

Creación y terminación de procesos

Práctica 2



ITESO, Universidad
Jesuita de Guadalajara

60
años

María del Carmen Martínez Nuño is703358

Mariana Sierra Vega is702782

Fundamentos de Sistemas Operativos Otoño 2018

4 de septiembre del 2018

Profesor José Luis Elvira Valenzuela

Creación y terminación de procesos

1. Revise el comando `ps`, y qué opciones tiene para su ejecución, explique ¿por qué cuando un usuario teclea el comando `ps` sin parámetros solo muestra unos cuantos procesos?

Cuando se ejecuta el comando `ps` únicamente se muestran los procesos que están corriendo en la terminal en la que se ejecutó el comando. Mostrando el `bash` (consola) y los procesos que ahí corren.

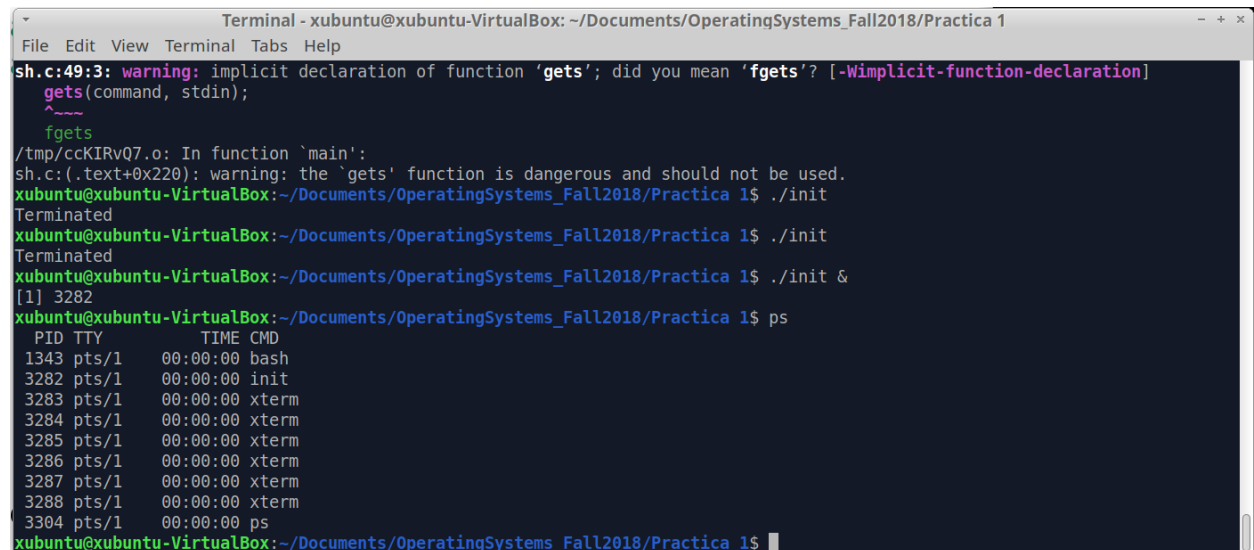
2. Al ejecutar el proceso `init` ¿Qué procesos nuevos se muestran en el sistema?

Al correr `init` en el background podemos observar cómo se crean 6 nuevos procesos a parte del `init` original que representan las 6 consolas que se abren desde consola.

3. Inicie al menos dos sesiones en las ventanas que creo `getty` y muestre la lista de procesos en la misma ventana donde ejecutó el proceso `init`, ¿qué procesos nuevos se muestran en el sistema?

