Documentación de Simple network library

Version de Simple network library: 0.0.1
Siglas de Simple network library: SNL
Fecha de la ultima revision: 7/6/2009
Total de paginas: 26

Copyright © 2009 Jesús Hernández Gormaz.
Se concede permiso para copiar, distribuir y/o modificar este documento bajo los términos de la Licencia de Documentación Libre de GNU, Versión 1.3 o cualquier otra versión posterior publicada por la Free Software Foundation; sin Textos de Cubierta Delantera ni Textos de Cubierta Trasera, siendo las Secciones Invariantes Comentarios del autor original. Una copia de la licencia está incluida en la sección titulada GNU Free Documentation License.

Introducción

La biblioteca Simple network library, o por sus siglas SNL, comenzó a ser escrita el 22 de Marzo del año 2009 d.c. con la intención de escribir una biblioteca para crear aplicaciones que se comunicasen con otras a través de una red, como por ejemplo internet, sin depender de otras bibliotecas sino realizando llamadas al sistema sin que esto supusiese tener que estar manejando complejas funciones ni escribir una gran cantidad de lineas de código para el programador, sino que fuese una biblioteca de fácil utilización y que proporcionase al programador las estructuras de datos necesarias para poder escribir aplicaciones de red de forma sencilla y funciones que simplifiquen la comunicación a través de la red.

La primera versión de SNL se consiguió el 5 de Abril del año 2009 d.c. Y aunque aun fuese una versión en pruebas en busca de posibles errores que pudiesen hallarse al usar SNL ya podía utilizarse para programar servidores y clientes que pudiesen comunicarse por una red de forma sencilla. Esta primera versión es la 0.0.0 y soportaba los protocolos IPv4 e IPv6 contaba con una función para resolver nombres de dominios mediante DNS, trabajaba con los protocolos TCP y UDP, permitía manejar grupos de sockets como grupos de conexiones con facilidad y de forma casi intuitiva. Los grupos de conexiones consiguió dar la facilidad para manejar comunicaciones a través de una red que se buscaba al comenzar a programar SNL, admitía añadir al grupo una conexión como servidor a la escucha de forma que recibiese un trato diferente al trabajar con el grupo, comprobar nuevas conexiones entrantes en una conexión a la escucha o nuevos datos que se recibiesen en una conexión sin quedar la ejecución bloqueada de forma indefinida, determinar fácilmente el tiempo máximo que se estaría a la espera de alguno de los sucesos anteriores, un byte con el que poder diferenciar un grupo o unos determinados grupos que pudiese asignarlo arbitrariamente el programador y otras características. Además de las funciones para la comunicación a través de redes o para manejar grupos de conexiones, SNL contaba con funciones con las que controlar la correcta ejecución de la biblioteca, la recuperación tras errores y la prevención de los mismos o de otros de mayor gravedad.

Entre otras cosas SNL puede impedir la ejecución de algunas funciones o todas para impedir que se puedan producir errores que pudiesen hacer inestable la ejecución del programa. Si bien el sistema de códigos de errores con los que el programador poder saber en tiempo de ejecución que error había tenido lugar era reducido todavía, como el funcionamiento del bloqueo de funciones de SNL lo llevaba a cabo automáticamente la propia biblioteca, el que aun los códigos de errores fuesen demasiado generales no suponía un problema grave ya que el programador no tenia que ser responsable de bloquear la ejecución de las funciones de SNL, solo de comprobar si había ocurrido algún error o no.

SNL se ha escrito buscando que sea portable, esto se ha conseguido ya que SNL puede compilarse en sistemas GNU/Linux, en Open Solaris, en otros sistemas derivados de Unix y en sistemas windows. En la versión 0.0.0 el makefile de SNL solo permitía compilar SNL en GNU/Linux pero el código fuente podía ser compilado en otros sistemas compilándolo manualmente e instalándola también de forma manual. El makefile además de compilar e instalar la biblioteca también permitía desinstalarla.

La versión de SNL que abordaremos en esta documentación va a ser esta misma, la versión 0.0.1 que es la ultima versión oficial de SNL actualmente a 15 de Junio del año 2009 d.c. (según el calendario gregoriano). Ahora sin más preámbulos comencemos con Simple network library. En esta versión las funciones para IPv4 se conservan y hay disponibles funciones para IPv6 mas las funciones para IPv6 están menos probadas que las de IPv4 actualmente por aun no estar usándose IPv6 de forma completa y total en todo internet antes del 2011, que es para cuando se espera pasar de IPv4 a IPv6 de una progresiva, se espera poder comprobar el funcionamiento de las funciones para IPv6 más y mejor y poder corregir cualquier error que surgiese. Se ha intentado que las diferencias al usar las funciones de IPv6 en contraste con las de IPv4 sea mínimo, consiguiéndose en la mayoría. En esta nueva versión, entre otras cosas, se ha optimizado el código fuente y se han añadido algunas funciones para trabajar con grupos de conexiones de forma que se pueda aprovechar mejor el uso de grupos, también se han modificado las declaraciones de las funciones con una compilación condicional para que la biblioteca pueda usarse igual en C que en C++.

Lista de funciones de SNL

aquí tienen la lista de todas las funciones de SNL que puede usar en su programa, cada función va acompañada de una breve explicación de la función:

