Introducción a GNU/Linux

Procesos

Material elaborado por:

M. en I. Yolanda Flores
yoli@unam.mx
Coordinación de Supercómputo, DGTIC, UNAM.

Procesos

- Un **programa** es un archivo ejecutable que reside en un sistema de archivos.
- Un **proceso** es una instancia de un programa en ejecución.

DGTIC

Procesos

Existe una metafora especial que se aplica a los procesos en el sistema UNIX:

Los procesos tienen vida: están vivos o muertos; son creados (nacidos) o mueren; pueden convertirse en zombies o convertirse en huérfanos. Son padres o hijos, y cuando deseamos suprimirlos, los matamos.

Procesos

- Cada proceso en UNIX tiene asociado un número que lo identifica y se denomina identificador de proceso o PID.
- Además del PID, los procesos tienen asignado otro número llamado identificador del proceso padre o PPID.

Creación de Procesos

- Un proceso es creado a partir de un archivo ejecutable.
- Se crea mediante una *llamada al sistema* (subrutina que hace que el kernel proporcione un cierto servicio a un programa).

DGTIC

Llamada al sistema fork

- Es la única forma de crear un nuevo proceso.
- Crea un duplicado exacto del proceso original.
- El proceso que hace la llamada se denomina proceso padre; el proceso que se crea se denomina proceso hijo.
- Después del fork, padre e hijo siguen caminos independientes.
- El código, como no se modifica, es compartido entre el hijo y el padre.

```
/*
 * child.c - Simple fork usage
 */
#include <unistd.h>
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
   pid_t child;

   if((child = fork()) == -1) {
                perror("fork");
                exit(EXIT_FAILURE);
   } else if(child == 0) {
                puts("En el hijo");
                printf("\tchild pid = %d\n", getpid());
                printf("\tchild ppid = %d\n", getppid());
                exit(EXIT_SUCCESS);
   } else {
                puts("En el padre");
                printf("\tparent pid = %d\n", getpid());
                printf("\tparent ppid = %d\n", getppid());
                sleep(2);
   }
   printf("\n");
   exit(EXIT_SUCCESS);
}
```


DGTIC

Llamada al sistema exec

- Sustituye la imagen de un proceso por el código por un nuevo archivo ejecutable (no crea un nuevo proceso)
- La llamada exec permite a un proceso hijo ejecutar un comando especificado.

Procesos

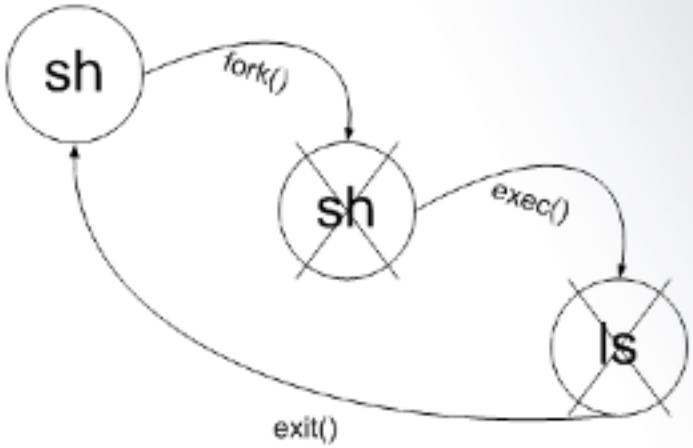

Tabla de procesos

- Arreglo de estructuras con un dato por proceso.
- Contiene toda la información de los atributos de un proceso
  - información del planificador (ID, prioridad, etc.)
  - estado actual (preparado, en espera...)
  - registros de la CPU (contador de programa, otros registros)
  - apuntadores a las zonas de memoria del proceso
  - info. de contabilidad (tiempo consumido, etc.)
  - etc.

ps (process status)

- Comando que sirve para ver el estado de los procesos. Proporciona información sobre:
  - Identificación del proceso (PID)
  - Identificación del proceso padre (PPID)
  - Usuario (USER o UID)
  - Tiempo de CPU (TIME)
  - Memoria (SZ, RSS)
  - Estado (S)
  - Comando (CMD)
  - Etc.

ps

- BSD (sin guión)

`a`

Procesos de todos los usuarios.

