libMainWindow

una libreria di supporto per LCS 2005/06

Francesco Nidito
nids@di.unipi.it

A.A. 2005/06

Indice

1 Introduzione

La libreria *libMainWindow* è stata implementata per fornire della semplici funzioni per interagire con la libreria *ncurses* agli studenti che devono svolgere il progetto del corso di LCS dell'anno accademico 2005/06.

La libreria fornisce un totale di 7 ("sette") funzioni per inizializzare, ridimensionare, terminare, leggere e scrivere dati da un terminale senza dover interagire direttamente con la libreria *ncurses*.

1.1 Cosa fa la libreria

La libreria può inizializzare un qualsiasi terminale normalmente fornito con Linux o Unix, anche grafico (xterm, gnome-terminal, kterm...), e predisporlo per l'invocazione delle funzioni *ncurses*.

La libreria consente, inoltre, di produrre output da stringhe gia formattate, gestendo automaticamente lo scorrimento della finestra di output. Inoltre fornisce una funzione di lettura di stringhe dal buffer di immissione delle.

1.2 Cosa non fa la libreria

La libreria **non** è stata scritta da un esperto di *ncurses*, quindi è sicuramente carente di qualche caratteristica che potrebbe renderla più funzionale.

La libreria fornisce un sistema di auto scorrimento della finestra di output che però non può essere comandato e quindi una volta che le righe sono uscite dal campo di vista non sono più recuperabili se non nel file di log, di cui si parla più avanti.

2 Strutture dati e funzioni

La libreria è molto semplice e consiste unicamente di una struttura dati e di sette funzioni.

2.1 Strutture dati

La libreria utilizza per il suo funzionamento una sola struttura dati che è rappresentata da una `stuct` che contiene tutti i dati necessari alla corretta visualizzazione e controllo dello schermo.

2.1.1 MainWindow_t

```
typedef struct
{
      WINDOW *InWin;
      WINDOW *OutWin;
      char* ClearString;
      int TermWidth;
      int TermHeight;
      pthread_mutex_t *Mutex;
      FILE *Log;
} MainWindow_t;
```

Il tipo `MainWindow_t` è aggregato e contiene i seguenti campi per mantenere lo stato dello schermo:

- **`WINDOW *InWin`** rappresenta il puntatore alla finestra di input nella quale vengono inserite le stringhe dall'utente;
- **`WINDOW *OutWin`** rappresenta il puntatore alla finestra di output nella quale sono visualizzate le stringhe;
- **`char* ClearString`** è una stringa di caratteri *spazio* usata per andare a ripulire la finestra di input dopo che questa è stata letta dal sistema;
- **`int TermWidth`** è la larghezza dell terminale in caratteri;
- **`int TermHeight`** è l'altezza dell terminale in caratteri;
- **`pthread_mutex_t *Mutex`** è la *mutex* usata per proteggere le sezioni di codice che spostano il cursore ed il focus per l'input;
- **`FILE *Log`** è il file sul quale vengono eseguite le operazioni di *logging* della finesta di output.

2.2 Funzioni

La libreria espone solo sette funzioni. È stato scelto un numero così ridotto al fine di semplificarne l'uso.

2.2.1 InitScreen

```
MainWindow_t* InitScreen( const char* logfile );
```

La funzione `InitScreen` provvede ad inizializzare il terminale e le stutture dati per il loro utilizzo tramite le *ncurses*.

La funzione riceve in ingresso il parametro:

- **`const char* logfile`** che rappresenta il nome del file nel quale verranno effettuate le operazioni di *logging* della finesta di output. Se tale parametro è `NULL` allora le operazioni di *logging* non sono effettuate.

La funzione ritorna il puntatore alla struttura dati che rappresenta il sistema di gestione delle finestre e che dovrà essere passato come argomento a tutte le altre funzioni della libreria. La funzione ritorna `NULL` nel caso in cui si sia verificato qualche errore durante l'inizializzazione.

2.2.2 PrintString

```
int PrintString( MainWindow_t* mw, const char* str );
```

La funzione `PrintString` provvede a stampare una stringa nella finestra di output.

La funzione riceve in ingresso i parametri:

- **`MainWindow_t* mw`** che rappresenta l'ambiente che contiene la finestra di output;
- **`const char* str`** che è la stringa che deve essere stampata nella finesta di output.

La funzione ritorna `OK` se l'operazione è stata completata con successo e `ERR` se lòperazione non è andata a buon fine.
`OK` e `ERR` sono definite dalla libreria *ncurses*.

2.2.3 PrintError

```
int PrintError( MainWindow_t* mw, const char* str );
```

La funzione `PrintError` provvede a stampare una stringa nella finestra di output dopo averla concatenata con la stringa `''ERROR''`.

