Los problemas que pueden presentar los hilos, como ya he dicho, los contemplaré en entradas posteriores. Por tanto a partir de ahora hablaré acerca de cómo funcionan en C. Las funciones necesarias para el manejo de hilos en C se encuentra en la biblioteca **pthread.h**. Estas funciones son:

1. **int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*(\*rutina)(void \*), void \*arg)**
2. **pthread\_t pthread\_self(void)**
3. **int pthread\_join(pthread\_t thread, void \*\*value)**
4. **int pthread\_exit(void \*value)**

En***pthread\_create*** tenemos 4 parametros. El **primer parámetro guardará el identificador del thread** que creamos. El **segundo parámetro** es un puntero a una struct con los **atributos del hilo** (podemos pasar NULL). Estos atributos los comentaré en otra entrada. El **tercer parámetro** es un tipo de puntero especial, este puntero es un puntero que recibirá una función. **Esta función será lo que ejecute nuestro hilo**. Debemos tener en cuenta que **se pasa la dirección de esta función no los parámetros**, para esto tenemos **el último argumento** que es un puntero  y tan solo podemos pasar un parámetro. Como tan solo se puede pasar un parámetro se recomienda que sea una struct que contenga todos los datos que deseamos.

La función **pthread\_t pthread\_self**devuelve el identificador de hilo del hilo que la ejecuta. Por otra parte tenemos la función **pthread\_join**esta función **servirá para esperar a otro hilo**. Recibe un identificador de hilo (parámetro thread) al que debemos esperar. El segundo parámetro es el valor de terminación del hilo. La última función es**pthread\_exit** y es la que finaliza el hilo. Recibe un parámetro que será el valor de terminación (en forma de puntero).

Tres de estas funciones devuelven un int, es decir en caso de error devuelve -1. Todo lo anteriormente mencionado se puede mostrar con un ejemplo bastante sencillo:

#include

#include

#include

struct param{

char \*frase;

int numero;

};

/\*

\* Función que ejecutarán los hilos.

\*/

void hiloMensaje(struct param \*mensa){

printf("%s %d\n",mensa->frase, mensa->numero);

}

int main() {

pthread\_t thd1, thd2;

struct param param1 = {"Soy el hilo: ",1};

struct param param2 = {"Digo otra cosa ",2};

/\*

\* Creamos dos hilos

\*/

/\* La función la pasaremos como (void\*)nombreFuncion.

\* Es decir, hacemos un CAST a void\*.

\* También es importante realizar esto con la dirección de memorio

\* de la variable que contiene los parámetros.

\*/

pthread\_create (&thd1, NULL, (void\*)hiloMensaje, (void\*)&param1);

pthread\_create (&thd2, NULL, (void\*)hiloMensaje, (void\*)&param2);

/\*

\* Esperamos la finalización de los hilos

\*/

pthread\_join(thd1, NULL);

pthread\_join(thd2, NULL);

printf("Han finalizado los thread.\n");

}

*En algunos casos aparece una línea 39 con etiquetas de cierre. Esto es debido a un error del script que escribe el código para que se pueda copiar y pegar. Es decir se debe ignorar esta línea de código.*  
Veamos una de las posibles ejecuciones de este programa:

[](http://2.bp.blogspot.com/-FABZKEuBf1k/Ts6mu84JY0I/AAAAAAAAADw/ycwFcO7UGo4/s1600/ejecucion.jpg)

Los hilos pueden ejecutarse en otro orden. Eso depende del algoritmo de planificación de nuestro sistema. Cabe mencionar que para compilarlo (mi SO es Debian 32) he empleado el comando: gcc hilos.c  -lpthread -o hilo