050242596fork() Abner Galindo González

La explicación que se da es que es una llamada al sistema operativo de UNIX, para estructurar un proceso padre-hijo, usa la cabecera #include <unistd.h>, su sintaxis es int fork(void).

También crea un proceso exactamente igual al invocado. Si ocure un error retorna -1 no creando al hijo, de lo contrario el proceso padre obtiene el pid del proceso hijo que acaba de nacer, y el proceso hijo recibe el valor 0.

Se muestra un ejemplo donde "se crea un proceso hijo que imprime en pantalla el pid de su proceso padre, mientras que el proceso padre imprime en pantalla su propio pid y el del proceso hijo que ha creado. Para ello, se utilizan las llamadas al sistema getpid() y getppid(). El proceso padre, antes de finalizar se suspende hasta que el hijo acaba, para evitar que éste se quede zombie. Para ello, utiliza la llamada al sistema wait(), que recibe en la variable status el estado en que el proceso hijo finalizó

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main() {
  int pid = 0, status = 0;
  if ((pid = fork()) == -1) {
    printf(``Error al crear proceso hijo\n");
    exit(1);
  }
  if (pid == 0) { /* Proceso Hijo */
    printf(``El PID de mi proceso padre es %d\n", getppid());
    exit(1);
  }
  else { /* Proceso Padre */
    printf(``Mi PID es el %d y he creado un proceso hijo cuyo pid es %d\n",
           getpid(), pid);
    wait(&status);
    printf(``\nEl proceso hijo finalizo con el estado %d\n", status);
    exit(0);
  }
}."
```

Fuentes referenciadas en Internet fork()