Sesión 3: Threads

Objetivo: Crear/destruir/gestionar threads en Linux

Ejercicio 1:

Escribe un programa, llamado ej1.c, que cree un thread. Este thread, cuando se ejecute tiene que mostrar un mensaje por la salida estándar y acabar. El primer thread tiene que esperar la finalización del thread creado para terminar.

Llamadas al sistema que se tienen que utilizar:

int pthread_create();

int pthread_exit();

int pthread_join();

Ejercicio 2:

Escribe un programa, llamado ej2.c, en el cual existe una variable global a, de tipo long. Este programa tiene que crear un nuevo thread. El thread principal tiene que decrementar de forma monótona el valor de la variable a, mientras que el nuevo thread tiene que incrementar el valor de esta variable. Después de incrementar/decrementar el valor de la variable, ambos tienen que mostrar por la salida estándar el valor de esta variable.

¿El resultado de las ejecuciones de estos threads muestran los resultados que esperabas? ¿Por qué la variable a va cogiendo esos valores?

Ejercicio 3:

Escribe un programa, llamado ej3.c, en el que declares un vector global de 10 posiciones donde cada posición es un long. Crea un thread nuevo. Tanto el thread principal como el nuevo thread, tienen que recorrer todas las posiciones del vector incrementando en 1 el valor de cada posición. Este proceso lo tienen que realizar, cada uno, 10000 veces. Para acabar, el thread principal, una vez haya acabado el nuevo thread, tiene que mostrar el contenido de cada posición del vector.

¿El resultado de las ejecuciones de estos threads muestran los resultados que esperabas?

Ejercicio 4:

Escribe un programa, llamado ej4.c, en el que declares un vector global de 10 posiciones donde cada posición es un long. Crea tantos threads secundarios como posiciones tiene el vector. Cuando crees cada uno de estos threads, se le tiene que pasar como parámetro la posición con la que tiene que trabajar de tal forma que dos o más threads no trabajen con la misma posición. Cada uno de estos threads secundarios tienen que incrementar 10000 veces el contenido de su posición. Finalmente, el thread principal, una vez todos los threads secundarios hayan terminado, tiene que mostrar el contenido de cada una de las posiciones del vector.