La funzione riceve in ingresso i parametri:

- **`MainWindow_t* mw`** che rappresenta l'ambiente che contiene la finestra di output;

- `const char* str` che è la stringa che deve essere stampata nella finesta di output come messaggio di errore.

La funzione ritorna `OK` se l'operazione è stata completata con successo e `ERR` se lòperazione non è andata a buon fine.
`OK` e `ERR` sono definite dalla libreria *ncurses*.

2.2.4 PrintMessage

```
int PrintMessage( MainWindow_t* mw, const char* str );
```

La funzione `PrintMessage` provvede a stampare una stringa nella finestra di output dopo averla concatenata con la stringa ``MESSAGE''.

La funzione riceve in ingresso i parametri:

- `MainWindow_t* mw` che rappresenta l'ambiente che contiene la finestra di output;
- `const char* str` che è la stringa che deve essere stampata nella finesta di output come messaggio da parte dell'applicazione.

La funzione ritorna `OK` se l'operazione è stata completata con successo e `ERR` se l'operazione non è andata a buon fine.
`OK` e `ERR` sono definite dalla libreria *ncurses*.

2.2.5 ReadString

```
int ReadString( MainWindow_t *mw, char *str, int length );
```

La funzione `ReadString` provvede a leggere una stringa immessa dall'utente nella finesta di input.

La funzione riceve in ingresso i parametri:

- `MainWindow_t* mw` che rappresenta l'ambiente che contiene la finestra di input;
- `char* str` il puntatore all'area di memoria nella quale deve essere memorizzata la stringa;
- `int length` il numero massimo di caratteri che devono essere letti dalla finestra di input.

La funzione ritorna il numero di caratteri letti, -1 in caso di errore.

NB: l'allocazione, e la successiva deallocazione, della memoria per la variabile `str` è lasciata all'utente.

2.2.6 DestroyScreen

```
void DestroyScreen( MainWindow_t* mw );
```

La funzione `DestroyScreen` provvede a ripulire la struttura dati `MainWindow_t`, chiudere la sessione *ncurses* . Inoltre provvede a scrivere il buffer di scittura e chiude il file di log se questo e' stato creato.

La funzione riceve in ingresso i parametri:

- **`MainWindow_t* mw`** che rappresenta l'ambiente di gestione delle finestre.

2.2.7 ResizeScreen

```
void ResizeScreen( MainWindow_t* mw );
```

La funzione `ResizeScreen` provvede a ridimensionare le finestre di input e output in relazione al ridimensionamento del terminale.

La funzione riceve in ingresso i parametri:

- **`MainWindow_t* mw`** che rappresenta l'ambiente di gestione delle finestre.

La funzione deve essere invocata alla ricezione di un segnale di tipo `SIGWINCH`[1]. Dato che la libreria è *reentrant* e lo stato è mantenuto in una variabile di tipo `MainWindow_t`, questo non può avvenire in maniera trasprente. Si deve quindi richiamare la funzione `ResizeScreen` da dentro un'altra funzione che gestisca l'arrivo del segnale, come mostrato nell'esempio seguente:

```
#include "MainWindow.h"

MainWindow_t *mw;

void ResizeHandler(int signum){
   ResizeScreen(mw);
}

...

int main(int argc, char** argv){
   struct sigaction saWhinch;
```

[1] SIGnal for WINdow size CHange

```
    ...
    saWhinch.sa_handler = ResizeHandler;
    sigemptyset(&saWhinch.sa_mask);
    saWhinch.sa_flags = SA_RESTART;

    sigaction(SIGWINCH, &saWhinch, NULL);
    ...
}
```

3 Installazione ed utilizzo

La libreria é pensata per essere collegata staticamante all'applicazione che ne fa uso. Il processo di compilazione produce infatti la libreria statica `libMainWindow.a`

3.1 Composizione Pacchetto

Il pacchetto di distribuzione che contiene la libreria è così organizzato:

```
MainWindow/
 |
 +--examples/
 |   |
 |   +--echo_shell.c
 |   +--Makefile
 |
 +--doc/
 |   |
 |   +--libMainWindow.tex
 |   +--Makefile
 |
 +--InitScreen.c
 +--MainWindow.h
 +--PrintError.c
 +--PrintMessage.c
 +--PrintString.c
 +--ReadString.c
 +--ResizeScreen.c
 +--Makefile
```

I file sorgenti della libreria sono contenuti nella directory principale insieme al makefile.

Nella directory `examples` sono presenti gli esempi di uso della libreria[2].

Nella directory `doc` è presente la documentazione in formato sorgente TEX.

3.2 Installazione Libreria

La libreria non ha un processo di installazione vero e proprio ma, una volta prodotto il file `libMainWindow.a` questo è utilizzabile dalle applicazioni.

