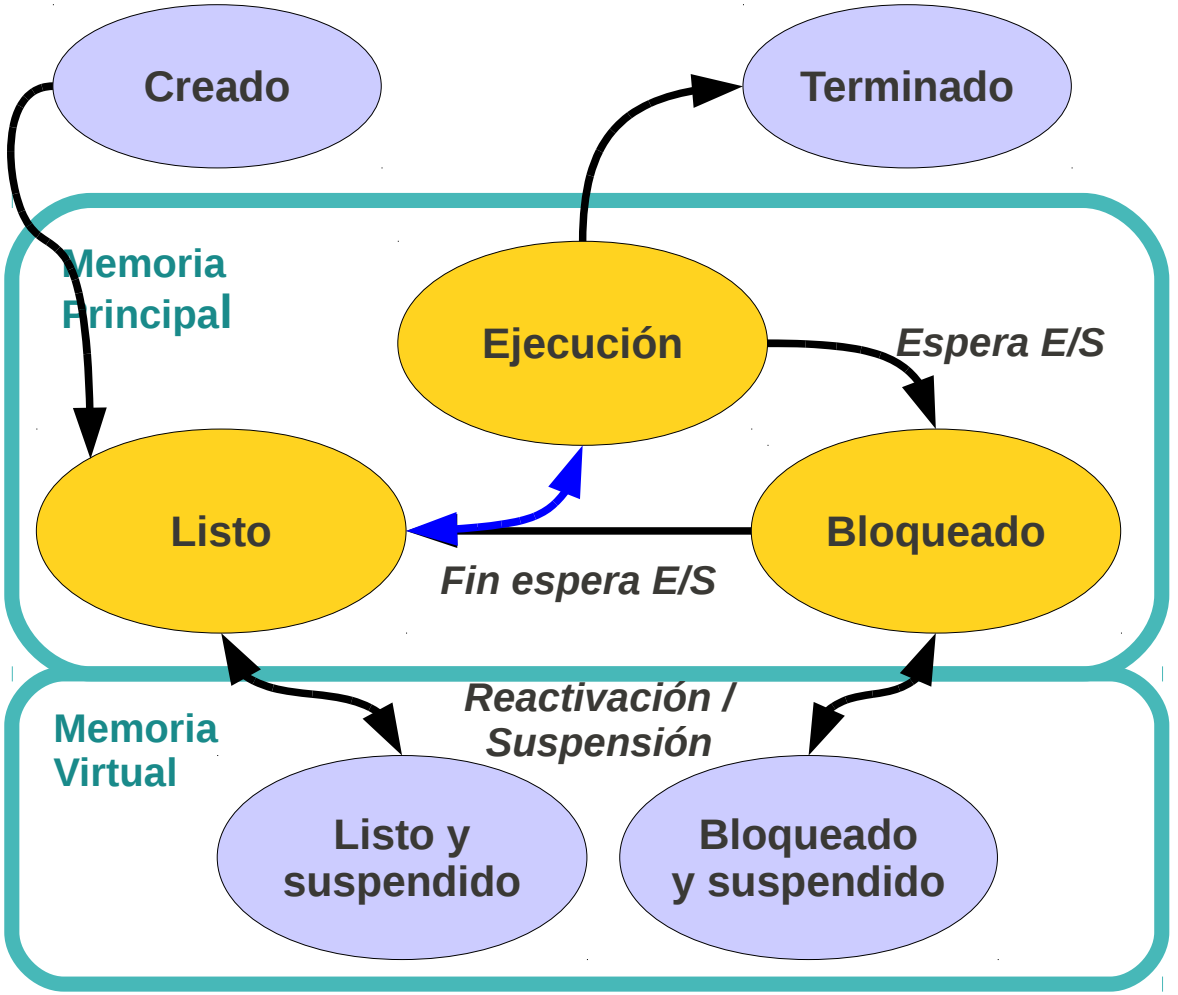
PLANIFICACIÓN DE PROCESOS
(Scheduling)