- int SNL_emergencia(void);
  - Esta función permite desbloquear la ejecución de todas las funciones de la biblioteca, no es recomendable usarla de forma arbitraria ya que podría provocar la inestabilidad de la ejecución del programa pudiendo provocar errores más graves que los que han provocado el que SNL bloquee algunas o todas las funciones. No es hay que pasarle ningún argumento al llamarla y devuelve 0 si todo fue bien, -1 si hubo algún problema durante la ejecución y -2 si incluso esta función ha sido bloqueada y en este caso se debería tener en cuenta el no usar ninguna función de la biblioteca durante el resto de la ejecución de un programa o incluso el acabar la ejecución del mismo.
- unsigned int SNL_ultimo_error(void);
  - No hay que pasarle ningún argumento y devuelve el valor del código de error del ultimo error de SNL que se produjo.
- int SNL_comprobado_ultimo_error(void);
  - Con esta función se puede comprobar si el ultimo error ha sido comprobado con la función SNL_ultimo_error() o no, devuelve 0 si el ultimo error ya fue comprobado y -1 en caso contrario, no hay que pasarle ningún argumento. Como todas las funciones que devuelven números enteros con signo, devuelve -2 si esta bloqueada la ejecución de la función.
- unsigned long int informacion_biblioteca_SNL(char *informacion, unsigned long int longitud);
  - Con esta función se puede obtener información de SNL como el nombre de la biblioteca, las siglas, la versión, la fecha de creación, el nombre del autor de SNL, la fecha y hora de la compilación de SNL y otra información de SNL. Recibe como argumentos un puntero a una cadena de caracteres donde se copiara la cadena de caracteres con toda la información de SNL y la longitud en caracteres de dicha cadena. Devuelve 0 si todo fue bien y en caso de error devuelve la longitud necesaria para poder copiar la cadena de caracteres.
- int EncriptarTexto_CifradoBitsIguales(unsigned char *texto, unsigned char *clave);
  - Esta es una función de cifrado simétrico muy simple y que no se recomienda usar para transmitir datos que tengan que ir cifrados de forma segura, el uso de esta función esta más orientado a videojuegos donde los datos que se cifran no son de demasiada importancia más allá del propio juego como por ejemplo para partidas en red en las que sea necesaria un clave para formar parte de la partida. Si necesita cifrar datos con un nivel de seguridad normal o alto necesitara una biblioteca con funciones criptográficas. La función cifra realizando un XOR a cada byte de la cadena a cifrar con cada byte de la cadena clave y un NOT a cada byte de la cadena obtenida tras el XOR anterior. La función recibe como primer argumento un puntero a la cadena que se quiere cifrar y como segundo argumento un puntero a la cadena que se usara de clave, la cadena cifrada la devolverá en el puntero que se le paso como primer argumento, tengalo en cuenta. Devuelve 0 si todo fue bien y -1 si ocurrió algún error; como en todas las funciones que devuelven un int si devuelve -2 es que no puede ejecutarse.
- unsigned int checksum(SNL_datos datos, unsigned long longitud);
  - Recibe como argumentos SNL_datos y la longitud en bytes de dichos datos y devuelve el checksum, o suma de verificación, de los bytes que se pasaron como primer argumento. Es igual a la que se utilizan en los paquetes TCP, o en otros tipos de paquetes, para verificar de forma sencilla la integridad de los datos del paquete para poder detectar la perdida de datos accidental.
- grupo_conexiones *SNL_nuevo_grupo_conexiones(int conexion_escucha);
  - Devuelve un puntero a un grupo de conexiones nuevo y recibe como argumento la conexión de escucha que se añadirá al grupo (solo conexiones de escucha) para añadirla a la vez que se reserva memoria para el nuevo grupo, o SIN_CONEXION si no se añadirá ninguna conexión de escucha. Cada grupo solo puede tener una o ninguna conexión de escucha pues los grupos

han sido diseñados para poder tratarlos como servidores en una aplicación servidor de forma que cada grupo estuviese formado por una conexión de escucha y todas las conexiones entrantes que se fuesen aceptando; o en el caso de una aplicación que funcionase como cliente, que la conexión de escucha fuese SIN_CONEXION y el grupo estuviese formado por todas las conexiones que tuviese abiertas con las aplicaciones servidores. Es recomendable no mezclar tipos de conexiones diferentes en un mismo grupo (como las conexiones TCP/IPv4, TCP/IPv6, UDP/IPv4 o UDP/IPv6) para evitar que el programador tenga que crear y mantener variables con las que diferenciar de que tipo es cada conexión, por lo que no mezcle conexiones de tipos diferentes en un mismo grupo de conexiones.

- int SNL_nueva_conexion_grupo(int conexion, grupo_conexiones *grupo);
  - Añade la conexión que se pasa como primer argumento al grupo de conexiones que se pasa como segundo argumento como puntero. Devuelve 0 si todo fue bien y -1 en caso contrario.
- int SNL_quitar_conexion_grupo(int conexion, grupo_conexiones *grupo);
  - Igual que SNL_nueva_conexion_grupo pero esta quita la conexión del grupo de conexiones sin cerrarla. Devuelve 0 si todo fue bien y -1 en caso contrario.
- int SNL_desconectar_conexion_grupo(int conexion, grupo_conexiones *grupo);
  - Exactamente igual que SNL_quitar_conexion_grupo pero esta función cierra la conexión después de quitarla del grupo de conexiones. Devuelve 0 si todo fue bien y -1 en caso contrario.
- int SNL_tiempo_espera_grupo(grupo_conexiones *grupo, int segundos, int microsegundos);
  - Con esta función se cambia el tiempo que se esperara a que un socket del grupo de conexiones reciba datos o tenga, el socket en escucha, una nueva conexión entrante. El asignar valores negativos como tiempo podría llevar a error o a que la función SNL_comprobar_grupo_conexiones quedara bloqueada a la espera de uno de los sucesos ya mencionados, el asignar 0 como tiempo de espera significaría que según se revisasen los sockets no se esperaría a ninguno de los sucesos anteriores. De forma predeterminada cada grupo tiene configurado de forma óptima para la mayoría de los casos este tiempo de espera. El primer argumento es el grupo pasado como puntero, el segundo los segundos de espera y el tercero los microsegundos de espera, téngase en cuenta que 1 μs son 1000000 s. Devuelve 0 si todo fue bien y -1 en caso contrario.
- int SNL_cambiar_id_grupo(grupo_conexiones *grupo, char id);
  - Recibe como primer argumento un puntero a un grupo y como segundo argumento un caracter (1 byte), devuelve 0 si todo fue bien y -1 en caso contrario. Cambia el byte de identificación de un grupo de conexiones el cual puede ser utilizado de forma arbitraria por el programador para diferenciar un grupo de conexiones de otro, para diferenciar que tipo de conexiones forman el grupo de conexiones, para distinguir unos determinados grupos de otros o cualquier otra utilidad que le encuentre el programador y necesite. Este byte de identificador es para el uso del programador y SNL no lo usa absolutamente para nada por lo que se puede cambiar sin miedo ya que no influye en el funcionamiento de SNL.
- char SNL_id_grupo(grupo_conexiones *grupo);
  - Recibe como único argumento un puntero a un grupo en conexiones y devuelve el caracter (1 byte) de identificación del grupo de conexiones pasado como argumento.
- int SNL_pertenece_a_grupo_conexiones(int conexion, grupo_conexiones *grupo);
  - Esta función sirve para saber si una conexión forma parte de un grupo de conexiones, recibe como primer argumento la conexión sobre la que se pregunta si pertenece al grupo, como segundo argumento el puntero al grupo de conexiones y devuelve 0 si la conexión pertenece al grupo, 1 si no pertenece al grupo y -1 si ocurrió algún error.
- int SNL_enviar_grupo_TCP(grupo_conexiones *grupo, const SNL_datos datos, int bytes, int flags);
  - Envía unos mismos datos a cada socket perteneciente al grupo de conexiones con los mismos flags (excepto a la conexión en escucha, solo puede usarse con grupos en los que todas las conexiones sean del tipo TCP pues si hay alguna conexión de otro tipo se produciría un error). Su primer argumento es un puntero a un grupo de conexiones, el segundo los datos a enviar, el

tercero es el numero de bytes de los datos que se enviaran y el cuarto es el flag que se enviara con el paquete TCP y puede dejarse como 0 si no se quiere usar. Devuelve 0 si todo fue bien y -1 si hubo algún problema enviando los datos a alguna de las conexiones del grupo.

