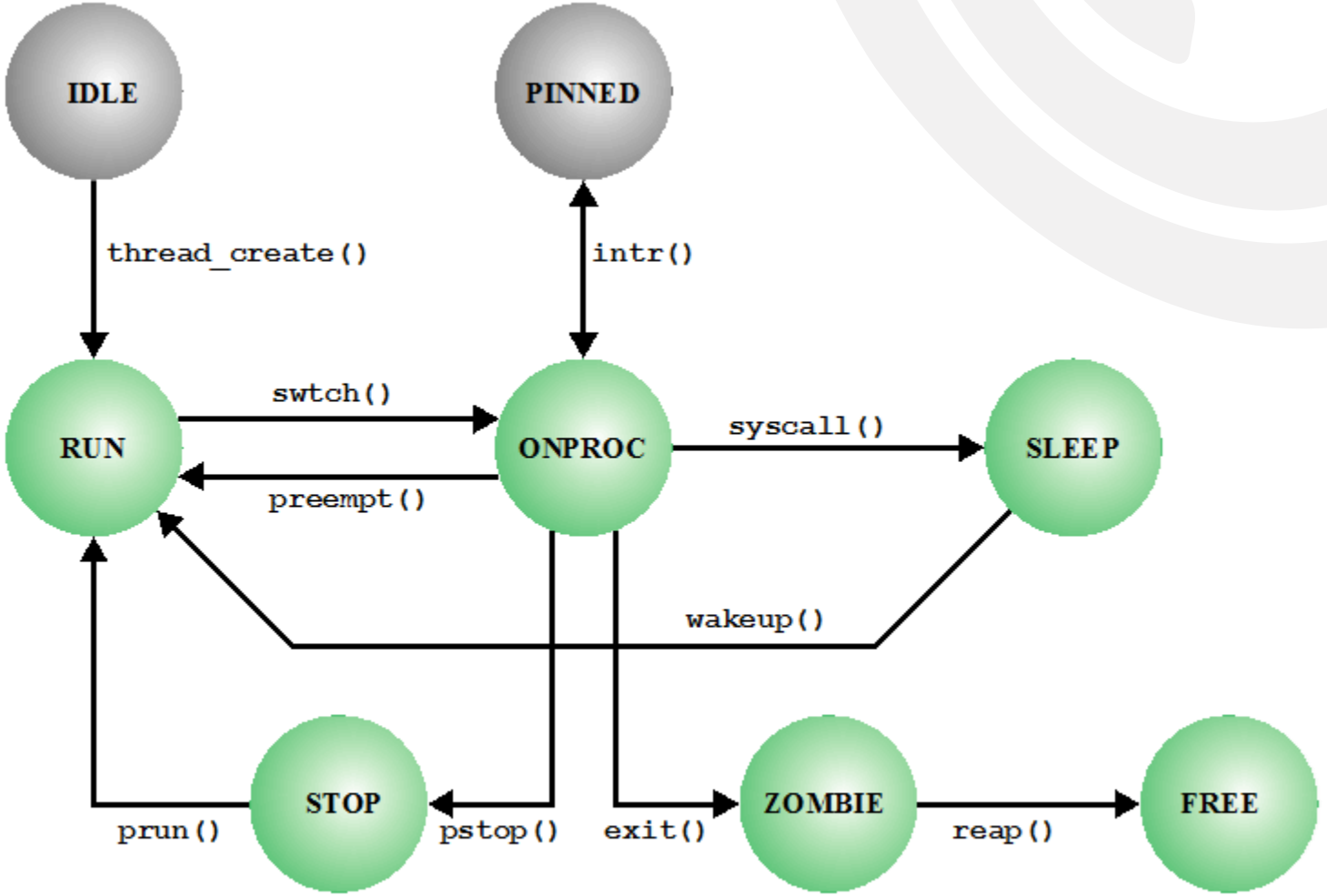
SOLARIS

Process	• Incluye el espacio de direcciones del usuario, stack y el PCB.
User-level Threads	• Unidad de ejecución creada por el usuario en el proceso.
Lightweight Processes (LWP)	• Mapeo entre ULTs y kernel threads.
Kernel Threads	• Entidad fundamental que puede ser planificada y despachada para ejecutar en uno de los procesadores del Sistema.