Introducción: Scheduling

Estados de un proceso



suspendido = swapeado

Introducción: Scheduling

- Conjunto de políticas y mecanismos construidos dentro del sistema operativo que gobiernan la forma de ejecutar los procesos

- Es el concepto central para los sistemas operativos multitarea
  En español: *planificación*

- El módulo del sistema operativo que lleva a cabo la planificación se denomina *scheduler* o planificador

- Permitir la ejecución “simultánea” de muchos más procesos que recursos de cómputo disponibles

- Consiste en la definición de cual proceso se ejecuta en cada momento y durante cuanto tiempo
  Seleccionar uno o más procesos de estado LISTO y ejecutarlos

- Un scheduling óptimo debería dar
  - máxima respuesta a los procesos interactivos
  - máximo *throughput* para procesos batch

Objetivos

• Diferentes formas de scheduling persiguen distintos objetivos:

- Utilización de la CPU – mantener la(s) CPU(s) en su máximo nivel de uso
- Máximo Throughput – número de procesos finalizados por unidad de tiempo
- Mínimo Turnaround – tiempo total en que un proceso se presenta y finaliza
- Mínimo Tiempo de Espera – tiempo total en que un proceso espera en LISTO
- Máximo Tiempo de Respuesta – tiempo en que una solicitud de ejecución del proceso se presenta y la primer respuesta producida (procesos interactivos)
- Equidad: Repartir las CPUs iguales cantidades de tiempo para cada proceso

- Cumplimiento con deadlines
- Prevenir postergación indefinida

Es fundamental el contexto: batch processing, interactive, real time

Tipos de planificación

- **Preemptive (apropiativo)**
  - Puede quitarse el procesador al proceso actual
  - Importante (necesaria) para procesos interactivos
  - Si un proceso se ejecuta por más de un tiempo predefinido, es suspendido
  - Requiere un *clock interrupt* para marcar los intervalos de tiempo con el fin de que el scheduler recupere el control de la CPU

- **Non-Preemptive (no apropiativo, cooperativo)**
  - Los procesos se ejecutan hasta su finalización o hasta que voluntariamente (por bloqueo esperando E/S) entrega el control de la CPU
  - Problema: Procesos “poco importantes” pueden bloquear a los “importantes”

El reloj de interrupciones

- Para que se ejecuten las actividades de planificación el sistema operativo debe apoderarse de la CPU periódicamente

- El sistema operativo gestiona un *reloj de interrupciones* que genera interrupciones cada cierto lapso de tiempo

- El reloj de interrupciones asegura que ningún proceso de usuario monopolice la CPU

- No es el reloj de la máquina o reloj hardware
  El reloj de interrupciones opera a una frecuencia mucho menor

- **Niveles de planificación**
- **Planificación a corto plazo o de la CPU**

  - ***Dispatcher*** programa que asigna la CPU a uno de los procesos de `LISTO` siguiendo un algoritmo determinado

  - El dispatcher conmuta el procesador entre dos procesos mediante un cambio de contexto

  - El dispatcher se activa cuando:
    - el proceso ejecutándose termina o no puede continuar haciéndolo
    - el proceso en ejecución agota su *quantum* de tiempo asignado

  - El dispatcher es un proceso de sistema, que para su ejecución obviamente emplea la CPU

  - Se asignan prioridades a los procesos

Niveles de planificación

• Planificación a medio plazo (Middle Term Scheduler)

- También conocido como planificador de swapping
- Sólo un número finito puede estar en memoria en cada momento
- Puede ocurrir que todos o un gran número de procesos se encuentren bloqueados
  Los procesos con posibilidad de ejecutarse deberían estar en memoria, mientras que los otros en memoria secundaria (swap space, generalmente el disco)
- El planificador o scheduler a medio plazo es el que transiciona los procesos de memoria principal a memoria secundaria
- Transiciona procesos de memoria a swap space (*swap-out*) a los procesos bloqueados y ubicando en memoria (*swap-in*) aquellos que se encuentran bloqueados sólo por no disponer de memoria
- El swapping genera sobrecarga en el sistema y generalmente se realiza cuando existe faltante de recursos (RAM)

