Gestión de procesos y señales

Planificación

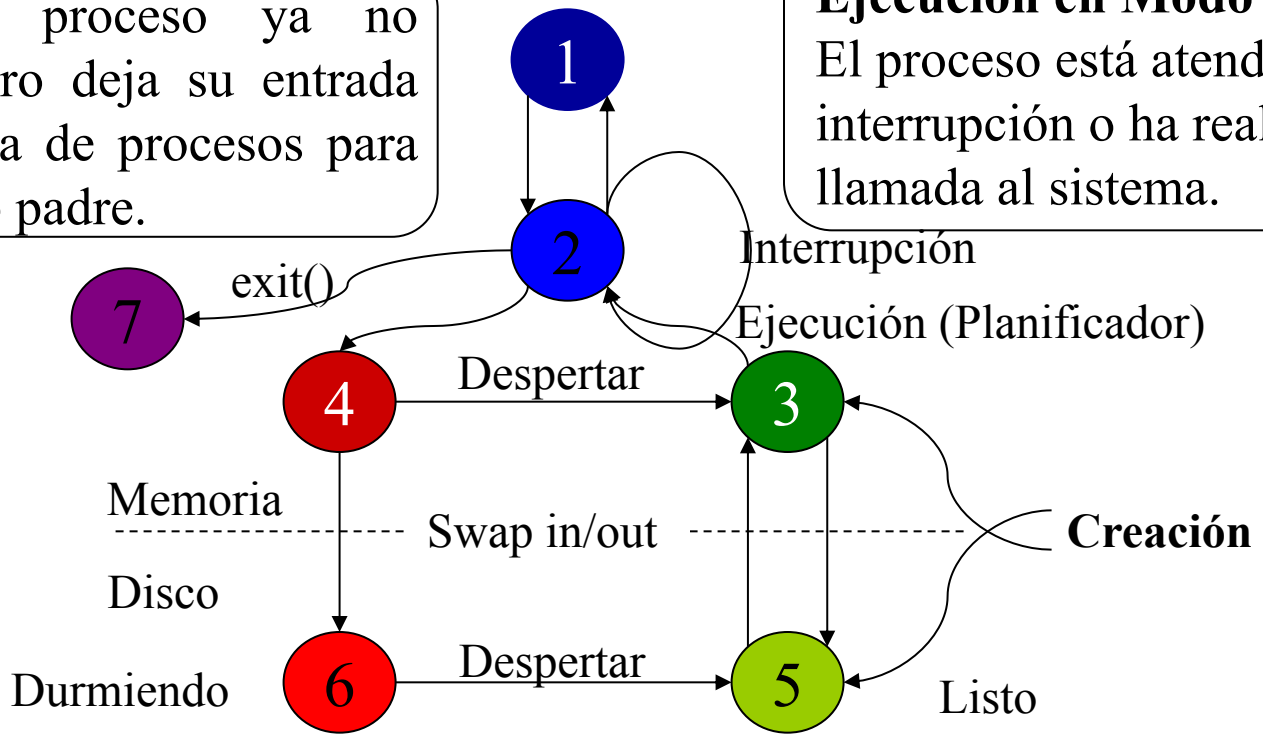
Autores:
•Rubén S. Montero (rsmontero@ucm.es)
•Ignacio M. Llorente (imllorente@ucm.es)
•José Ignacio Gómez (jigomez@ucm.es)

Estado de un Proceso

Ejecución en Modo Usuario:El proceso está activo y ejecutándose.

Zombi:El proceso ya no existe, pero deja su entrada en la tabla de procesos para el proceso padre.

Ejecución en Modo Supervisor: El proceso está atendiendo a una interrupción o ha realizado una llamada al sistema.



Durmiendo: El proceso está esperando a que termine una operación de E/S

Espera: El proceso está listo para ejecutarse, esperando que quede libre la CPU

Planificación de un Proceso

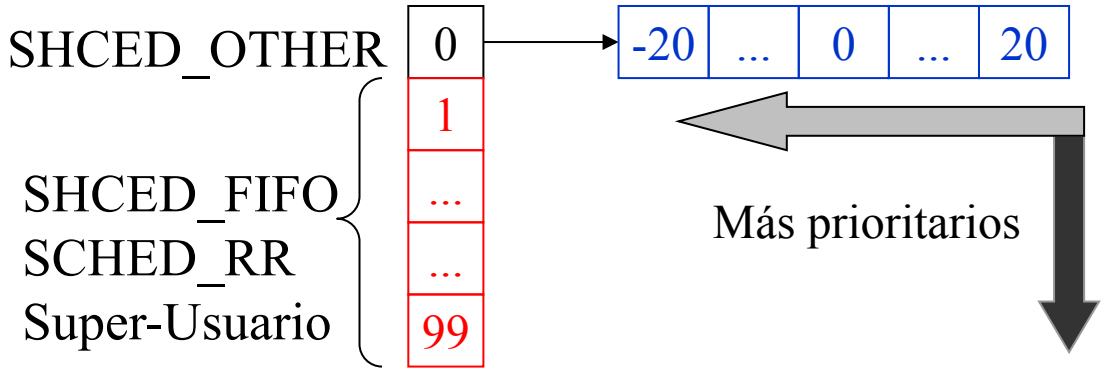
Planificador: Algoritmo del núcleo que determina el orden de ejecución de los procesos en función de la prioridad estática (modificable sólo mediante llamadas al sistema) y dinámica del proceso (modificable también mediante comandos, nice, renice, ...). El sistema es expropiativo