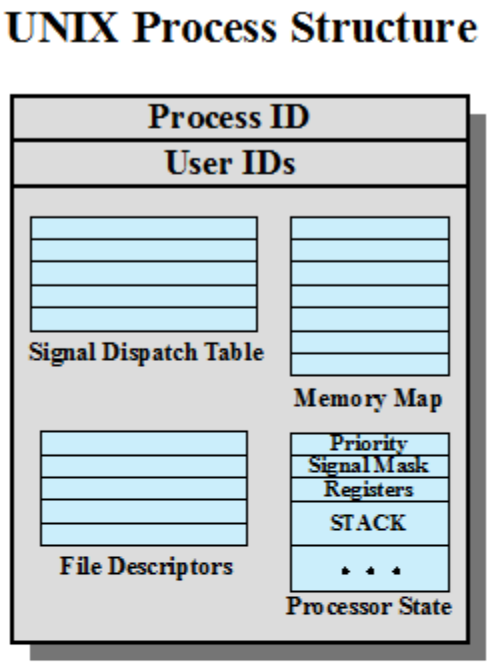
SOLARIS

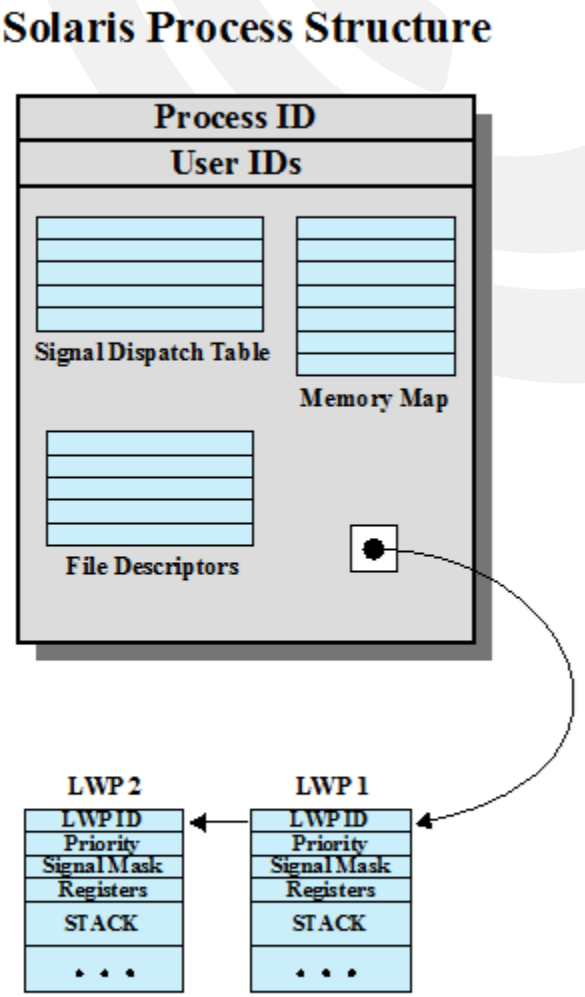

Process
User thread
Lightweight process (LWP)
User thread
Lightweight process (LWP)
syscall()
syscall()
Kernel thread
Kernel thread
System calls
Kernel
Hardware