- int SNL_comprobar_grupo_conexiones(grupo_conexiones *grupo);
  - Comprueba todas las conexiones del grupo pasado como puntero en el primer argumento para posteriormente poder saber si alguna conexión ha recibido datos o si hay alguna conexión entrante en espera. Devuelve 0 si todo fue bien y -1 en caso contrario.
- int SNL_conexion_entrante_grupo(grupo_conexiones *grupo);
  - Comprueba si hay alguna conexión entrante esperando ser aceptada (para lo que es necesario que el grupo tenga una conexión de escucha), recibe como argumento un puntero a un grupo y devuelve 0 si no hay ninguna conexión entrante, 1 si hay conexiones entrantes y -1 en caso de error. Con esta función no es necesario llamar antes a SNL_comprobar_grupo_conexiones ya que lo hace la propia función
- int SNL_conexion_activa_grupo_conexiones(grupo_conexiones *grupo);
  - Recibe como argumento un puntero a un grupo de conexiones y devuelve una conexión activa del grupo, una conexión que haya recibido datos o que se cerrara por la maquina remota. En caso de no haber ninguna conexión activa devuelve SIN_CONEXION.
- int SNL_conexion_escucha_grupo(grupo_conexiones *grupo);
  - Recibe como argumento un puntero a un grupo de conexiones y devuelve la conexión de escucha del grupo o SIN_CONEXION si no hay ninguna conexión de escucha.
- int SNL_cambiar_escucha_grupo(grupo_conexiones *grupo, int conexion);
  - Recibe como argumentos un puntero a un grupo de conexiones y una conexión y cambia la conexión de escucha del grupo pasado como primer argumento a la conexión pasada como segundo argumento. Si el grupo ya tenia una conexión de escucha anteriormente esta función no se encarga de cerrarla, por lo que esto deberá hacerlo el programador antes de cambiar la conexión de escucha del grupo.
- int SNL_liberar_grupo_conexiones(grupo_conexiones *grupo);
  - Recibe como argumento un puntero a un grupo y libera la memoria dinámica de dicho grupo, pero no cierra las conexiones del grupo. Devuelve 0 si todo fue bien y -1 en caso contrario.
- int SNL_cerrar_grupo_conexiones(grupo_conexiones *grupo);
  - El funcionamiento de esta función es igual al de SNL_liberar_grupo_conexiones pero antes de liberarlo cierra todas las conexiones del grupo. Devuelve 0 si todo fue bien y -1 en caso contrario.
- int SNL_conectar_TCP_IPv4(char *direccion, unsigned short int puerto);
  - Con esta función es posible conectarse a un servidor usando el protocolo TCP/IPv4, recibe como argumentos una cadena de caracteres con la dirección IP a la que conectarse (en el caso de IPv4 seria algo como “192.1.1.67”, por ejemplo) y el puerto al que se debe conectar (recuerde que los puertos hasta el 1023 son usados para fines específicos como puede ser el protocolo HTTP en el 80 o el SMTP en el 25 por lo que para usar puertos inferiores al 1023 tendrá que ejecutar la aplicación con los privilegios y configuración que sea necesaria en el sistema en que se ejecute, para evitar este problema puede usar puertos superiores al 1023 y el sistema le dejara usarlo sin problemas). Devuelve la conexión si todo fue bien o SIN_CONEXION si hubo algún problema.
- int SNL_conectar_UDP_IPv4(unsigned short int puerto);
  - Abre un socket UDP en el puerto que se especifique (véase la función SNL_conectar_TCP_IPv4 para el comentario sobre los puertos), el único argumento es el puerto que usar para la conexión ya que UDP no es un protocolo orientado a conexión como lo es TCP. Devuelve el socket UDP si todo fue bien o SIN_CONEXION si hubo algún error.
- int SNL_recibir_UDP_IPv4(int conexion, SNL_datos datos, int bytes, int flags, char *dirección_remota, unsigned short int *puerto_remoto, unsigned long int longitud_d);
  - Recibe como argumentos un socket UDP del que se van a leer los datos recibidos, el puntero donde se guardaran los datos (que puede ser simplemente una variable del tipo SNL_datos o un puntero a una cadena de caracteres), la longitud máxima en bytes de datos que puede leerse

de una sola vez, flags de la operación de lectura del socket (se puede dejar a 0 si no se quiere o sabe usar), un puntero a una cadena de caracteres donde se copiara la dirección que nos envío los datos, un puntero a un tipo unsigned short int donde se copiara el puerto de la maquina remota y, como ultimo argumento, la longitud en caracteres que puede tener como máximo la dirección de la maquina remota que se copiara al antepenúltimo argumento. Devuelve el numero de bytes recibidos, 0 si no se han recibido datos y -1 en caso de error en la función

- int SNL_enviar_UDP_IPv4(int conexion, SNL_datos datos, int bytes, int flags, char *dirección_remota, unsigned short int puerto_remoto);
  - Esta función es muy parecida a SNL_enviar_UDP_IPv4 pero con un argumento menos. Recibe como argumentos un socket UDP, los datos a enviar, la longitud de los datos a enviar en bytes (o en caracteres si se pasa un puntero a una cadena de caracteres), el flag que se usara en el para escribir en el socket el envío (se puede dejar a 0 si no se va a usar), un puntero a una cadena de caracteres con la dirección a la que se debe enviar y el puerto de la maquina remota al que debe enviarse. Devuelve 0 si todo fue bien o -1 en caso de error.
- int SNL_escuchar_TCP_IPv4(char *dirección, unsigned short int puerto, int cola);
  - Devuelve una conexión del tipo TCP/IPv4 que estará a la escucha de nuevas conexiones entrantes. Como argumentos hay que pasarle la dirección en la que estará a la escucha (puede pasarse NULL para que se escoja automáticamente, es lo más recomendable), el puerto en el que estará a la escucha y el numero de conexiones entrantes que puede haber en cola esperando ser aceptadas (lo normal es 20, y en algunos sistemas no se puede pasar de 20 o 25, más conexiones en cola es innecesario ya que una aplicación que este funcionando como servidor debería estar encargándose de aceptar las conexiones entrantes y atenderlas para evitar llegar a cubrir el máximo de conexiones en cola). En caso de algún error devuelve SIN_CONEXION.
- int SNL_aceptar_conexion_IPv4(int conexion_escucha, char *dirección_remota, int *puerto_remoto);
  - Devuelve una conexión entrante y recibe como argumentos la conexión de escucha, un puntero a una cadena de caracteres donde se copiara la dirección IPv4 de la maquina remota y un puntero donde se copiara el numero a un tipo int donde se copiara el puerto de la maquina remota. Devuelve SIN_CONEXION en caso de error.
- int SNL_DNS_IPv4(char *dirección, char *host);
  - Recibe como argumentos un puntero a una cadena de caracteres donde copiara la dirección IPv4 del host que se pase como segundo argumento y como segundo argumento el puntero a la cadena de caracteres con el host del que se quiere consultar la dirección IPv4 Devuelve 0 si todo fue bien y -1 en caso contrario.
- int SNL_conectar_TCP(char *dirección, unsigned short int puerto);
  - Exactamente igual que SNL_conectar_TCP_IPv4 pero esta función trabaja con el protocolo IPv6 en lugar de su antecesor IPv4, esto significa que la dirección IP de la cadena de caracteres que se pase como primer argumento tendría un formato de dirección IPv6 (por ejemplo, “FFFF:0000:0000:0000:0000:0000:0000:0012” o “FFFF::12”); de resto el funcionamiento de esta función es igual que su correspondiente para IPv4
- int SNL_conectar_UDP(unsigned short int puerto);
  - Absolutamente igual a SNL_conectar_UDP_IPv4 solo que trabaja sobre el protocolo IPv6.
- int SNL_recibir_UDP(int conexion, SNL_datos datos, int bytes, int flags, char *dirección_remota, unsigned short int *puerto_remoto, unsigned long int longitud_d);
  - Igual que SNL_recibir_UDP_IPv4 con la diferencia del formato de la dirección IP, que en este caso sera una dirección IPv6 (véase los comentarios sobre la dirección IPv6 en SNL_conectar_TCP).
- int SNL_enviar_UDP(int conexion, SNL_datos datos, int bytes, int flags, char *dirección_remota, unsigned short int puerto_remoto);
  - Igual que SNL_enviar_UDP_IPv4 con la diferencia del formato de la dirección IP, que en este caso sera una dirección IPv6 (véase los comentarios sobre la dirección IPv6 en SNL_conectar_TCP).

