Creación de procesos con Fork();

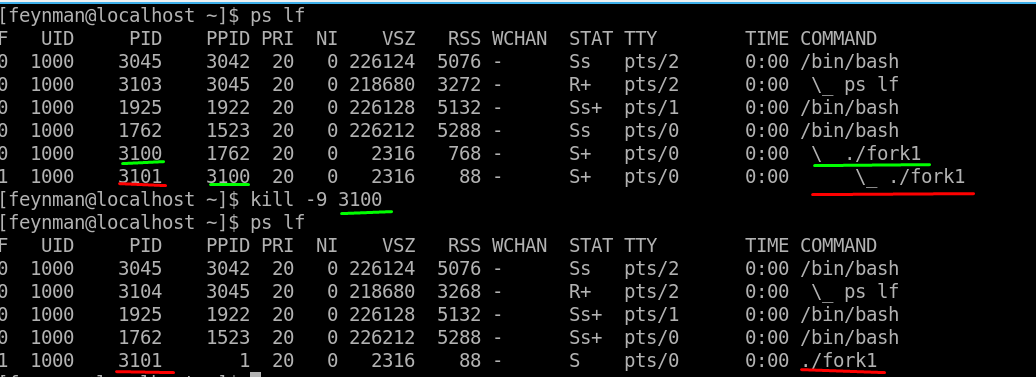
pid\_t fork(void)

* Crea un nuevo proceso (hijo)
* El proceso hijo es un duplicado del proceso “padre” (proceso que llamó a la función fork() )
* Los dos procesos:
  + Tienen PID diferentes (Identificador del Proceso)
  + Corren en espacios de memoria separados

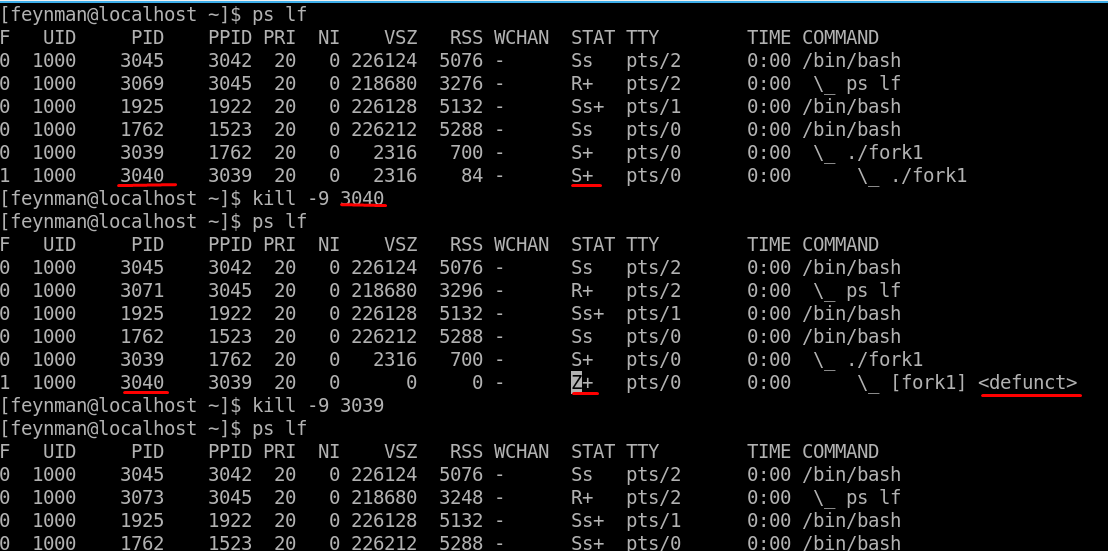
Retorno del fork()

* Si todo va bien:
  + Retorno al padre: PID del hijo
  + Retorno al hijo: 0
* Si hay errores
  + Retorno al padre: -1
  + El hijo no se crea

Proceso huérfano: Se termina la ejecución del proceso padre mediante “kill -9 PIDPadre”. Esto asignará al huérfano un nuevo PIDPadre.



Proceso Zombie: Se termina la ejecución de un proceso hijo creado mediante fork(). Este proceso deja de ejecutarse, pero sigue consumiendo un espacio residual en memoria hasta que se termina el proceso padre.



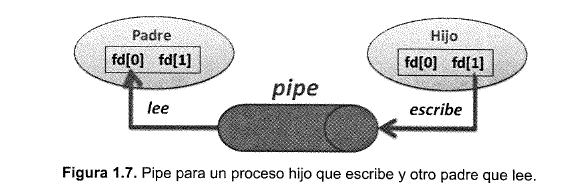
pid\_t wait(int \*wstatus)

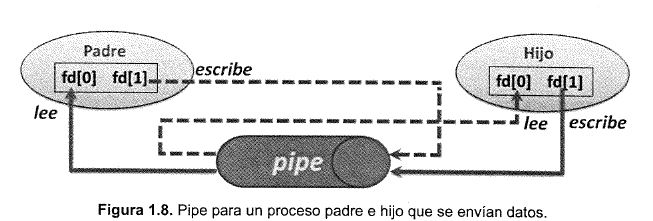
Será llamado por el proceso padre para esperar a que un proceso termine. Como argumento de entrada recibe un puntero a una variable status

WEXITSTATUS(wstatus)

Pipe

* Mecanimos IPC(comunicación entre procesos) unidireccional
* Comunican procesos “relacionados” (padre- hijo; hijo- hijo)
* Envío de “byte streams”
* Pipe asociado a dos fd (uno para escritura y otro para lectura)





Funciones:

* Crear pipe: int pipe(int fd[2]); /\* fd[0] : READ; fd [1] WRITE \*/
* Escribir: ssize\_t write(int fd, const void \*buf, size\_t nr);
* Leer: ssize\_t read(int fd, void \*buf, size\_t nr);
* Cerrar: int close (int fd);