- **Niveles de planificación**
- **Planificación a largo plazo (Long Term Scheduler)**
  - Conocido como scheduler de admisión
  - Actúa cuando se crean los procesos
  - Decide que procesos serán admitidos en la cola de procesos `LISTO`
  - Controla el grado de multiprogramación en los sistemas multitasking regulando la cantidad máxima de procesos ejecutándose concurrentemente en cada momento
  - Se encarga de ingresar y finalizar a los procesos

Algoritmos de planificación

- Simplificando, un proceso puede considerarse como una serie de ciclos de ejecución en CPU y tiempos de espera por E/S

- Los procesos alternan entre estos dos estadíos
  Proceso (ráfagas variables):
  Ráfaga de CPU – Ráfaga E/S - Ráfaga de CPU – Ráfaga E/S - Ráfaga de CPU ...

- Para estudiar los algoritmos haremos algunas suposiciones en aras de la simplicidad de los procesos
  - No hay cambios de contexto
  - La CPU empleada por el S.O. es despreciable
  - Los procesos sólo consisten en ráfagas de CPU
  - Métrica: Tiempo de Espera promedio

$$\frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2$$

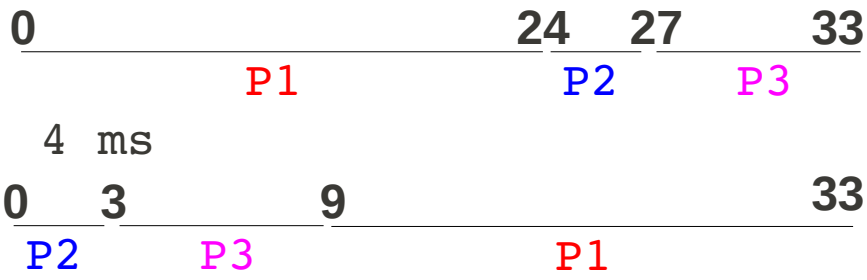
5, 4, 6, 5 Promedio=5, Varianza=0,5
1, 8, 7, 4 Promedio=5, Varianza=7,5

First-Come First-Serve (FCFS)

- Servicio por orden de llegada
- Se implementa con una cola FIFO
- El tiempo de espera promedio suele ser largo

Proceso	Tiempo (ms)
P1	24
P2	3
P3	6

Orden de llegada	T. Espera Promedio
P1, P2, P3	27 ms
P2, P3, P1	4 ms



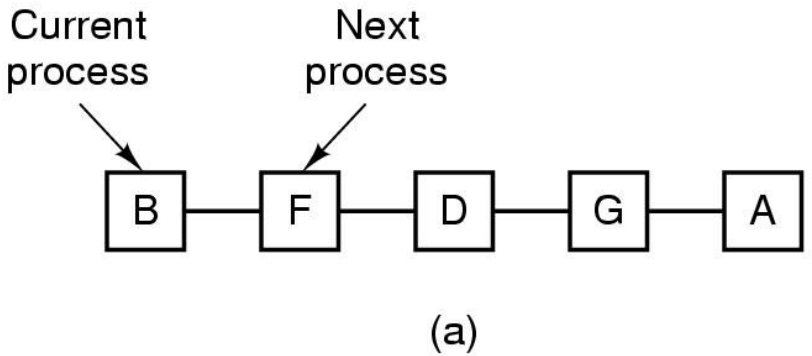
- Pros & cons
  - Simple. Fácil de entender y programar
  - Inútil para ambientes interactivos