- int SNL_escuchar_TCP(char *dirección, unsigned short int puerto, int cola);
  - Igual que SNL_escuchar_TCP_IPv4 con la diferencia del formato de la dirección IP, que en este caso sera una dirección IPv6 (véase los comentarios sobre la dirección IPv6 en SNL_conectar_TCP).
- int SNL_aceptar_conexion(int conexion_escucha, char *dirección_remota, int *puerto_remoto, unsigned long int longitud_d);
  - Igual que SNL_aceptar_conexion con las diferencias de que como ultimo argumento hay que indicar la longitud en caracteres máxima que puede almacenar la cadena que se pasa como puntero como segundo argumento y la diferencia del formato de la dirección IP, que en este caso sera una dirección IPv6 (véase los comentarios sobre la dirección IPv6 en SNL_conectar_TCP).
- int SNL_desconectar(int conexion);
  - Finaliza una conexión tanto de sockets TCP como UDP tanto de IPv4 como de IPv6, recibe como único argumento la conexión a cerrar y devuelve 0 si todo fue bien y -1 en caso contrario.
- int SNL_enviar_TCP(int conexion, const SNL_datos datos, int bytes, int flags);
  - Recibe como argumentos la conexión por la que enviar datos, los datos a enviar, el numero de bytes (o de caracteres si como segundo argumento se pasa un puntero a una cadena) de los datos a enviar y el flag que indicar al escribir los datos en el socket. Funciona tanto con IPv4 como con IPv6. Devuelve 0 si todo fue bien o el numero de bytes que no se pudieron enviar.
- int SNL_recibir_TCP(int conexion, SNL_datos datos, int bytes, int flags);
  - Igual que SNL_enviar_TCP pero con esta función leemos los datos recibidos por la conexión que se pasa como primer argumento, los datos leídos los copia a la variable SNL_datos pasada como segundo argumento (o a la cadena de caracteres pasada como puntero), el tercer argumento es el numero de bytes que pueden leerse de una sola vez y el ultimo argumento el flag a tener en cuenta al leer del socket. Devuelve el numero de bytes leídos o 0 si la conexión fue cerrada por la maquina remota y en este caso cierra automáticamente el socket, también puede devolver -1 en caso de error.
- int SNL_DNS(char *dirección, char *host, char *servicio, unsigned long int longitud_d, unsigned long int longitud_s);
  - Esta es la función de DNS para IPv6, el primer argumento es un puntero a donde se guardara la cadena de caracteres de la dirección IP, el segundo es un puntero la cadena de caracteres del dominio que se quiere consultar, el tercero es un puntero a una cadena de caracteres de el servicio (puede dejarlo como NULL), el cuarto es la longitud máxima de la cadena de caracteres de la IP y el ultimo la longitud máxima de la cadena de caracteres del servicio. Devuelve 0 si todo fue bien y -1 en caso contrario.
- int SNL_iniciar_W(void);
  - Esta función solo esta en SNL cuando se compila para los sistemas windows y hay que llamarla en dichos sistemas antes de usar la biblioteca, no tiene ningún argumento y devuelve 0 si todo fue bien y -1 en caso contrario.
- int SNL_acabar_W(void);
  - Esta función solo esta en SNL cuando se compila para los sistemas windows y hay que llamarla en dichos sistemas una vez se acaba de usar la biblioteca SNL y no se va a volver a usar en esa ejecución Devuelve 0 si todo fue bien y -1 en caso contrario. Si se llama a esta función y se quiere volver a usar alguna función de SNL hay que volver a llamar a SNL_iniciar_W.

