Técnicas Digitales III

Trabajo práctico: Procesos

Creación de procesos en C

1. Compile y ejecute proc_01.c

Compile el programa	`$ gcc -o proc_01 proc_01.c`
Ejecute	`$ ./proc_01`

   ¿Qué es el PID ? ¿Cómo es el PID del padre respecto del PID del hijo? ¿Por qué?

2. Compile y ejecute proc_02.c. Luego, desde otra consola, ejecute el comando "`pstree -p`" e identifique los procesos en ejecución.

Compile el programa	`$ gcc -o proc_02 proc_02.c`
Ejecute	`$ ./proc_02`
Visualice el árbol de procesos	`$ pstree -p`

3. El programa proc_03.c ejecuta 2 veces la función `fork()` y hace una espera activa de 30 segundos con la función sleep(). Compile y ejecute proc_03.c. Luego, desde otra consola ejecute el comando `pstree -p` e identifique los procesos en ejecución.

4. La función `fork()` devuelve el pid del proceso hijo cuando lo ejecuta el padre; y devuelve 0 (cero) cuando lo ejecuta el proceso hijo. ¿Qué estructura de bifurcación de C le parece más conveniente para implementar que padre e hijo ejecuten diferente código (`if, while, for, case`)? Modifique proc_02.c con la estructura de bifurcación seleccionada. Compile y ejecute el programa.

5. Compile y ejecute proc_05.c. La variable *x* es inicializada en 100. Luego es decrementada por el proceso hijo e incrementada por el proceso padre. ¿Por qué *x* nunca retorna a su valor original?

6. Tome proc_02.c y ponga las funciones de las líneas 14 y 15 dentro de un bucle que se repita 3 veces. Imprima también el valor de la variable de control del bucle (variable i). Analice y deduzca cuántos hijos son creados. Justifique su respuesta. ¿Qué sucede con el valor de i?.

Creación de procesos en Python

7. Analice el contenido del archivo proc_01.py y ejecutelo. ¿Qué diferencias observa respecto a la creación de procesos en C?

8. Analice el contenido del archivo proc_02.py y ejecutelo. ¿Qué tipo de variable es la variable suma?

Procesos huérfanos y zombies en C

9. Compile y ejecute el programa proc_09.c. Ejecute en otra consola:

```
$ pstree -p > pstree.txt
$ ps aux | grep proc_09
```

¿Qué observa a la salida de este comando?

10. Tome el programa proc_09.c y fuerce a que el proceso hijo haga una espera activa de 30 segundos con la función `sleep()`. El proceso padre debe terminar antes que el proceso hijo.

Ejecute en otra consola:

```
$ pstree -p > pstree.txt
$ ps aux | grep proc_09
```

¿Qué sucede con el proceso hijo?. Identifique si persisten los procesos en cuestión. Observe los números de pid.

11. Tome el programa proc_09.c y fuerce a que el proceso hijo haga una espera activa de 30 segundos con la función `sleep()` y agregue al final del código del proceso padre la función `wait(NULL)`.

Ejecute en otra consola:

```
$ pstree -p > pstree.txt
$ ps aux | grep proc_09
```

Identifique si persisten los procesos en cuestión. Observe los números de pid.

12. Tome el programa proc_09.c y fuerce a que el proceso padre entre en un bucle infinito con la función `while(1).` Visualice luego los procesos en ejecución con `$`

```
$ pstree -p > pstree.txt
$ ps aux | grep proc_09
```

Identifique el proceso “zombie”.

Función execl() en C

13. Compile y ejecute proc_15.c. ¿Qué sucede al ejecutar la función execl()?

14. En el archivo proc_15.c, comente la línea 13 y descomente la línea 14. Compile y ejecute. ¿Qué observa por consola? ¿Por qué ha cambiado la salida respecto al Ej. 13?.