UNIVERSIDAD DEL CARIBE

Ingeniería en Telemática

Diseño de Sistemas Operativos

"Procesos y Señales"

070300610 Edgar Zaldivar Rodríguez
090300113 Claussell Albornoz Flavio Antonio

PROFESOR:
SAN MARTIN ALEJANDRO MARTIN CANUL

Cancún, Quintana Roo Miercoles 28 Enero del 2015

Introducción

¿ Que es un proceso ?

Un proceso simplemente es un programa en ejecución. Los procesos además de la información propia del programa contienen la información necesaria para que el programa interaccione con el sistema.

Tipos de procesos

Child (hijos)

Un proceso hijo es un proceso creado por otro proceso, estos se crean mediante la llamada al sistema fork() y en realidad, todos los procesos en algún momento son hijos, todos menos el proceso init. En el caso de que un proceso sea creado mediante la shell, la shell sera el padre.

Orphan (huérfanos)

Normalmente un proceso hijo termina antes que un proceso padre, pero se puede dar la situación de que se mate a un proceso padre (killed) y el hijo se quede sin padre. Entonces el proceso init lo adoptara como hijo, pero como su padre original no existe, es considerado huérfano.

Daemon (demonios)

Es un tipo especial de proceso que se ejecuta en segundo plano y no esta asociado a ninguna shell. Esto se consigue matando la shell que crea el proceso, de esta forma el padre de este proceso pasa a ser el proceso init. Estos corren con permisos de root y su cometido es proveer servicios a otros procesos.

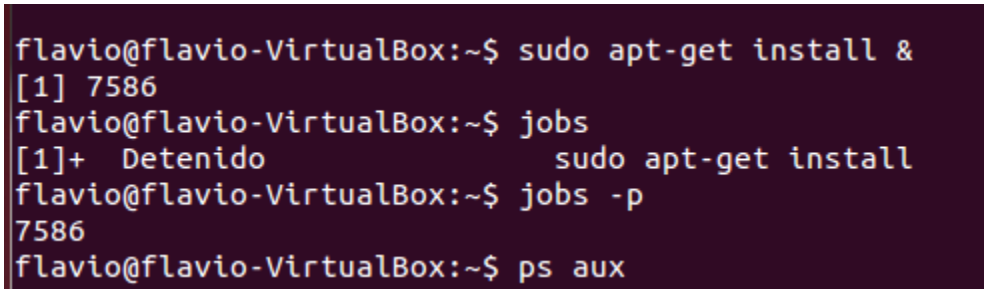
Zombie

Cuando un proceso hijo termina, el sistema guarda el PID (Identificador) y su estado (un parámetro) para dárselo a su padre. Hasta entonces el proceso finalizado entra en estado zombie. Cuando un proceso finaliza toda la memoria y recursos asociados con dicho proceso son liberados, pero la entrada del mismo en la tabla de procesos aún existe, para cuando su padre llame a la función wait() devolverle su PID y estado.

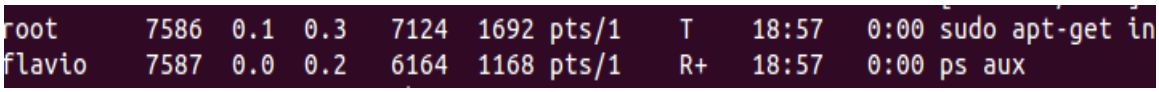
Reporte de practica

En esta practica se deberá identificar la forma en que se puede mandar un proceso a segundo plano, como regresarlo de segundo plano, matar dichos procesos y las señales que se utilizan en estas operaciones.