Ejemplos de uso de SNL

Para comenzar escribiremos el código necesario para conectar un cliente a un servidor usando SNL para ello. Veamos el código primero:

```
/*
testSNL Copyright (C) 2009 Jesús Hernández Gormaz
```

```
Fecha de creación: 5 de abril del 2009 (Siglo XXI)

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License as
published by the Free Software Foundation; either versión 3, or
(at your option) any later versión
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Este programa es software libre. Puede redistribuirlo y/o
modificarlo bajo los términos de la Licencia Pública General
de GNU según es publicada por la Free Software Foundation,
bien de la versión 3 de dicha Licencia o bien (según su
elección) de cualquier versión posterior.
Este programa se distribuye con la esperanza de que sea
útil, pero SIN NINGUNA GARANTÍA, incluso sin la garantía
MERCANTIL implícita o sin garantizar la CONVENIENCIA PARA UN
PROPÓSITO PARTICULAR. Para más detalles, véase la Licencia
Pública General de GNU.

Debería haber recibido una copia de la Licencia Pública
General junto con este programa. En caso contrario, escriba
a la Free Software Foundation, Inc., en 675 Mass Ave,
Cambridge, MA 02139, EEUU.
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <SNL/SNL.h>

int main(int argc, char *argv[]){
if(argc < 2){
     printf("Para usar este ejemplo de SNL debe de indicar un
dominio");
     printf(" después del comando con el que ejecuta el programa,
por");
     printf(" ejemplo:\n %s www.fsf.org\n", argv[0]);
     exit(-1);
}else{
     /*la variable conexión guardara el identificador de la conexión
     el cual deberemos de pasarlo a las funciones cuando enviemos
     o recibamos datos, etc, etc, etc.*/
     int conexion;
     /*texto es el array que usaremos como buffer para enviar
     y recibir datos, con medio kilobyte es suficiente ya que
     normalmente las paginas pesan unos pocos kilobytes (sin
     contar las imágenes) y así nuestro ejemplo leerá varias
     veces el socket para poder recibir toda la pagina*/
     char texto[512];
```

```
    /*las dos variables siguientes son respectivamente para guardar
    la dirección del host, la IP, y para guardar un texto con la
    información de SNL*/
    char direccion[512], informacion[501];
    /*para trabajar con la conexión usaremos un grupo de conexiones
    por ser mejor que trabajar con la conexión directamente
    incluso cuando solo necesitamos una conexión, el grupo nos
    permite comprobar si hay datos esperando que los leamos en
    algun socket sin que la aplicación se quede a la espera de que
    haya datos en el socket*/
    grupo_conexiones *a;
    /*mostramos en terminal un texto que indica que el programa
    esta ejecutandose y que vamos a empezar con la conexion*/
    printf("\nEjecucion\n");
    /*empezamos a llenar texto con la peticion HTTP que enviaremos
    despues*/
    strcpy(&texto[0], "GET ");
    /*mostramos en pantalla el host al que nos conectaremos*/
    printf("Host: %s\n", argv[1]);
    /*usamos la función de DNS para IPv6 de forma que si se
    obtiene una dirección parecida a 1F::F0:AB es que la maquina en
    la que ejecutamos este ejemplo ya funciona con IPv6, pero este
    ejemplo es de IPv4, por lo que esta es la única función de IPv6
    que usaremos en este ejemplo por ser la que más difiere de su
    equivalente para IPv4, el primer argumento es la cadena donde
    queremos guardar la dirección IP, el segundo el host, el tercero
    es el servicio pero lo dejamos como null ya que no necesitamos
    conocer el puerto del servicio ni el servicio de un puerto
    en este ejemplo además de no haber modificado los ficheros de
    configuración del sistema para configurar este ejemplo como un
    servicio (esto limita los puertos que podemos usar), el cuarto
    la longitud máxima de la cadena de caracteres de la dirección
    (igual que antes al obtener el texto de la información de SNL)
    y el ultimo es la longitud del servicio, 0 como le pasamos NULL*/
    SNL_DNS(&direccion[0], argv[1], NULL, 1023, 0);
    /*imprimimos en la terminal el host y la Ipv6 obtenida*/
    printf("DNS IPv6 obtenido para %s: %s\n", argv[1], &direccion[0]);
    /*usamos la funcion de DNS para IPv4, el primer argumento es
    donde se copiara la direccion IPv4 y el segundo el host*/
    SNL_DNS_IPv4(&direccion[0], argv[1]);
    /*igual que antes, mostramos el host y la IPv4 en pantalla*/
    printf("DNS IPv4 obtenido para%s: %s\n", argv[1], &direccion[0]);
    /*continuamos rellenando el buffer con la peticion GET de HTTP
    que enviaremos*/
    strcat(&texto[0], "http://");
    strcat(&texto[0], argv[1]);
    strcat(&texto[0], "/");
    strcat(&texto[0], " HTTP/1.1\n\rHost: ");
    strcat(&texto[0], argv[1]);
    strcat(&texto[0], "\n\rAccept: text/html\n\rKeep-Alive:
300\n\rProxy-Connection: keep-alive\n\r\n\r");
    /*iniciamos una conexión de TCP sobre IPv4 en la direccion IP
    que obtuvimos antes en el puerto 80, el puerto usado para el
    protocolo HTTP*/
    conexion= SNL_conectar_TCP_IPv4(&direccion[0], 80);
    /*comprobamos si conseguimos conectarnos*/
```

```
    if(conexion == SIN_CONEXION){
        printf("problemas conectando\n");
        exit(-1);
    }
    printf("conectado\n");
    /*creamos un nuevo grupo de conexiones indicando como conexión
    de escucha SIN_CONEXION ya que en este ejemplo no aceptaremos
    conexiones como si fuese un servidor*/
    a= SNL_nuevo_grupo_conexiones(SIN_CONEXION);
    /*añadimos la conexión abierta al grupo*/
    SNL_nueva_conexion_grupo(conexion, a);
    /*enviamos el buffer que preparamos con la peticion HTTP
    y comprobamos que todo el buffer se enviara*/
    if(SNL_enviar_grupo_TCP(a, &texto[0], strlen(&texto[0]), 0) == 0){
        printf("enviado:\n\n");
        printf("%s\n\n", &texto[0]);
    }else{
        exit(-1);
    }
    /*cambiamos el tiempo de espera del grupo a que suceda algo en
    los sockets*/
    SNL_tiempo_espera_grupo(a, 2, 0);
    /*esta funcion se encargara de comprobar todas las conexiones
    del grupo menos la conexión de escucha*/
    SNL_comprobar_grupo_conexiones(a);
    /*volvemos a cambiar el tiempo de espera del grupo para dar
    mas tiempo al servidor para enviar la respuesta*/
    SNL_tiempo_espera_grupo(a, 5, 500000);
    /*imprimimos texto en la terminal*/
    printf("Esperando respuesta del servidor:\n\n");
    /*iniciamos un bucle while en el que la condicion es que
    la conexión en la que ha sucedido algo del grupo sea
    la conexión que iniciamos antes (esto siempre se cumplira
    ya que es la unica del grupo, a no ser que no haya ninguna
    y el retorno sea SIN_CONEXION*/
    while(SNL_conexion_activa_grupo_conexiones(a) == conexion){
        /*intentamos leer datos recividos en el socket y
        comprobamos si da error al leer datos recividos lo
        cual puede ser por que el servidor cerrase la conexión
        si devuelve 0 o por error en la recepcion de los datos
        o error en la conexión si es -1*/
        if(SNL_recibir_TCP(conexion, &texto[0], 512, 0) <= 0){
            printf("\n\nproblemas reciviendo datos del servidor,");
            printf(" puede que el servidor hubiese cerrado la
conexion");
            /*salimos del bucle*/
            break;
        }else{
            /*como conseguimos recibir datos los mostramos en
            la terminal*/
            printf("%s", &texto[0]);
        }
        /*comprobamos si no hay ninguna conexión que este aun
        activa*/
        if(SNL_comprobar_grupo_conexiones(a) == -1){
            /*salimos del bucle*/
```

```
                break;
            }
        }
        printf("\n\nsi no se muestran datos recibidos desde el servidor
puede ser");
        printf(" que el tiempo de espera se excedio del esperado");
        /*cerramos todas las conexiones del grupo y eliminamos el grupo
        liberando la memoria ocupada por este*/
        SNL_cerrar_grupo_conexiones(a);
        printf("\n\ndesconectado\n");
        /*copiamos a informacion el texto con la informacion de SNL
        y el 500 indica que el array puede guardar como maximo una
        cadena de 500 caracteres, sin contar el carácter nulo de fin
        de cadena de caracteres*/
        informacion_biblioteca_SNL(&informacion[0], 500);
        /*mostramos la informacion de SNL obtenida al principio*/
        printf("\n%s\n\n", &informacion[0]);
        printf("El texto anterior cifrado:\n");
        /*encriptamos el texto usando la funcion de cifrado de bits
        iguales que tiene SNL, esto es solo para mostrar un ejemplo de
        esta funcion*/
        EncriptarTexto_CifradoBitsIguales(&informacion[0], "Free software
as in freedom");
        printf("\n%s\n\n", &informacion[0]);
        printf("Fin\n");
}
exit(0);
}
```

