Gestión de Procesos en Linux

Introducción

- **Procesos**: Programas en ejecución.
- Siempre tienen un proceso padre
- Tienen un ID asociado : PID
- Algunos comandos: `ps`, `kill`, `fg`, `bg`.

`sleep`

Para probar estos comandos utilizaremos el comando `sleep` para simular procesos que requieren tiempo.

- `sleep 10` → Espera 10 segundos antes de acabar
- `sleep 2m` → Espera 2 minutos antes de acabar

Ejecutar procesos en segundo plano

Normalmente no se puede introducir un nuevo comando hasta que acaba el anterior, podemos indicar que un comando se ejecute en segundo plano añadiendo `&` al final:

```
sleep 10 &
[1] 27752 # ID de trabajo (job) y PID
# ... Podemos seguir ejecutando comandos
# Nos avisan cuando acaba
[1]   Done                    sleep 10
```

Job vs PID

Concepto	Job	PID
Definición	Trabajo gestionado por el *shell*.	ID único de un proceso en el sistema.
Identificador	Número precedido por `%` (e.g., `%1`).	Número entero (e.g., `1234`).
Contexto	Usado con comandos como `fg` y `bg`.	Usado con comandos como `ps` y `kill`.
Visibilidad	Sólo visible en la sesión actual.	Visible globalmente en el sistema.
Relación	Un Job puede incluir uno o más PID.	Cada PID representa un proceso único.

`ps`: Visualización de Procesos

Uso básico:

- `ps` - Muestra procesos del usuario actual.
- `ps -ef` - Muestra **TODOS** los procesos con información adicional.

La opción `e` es la que indica todos los procesos y `f` (*full*) la información completa,pero suelen usarse conjuntamente.

(Hay muchas más opciones pero `ps -ef` es la manera más normal de utilizarlo)

Ejemplo

```
ps -ef

UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 Nov16 ?        00:00:02 /sbin/init splash
root           2       0  0 Nov16 ?        00:00:00 [kthreadd]
root           3       2  0 Nov16 ?        00:00:00 [pool_workqueue_release]
root           4       2  0 Nov16 ?        00:00:00 [kworker/R-rcu_g]
root           5       2  0 Nov16 ?        00:00:00 [kworker/R-rcu_p]
root           6       2  0 Nov16 ?        00:00:00 [kworker/R-slub_]
...
```

Columna	Descripción
UID	Usuario que ejecuta el proceso.
PID	ID del proceso.
PPID	ID del proceso padre.
C	Porcentaje de CPU usado.
STIME	Hora de inicio del proceso.
TTY	Terminal asociado (si aplica).
TIME	Tiempo total de CPU usado por el proceso.
CMD	Comando que inició el proceso.

`kill`: Finalizar Procesos

Uso básico:

- `kill [PID]` - Finaliza un proceso por ID.

Realmente kill sirve para enviar cualquier señal y por defecto se envía `SIGTERM`.

- Señales comunes:
  - `SIGTERM` (15): Solicita finalización.
  - `SIGKILL` (9): Forza terminación.

Trabajo en Segundo Plano: `bg`

Uso básico:

- `bg [job_id]` - Continua un trabajo que estaba parado en segundo plano.

```
sleep 10
[CTRL+Z] # Para el proceso
bg # Lo hace continuar (Por defecto el último job)
# Pero en segundo plano, podemos seguir usando comandos
```

Trabajo en Primer Plano: `fg`

Uso básico:

- `fg [job_id]` - Trae un trabajo al primer plano.

Ejemplo:

```
sleep 10 &
fg # Lo trae a primer plano (Por defecto el último job)
# Ahora ya no podemos seguir usando comandos, sleep esta en primer plano
```

Resumen

- `sleep`: Esperar un tiempo dado.
- `[comando] &`: Ejecutar un comando en segundo plano.
- `ps`: Lista procesos.
- `kill`: Controla terminación.
- `bg` / `fg`: Alterna entre fondo y primer plano.