SOLARIS - DIAGRAMA ESTADO PROCESOS

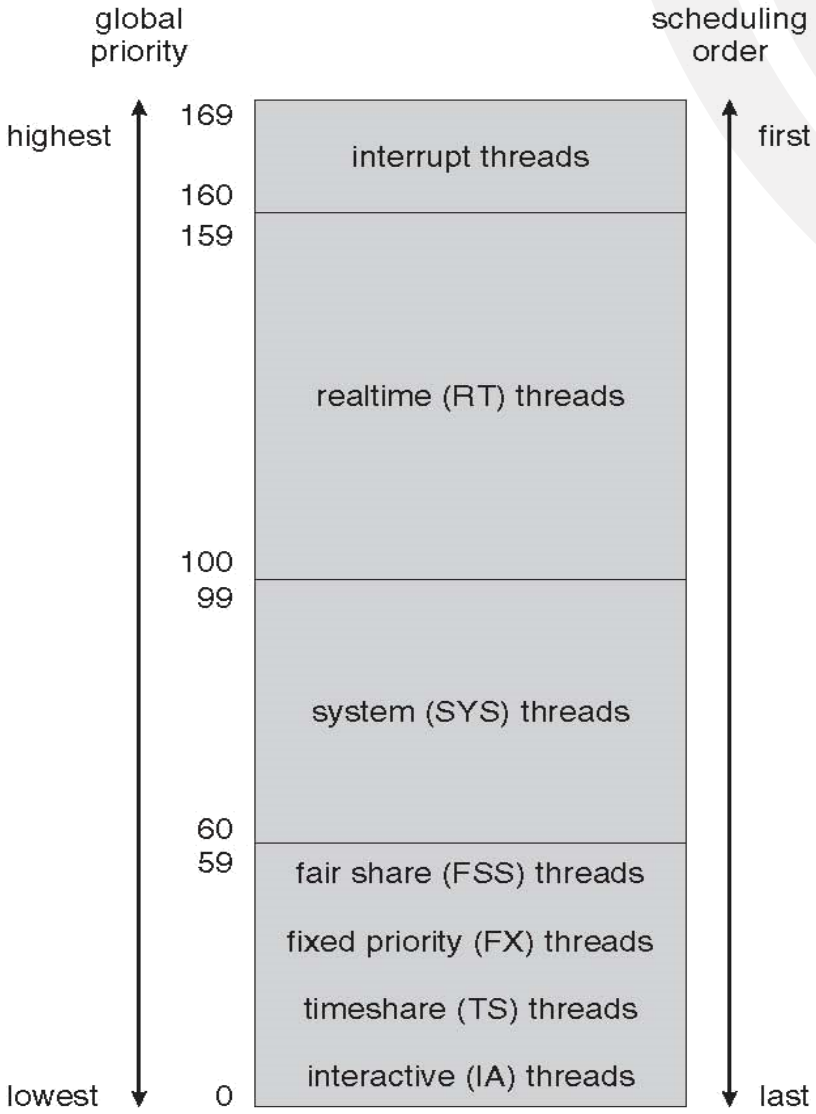


SOLARIS


UNIX Process Structure
Process ID
User IDs
Signal Dispatch Table
Memory Map
File Descriptors
Priority
Signal Mask
Registers
STACK
. . .
Processor State