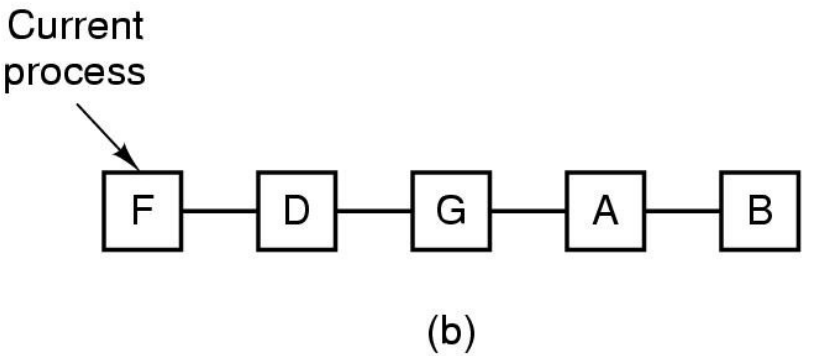
Shortest Job First (SJF)

- También es *non-preemptive*
- Se ejecutan los procesos en orden ascendente según el tiempo de ejecución de cada uno
- Se asigna la CPU al proceso más corto, en caso de “empate” se usa FCFS
- Es óptimo con respecto al tiempo de espera promedio mínimo para un conjunto de procesos
- Apropiado para procesos batch donde los tiempos de ejecución se conocen de antemano
- Ofrece un límite teórico (para el tiempo de espera promedio) frente al cual pueden compararse otros algoritmos

Round Robin

- Algoritmo *Preemptive*, apto para procesos interactivos
- Algoritmo simple, justo y ampliamente usado
- A cada proceso se le asigna un intervalo de tiempo llamado *quantum*
- Si un proceso continúa su ejecución superando su quantum, es *preempted* y la CPU es dada a otro proceso
- Si un proceso se bloquea o finaliza ante de que expire el quantum, la CPU se re-asigna en ese momento
- Cuando un proceso usa todo su quantum se lo coloca al final de la lista de procesos listos – suposición: todos los procesos tienen la misma importancia





Round Robin

|quantum| = X ms, ¿X?

- El cambio de contexto consume tiempo, supongamos 5 ms (1000ms = 1 segundo)

Si |quantum| = 20 ms ⇒ 25% de CPU para el sistema

Si |quantum| = 500 ms ⇒ 1% de CPU para el sistema

```
 1er usuario espera   500 ms
 2do usuario espera  1000 ms
 3er usuario espera  1500 ms
 ...
 9no usuario espera  4500 ms
10mo usuario espera  5000 ms       5 segundos!!!
```

- Un valor aceptado es 100 ms

- En muchos sistemas operativos es variable

Algoritmo basado en prioridades

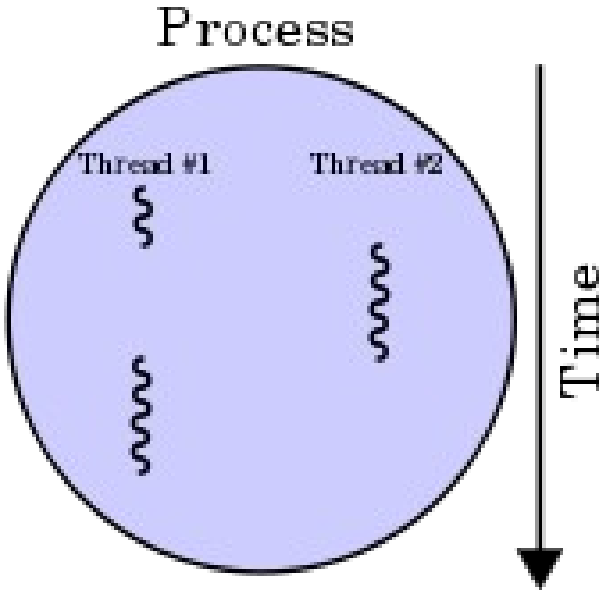
- A cada proceso en el sistema se le asigna un entero basándose en algún criterio
- Generalmente, un valor numérico bajo indica alta prioridad
- Se favorece a los procesos de alta prioridad al momento de asignar la CPU
- No apunta a reducir el tiempo de espera promedio general
  Los procesos importantes serán atendidos sin caer en retardos innecesarios
- Las prioridades pueden ser estáticas o dinámicas