`u`

Listado específico, incluye login del usuario.

`x`

Procesos con terminal asociada

`aux`

Lista todos los procesos del sistema.

- SV (con guión)

`-e`

Lista cada proceso corriendo en el sistema.

`-l`

Formato largo.

`-f`

Formato específico, incluye el login del usuario.

`-u login`

Procesos de un usuario en particular

`-fe`

Lista todos los procesos del sistema

DGTIC

Procesos

- Un proceso termina cuando invoca a una llamada al sistema específica (exit)
- También si se genera una excepción y el S.O. decide abortarlo
- Generalmente, cuando un proceso termina, con él muere su descendencia (genocidio).
- Si un hijo sobrevive a la muerte del proceso padre, es adoptado por el proceso init o, en GNU/Linux (kernel 3.4 o superior) por un proceso de tipo recolector o *subreaper* (el cual tiene activada la opción de convertirse en padre de sus procesos descendientes).

DGTIC

Procesos Zombies <defunct>

- Unix maneja una relación explícita entre procesos padres e hijos.
- Cuando un hijo muere, el padre recibe una notificación y es tarea del padre tener control exacto de cuales de sus hijos aún están activos.
- El proposito de un *proceso zombie* es recordar los PID e informar a los padres sobre el estado de los hijos.
- Si un proceso padre no recolecta la información de los hijos, el zombie permanecerá en la tabla de procesos hasta que muera el padre.

DGTIC

Señales

- Interrupción de software que se envía a un proceso para informarle de algún evento.
- Cada señal tiene asociado un número entero positivo que la identifica.

SIGHUP	1	Hangup	Señala al proceso que lea de nuevo sus archivos de configuración
SIGKILL	9	Kill	Muerte. Provoca la terminación de un proceso.
SIGTERM	15	Termination	Indica que un proceso debe terminar su ejecución
SIGSTOP	19	Stop	Indica al proceso que pare momentáneamente
SIGCONT	18	Continue	Indica al proceso que continúe con su ejecución

kill

- El comando kill sirve para **enviar señales** a uno o varios procesos.

```
kill [-señal] PID [PID … ]
```


DGTIC

Ejecución de procesos

- **Primer plano (foreground)**
  - El proceso es totalmente visible al usuario.
  - El proceso toma el control de la terminal.
  - El usuario debe esperar a que el proceso termine para poder ejecutar otro proceso.
  - El usuario puede interactuar con el proceso.

```
$ comando
```


DGTIC

Ejecución de procesos

- **Segundo plano (background)**
  - El proceso no es totalmente visible al usuario.
  - Generalmente este tipo de procesos no despliegan en pantalla.
  - El proceso libera la terminal por lo que el usuario puede ejecutar otro proceso mientras éste se está ejecutando.
  - Este tipo de procesos generalmente no son interactivos.
  - Para enviar directamente un proceso a ejecución en segundo plano se le pone un signo de & al final.

```
$ comando &
```

Control de trabajos

- **Jobs**
  - Es una forma más práctica introducida por C-Shell para controlar los procesos asociados a un shell en particular.
  - Para consultar los procesos asociados al shell o a la terminal, se debe dar el comando:

```
$ jobs
```

Control de trabajos

- **Envío de un job a primer plano**:

```
$ fg %JOBnumber
```

- **Envío de un job a segundo plano**:

```
$ bg %JOBnumber
```

- **Eliminar un job**:

```
$ kill %JOBnumber
```

Ejemplos

```
$ uname
Linux

$ comando
^Z
[1]+  Stopped comando

$ jobs
[1]+ Stopped comando

$ bg %1
[1]+ comando &
% jobs
[1]+ Running    comando &
% kill %1
% jobs
[1]+ Exit señal comando
```