Per costruire la libreria basta procedere come segue, partendo dall'archivio `libMainWindow.tar.gz`:

```
$ tar -zxf libMainWindow.tar.gz
$ cd MainWindow
$ make all
```

L'ultimo comando provvede a costruire la libreria, la documentazione e gli esempi.

3.3 Uso della Libreria

La libreria per essere usata deve essere collegata all'applicazione che ne fa uso durante la compilazione. Inoltre i sorgenti dell'applicazione devono includere il file `MainWindow.h`.

Per esempio, supponendo che il vostro progetto (che con enorme sforzo di fantasia si chiamerà `progetto`), sia contenuto nella directory `MioProgetto` e che sia formato dai file `tuo_file_1.c`, `tuo_file_2.c` ... `tuo_file_n.c`.
Inoltre supponendo che il file `libMainWindow.a` sia stato copiato nella directoy `MioProgetto/lib` e il file `MainWindow.h` sia stato copiato nella directory `MioProgetto/include`.
Per poter costruire `progetto` si deve procedere come segue:

```
$ cd MioProgetto
$ gcc -I ./include -c tuo_file_1.c
$ gcc -I ./include -c tuo_file_2.c
...
```

[2] Allo stato attuale dei fatti un esempio soltanto

```
$ gcc -I ./include -c tuo_file_n.c
$ gcc -L ./lib -o progetto *.o -lMainWindow -lcurses
```

L'opzione `-I` istruisce il compilatore `gcc` in modo che i file headers vengano cercati anche nella directory `./include`. Analogamente l'opzione `-L` fa in modo che le librerie da collegare vengano cercate anche nella directory `./lib`. L'opzione `-l` fa in modo che il linker colleghi all'applicazione la libreria associata (`-lMainWindow` collega la libreria `libMainWindow.a`).

Per ulteriori chiarimenti sulle precedenti opzioni del compilatore `gcc`, e le restanti 198 opzioni[3], si faccia riferimento a:

```
$ man gcc
```

4 Esempio

Adesso viene presentato un esempio di uso della libreria costruendo una banale applicazione che legge dall'input dell'interfaccia e ciclicamente stampa la stringa letta così come è stata inserita, come messaggio e come messaggio di errore. L'applicazione termina inserendo la stringa quit.

4.1 Esempio Passo Passo

Per prima cosa si devono inserire i file header richiesti dall'applicazione. Sono necessari:

- ovviamente il file header della libreria `libMainWindow`;
- `string.h` per la funzione `strcmp`;
- `signal.h` per gestire i segnali;
- `stdlib.h` per la funzione `exit`.

```
#include "MainWindow.h"
#include <string.h>
#include <signal.h>
#include <stdlib.h>
```

[3]Il `gcc-4.0.1` ha 201 opzioni in totale

La variablile che contiene lo stato della finestra `mw` è dichiarata globale in modo da poter essere riferita anche dalle funzioni che gestiscono i segnali. Vengono dichiarate anche due funzioni per la gestione dei segnali:

- `SigIntHandler` gestisce il segnale `SIGINT` provvedendo a distuggere lo stato della finestra e ad uscire;
- `ResizeHandler` gestisce il segnale `SIGWINCH` provvedendo a ridimensionare l'interfaccia e successivamente a reimpostare il gestore.

```
MainWindow_t *mw;

void
SigIntHandler(int segnum)
{
   DestroyScreen(mw);
   _Exit(0);
}

void
ResizeHandler(int segnum)
{
   ResizeScreen(mw);
}
```

All'inizio della funzione principale dell'applicazione vengono dichiarate le variabili in uso all'applicazione:

- l'array `f` contiene i puntatori alle funzioni che vengono invocate ciclicamente;
- `i` è l'indice della funzione da usare;
- `str` è la stringa sulla quale viene copiato l'input e è anche la stringa che viene stampata;
- `saWhinch` è la variabile usata per gestire il segnale `SIGWHINCH` (ridimensionamento della finestra);
- `saInt` è la variabile usata per gestire il segnale `SIGINT`.

```
int
main(int argc, char** argv)
{
```

```
int (*f[3])(MainWindow_t*, const char*) = {PrintString,
                                           PrintError,
                                           PrintMessage};
int i = 0;
char str[256];

struct sigaction saWhinch;
struct sigaction saInt;
```

Si inizializza la variabile che contiene lo stato dell'interfaccia, facendo in modo che l'output venga copiato anche nel file `echo_shell.log`.

```
mw = InitScreen("echo_shell.log");
```