Prioridades estáticas vs dinámicas

- **Prioridades estáticas**
  - La prioridad asignada a un proceso no cambia mientras existe
  - Simple de implementar (similar a asignar un PID)
  - No sobrecarga al sistema
  - Problema: Starvation, un proceso de baja prioridad se puede ver perjudicado
    http://en.wikipedia.org/wiki/Resource_starvation

- **Prioridades dinámicas**
  - La prioridad cambia durante la ejecución del proceso
  - Priority aging: aumentar la prioridad de los procesos que esperan mucho tiempo
    http://en.wikipedia.org/wiki/Aging_%28scheduling%29

Caso Solaris 10



Kernel thread model

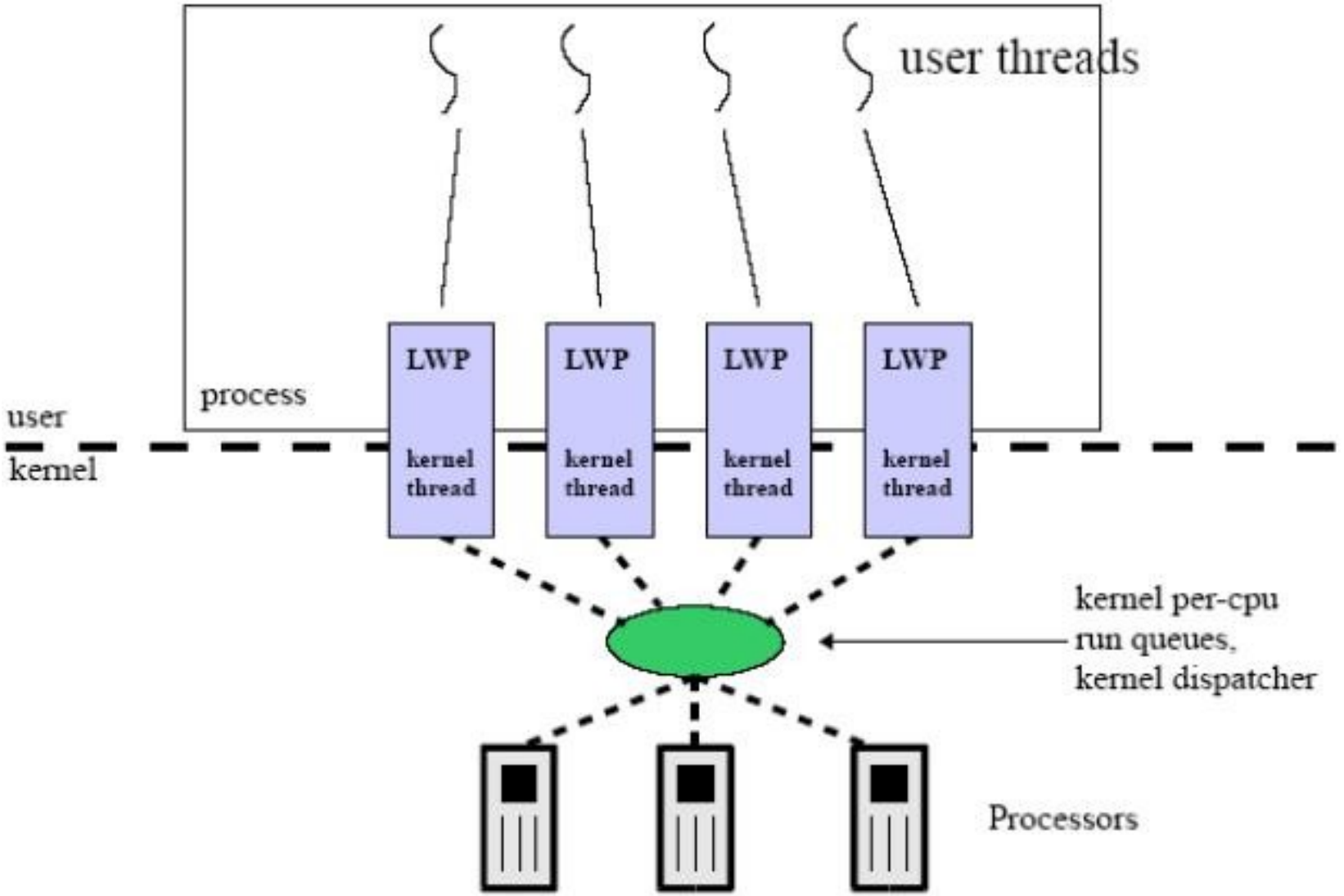
- **kernel threads**
  Es lo que es planificado/ejecutado en un procesador