Como el código anterior esta comentado linea por linea sobran las palabras ya, este ejemplo lo que hace básicamente es pedir a un servidor HTTP la pagina index.htm, index.html o index.php del host indicado usando una conexión TCP sobre IPv4. Solo queda recordar un detalle, los grupos pueden usarse tanto con conexiones TCP como con conexiones UDP pero se debe usar cada grupo para un solo tipo de sockets, es decir, si necesitamos usar sockets TCP y UDP tendremos que usar un grupo para las conexiones TCP y otro para los sockets UDP, no debemos mezclarlos ya que despues no sabríamos cuales son las conexiones TCP y cuales las UDP.

A continuación vamos a ver el código de un servidor de chat sencillo que usaremos de ejemplo de como usar SNL para programar servidores, como anteriormente, el código esta comentado de forma que con ver el código es suficiente para, junto con los comentarios del código, entender que se hace en cada momento. El ejemplo que vamos a ver acepta todas las conexiones entrantes, comprueba si algún cliente ha enviado algo y lo reenvía a todos los clientes, muestra en la terminal algunos datos como el numero de clientes conectados y algunos detalles que no son importantes:

```
/*
chatserverSNL Copyright (C) 2009 Jesús Hernández Gormaz

Fecha de creacion: 8 de Junio del 2009 (Siglo XXI)

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License as
published by the Free Software Foundation; either version 3, or
(at your option) any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Este programa es software libre. Puede redistribuirlo y/o
modificarlo bajo los términos de la Licencia Pública General
de GNU según es publicada por la Free Software Foundation,
bien de la versión 3 de dicha Licencia o bien (según su
elección) de cualquier versión posterior.
Este programa se distribuye con la esperanza de que sea
útil, pero SIN NINGUNA GARANTÍA, incluso sin la garantía
MERCANTIL implícita o sin garantizar la CONVENIENCIA PARA UN
PROPÓSITO PARTICULAR. Para más detalles, véase la Licencia
Pública General de GNU.

Debería haber recibido una copia de la Licencia Pública
General junto con este programa. En caso contrario, escriba
a la Free Software Foundation, Inc., en 675 Mass Ave,
Cambridge, MA 02139, EEUU.
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <SNL/SNL.h>

/*tipo de variable enumeracion para especificar un color de algo que
el
 programa debe comunicar al usuario*/
typedef enum color {NoColor, normal, rojo, cyan, verde, azul, negro,
amarillo, \
blanco, gris_claro, marron, purpura, gris_oscuro, verde_claro,
cyan_claro, \
rojo_claro, purpura_claro} color;

/*funcion que manda a la terminal secuencias de escape ANSI para
borrar la pantalla y colocar el cursor al comienzo de la misma*/
int secuencia_borrar_terminal();
/*funcion que manda a la terminal secuencias de escape ANSI para dar
formato de color al texto en la terminal (el cambio se mantiene hasta
que se vuelva a cambiar)*/
int secuencia_color_terminal(color c);

int main(int argc, char *argv[]){
/*iniciamos las variables*/
int conexion, conexion_escucha, puerto, e=0, i=0, p=1;
unsigned long int numero_clientes=0;
char buffer[1537], direccion[21];
/* Esta sera la publicidad que mostrara el servidor*/
char publicidad[]="\n\nPublicidad del servidor:\nVisita
www.snl.ya.st\n\n\n\0";
/*el grupo de conexiones, con uno es suficiente*/
grupo_conexiones *clientes;
/*esto es puramente por decoracion y que quede bonito,
```

```
 solo borra la ventana de la terminal*/
secuencia_borrar_terminal();
/*escribimos en pantalla*/
printf("Conectando el servidor en el puerto 1500, espere.");
/*iniciamos una conexion de escucha (este ejemplo es usando IPv4,
 pero con IPv6 es practicamente igual pero usando las funciones que no
 acaban con _IPv4 el nombre de la funcion, consulte la documentacion
para
 más información*/
conexion_escucha= SNL_escuchar_TCP_IPv4(NULL, 1500, 20);
/*comprobamos que el servidor este conectado*/
if(conexion == SIN_CONEXION){
     /*esta es otra funcion que simplemente es para que quede bonito
el
      ejemplo, imprime en pantalla una secuencia de escape ANSI que
hace
      que todo lo que se escriba despues se vea en rojo claro*/
     secuencia_color_terminal(rojo_claro);
     printf("Error iniciando la conexion del servidor en el puerto
1500.\n");
     /*con esto hacemos que al escribir no se escriba en rojo en
pantalla
      sino en el color predeterminado*/
     secuencia_color_terminal(NoColor);
     return -1;
}else{
     printf("Servidor conectado y aceptando clientes en el puerto ");
     secuencia_color_terminal(cyan);
     printf("1500");
     secuencia_color_terminal(NoColor);
     printf(".\n");
     /*esto si nos interesa, iniciamos un nuevo grupo de conexiones
      pasando como argumento la conexion que usaremos para escuchar
      conexiones entrantes*/
     clientes= SNL_nuevo_grupo_conexiones(conexion_escucha);
     /*comprobamos que se creara el nuevo grupo*/
     if(clientes == NULL){
          /*esto es que no se a creado, lo indicamos y acabamos el
           programa*/
          secuencia_color_terminal(rojo_claro);
          printf("Error de memoria insuficiente.\n");
          secuencia_color_terminal(NoColor);
          return -1;
     }else{
          /*se creo correctamente el grupo, cambiamos el tiempo de
espera
           a que suceda algo en los sockets del grupo a un segundo y
           500000 microsegundos (un segundo y medio)*/
          SNL_tiempo_espera_grupo(clientes, 1, 500000);
          /*esto es un bucle con el que estar atendiendo a los clientes
          continuamente sin que acabe el programa, deberemos acabarlo
          pulsando CONTROL+C o cerrando la terminal*/
          while(e == 0){
               printf("Comprobando conexiones entrantes, espere.\n");
               /*comprobamos si hay alguna conexion entrante*/
               while(SNL_conexion_entrante_grupo(clientes) == 1){
```

```
                    printf("Nueva conexion entrante, espere.\n");
                    /*aceptamos la conexion entrante como un buen
                     servidor*/
                    conexion=
SNL_aceptar_conexion_IPv4(conexion_escucha, &direccion[0], &puerto);
                    /*comprobamos que se ha podido establecer la
                     conexion*/
                    if(conexion == SIN_CONEXION){
                         secuencia_color_terminal(rojo_claro);
                         printf("Error aceptando conexion
entrante.\n");
                         secuencia_color_terminal(NoColor);
                    }else{
                         printf("Conexion entrante nueva aceptada:\n");
                         secuencia_color_terminal(amarillo);
                         printf("%s %i\n", &direccion[0], puerto);
                         secuencia_color_terminal(NoColor);
                         /*añadimos la conexion al grupo de
                          conexiones y comprobamos que no
                          hubo errores*/
                         if(SNL_nueva_conexion_grupo(conexion,
clientes) != 0){
                              secuencia_color_terminal(rojo_claro);
                              printf("Error de memoria en la conexion
entrante.\n");
                              secuencia_color_terminal(NoColor);
                         }else{
                              printf("La nueva conexion entrante ha
sido aceptada sin problemas.\n");
                              /*aumentamos el numero de clientes*/
                              numero_clientes++;
                              strcpy(&buffer[0], "Un cliente se ha
conectado al servidor.\n");
                              strcat(&buffer[0], "IP del cliente que se
ha conectado: ");
                              strcat(&buffer[0], &direccion[0]);
                              strcat(&buffer[0], "\n\0");
                              /*enviamos a todos los clientes
                               la direccion IP del nuevo
                               cliente y comprobamos que
                               no hubo ningun error enviando*/
                              if(SNL_enviar_grupo_TCP(clientes,
&buffer[0], 1536, 0) == 0){
                                   printf("Datos reenviados a todos los
clientes sin problemas.\n");
                              }else{

     secuencia_color_terminal(rojo_claro);
                                   printf("Error reenviando datos a
todos los clientes.\n");
                                   secuencia_color_terminal(NoColor);
                              }
                         }
                    }
               }
               printf("Conexiones entrantes comprobadas.\n");
```

```
               printf("Comprobando clientes conectados en busca de
actividad en la conexion, espere.\n");
               conexion= SIN_CONEXION;
               /*comprobamos si hay alguna conexion activa en el
                grupo de conexiones*/
               if(SNL_comprobar_grupo_conexiones(clientes) == 0){
                    /*ahora obtenemos la conexion activa del grupo*/
                    conexion=
SNL_conexion_activa_grupo_conexiones(clientes);
                    /*este bucle es para despues intentar obtener
                     otra conexion activa hasta atender a todas*/
                    while(conexion > SIN_CONEXION){
