

Fundamentos de Sistemas Operativos

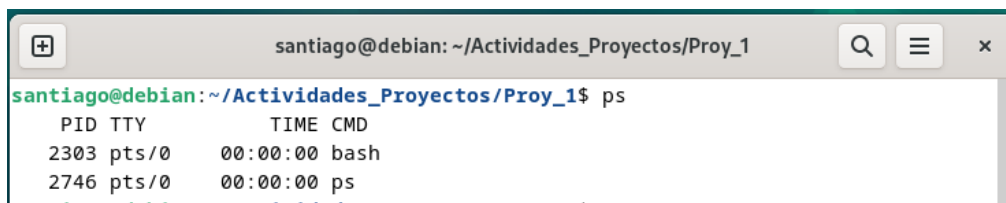
Práctica 1 – Procesos: Emulación de login del SO

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?

El comando `ps` se usa para mostrar información sobre los procesos en ejecución. Cuando se teclea los procesos que muestra son:

- **`ps`**: este dice que procesos que están asociados con el terminal actual. Solo se muestran los procesos del usuario que ejecutó el comando en la terminal donde se ejecutó.
- **`ps -e` o `ps -a`**: este muestra todos los procesos en el sistema, pero sin filtrar por terminal.
- **`ps -f`**: muestra la información en un formato más completo, incluyendo detalles como el ID del proceso padre (PPID).
- **`ps -u usuario`**: dice los procesos que están siendo ejecutados por un usuario específico.
- **`ps -aux`**: te dice todos los procesos en ejecución con detalles, incluyendo procesos de otros usuarios.

Solo se muestran algunos porque, por defecto, el comando `ps` solo muestra los procesos asociados con la terminal o shell en la que se ejecuta el comando, por lo que solo muestra los procesos que el usuario ha iniciado en esa sesión o terminal y no incluye procesos de otros usuarios ni procesos de otros terminales, ya que no tiene acceso a la información completa del sistema a menos que se usen opciones como `ps -e` o `ps -aux`.



The screenshot shows a terminal window with the title bar "santiago@debian: ~/Actividades_Proyectos/Proy_1". The prompt is "santiago@debian: ~/Actividades_Proyectos/Proy_1\$". The command "ps" has been entered, and the output is displayed as follows:

PID	TTY	TIME	CMD
2303	pts/0	00:00:00	bash
2746	pts/0	00:00:00	ps

```
login: Pedro
password: PicaPiedra
Login successful!
Si le atinaste
Shell started for user: 2707
sh > ps
  PID TTY          TIME CMD
  2713 pts/5        00:00:00 getty
  2714 pts/5        00:00:00 sh
  2715 pts/5        00:00:00 ps
sh > █
```

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

El primer proceso que se ejecuta es el proceso init y los 6 procesos hijos, es decir los getty y aparte de los xterm para abrir las ventanas

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?

Después de que el init haya creado a los procesos getty, al abrir las dos sesiones getty va a iniciar nuevos procesos de shell

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

Se muestra el ps, el proceso sh que se está ejecutando en esa terminal además de otros procesos que estén en ejecución desde la terminal o el shell, entonces sólo se mostrarán los procesos de esa sesión.

