Sockets y Threads

Práctica 1

1) Tomar el ejemplo “server_tcp.c” y modificarlo para que el mismo pueda aceptar múltiples conexiones. Para ello se lanzará un thread cada vez que se acepte una nueva conexión, dicho thread se encargará de recibir datos del cliente y enviar como respuesta lo mismo que recibió (echo).

Se deberá utilizar la biblioteca “ClientData.c” provista para el ejercicio, la cual modeliza un cliente que se conecta al servidor. En “ClientData.h” se define la siguiente estructura:

```
struct S_ClientData
{
	int fd;
	pthread_t thread;
	int flagFree;
};
typedef struct S_ClientData ClientData;
```

Se deberá definir una lista (array) del tipo ClientData de 100 posiciones.

Los campos se deberán cargar luego de aceptar la conexión, de la siguiente manera:

- fd: File descriptor devuelto por la función “accept()”
- thread: variable pthread_t utilizada para crear el thread
- flagFree: En 0 para indicar que la posición del array está ocupada (se utilizará una lista de ítems del tipo ClientData)

Cuando se acepta una conexión, se utilizará la función “cd_getFreeIndex()” para obtener el índice de la lista de un ítem libre (sin datos de ningún cliente). El mismo se utilizará para cargar los datos y luego lanzar el thread.

El thread deberá recibir como argumentos un puntero al ítem ClientData, el cual posee en uno de sus campos el filedescriptor de la conexión aceptada, el cual se utilizará para enviar y recibir datos.

La biblioteca ClientData también posee la función “cd_init()” la cual debe ejecutarse al comienzo del programa para inicializar la lista vacía.

2) Capturar la signal SIGINT (ctrl+c) y finalizar todos los threads y sockets abiertos antes de finalizar el proceso.