Laboratorio 3 - Procesos, Hilos y Planificación

Nota: respondan las preguntas que se realizan en el laboratorio, en el archivo `respuestas.txt`, indicando el ejercicio.

Ejercicio 1

El programa `proc.c` crea tantos procesos hijos como se le indique por la línea de comandos. Ejecutarlo creando 5, 10 y 20 procesos hijos. ¿Por qué es posible que el orden de terminación de los procesos no sea el de su creación?

Ejercicio 2

Completar el programa `thread.c`, que debe crear un hilo que imprime `¡Hola!` múltiples veces por pantalla. Estas son las funciones que deben completar:

- `pthread_create` (*manual*): crea un hilo.
- `pthread_join` (*manual*): espera por que un hilo específico finalice.
- `pthread_exit` (*manual*): termina con la ejecución de un hilo.

Una vez que el programa este completo, al ejecutarlo tendrían que ver una salida similar a la siguiente:

```
$ bin/thread 3
[0] ¡Hola!
[1] ¡Hola!
[2] ¡Hola!
El hilo finalizó.
$
```

Una vez que tengan el programa funcionando, responder:

1. ¿Que sucede si comentan la funcion `pthred_join()`, y vuelven a ejecutar el programa? ¿Por qué?
2. Mantener comentado `pthread_join()`, y reemplazar en `main()` la invocación a `exit()` por una invocación a `pthread_exit()`. ¿Qué sucede ahora? ¿Por qué?

Ejercicio 3

Completar el programa `threads.c`, que crea tantos hilos como se le indique por la línea de comandos. Cada hilo imprime un mensaje, y un número que le es pasado como parámetro al momento de crearlo. Deben completar la invocación de las mismas funciones que en el **Ejercicio 2**, más la función `pthread_detach` (*manual*).

En este programa al hilo `main` no le interesa el resultado de los hilos creados, por lo que no realiza luego un *join* sobre cada hilo. La funcion `pthread_detach()` indica esto, de manera que, cuando el hilo invoca `pthread_exit()`, el sistema operativo sabe que no debe mantener ningún dato del hilo (por ejemplo, para que otro hilo pueda realizar un *join* sobre el mismo).

Ejercicio 4

Completar el programa `forkvsthread.c`, que crea una cierta cantidad de procesos hijos o de hilos, según se le indique. Cada vez que crea un proceso hijo o hilo, espera a que el mismo termine antes de continuar creando el resto. Deben completar las invocaciones necesarias para crear los hilos (las mismas que en el **Ejercicio 2**).

Lo que vamos a hacer en este ejercicio es comparar cuanto cuesta crear un mismo número de hilos comparado con el mismo número de procesos hijos. Para esto vamos a utilizar el comando `time`, que nos indica cuanto tiempo consumió la ejecución de un programa.

Una vez que tengan el programa funcionando, ejecutar los siguientes comandos:

```
$ /usr/bin/time -p bin/forkvsthread 1 1000
$ /usr/bin/time -p bin/forkvsthread 2 1000
```

Nota: es importante usar el *path* completo (`/usr/bin/time`), para no ejecutar el comando `time` provisto por el *shell*.

Responder:

1. ¿Cual de las dos variantes tuvo menos costo, la creación de hilos o la creación de procesos? ¿Por qué?

¡Fin del Laboratorio 3!