<sched.h>

POSIX

- **SCHED_OTHER:** Política estándar, la prioridad estática es 0. La prioridad dinámica se determina mediante el *nice* del proceso.
- **SHCED_FIFO:** Prioridad estáticas mayores que 0, siempre expriopiará a los procesos de la clase anterior.
- **SHCED_RR:** Igual que la anterior, pero a cada proceso se le aplica un cuanto de tiempo de ejecución.

Procesos críticos en tiempo



Planificación de un Proceso

Funciones para **fijar** y **consultar la política de planificación**:

- `int sched_setscheduler(pid_t pid, int policy, const struct sched_param *p);`
- `int sched_getscheduler(pid_t pid);`

<sched.h>
POSIX

```
struct sched_param{
...
int sched_priority;
...
};
```

- `pid:` Identificador del proceso, `pid=0` para el proceso que realiza la llamda
- `policy:` Política de planificación del proceso.
- `p:` Sirve para fijar la nueva prioridad.

- Un proceso creado con `fork()` **hereda** las propiedades del **planificador**
- Si un proceso con una prioridad >1, entra en un bucle infinito **bloqueará los procesos con menor prioridad para siempre**.

Planificación de un Proceso

Funciones para **fijar** y **consultar** los **parámetros del planificador**:

- `int sched_setparam(pid_t pid, const struct sched_param *p);`
- `int sched_getparam(pid_t pid, const struct sched_param *p);`

<sched.h>
POSIX

- `pid`: Identificador del proceso, `pid=0` para el proceso que realiza la llamada
- `p`: Sirve para fijar la nueva prioridad.

Funciones para **consultar** el **rango de prioridades estáticas**:

- `int sched_get_priority_max(int policy);`
- `int sched_get_priority_min(int policy);`

- `policy`: Especifica el planificador

Planificación de un Proceso

Códigos de error asociados a las funciones del planificador:

- **ESRCH:** No se pudo encontrar el proceso especificado por `pid`.
- **EPERM:** El proceso no tiene los privilegios adecuados. El uid del proceso que realiza la llamada a `sched_setscheduler()`, ha de ser igual al uid del proceso especificado por `pid`. Sólo el super-usuario puede establecer las políticas de "tiempo-real".
- **EINVAL:** La política indicada no es válida, o la nueva prioridad no es válida, min< priority < max.

Planificación de un Proceso

Funciones para **fijar** y **consultar** la **prioridad de planificación** de un proceso SCHED_OTHER:

`<sys/time.h>`

SV+BSD

- `int getpriority(int which, int who);`
- `int setpriority (int which, int who, int prio);`

- **`which`**: Hace referncia a la clase que verá modificada su prioridad PRIO_PROCESS, PRIO_PGRP ó PRIO_USER.
- **`who`**: En función de la clase es un identificador de un proceso, un identificador de un grupo de procesos ó un identificador de usuario.
- **`prio`**: Valor de la nueva prioridad (0 por defecto) → 20>`prio`>-20, valores más bajos corresponden a mayor prioridad.

Dado que `getpriority` puede devolver valores -1, para ver si se ha producido un error hay que consultar `errno`:

- **`ESRCH`**: No se encontró el proceso,
- **`EINVAL`**: Valor no válido en `which`.
- **`EPERM`**: El ID efectivo no permite modificar la prioridad
- **`EACCES`**: Un proceso de usuario intentó reducir la prioridad

Influir en la planificación de un proceso

- Modificando su prioridad estática
  - Mediante *sched_setscheduler()*
- Para los procesos SCHED_OTHER, modificando su prioridad dinámica
  - Mediante *setpriority*()
  - Comando *nice* permite ejecutar un proceso con una determinada prioridad (entre -20 y 19)
  - Comando *renice* permite variar la prioridad de un proceso enejecución

```
• int nice(int inc);
```

<unistd.h>
SV+BSD+POSIX

Llamada al sistema *nice*: permite incrementar/decerementar la prioridad del proceso en ejecución

Sólo *root* puede especificar incrementos negativos

Influir en la planificación de un proceso

- Influyendo en el procesador en que se ejecutará el proceso
  - Comando *tasket* permite especificar en qué CPUs queremos permitir que se ejecute el proceso que se creará

<sched.h>

```
• int sched_setaffinity(pid_t pid, unsigned int len,
                        unsigned long *mask);

• int sched_getaffinity(pid_t pid, unsigned int len,
                        unsigned long *mask);
```

Llamadas al sistema con funcionalidad similar a *taskset*.
`pid` → proceso al que se quiere cambiar/consultar la afinidad
`mask` → nueva afinidad proceso-CPU expresada como máscara de bits:

mask == 0x00000001 -→ CPU 0
mask == 0x00000003 → CPUs 0 y 1