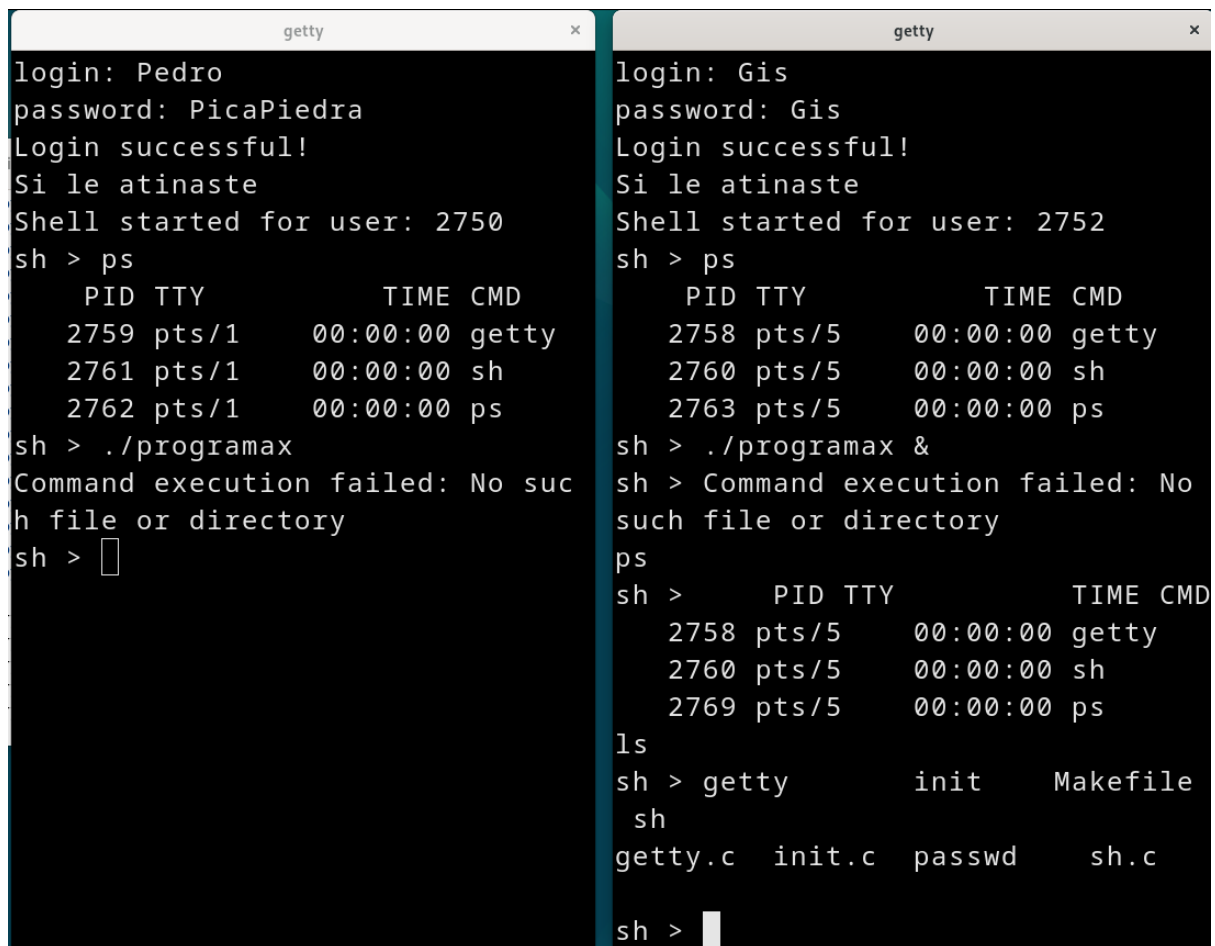
¿Qué efecto tiene la espera de un proceso hijo en el proceso getty?, ¿qué sucedería si no existiera esa espera?

Pues que si no ponemos el waitpid el proceso getty se sigue ejecutando y podría crear nuevos shells antes de haber terminado los anteriores, osea que los procesos se conviertan en zombies por lo que al final terminan tomando más recursos.

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.

La llamada exec reemplaza la imagen del proceso actual con el programa especificado en el argumento, y para encontrar ese programa, busca en los directorios en la variable de entorno PATH, la cual es heredada por los procesos hijos del proceso que la crea. Por ejemplo, cuando el proceso init inicia un proceso getty, el valor del PATH se transmite al proceso getty, y luego a los procesos que getty cree (como sh). No es necesario reestablecer PATH en cada uno de estos procesos porque el sistema operativo lo maneja automáticamente al crear los procesos hijos.



The image shows two terminal windows side-by-side, both titled 'getty'. The left window shows a login for 'Pedro' with password 'PicaPiedra'. The right window shows a login for 'Gis' with password 'Gis'. Both users are prompted to 'Si le atinaste' (If you guessed it) and then 'Shell started for user: 2750' (left) or '2752' (right). Both users run 'ps' and see a table of processes. In the left window, the processes are getty (PID 2759), sh (PID 2761), and ps (PID 2762). In the right window, the processes are getty (PID 2758), sh (PID 2760), and ps (PID 2763). Both users then run './programax' and receive the error 'Command execution failed: No such file or directory'. The right window then shows the output of 'ls' and 'sh > getty', which lists files in the current directory: 'sh', 'getty.c', 'init.c', 'passwd', and 'sh.c'.

```
login: Pedro
password: PicaPiedra
Login successful!
Si le atinaste
Shell started for user: 2750
sh > ps
  PID TTY          TIME CMD
 2759 pts/1        00:00:00 getty
 2761 pts/1        00:00:00 sh
 2762 pts/1        00:00:00 ps
sh > ./programax
Command execution failed: No such
file or directory
sh > 
```

```
login: Gis
password: Gis
Login successful!
Si le atinaste
Shell started for user: 2752
sh > ps
  PID TTY          TIME CMD
 2758 pts/5        00:00:00 getty
 2760 pts/5        00:00:00 sh
 2763 pts/5        00:00:00 ps
sh > ./programax &
sh > Command execution failed: No
such file or directory
ps
sh >      PID TTY          TIME CMD
 2758 pts/5        00:00:00 getty
 2760 pts/5        00:00:00 sh
 2769 pts/5        00:00:00 ps
ls
sh > getty      init      Makefile
sh
getty.c  init.c  passwd  sh.c
sh > 
```