- **user threads**
  Hilo de ejecución de un proceso de usuario

- **process**
  El objeto que representa el entorno de ejecución de un programa

- **lightweight process (lwp)**
  Contexto de ejecución para un thread de usuario.
  Asocia un user thread con un kernel thread.

- Caso Solaris 10



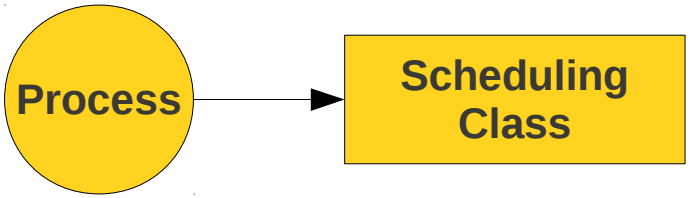
Caso Solaris 10

- El kernel de Solaris es *fully preemptible*.
  Todos los threads (incluyendo los que soportan las actividades del kernel) pueden ser interrumpidos para dar lugar a un thread de mayor prioridad

- Todo proceso de usuario es controlado por una *clase de scheduling*

- Cada clase de scheduling asigna una prioridad a los procesos que administra

- Las clases conforman un rango global de prioridades y determinan el orden de ejeución de los procesos



- Solaris 10 reconoce 170 prioridades diferentes (0-169)
  Estas prioridades de distribuyen en intervalos que corresponden a las distintas clases de scheduling.

Clases de Scheduling

TS (Timeshare)

- Clase por defecto para los procesos de usuario y sus kernel threads asociados
- Ajusta las prioridades en el tiempo
- Los procesos aquí usan time-slicing
- Rango de prioridades: 0-59

IA (Interactive)

- Similar a TS que prioriza a los procesos activos en ambiente gráfico.
- Rango: 0-59

SYS (System)

- Empleada para planificar los kernel threads
- No se emplea time-slicing, los procesos inician y se terminan o bloquean
- Prioridades fijas
- sched, pageout, fsflush
- Rango de prioridades: 60-99

RT (Real Time)

- Los procesos de esta clase tienen la máxima prioridad, excepto las threads de interrupciones
- Time-slicing
- Quantum fijo
- Rango: 100-159

