Práctica 1: Creación de procesos

PROGRAMACIÓN CONCURRENTE

Dra. Elizabeth Pérez Cortés

Estándar POSIX (http://pubs.opengroup.org/onlinepubs/9699919799/nframe.html)

POSIX es una norma escrita y una marca registrada por la Institute of Electrical and Electronics Engineers.[3] Dicha norma define una **interfaz estándar** del sistema operativo y el entorno, incluyendo un intérprete de comandos (o "shell"), y programas de utilidades comunes para apoyar la portabilidad de las aplicaciones a nivel de código fuente.

fork()

```
#include <unistd.h>

pid_t fork(void);
```

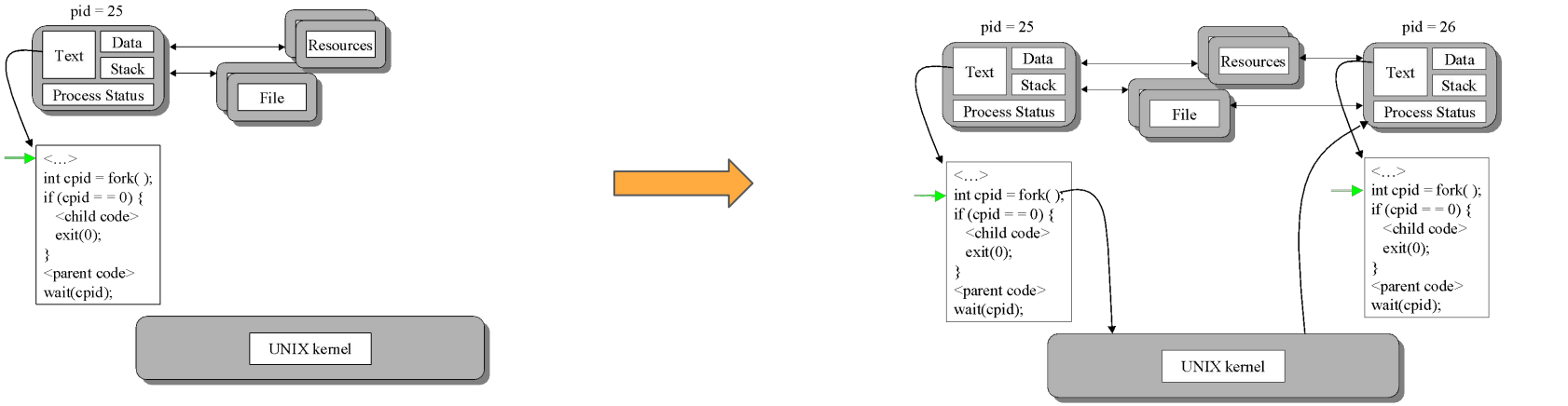
El llamado al sistema fork() crea un proceso hijo que únicamente difiere del padre en su identificador de proceso (PID) y el identificador de su proceso padre (PPID), y en el hecho de que la utilización de recursos del proceso recién creado se establece en 0.

Si la invocación a fork() es exitosa, el valor de regreso en el flujo de ejecución del padre es el PID del proceso hijo, y el valor de regreso en el flujo de ejecución del proceso hijo es 0. En caso de error, se regresa –1 en el contexto del padre y no se creará el proceso hijo.

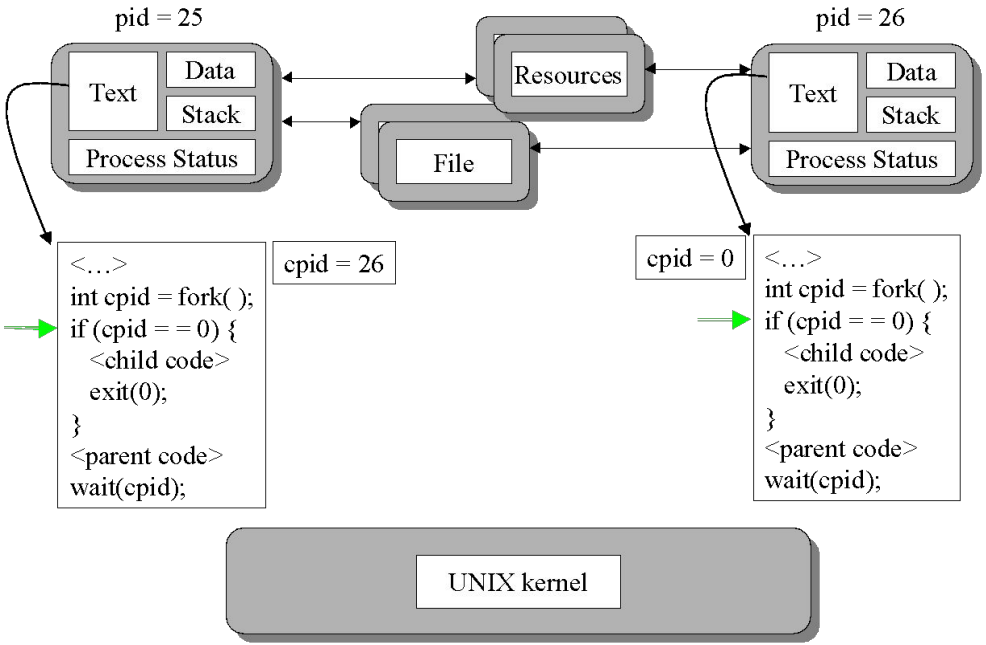
Ejemplo:

```
pid_t pid=fork();
if (pid==-1)
    printf("Error");
else if (pid==0)
    printf("Soy el Hijo");
else
    print("Soy el padre");
```

Internamente



Internamente



wait()

```
#include <sys/wait.h>
pid_t wait(int *stat_loc);
```

wait() suspende la ejecución del proceso actual, hasta que un hijo ha terminado, o hasta que se entrega una señal cuya acción es terminar el proceso actual. Si un proceso hijo ya ha terminado al momento de la llamada (proceso “zombie”), la función regresa inmediatamente. Los recursos del sistema utilizados por el hijo son liberados.

Si status no es NULL, wait guardará información del estado del proceso en el lugar al que apunta. Este estatus se puede evaluar con alguna de las siguientes macros:
WEXITSTATUS(status): evalúa los ocho dígitos menos significativos del valor de regreso del hijo que terminó, que pudieron haber sido especificados como el argumento de una llamada a exit() o como el argumento para una sentencia return en el programa main.
El valor de regreso es el PID del hijo que terminó, o bien –1 en caso de error.

getpid() y getppid()

```
#include <unistd.h>

pid_t getpid(void);
```

getpid() regresa el identificador del proceso que lo invoca.

```
#include <unistd.h>

pid_t getppid(void);
```

getppid() regresa el identificador del padre del proceso que lo invoca.