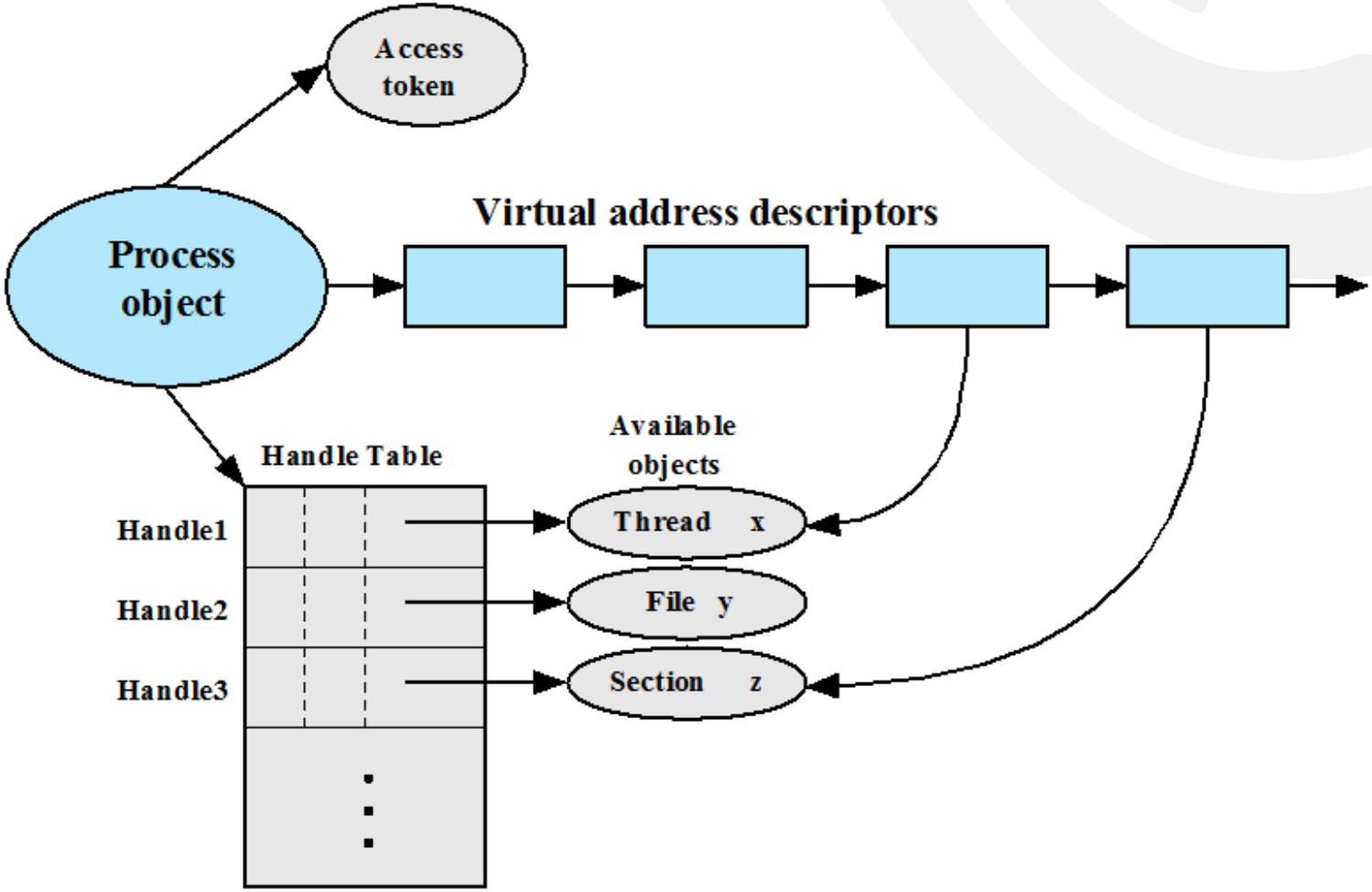

Solaris Process Structure
Process ID
User IDs
Signal Dispatch Table
Memory Map
File Descriptors
LWP 2
LWP 1
LWP ID
Priority
Signal Mask
Registers
STACK
. . .