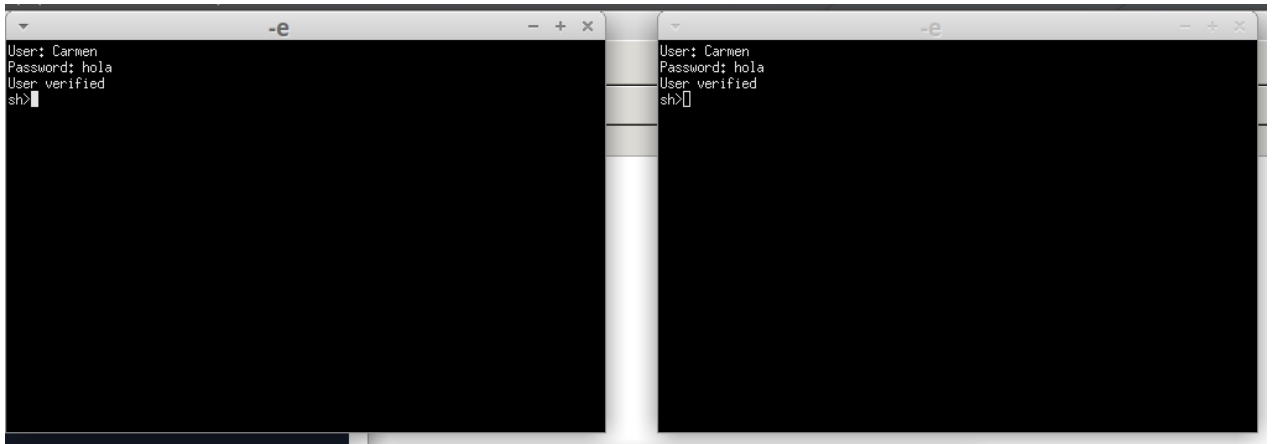
Al correr `init` en el background y acceder a dos diferentes sesiones en las ventanas que se crearon de `getty` no se crea ningún nuevo proceso con el comando `ps` normal, ya que no están en el alcance de la terminal que se está ejecutando en ese momento, si pusiéramos el mismo comando en la nueva `sh` que ese abre ahí sí se verán los nuevos procesos creados.

- a. Proceso iniciado en background y cantidad de procesos que hay



```
Terminal - xubuntu@xubuntu-VirtualBox: ~/Documents/OperatingSystems_Fall2018/Practica 1
File Edit View Terminal Tabs Help
sh.c:49:3: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [-Wimplicit-function-declaration]
  gets(command, stdin);
  ^
  fgets
/tmp/ccKIRvQ7.o: In function 'main':
sh.c:(.text+0x220): warning: the 'gets' function is dangerous and should not be used.
xubuntu@xubuntu-VirtualBox:~/Documents/OperatingSystems_Fall2018/Practica 1$ ./init
Terminated
xubuntu@xubuntu-VirtualBox:~/Documents/OperatingSystems_Fall2018/Practica 1$ ./init
Terminated
xubuntu@xubuntu-VirtualBox:~/Documents/OperatingSystems_Fall2018/Practica 1$ ./init &
[1] 3282
xubuntu@xubuntu-VirtualBox:~/Documents/OperatingSystems_Fall2018/Practica 1$ ps
  PID TTY          TIME CMD
 1343 pts/1    00:00:00 bash
 3282 pts/1    00:00:00 init
 3283 pts/1    00:00:00 xterm
 3284 pts/1    00:00:00 xterm
 3285 pts/1    00:00:00 xterm
 3286 pts/1    00:00:00 xterm
 3287 pts/1    00:00:00 xterm
 3288 pts/1    00:00:00 xterm
 3304 pts/1    00:00:00 ps
xubuntu@xubuntu-VirtualBox:~/Documents/OperatingSystems_Fall2018/Practica 1$
```