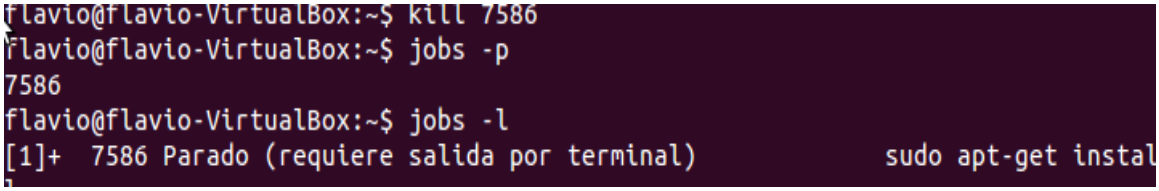
Como se aprecia en la imagen de abajo existe un método para mandar los procesos en segundo plano y es agregar un "&" al final del comando para mandarlo en segundo plano, también se puede utilizar el comando "bg" al inicio del comando y realizara el mismo proceso que el "&".



Después de realizar un "ps aux" se enlistan los procesos que están corriendo y se puede notar el proceso en segundo plano que se realizo como se aprecia en la imagen de abajo.



En la imagen de abajo se puede notar que intentamos matar el proceso usando el comando "kill" pero no tuvimos éxito, si ejecutamos el comando "Jobs -p" o "Jobs - l" que es para listar los procesos en segundo plano notaremos que el proceso sigue vivo.



Existen dos formas de terminar estos procesos en segundo plano, uno es traerlos de vuelta a primer plano dejar que se ejecuten, esto se puede lograr con el comando "fg" o matarlos con el comando "kill id", pero si quisiéramos matarlo desde segundo plano solo debemos agregar al comando -9 = "kill -9 id"enviar una señal definitiva junto con el comando para que nos deje matar el proceso sin ningún problema

```
flavio@flavio-VirtualBox:~$ fg sudo apt-get install
sudo apt-get install
Terminado
```

```
flavio@flavio-VirtualBox:~$ fg 1
sudo apt-get install
[sudo] password for flavio:
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
```

Existe una forma de pasar señales por medio de la consola y en la siguiente imagen se aprecia como saber como esta configurado tu consola para transmitir las señales por combinaciones de teclas, al ejecutar el comando en la imagen podrás saber dichas combinaciones.

```
flavio@flavio-VirtualBox:~$ stty -a | grep intr
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = M-^?; eol2 = M-^?;
flavio@flavio-VirtualBox:~$
```

Algunas de las señales mas importantes son:

Signal	Code on Linux	Default Action	Description
SIGABRT	6	A	Process abort signal
SIGALRM	14	T	Alarm clock
SIGBUS	10	A	Access to an undefined portion of a memory object
SIGCHLD	18	I - Ignore the Signal[8]	Child process terminated, stopped,
SIGCONT	25	C - Continue the process	Continue executing, if stopped.
SIGFPE	8	A	Erroneous arithmetic operation.

SIGHUP	1	T	Hangup.
SIGILL	4	A	Illegal instruction.
SIGINT	2	T	Terminal interrupt signal.
SIGKILL	9	T	Kill (cannot be caught or ignored).
SIGPIPE	13	T - Abnormal termination of the process	Write on a pipe with no one to read it.
SIGQUIT	3	A - Abnormal termination of the process	Terminal quit signal.
SIGSEGV	11	A	Invalid memory reference.
SIGSTOP	23	S - Stop the process	Stop executing (cannot be caught or ignored).
SIGTERM	15	T	Termination signal.
SIGTSTP	20	S	Terminal stop signal.
SIGTTIN	26	S	Background process attempting read.
SIGTTOU	27	S	Background process attempting write.
SIGUSR1	16	T	User-defined signal 1.

SIGUSR2	17	T	User-defined signal 2.
SIGPOLL	22	T	Pollable event.
SIGPROF	29	T	Profiling timer expired.
SIGSYS	12	A	Bad system call.
SIGTRAP	5	A	Trace/breakpoint trap.
SIGURG	21	I	High bandwidth data is available at a socket.
SIGVTALRM	28	T	Virtual timer expired.
SIGXCPU	30	A	CPU time limit exceeded.
SIGXFSZ	31	A	File size limit exceeded