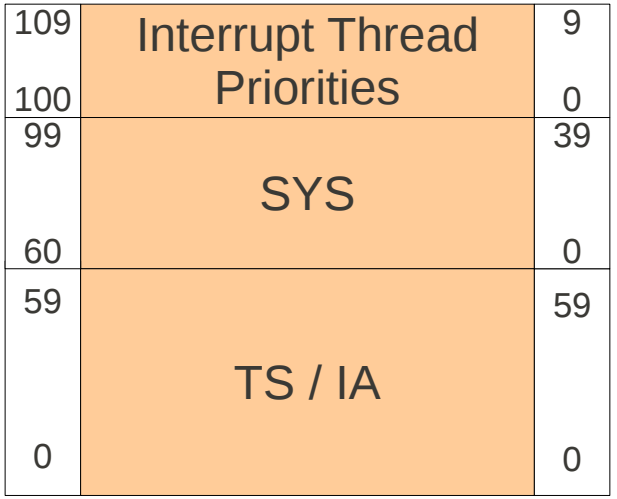
FX (Fixed Priority)

- Los procesos de esta clase tienen prioridades fijas
- Rango: 0-59

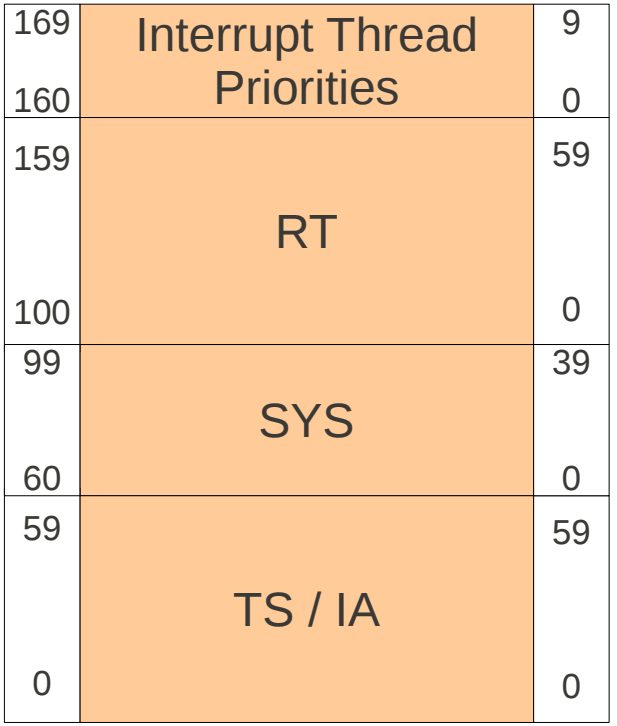
Instituto Politécnico Superior

Caso Solaris 10- Comandos

- La clase RT no se carga por defecto



Sin clase RT habilitada



Con clase RT habilitada

Caso Solaris 10- Comandos

- **dispadmin**
  - Permite inspeccionar y cambiar los parámetros de planificación para las clases TS, FX, FSS (Fair Share Scheduler) y RT.

  - Mostar los parámetros configurados de una clase de scheduling

  - Mostrar y cambiar la clase de scheduling por defecto

```
bash-3.00# dispadmin -l
CONFIGURED CLASSES
==================

SYS       (System Class)
TS        (Time Sharing)
FX        (Fixed Priority)
IA        (Interactive)
```

```
# dispadmin -l
# dispadmin -d
# dispadmin -d TS
# dispadmin -d
# cat /etc/dispadmin.conf
```

Tabla de planificación

• Comando

```
# dispadmin -c TS -g -r 1000  (quantums en ms)
```

• Cabeceras de la tabla

ts_quantum: quantum asignado por defecto al proceso según su prioridad
ts_tqexp: prioridad asignada al proceso cuando consume todo su quantum
ts_slpret: prioridad asignada a un proceso que se bloquea antes de consumir su quantum
ts_maxwait: si el proceso no recibe CPU luego de un intervalo de tiempo ts_maxwait, su prioridad es elevada a **ts_lwait**
PRIORITY LEVEL: indica el nivel de prioridad de estos parámetros.

Caso Solaris 10- Comandos

- **priocntl**
  - Administra los parámetros de scheduling para los procesos.
  - Lista las clases de scheduling cargadas
  - Muestra los parámetros de scheduling de un proceso
  - Asigna una clase de scheduling a un nuevo proceso