                         /*vaciamos el buffer, esto no es
                          relevante para el uso de SNL, pero
                          como este programa lo probare usando
                          telnet para conectarme a el consigo
                          asi que al enviar un mensaje no
                          muestre fragmentos de mensajes
                          anteriores*/
                         for(i=0; i<1537; i++){
                              buffer[i]= '\0';
                         }
                         /*recibimos los datos del cliente y
                          guardamos el valor retornado*/
                         i= SNL_recibir_TCP(conexion, &buffer[0], 1536,
0);
                         /*comprobamos que sucedio con el valor
                          retornado*/
                         if(i == 0){
                              printf("Desconexion del cliente.\n");
                              /*quitamos la conexion del grupo
                               como la funcion de recivir datos
                               ya la cerro automaticamente*/
                              if(SNL_quitar_conexion_grupo(conexion,
clientes) == 0){
                                   printf("Conexion cerrada sin
problemas.\n");
                                   /*ahora tenemos un cliente menos
                                    conectado*/
                                   numero_clientes--;
                                   strcpy(&buffer[0], "Un cliente se ha
desconectado del servidor.\n\0");
                                   /*informamos de esto a los
                                    demas clientes*/
                                   if(SNL_enviar_grupo_TCP(clientes,
&buffer[0], 1536, 0) == 0){
                                        printf("Datos reenviados a
todos los clientes sin problemas.\n");
                                   }else{

     secuencia_color_terminal(rojo_claro);
                                        printf("Error reenviando datos
a todos los clientes.\n");

     secuencia_color_terminal(NoColor);
                                   }
```

```
                              }else{

     secuencia_color_terminal(rojo_claro);
                                   printf("Error cerrando la conexion.\
n");
                                   secuencia_color_terminal(NoColor);
                                   numero_clientes--;
                                   strcpy(&buffer[0], "Un cliente se ha
desconectado del servidor.\nOcurrio un error en la desconexion.\n\0");
                                   if(SNL_enviar_grupo_TCP(clientes,
&buffer[0], 1536, 0) == 0){
                                        printf("Datos reenviados a
todos los clientes sin problemas.\n");
                                   }else{

     secuencia_color_terminal(rojo_claro);
                                        printf("Error reenviando datos
a todos los clientes.\n");

     secuencia_color_terminal(NoColor);
                                   }
                              }
                         }else if(i == -1){
                              secuencia_color_terminal(rojo_claro);
                              printf("Error recibiendo datos del
cliente.\n");
                              secuencia_color_terminal(NoColor);
                              strcpy(&buffer[0], "Ocurrio un error
reciviendo datos de un cliente.\n\0");
                              /*informamos a todos los clientes
                               del error reciviendo los datos*/
                              if(SNL_enviar_grupo_TCP(clientes,
&buffer[0], 1536, 0) == 0){
                                   printf("Datos reenviados a todos los
clientes sin problemas.\n");
                              }else{

     secuencia_color_terminal(rojo_claro);
                                   printf("Error reenviando datos a
todos los clientes.\n");
                                   secuencia_color_terminal(NoColor);
                              }
                         }else{
                              printf("Datos recividos del cliente:\n");
                              secuencia_color_terminal(verde);
                              printf("%s\n", &buffer[0]);
                              secuencia_color_terminal(NoColor);
                              printf("Reenviando datos a todos los
clientes, espere.\n");
                              /*reenviamos los datos recividos
                               a todos los clientes*/
                              if(SNL_enviar_grupo_TCP(clientes,
&buffer[0], 1536, 0) == 0){
                                   printf("Datos reenviados a todos los
clientes sin problemas.\n");
                                   /*esto es para despues
```

```
                                    mostrar publicidad*/
                                   p++;
                              }else{

     secuencia_color_terminal(rojo_claro);
                                   printf("Error reenviando datos a
todos los clientes.\n");
                                   secuencia_color_terminal(NoColor);
                              }
                         }
                         /*este es el motivo del while dentro del
                          que estamos, comprobamos nuevamente
                          las conexiones activas para atender a
                          todas*/
                         conexion= SIN_CONEXION;
                         SNL_comprobar_grupo_conexiones(clientes);
                         conexion=
SNL_conexion_activa_grupo_conexiones(clientes);
                    }
               }
               printf("Actividad en las conexiones comprobadas.\n");
               /*esto se encarga de la publicidad*/
               if(p >= 20){
                    p= 1;
                    strcpy(&buffer[0], &publicidad[0]);
                    strcat(&buffer[0], "Datos del servidor:\n - Ultima
IP conectada: ");
                    strcat(&buffer[0], &direccion[0]);
                    strcat(&buffer[0], "\n\n\n\0");
                    if(SNL_enviar_grupo_TCP(clientes, &buffer[0], 1536,
0) == 0){
                         printf("Publicidad enviada a todos los
clientes sin problemas.\n");
                    }else{
                         secuencia_color_terminal(rojo_claro);
                         printf("Error enviando publicidad a todos los
clientes.\n");
                         secuencia_color_terminal(NoColor);
                    }
               }
               /*esto es tambien para que quede bonito, y para evitar
                consumir demasiada CPU, para el programa 1 segundo,
                esto en GNU/Linux funciona, pero en otros sistemas no
                puedo asegurarlo, como no es relevante puede eliminarse
                para compilar si da error solo que la pantalla de
                la terminal parpadeara al borrar el texto y
                escribir nuevamente*/
               sleep(1);
               secuencia_borrar_terminal();
               printf("Servidor conectado y aceptando clientes en el
puerto ");
               secuencia_color_terminal(cyan);
               printf("1500");
               secuencia_color_terminal(NoColor);
               printf(".\n");
               printf("Total de clientes conectados: ");
```

```
                secuencia_color_terminal(cyan);
                printf("%lu", numero_clientes);
                secuencia_color_terminal(NoColor);
                printf(".\n");
                printf("Ultimos datos recividos del cliente:\n");
                secuencia_color_terminal(verde);
                printf("%s", &buffer[0]);
                secuencia_color_terminal(NoColor);
                printf("\n");
           }
           return 0;
     }
}
}

/*funcion que manda a la terminal secuencias de escape ANSI para
borrar la pantalla y colocar el cursor al comienzo de la misma*/
int secuencia_borrar_terminal(){
printf("\n");
printf("%c[2J", 27);
printf("%c[0;0f", 27);
secuencia_color_terminal(normal);
return 0;
}

/*funcion que manda a la terminal secuencias de escape ANSI para dar
formato de color al texto en la terminal (el cambio se mantiene hasta
que se vuelva a cambiar)*/
int secuencia_color_terminal(color c){
     register int r=0;
     printf("%c[", 27);
     switch(c){
          case rojo: printf("0;31");
               break;
          case rojo_claro: printf("1;31");
               break;
          case verde: printf("0;32");
               break;
          case verde_claro: printf("1;32");
               break;
          case amarillo: printf("1;33");
               break;
          case marron: printf("0;33");
               break;
          case gris_oscuro: printf("1;30");
               break;
          case gris_claro: printf("0;37");
               break;
          case azul: printf("0;34");
               break;
          case purpura: printf("0;35");
               break;
          case purpura_claro: printf("1;35");
               break;
          case cyan: printf("0;36");
               break;
```

```
            case cyan_claro: printf("1;36");
                 break;
            case blanco: printf("1;37");
                 break;
            default: printf("0;00");
                 break;
      }
      printf("m");
      return r;
}
```

