Auxiliar 6: Estrategias de Scheduling

CC4302 - Sistemas Operativos
José Astorga

Conceptos Importantes

- Procesos livianos (threads) vs procesos pesados (procesos Unix)
- Preemptiveness
- Interrupciones y el timer
- Estados de un proceso
- El descriptor de proceso
- Cambio de contexto
- Colas de scheduling
- Cambios de contexto implícito vs. cambio de contexto explícito (cuando el cambio de contexto ocurre producto de una acción del mismo proceso, y no producto de una interrupción del timer o del disco)
- El identificador de proceso (pid)
- Ráfagas de CPU

Estrategias de Scheduling:

- First Come First Served
- Shortest Job First
- Prioridades
- Round Robin

Pueden ser de tipo preemptive o non-preemptive.

Estrategias de Scheduling:

Estrategia	Pros	Cons
First Come First Served		
Shortest Job First		
Prioridades		
Round Robin		

Estrategias de Scheduling:

Estrategia	Pros	Cons
First Come First Served	Simpleza	Mal tiempo de despacho promedio. No sirve para sistemas interactivos
Shortest Job First		
Prioridades		
Round Robin		

Estrategias de Scheduling:

Estrategia	Pros	Cons
First Come First Served	Simpleza	Mal tiempo de despacho promedio. No sirve para sistemas interactivos
Shortest Job First	Menor tiempo de despacho promedio	Hambruna para los procesos intensivos en CPU
Prioridades		
Round Robin		

Estrategias de Scheduling:

Estrategia	Pros	Cons
First Come First Served	Simpleza	Mal tiempo de despacho promedio. No sirve para sistemas interactivos
Shortest Job First	Menor tiempo de despacho promedio	Hambruna para los procesos intensivos en CPU
Prioridades	Se atienden procesos “importantes” primero.	Hambruna. (Se puede solucionar aumentando prioridad de procesos en estado READY)
Round Robin		

Estrategias de Scheduling:

Estrategia	Pros	Cons
First Come First Served	Simpleza	Mal tiempo de despacho promedio. No sirve para sistemas interactivos
Shortest Job First	Menor tiempo de despacho promedio	Hambruna para los procesos intensivos en CPU
Prioridades	Se atienden procesos “importantes” primero.	Hambruna. (Se puede solucionar aumentando prioridad de procesos en estado READY)
Round Robin	Minimiza el tiempo de respuesta	Muchos cambios de contextos implícitos (se soluciona fijando la el tamaño de la tajada de manera adecuada)

Diagrama de scheduling

El diagrama muestra las decisiones de scheduling para 3 procesos. A la izquierda se indica para cada proceso las duraciones de sus ráfagas de CPU y entre paréntesis las duraciones de sus estados de espera.

La línea punteada indica que el estado del proceso es READY. Si el proceso está en estado de espera el espacio aparece en blanco.

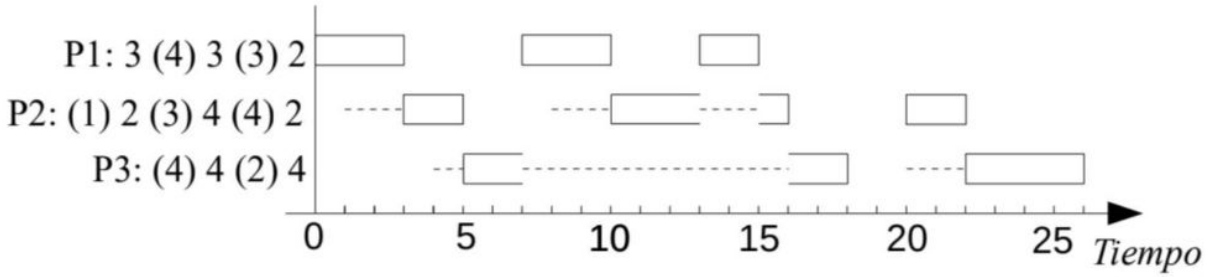


Diagrama de scheduling

¿De qué estrategia de scheduling se trata?

