



Escuela Superior de Cómputo  
Instituto Politécnico Nacional

Sistemas Operativos  
Procesos

## Práctica 2

---

Proceso zombie, Kill y Waitpid

Hernández Herrera Daniel Alejandro  
Ojeda Galván René  
Sánchez Gutiérrez Luis Arturo  
Torres Rodríguez Mauricio Alberto  
2CV7

## 1. zombie.c

Muestre la pantalla de ejecución del programa y ejecute de manera adicional en una terminal diferente el comando `ps -ae`.

```
root@R2D23sUbuntu:/home/arturo/2_2# ./zombie
root@R2D23sUbuntu:/home/arturo/2_2# ./zombie
-
906 ?      00:00:00 systemd-logind
911 ?      00:00:00 dbus-daemon
935 ?      00:00:00 rsyslogd
937 ?      00:00:00 lxcfs
940 ?      00:00:00 acpid
981 ?      00:00:00 polkitd
990 ?      00:00:00 mdadm
1062 ?     00:00:00 kworker/0:4
1067 ?     00:00:00 iscsid
1068 ?     00:00:00 iscsid
1127 tty1   00:00:00 login
1164 ?     00:00:00 apt-get
1167 ?     00:00:00 http
1168 ?     00:00:00 http
1174 ?     00:00:00 gpgv
1212 tty1   00:00:00 login
1263 ?     00:00:00 systemd
1273 ?     00:00:00 (sd-pam)
1278 tty1   00:00:00 bash
1302 tty1   00:00:00 zombie
1303 tty1   00:00:00 zombie <defunct>
1304 tty2   00:00:00 login
1340 tty2   00:00:00 bash
1352 tty2   00:00:00 ps
arturo@R2D23sUbuntu:~$ _
```

Realice lo siguiente

- Investigue en la bibliografía proporcionada para la asignatura, el diagrama de estados de procesos en LINUX y describa de manera cuáles son las condiciones que hacen que se pase a cada estado

Estos son los estados de los procesos de Linux[1]:

- **TASK\_RUNNING**

El proceso ya está siendo ejecutado o está en espera de ser ejecutado.

- **TASK\_INTERRUPTIBLE**

Proceso suspendido que puede reanudar a TASK\_RUNNING dada una señal.

- **TASK\_UNINTERRUPTIBLE**

Proceso suspendido que no puede reanudar normalmente a TASK\_RUNNING dada una señal.

- **TASK\_STOPPED**

Estado que indica que un proceso ha acabado.

- **TASK\_ZOMBIE**

Estado que indica que un proceso ha acabado, pero no puede ser removido apropiadamente pues el padre no puede ser reportado de datos del proceso que podría necesitar. (Según la bibliografía, esto sucede cuando el proceso padre no sale de *wait()*)

- Tomando como referencia el diagrama obtenido en el punto anterior, explique de manera detallada los estados por los que pasa el proceso padre y el proceso hijo durante la ejecución del programa

- El proceso padre empieza en estado `TASK_RUNNING`.
- El proceso padre llama la función `fork()` y obtiene un hijo de mismo código.
- Ambos procesos en estado `TASK_RUNNING` entran al switch case.
- El padre pasa a `TASK_UNINTERRUPTIBLE` con `sleep()`.
- El proceso hijo termina, pero pasa a `TASK_ZOMBIE`, pues su padre sigue en `TASK_UNINTERRUPTIBLE`.
- Ambos procesos pasan a `TASK_STOPPED`

■ `perror`

Esta función produce un mensaje que describe al último error obtenido durante una llamada al sistema o una función de biblioteca.

■ `fork`

Crea un proceso hijo duplicando el proceso padre.

## 2. kill.c

Muestre la pantalla de ejecución del programa.

```
root@R2D23sUbuntu:/home/arturo/2_2# ./kill
Se envia la senal SIGCHLD a mi hijo 1572
Mi hijo 1572 esta vivo
Se envia la senal SIGTERM a mi hijo 1572
Mi hijo 1572 ha culminado por medio de la senal SIGTERM
root@R2D23sUbuntu:/home/arturo/2_2# _
```

Describa cuál es el propósito de las siguientes bibliotecas:

- signal.h

Esta biblioteca es utilizada para manejar señales.

- stdlib.h

Biblioteca estándar que utilizamos para manejar los procesos. Nos define las macros EXIT\_SUCCESS y EXIT\_FAILURE y la función *exit()*.

Para las siguientes funciones, mencione dónde están definidas, qué es lo que proporcionan de salida y qué argumentos necesitan de entrada:

- exit

Está definida en la biblioteca *stdlib.h*. No regresan nada y requieren un entero que representará el estado de salida del proceso. Termina el proceso.

- kill

Está definida en la biblioteca *signal.h*. Regresa 0 si tuvo éxito, -1 si falló. Pide como parámetros la señal a enviar y el pid del proceso a enviar la señal. Si es ¿0, se enviará la señal al proceso específico. Igual a 0 enviará la señal a todos los procesos del grupo del emisor. -1 enviará la señal a todos los procesos.

Mencione el significado de los siguientes valores de salida en exit:

- EXIT SUCCESS

El valor a regresar si el proceso ha acabado con éxito.

- EXIT FAILURE

El valor a regresar si el proceso ha acabado con errores.

### 3. waitpid.c

Muestre la pantalla de ejecución del programa.

```
root@R2D23sUbuntu:/home/arturo/2_2# ./waitpid
PROCESO HIJO
    pid hijo: 1209
    pid del padre: 1208
PROCESO PADRE:
    pid de proceso padre = 1208
    pid de proceso abuelo = 1184
    mi hijo envio la condicion = 0
root@R2D23sUbuntu:/home/arturo/2_2# _
```

Describa cuál es el propósito de la siguiente biblioteca:

- sys/wait.h

Proporciona lo necesario para trabajar con *waitpid()*.

Mencione dónde está definida, qué es lo que proporciona de salida y qué argumentos necesita de entrada la función *waitpid*:

- *waitpid*

Se encuentra definido en `sys/wait.h`. Espera a que un proceso hijo acabe.



## Referencias

- [1] Bovet D. & Cesati M. (2000), *Understanding the Linux Kernel*, O'Reilly, Primera Edición, 65-66, 1994.