El ejemplo en si esta todo dentro de la función main (recordar que esto no es bueno, sino que hay que dividir el código en funciones de forma correcta para que luego sea más fácil de modificar y para poder reutilizar código llamando a la función en lugar de tener trozos de código iguales aumentando el tamaño del programa innecesariamente) y las dos funciones únicamente son para que el ejemplo tenga un mejor aspecto en la terminal.

Como por el ancho del documento el código de los ejemplos aparecen las lineas ajustadas de forma que una linea de código queda dividida en varias lineas es aconsejable descargarse el código fuente de SNL donde encontrara el código de los dos ejemplos que vimos.

Tabla de códigos de errores de SNL

A continuación tienen la tabla con los códigos de errores que puede devolver SNL_ultimo_error() y cada código de error indica un error diferente y conlleva diferentes mecanismos anti-error que SNL pondrá automáticamente en funcionamiento:

Código de error	Descripción
0	Sin error, es un error nulo, es decir, no hay error.
1	Error grave irrecuperable que impide la correcta ejecución de todo el programa, pudiendo llegar a finalizarse el programa con exit(-1) de ser necesario.
2	Error grave irrecuperable que impide la correcta ejecución de cualquier función de la biblioteca.
3	Error grave irrecuperable que impide la correcta ejecución de cualquier función de la biblioteca excepto las funciones de liberación de memoria dinámica.
4	Error producido al restaurar el nivel de ejecución de todas las funciones de la biblioteca por parte del programador con la función SNL_emergencia(), este error siempre se produce al usar la mencionada función.
5	Error grave irrecuperable que impide la correcta ejecución de cualquier función de la biblioteca excepto las funciones para el tratamiento de errores.
6	Error al intentar ejecutar una función cuyo nivel de ejecución ha sido bloqueado.
100	Error desconocido reservando memoria dinámica.
200	Error desconocido liberando memoria dinámica.
300	Error desconocido conectando.
400	Error desconocido desconectando.

Contacto

Para tratar cualquier tema respecto a la biblioteca Simple Network Library (SNL) puede ponerse en contacto con el autor, Jesús Hernández Gormaz, en el correo electrónico elrinconjhg@gmail.com indicando en el asunto que es para tratar un tema respecto a SNL. También pueden visitar la web de SNL en www.snl.ya.st.

Comentarios del autor original

Simple network library ha sido la primera biblioteca para la comunicación a través de redes que he programado, y también la primera biblioteca del nivel de SNL. SNL es un orgullo como programador y el mayor logro hasta el momento (a fecha de 7 de Abril del 2009 d.c.), es una biblioteca que esta ya desde sus inicios preparada para el protocolo IPv6 que reemplazara al IPv4 para 2011, si no cambian las cosas, y es una biblioteca que se actualizara para mantenerse al día en lo referente a comunicaciones en redes. Aunque actualmente SNL ha sido programada íntegramente por mi, Jesús Hernández Gormaz, como único programador de la biblioteca al ser esta software libre bajo la licencia GPL están abiertas las puertas a otros programadores que puedan aportar código con el que mejorarla, pueden contactar a mi correo electrónico elrinconjhg@gmail.com con cualquier tema relacionado con SNL o a través de la web de SNL. También pueden realizar donaciones por el trabajo realizado como autor de SNL en la web de SNL o en el enlace https://www.paypal.com/cgi-bin/webscr?cmd=_s-xclick&hosted_button_id=4538700 que les llevara a una pagina segura de Paypal, la donación por el trabajo realizado creando SNL es opcional y cada cual deberá valorar si lo hace y cuanto, SNL puede descargarse de forma gratuita en la web de SNL o si tienen una conexión a internet demasiado lenta pueden encargar a través de la web o a mi por mi correo electrónico una copia en CD o DVD de SNL la cual tendrá un cobro por el disco, por la grabación del mismo y el envío por correo postal.

Jesús Hernández Gormaz

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-

wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the

name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  YEAR  YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled "GNU
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with … Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.