Si registrano le due funzioni per gestire i segnali.

```
saInt.sa_handler = SigIntHandler;
sigemptyset(&saInt.sa_mask);
saInt.sa_flags = SA_RESTART;

sigaction(SIGINT, &saInt, NULL);

saWhinch.sa_handler = ResizeHandler;
sigemptyset(&saWhinch.sa_mask);
saWhinch.sa_flags = SA_RESTART;

sigaction(SIGWINCH, &saWhinch, NULL);
```

Si entra nel ciclo principale dell'applicazione nel quale viene letto l'input e il numero di caratteri letti è inserito nella variabile `c`.

```
for(;;)
{
   int c;
   c = ReadString(mw,str,256);
```

Se il numero di caratteri letti è maggiore di zero si procede nella seguente maniera: se la stringa letta equivale a quit si esce dal ciclo principale dell'applicazione; altrimenti si invoca la funzione con indice `i`, passandogli come parametro la stringa letta e si aggiorna l'indice della funzione attuale.

```
      if(c > 0)
      {
        if(!strcmp(str,"quit")) break;

        f[i++](mw, str);
        i %= 3;
      }
   }
```

All'uscita dal ciclo si provvede a distuggere lo stato della finestra e ad uscire.

```
   DestroyScreen(mw);

   return 0;
}
```

4.2 Esempio Completo

L'esempio precedente è qua di seguito presentato per intero. È inoltre presente nel pacchetto nel file `examples/echo_shell.c`.

```
#include "MainWindow.h"
#include <string.h>
#include <signal.h>
#include <stdlib.h>

MainWindow_t *mw;

/* Esce da ncurses e dall'applicazione  */
void
SigIntHandler(int segnum)
{
   DestroyScreen(mw);
   _Exit(0);
}

/* Gestisce il ridimensionamento della finestra */
void
ResizeHandler(int segnum)
{
```

```
   ResizeScreen(mw);
}

int
main(int argc, char** argv)
{
   /* Array delle funzioni da chiamare */
   int (*f[3])(MainWindow_t*, const char*) = {PrintString,
                                              PrintError,
                                              PrintMessage};
   int i = 0;
   char str[256];

   struct sigaction saWhinch;
   struct sigaction saInt;

   /* Creazione schermo e file di log */
   mw = InitScreen("echo_shell.log");

   /* Impostazione dei gestori delle interruzioni per il
    * ridimensionamento della finesta e del ^C
    */
   saWhinch.sa_handler = ResizeHandler;
   sigemptyset(&saWhinch.sa_mask);
   saWhinch.sa_flags = SA_RESTART;

   sigaction(SIGWINCH, &saWhinch, NULL);

   saInt.sa_handler = SigIntHandler;
   sigemptyset(&saInt.sa_mask);
   saInt.sa_flags = SA_RESTART;

   sigaction(SIGINT, &saInt, NULL);

   for(;;)
   {
      int c;
      c = ReadString(mw,str,256);

      if(c > 0)
      {
        /* Se la stringa letta e' "quit" esce*/
```

```
        if(!strcmp(str,"quit")) break;

        /* Chiama ciclicamente le funzioni */
        f[i++](mw, str);
        i %= 3;
      }
   }

   /* Esce da ncurses */
   DestroyScreen(mw);

   return 0;
}
```

5 Licenza

In ogni file che compone il codice sorgente della libreria, della documentazione, degli esempi e nei file `Makefile` è riportata la seguente nota di copyright e licenza :

> Except as contained in this notice, the name of the author shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from the author.

Riassumendo brevente si può:

- usare, copiare, modificare, distribuire e vendere[4] la libreria **senza** dovere niente all'autore della stessa, con il solo obbligo di riportare la nota di copyright e la licenza sulle opere derivate e relativa documentazione.

Riassumendo brevemente **non** si può:

- usare, copiare, modificare, distribuire e vendere[5] la libreria **senza** riportare la nota di copyright e la licenza sulle opere derivate e relativa documentazione;
- perseguire legalmente l'autore per alcun danno provocato a cose o persone dall'uso della libreria o sue opere derivate;
- usare nella promozione e/o commercializzazione della libreria stessa o di opere e prodotti derivati il nome dell'autore senza previa autorizzazione scritta dell'autore stesso.

6 Bug Noti

La libreria è in continuao miglioramento e aggiornamento. Allo stato attuale sono presenti i seguenti bug irrisolti.

1. Durante il ridimensionamento di un terminale grafico la vecchia riquadratura della finestra di output resta presente: questa viene rimossa man mano che l'output cresce.

Note

Per suggerimenti, impressioni e notifiche di possibili errori, sia per quanto riguarda la libreria che per quanto riguarda questa documentazione, si prega di contattare l'autore tramite posta elettronica all'indirizzo `nids@di.unipi.it` anteponendo al soggetto della mail la stringa `[libMainWindow]`.

[4] Ammesso e non concesso che troviate qualcuno che la compra
[5] Vedi nota precedente