Procesos e Hilos en C

Lucas Cedric Cervantes Beutelspacher

September 2021

1 Fork

La función fork se usa para la creación de procesos. La función fork genera un proceso adicional además de conservar el proceso original. El proceso original se denomina padre y el proceso adicional se denomina hijo.

Esta función de regresa un número que puede tomar tres valores: -1, 0, un numero positivo. Si regresa -1 significa que hubo un error al crear el proceso. Si regresa un numero positivo significa que nos encontramos en el proceso padre, y este número es el id del proceso padre. Si regresa 0 significa que estamos en el proceso hijo.

La funcion se invoca de la siguiente manera:

$$\text{pid} = \text{fork}()$$

donde pid es una variable que va a guardar el valor que regresa la funcion

2 Getpid

La función getpid se usa para conocer el id del proceso en el que se esté llamando y regresa un entero. También está la función getppid que regresa el id del proceso padre al proceso en el que se llama a la función. Si quisiéramos saber quién es el usuario propietario usaríamos la función getuid.

3 Wait

Wait es una función que pone en estado de espera a un proceso padre hasta que termine uno de sus procesos hijos. La función toma como parámetros una dirección de memoria (apuntador) donde se va a almacenar lo que sea que regrese el proceso hijo, si es que regresa algo. Regresa el id del proceso hijo que termina.

4 Pipe

Un pipe cuenta con funciones read, write y close, como si fuera un archivo. En el caso de read se necesitan pasar como parámetros el pipe que se va a estar

usando (en particular el primer elemento del arreglo), el lugar donde se va a guardar mensaje, y el tamaño del mensaje que se quiere leer. Write toma como parámetros el segundo elemento del pipe, el mensaje que se quiere mandar y la longitud del mensaje.

Para crear un pipe se tiene que declarar un arreglo de enteros de dos elementos y pasar este arreglo a la función pipe.

```
int pipeArr[2];
pipe(pipeArr);
```

Close se usa para cerrar el canal de conexión entre los procesos. Deponiendo de la acción se va a requerir un parámetro diferente para la función close. Si se está escribiendo se pasara el primer parámetro del pipe, y se está leyendo se pasara el segundo parámetro del pipe.

Para escribir y leer:

```
write(pipeArr[1], mensaje, longitud del mensaje)
close(pipeArr[0])
read(pipeArr[0], mensaje, longitud del mensaje)
close(pipeArr[1])
```

5 Hilos

Para poder usar hilos se necesita usar

$$include < pthread.h >$$

Un hilo, a diferencia de un proceso necesita ser definido. Por esto nos referimos, a las acciones que va a realizar, que recursos va a usar, etc. Al no ser una copia del proceso original no va a ser creado con ningún recurso del proceso original. En código esto se hace definiendo funciones en las cuales definiremos que hará el hilo y en los parámetros definiremos que recursos del proceso original va a necesitar.

Para la creación de hilos es necesario primer crear un objeto de tipo pthread_t y posteriormente iniciar dicho hilo.

```
pthread_t h1;
pthread_create(h1, NULL, function, parametrons de la function);
```