Técnicas Digitales III

Trabajo práctico: Señales

1. Compile y ejecute el programa (sig01.c):

   ```
   Compile el programa        gcc -o sig01 sig01.c
   Ejecute                    ./sig01
   ```

   El programa ejecuta una espera activa con la función **`sleep()`**. Espere a que el programa finalice normalmente. Para observar el estado de salida (exit status) ejecute en consola:

   ```
   > echo $?
   ```

   Ejecute nuevamente el programa sig01, pero finalice el mismo con combinación de teclas CONTROL+C en consola. Vuelta a verificar el estado de salida. ¿Qué observa? ¿Varía este valor si el programa termina en forma normal o en forma abrupta?

2. Compile el programa seg02.c. Este programa imprime en pantalla su PID, y luego ejecuta un bucle infinito. Con la función **`signal()`** se ignora la señal SIGKILL. Ejecute el programa y lea el PID del proceso y desde otra consola ejecute,

   ```
   kill -SIGKILL PID
   ```

   ¿Qué sucede?.

3. En el programa seg02.c del ejercicio 2 ignore la señal SIGTSTP. Ejecute el programa y desde la misma consola ejecute la combinación de teclas CONTROL+Z. ¿Qué sucede?. Ejecute la combinación de teclas CONTROL+C. ¿Qué observa?.

4. Escriba un programa seg03.c que imprima su PID y ejecute un bucle infinito. Además, programe un manejador para la señal SIGKILL que imprima el texto “me rehuso a terminar” cada vez que reciba dicha señal:
   `write(STDOUT_FILENO,"Me rehuso a terminar\n", sizeof("Me rehuso a terminar\n"));`

   Ejecute el programa, lea el PID del proceso y desde otra consola ejecute,

   ```
   kill -SIGKILL PID
   ```

   ¿Qué sucede?.

5. Modifique el programa seg03.c del ejercicio anterior para que el manejador sea para la señal SIGUSR1. Ejecute el programa, lea el PID del proceso y desde otra consola ejecute,

   ```
   kill -SIGUSR1 PID
   ```

¿Qué observa?.

6. Escriba un programa seg04.c que cree tres procesos hijos que ejecuten un bucle infinito. Luego de una espera activa, el proceso padre debe finalizar cada uno de los procesos enviando a cada uno de ellos la señal SIGKILL. Para ello utilice la función **`kill()`**.

7. Escriba un programa seg05.c que realice las siguientes tareas:

   a. Cree un proceso hijo que haga una espera activa de dos segundos y finalice con la función **`exit(0)`**.

   b. Luego de crear el proceso hijo, el padre debe ejecutar una espera activa por más de 10 segundos. Luego de la finalización del proceso hijo, verifique con **`pstree`** si este está en estado zombie.

   c. Establezca en el proceso padre un manejador para la señal SIGCHLD. En el manejador de esta señal se debe leer el estado de finalización del hijo con la función **`wait(&status)`**. Además se debe mostrar lo devuelto por la función `wait()` y el valor de status. ¿En este caso, el proceso hijo no queda en estado zombie?