Boletín 5- Señales

Departamento de Lenguajes y Sistemas Informáticos

Indice

1. Introducción
2. Envío de señales desde la shell: kill
3. Llamadas al Sistema
   - kill: envío de señal a un proceso
   - raise: autoenvío de una señal
   - alarm: planificar envío de una señal de alarma
   - signal: registrar manejador para una señal
   - pause: espera indefinida hasta recepción de señal

Introducción: Señales

- -Definición de señal en UNIX

  Una ***Señal*** es una interrupción por software que permite la comunicación entre procesos.

- Ante una señal un proceso puede responder de diferentes maneras:
  - ignorarla
  - invocar a la rutina de manejo por defecto: Produce la finalización
  - invocar una rutina registrada por el programador

Introducción: Señales (2)

- Cada señal está identificada por un número entero. Véamos algunas de ellas:

```
#define   SIGHUP    1    /* Hangup (POSIX).  */
#define   SIGINT    2    /* Interrupt (ANSI).  */
#define   SIGKILL   9    /* Kill, unblockable (POSIX).  */
#define   SIGUSR1   10   /* User-defined signal 1 (POSIX).  */
#define   SIGSEGV   11   /* Segmentation violation (ANSI).  */
#define   SIGUSR2   12   /* User-defined signal 2 (POSIX).  */
#define   SIGPIPE   13   /* Broken pipe (POSIX).  */
#define   SIGALRM   14   /* Alarm clock (POSIX).  */
#define   SIGTERM   15   /* Termination (ANSI).  */
#define   SIGCHLD   17   /* Child status changed  (POSIX).  */
#define   SIGCONT   18   /* Continue (POSIX).  */
#define   SIGSTOP   19   /* Stop, unblockable (POSIX).  */
```

Envío de señales desde la shell: kill

- Sintaxis:
  $ kill -señal pid

- Ejemplos:
  $ cat > prueba.txt
  $ ps -aux | grep cat

```
practica   237    1  0   Aug 19 ?        0:01     cat
```

$ kill -STOP 237 # interrumpe la ejecución de *cat*
$ kill -CONT 237 # reanuda la ejecución de *cat*
$ kill -KILL 237 # terminación abrupta de la ejecución de *cat*

$ kill -9 237 # equivalente a kill -KILL 237

Llamadas al sistema: Kill

- int kill(int pid, int sig)

- **Parámetros:**
  @pid: número de identificación del proceso
  pid = 0: se envía a todos los procesos del sistema
  pid = -1: se envía a todos excepto al proceso init
  pid < -1: se envía a todos los procesos del mismo grupo
  @sig: número de la señal que se quiere enviar

- **Acción:** Envía una cierta señal a un proceso
  Valor devuelto: En caso de error devuelve -1 y *errno* toma el valor que identifica el error producid*o*

Llamadas al sistema: Raise

- int raise(int sig)

- **Parámetros:**
  @sig: número de la señal que se quiere enviar

- **Acción:** Envía una cierta señal al propio proceso llamante
  Valor devuelto: En caso de error devuelve -1 y *errno* toma el valor que identifica el error producid*o*

- **Importante:** Es equivalente a *kill(getpid(), sig);*

Llamadas al sistema: Signal

- sighandler_t signal(int signum, sighandler_t handler)

- **Parámetros:**
  @signum: número de la señal que se quiere enviar
  @handler: función que maneja una señal recibida
  typedef void (*sighandler_t)(int);

- **Acción:** Registra un manejador para una cierta señal
  Valor devuelto: En caso de error devuelve SIG_ERR

Llamadas al sistema: Signal (2)

```
void manejador(int sig)
{
  printf("recibida señal número %d\n", sig);
  signal(SIGINT, manejador);    /* Sólo requerida sobre Solaris */
}

void main()
{
  if (signal(SIGINT, manejador) == SIG_ERR) {
        perror("signal");
        exit(EXIT_FAILURE);
  }
  ... /* código del programa */
}
```

Llamadas al sistema: Signal (3)

- **Importante:**
  - Por defecto las señales que un proceso recibe son manejadas por SIG_DFL que es el manejador por defecto: Aborta la ejecución del programa
  - Para ignorar una señal: SIG_IGN
  - La llamada a signal no es apilable: una llamada a signal reemplaza el manejador anteriormente asignado.
  - Comportamiento dependiente de la plataforma: Sobre **Solaris**, una vez manejada una cierta señal, se restablece de nuevo el manejador por defecto. Sobre **Linux** el manejador se mantiene hasta que el programador explícitamente diga lo contrario.

Llamadas al sistema: Alarm

- int alarm(unsigned int n)

- **Parámetros:**
  @n: número de segundos a esperar para enviar la señal

- **Acción:** Envía la señal SIG_ALRM pasados **n** segundos
  Valor devuelto: Número de segundos pendientes de espera de la llamada previa a alarm().

- **Importante:** La llamada no es apilable: si previamente se realizó una llamada a *alarm()* el valor del temporizador se actualiza, por tanto sólo se puede mantener una única alarma por proceso a la vez.

Llamadas al sistema: Pause

- int pause(void)

- **Acción:** Pone al proceso en espera indefinida hasta que reciba *cualquier* señal.
  Valor devuelto: En caso de éxito devuelve -1 y se establece a EINTR