- b. Entrar con usuario y contraseña correctos en ambos `getty`



- c. Mostrar de nuevo el comando `ps` demostrando que no aparecen nuevos procesos en el alcance de esta terminal.

```
Terminal - xubuntu@xubuntu-VirtualBox: ~/Documents/OperatingSystems_Fall2018/Practica 1
File Edit View Terminal Tabs Help
[1] 3282
xubuntu@xubuntu-VirtualBox:~/Documents/OperatingSystems_Fall2018/Practica 1$ ps
  PID TTY          TIME CMD
 1343 pts/1        00:00:00 bash
 3282 pts/1        00:00:00 init
 3283 pts/1        00:00:00 xterm
 3284 pts/1        00:00:00 xterm
 3285 pts/1        00:00:00 xterm
 3286 pts/1        00:00:00 xterm
 3287 pts/1        00:00:00 xterm
 3288 pts/1        00:00:00 xterm
 3304 pts/1        00:00:00 ps
xubuntu@xubuntu-VirtualBox:~/Documents/OperatingSystems_Fall2018/Practica 1$ ps
  PID TTY          TIME CMD
 1343 pts/1        00:00:00 bash
 3282 pts/1        00:00:00 init
 3283 pts/1        00:00:00 xterm
 3284 pts/1        00:00:00 xterm
 3285 pts/1        00:00:00 xterm
 3286 pts/1        00:00:00 xterm
 3287 pts/1        00:00:00 xterm
 3288 pts/1        00:00:00 xterm
 3311 pts/1        00:00:00 ps
xubuntu@xubuntu-VirtualBox:~/Documents/OperatingSystems_Fall2018/Practica 1$
```

4. En una de las ventanas del `shell` creada por el proceso `getty`, teclee el comando `ps`, ¿qué procesos se muestran?

```
User: Carmen
Password: hola
User verified
sh>ps
  PID TTY          TIME CMD
 3230 pts/3        00:00:00 getty
 3233 pts/3        00:00:00 sh
 3234 pts/3        00:00:00 sh
 3235 pts/3        00:00:00 ps
sh>
```

Podemos ver en la imagen anterior que tenemos el proceso **getty**, que a su vez crea el proceso **sh**, esto fue un hijo que fue reemplazado por un nuevo proceso que ejecuta un **shell** que es el que estará ejecutando a su vez el proceso **ps** que ingreso el usuario.

Viendo progresivamente los procesos tendrían las siguientes razones de ser:

3230 getty: Es el proceso padre del shell

3233 sh: Shell generado por el proceso getty

3234 sh: Sh ejecutado por el **execv**

3235 ps: Comando ejecutado por el sh

5. ¿Qué efecto tiene la espera de un proceso hijo en el proceso **getty**?, ¿qué sucedería si no existiera esa espera?

Utilizar métodos de espera, como **wait()** generan que el proceso padre no continúe con su ejecución hasta que su hijo se haya terminado de ejecutar; si esto no se lleva a cabo el proceso padre después de crear al hijo continuará con su ejecución y al terminar, el proceso se destruirá, destruyendo a su vez el proceso hijo sin importar el estado en el que esté. El problema con el proceso **getty** es que es un ciclo infinito que está esperando comandos en consola por lo que ambas terminales se empezarán a encimar una con otra haciendo que sea complicada su gestión, creando procesos innecesarios y procesos zombies, además de no permitir ver al usuario cuál es la terminal en la que se está trabajando.

6. En los procesos anteriores está utilizando la llamada **exec** que para reemplazar la imagen de un proceso busca el programa ejecutable en los directorios especificados en la variable de ambiente **PATH**. Investigue (buscar en documentación de UNIX) por qué esta llamada pueda hacer uso de los valores en la variable **PATH** sin que sea necesario inicializar esta variable en cada uno de sus procesos.

Esto es porque **exec** normalmente se ejecuta después de un **fork()** lo que implica que es un proceso hijo que, a menos que se especifique lo contrario, va a tener las mismas variables de entorno del padre, por lo que al heredar esta información tiene acceso a la variable de ambiente **PATH** y así puede ejecutar estos procesos sin la necesidad de inicializar esta variable en cada proceso hijo.

Notas:

Algunos programas pueden que aparezcan con warnings al momento de la compilación sin embargo no han afectado directamente a la ejecución de los procesos.