SOLARIS - PLANIFICIÓN

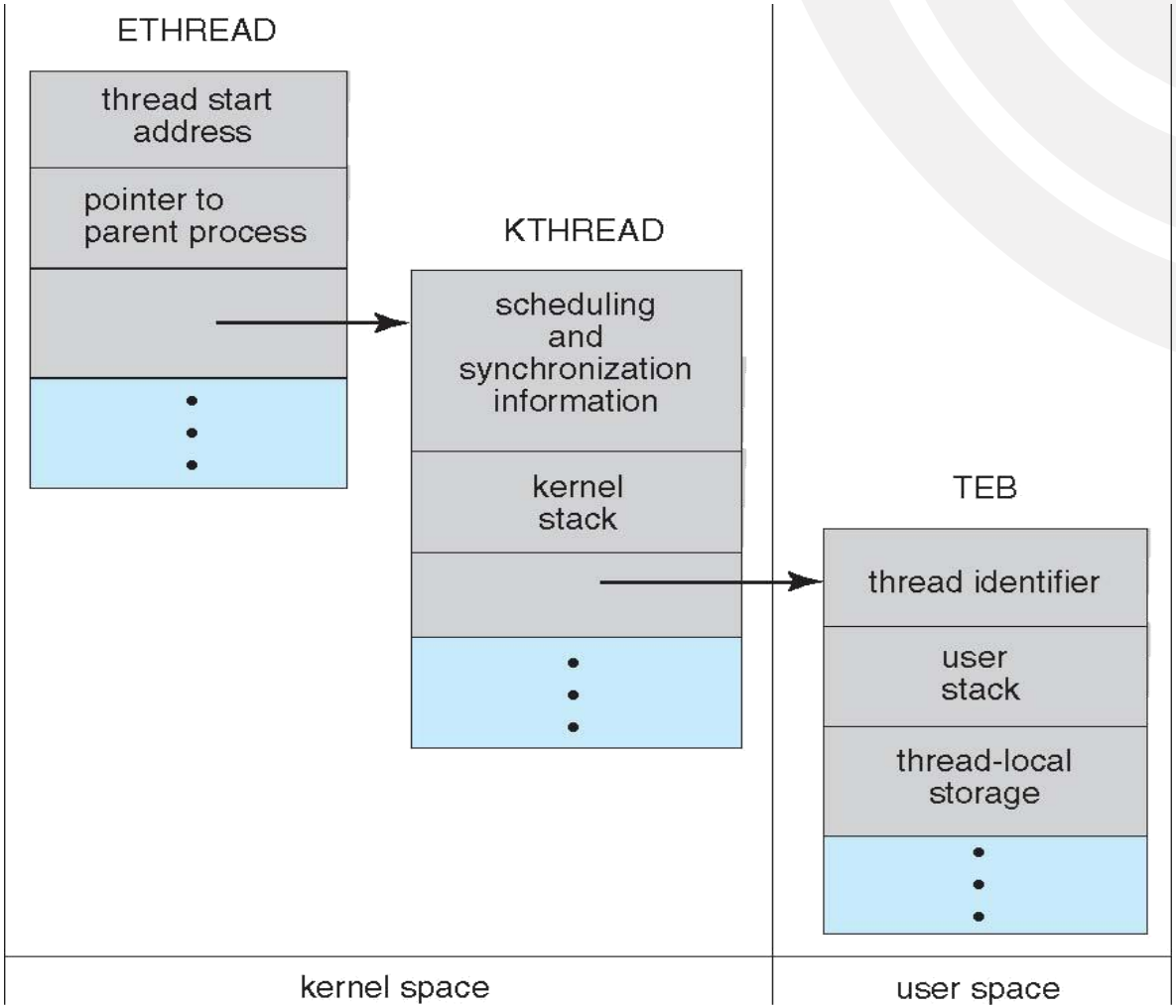


WINDOWS

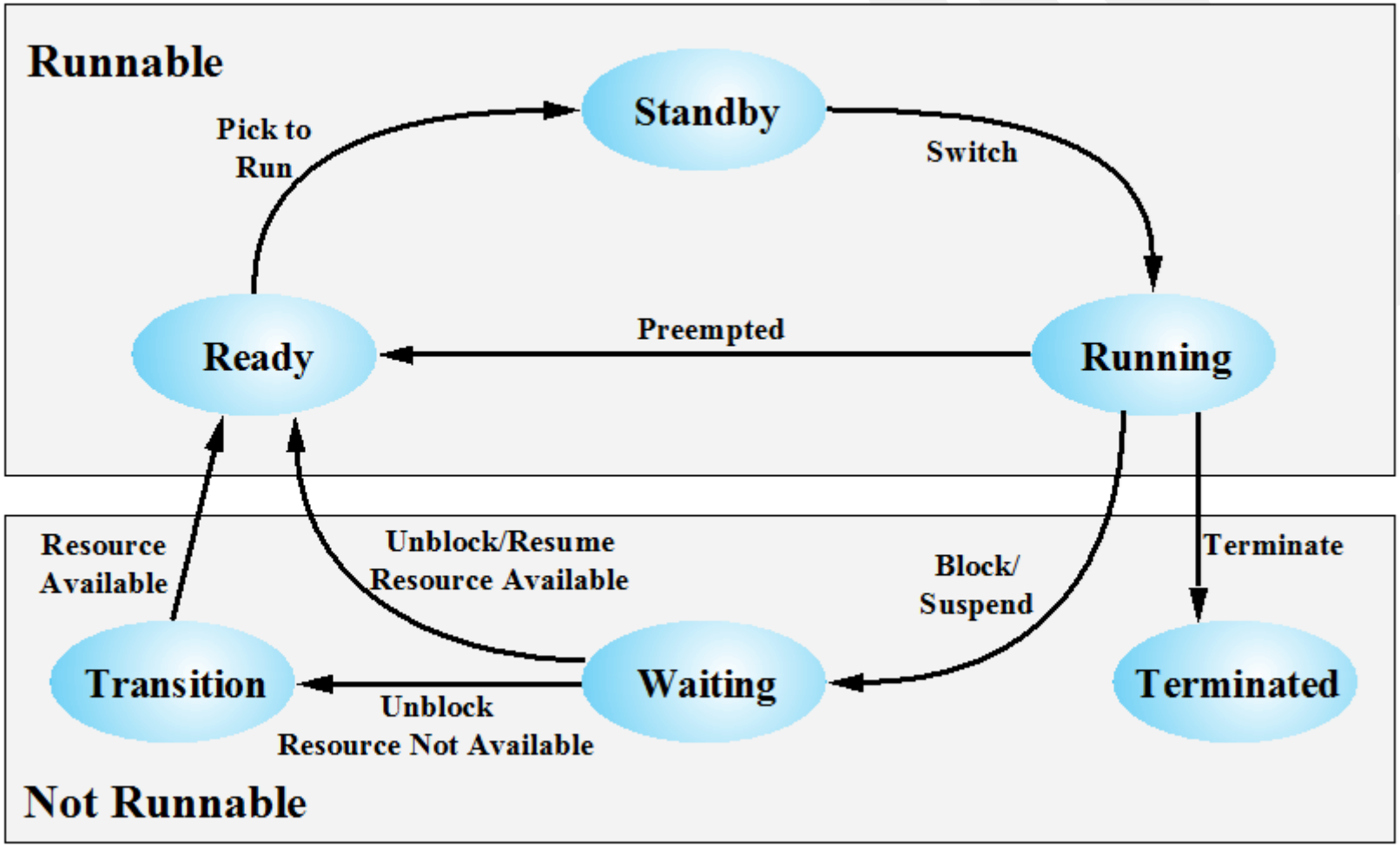
Un proceso en Windows y sus recursos



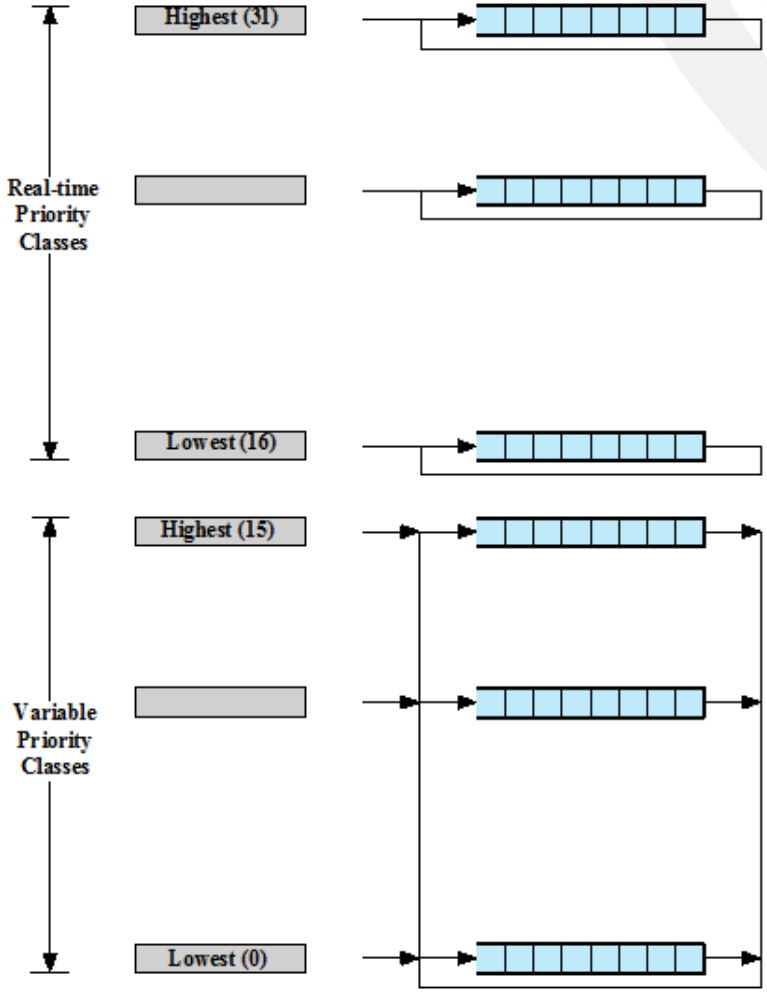
WINDOWS


ETHREAD
thread start address
pointer to parent process
KTHREAD
scheduling and synchronization information
kernel stack
TEB
thread identifier
user stack
thread-local storage
kernel space
user space

WINDOWS – DIAGRAMA DE ESTADO THREADS



WINDOWS - PLANIFICACIÓN



Bibliografía:

- Silberschatz, A., Gagne G., y Galvin, P.B.; "Operating System Concepts", 7ma Edición 2009; 9na Edición 2012; 10ma Edición 2018.
- Stallings, W. "Operating Systems: Internals and Design Principles", Prentice Hall, 6ta Edición 2009; 7maEdición 2011; 9na Edición 2018.