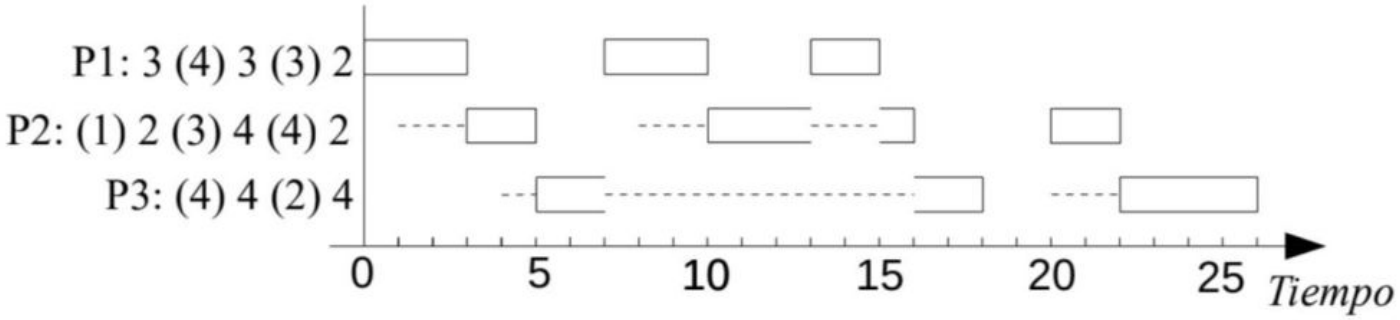
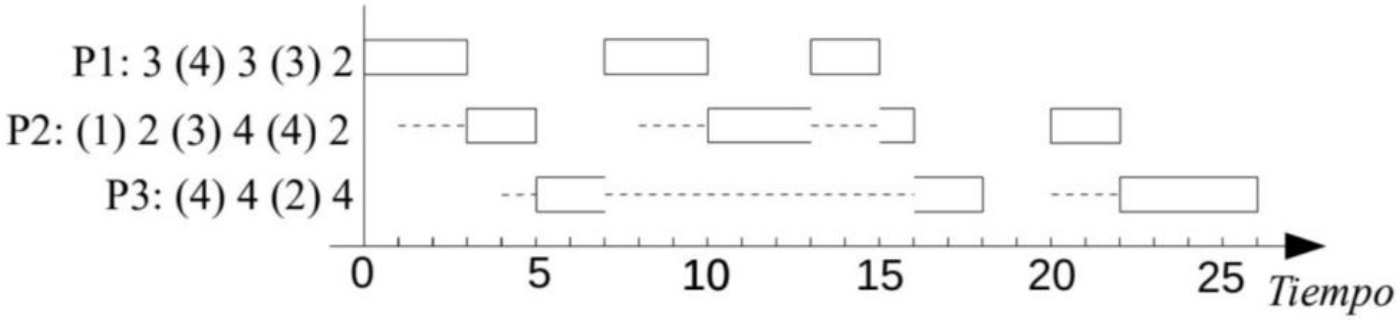


Diagrama de scheduling

¿De qué estrategia de scheduling se trata?



- Preemptive
- Estrategia de prioridades

Programar Estrategia de Scheduling: Prioridades para 1 core

Prioridades para 1 core

- Añadiremos una estrategia de scheduling por prioridades para una máquina de 1 core en el núcleo de nano system.

- Lo programaremos en el archivo nKernel/sched-pri-1core.c

Partes de Scheduler

- Cola Ready

```
static NthQueue *nth_pri1ReadyQueue[MAXPRI];
```

- Setear prioridad a un thread

```
static void nth_pri1SetReady(nThread th) {
  CHECK_CRITICAL("nth_fcfsSetReady")

  if (th->status==READY || th->status==RUN)
      nFatalError("nth_fcfsReady", "The thread was already in READY status\n");

  th->status= READY;
  nth_putBack(nth_pri1ReadyQueue[th->pri], th);
}
```

Partes de Scheduler

- Pasar un thread a estado READY

```
static void nth_pri1SetReady(nThread th) {
  CHECK_CRITICAL("nth_fcfsSetReady")

  if (th->status==READY || th->status==RUN)
      nFatalError("nth_fcfsReady", "The thread was already in READY status\n");

  th->status= READY;
  nth_putBack(nth_pri1ReadyQueue[th->pri], th);
}
```

Partes de Scheduler

- Pasar un thread a estado de espera

```
static void nth_pri1Suspend(State waitState) {
  CHECK_CRITICAL("nth_fcfsSuspend")

  nThread th= nSelf();
  if (th->status!=RUN && th->status!=READY)
      nFatalError("nth_fcfsSuspend", "Thread was not ready or run\n");
  th->status= waitState;
}
```

Partes de Scheduler

- Ahora el Scheduler mismo:

```
static void nth_pri1Schedule(void) {
      ...

}
```

Función que se llama cada vez que hay cambios en la cola ready (ej: se agrega un nuevo thread). Acá se determina cuál es el siguiente thread a ejecutarse.