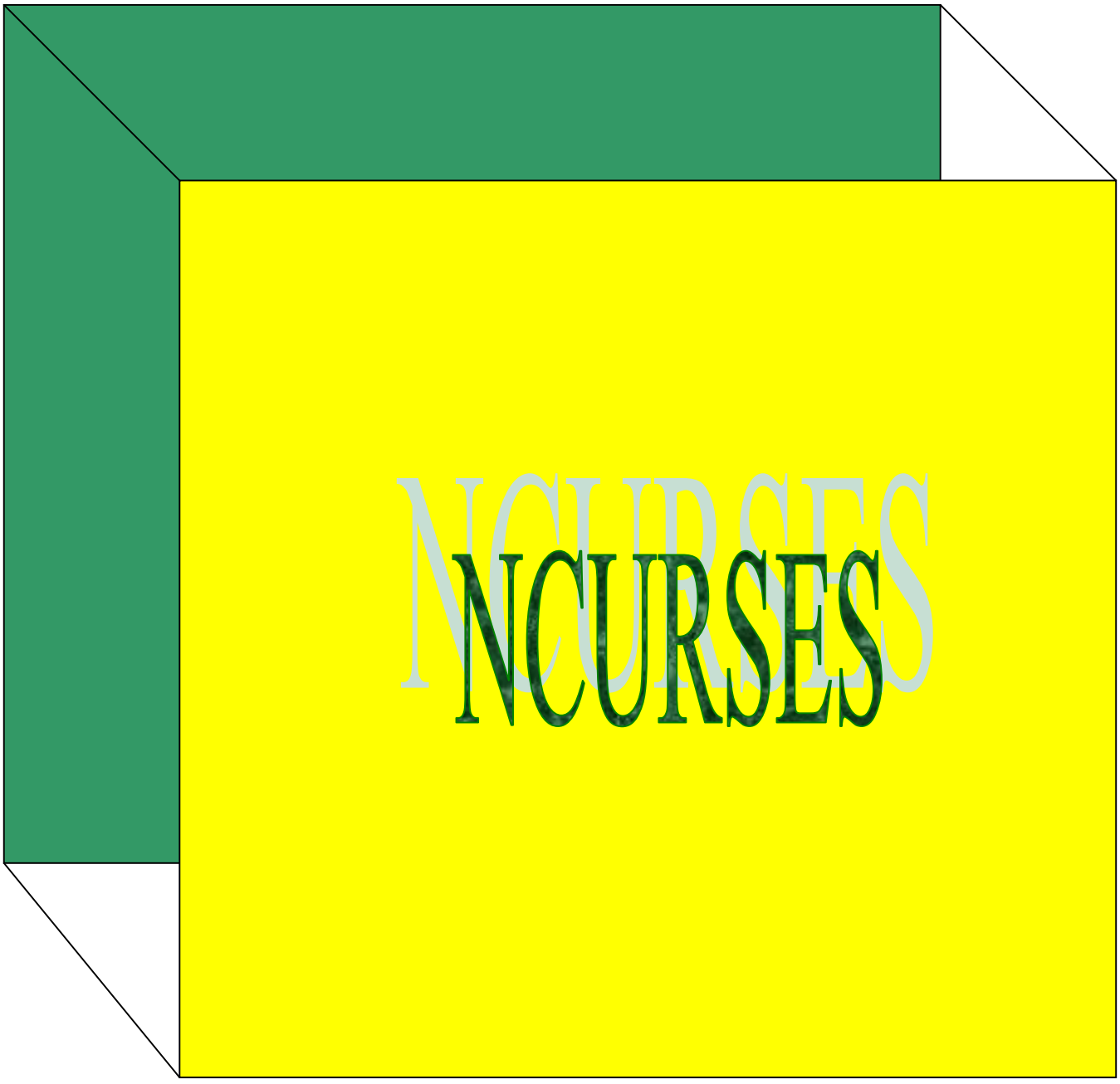
NCURSES

NCURSES

Instruções básicas de NCURSES

Escrito por Fabiano Barbosa Damasceno, aluno do curso Ciência da Computação da Universidade Católica de Goiás.
Instrutor: Alexandre Ribeiro, professor da Universidade Católica de Goiás.

NOTA:
Todos os exemplos contidos nesta apostila foram feitos e testados em um computador com a seguinte configuração:

Pentium 200 MMX
128MB de RAM
OS – Conectiva LINUX 6.0 – 7.0
Ncurses instalado apartir do pacote rpm: ncurses4-4.2-24cl.i386.rpm

Para poder usar a maioria das funções do ncurses, é necessário declarar o cabeçalho curses.h (curses.h aponta para /usr/include/ncurses/ncurses.h). A ncurses controla o terminal, portanto, você não usará diretamente as funções padrão ANSI de entrada e saída. Para poder compilar um código fonte que usa ncurses faça o seguinte:

```
$>gcc <nome.do.arq> -lcurses
```

Você também pode usar as demais opções do gcc que julgar necessário.
Logo abaixo estão algumas funções básicas com os seus respectivos protótipos e explicações.

CABEÇALHO:

```
<curses.h>
```

FUNÇÃO:

```
initscr();
```

PROTOTIPO:

```
WINDOW *initscr(void);
```

A função initscr() é normalmente (o que é quase sempre) a primeira função a ser declarada.
É ela que inicia o controle do terminal.

EXEMPLO 001:

```
#include<curses.h>
void main(void)
{
      (void)initscr(); //iniciando o controle do terminal,..
      mvprintw(LINES/2,(COLS-9)/2,"Olá LINUX");
      refresh();
      endwin();  //encerrando o controle da ncurses,..
}
```

FUNÇÃO:

```
endwin();
```

PROTOTIPO:

```
int endwin(void);
```

A função endwin() é usada sempre que encerra um programa que use ncurses, ou pelo menos sempre que se deseja desativar o controle do terminal através da ncurses. Se você encerrar o seu programa sem chamar endwin() ocorrerá um erro desconhecido.(CRASH NO BASH)

EXEMPLO 002:

Ver exemplo 001...

FUNÇÃO:

```
refresh();
wrefresh();
```

PROTOTIPO:

```
int refresh(void);
int wrefresh(WINDOW *win);
```

A função refresh() deve ser chamada sempre que se faz uma alteração na stream de saída do terminal, wrefresh() é similar a refresh(), ela atualiza a saída de uma janela.

No linux execute no shell (man refresh) para mais informações.

EXEMPLO 003:

```
#include<curses.h>
void main(void)
{
      WINDOW *janela;
      (void)initscr();
      printw("Não se esqueça do refresh");
      janela = newwin(5,20,(LINES-5)/2,(COLS-10)/2);
      box(janela,'|','-');
      wprintw(janela,"| Olá LINUX |"); //não se esqueça de passar o arg.
                                        //do ponteiro.
      refresh();
      wrefresh(janela);
      delwin(janela);
      endwin();  //não esqueça,..
}
```

FUNÇÃO:

```
printw();
```

PROTOTIPO:

```
int printw(char *fmt[, arg] ...);
```

A função printw(), é análoga à printf. Ver referência sobre printf C ANSI.

A diferença é que em vez de usa printf será usada printw para mostrar o resultado na stream de saída.

EXEMPLO 004:

```
#include<curses.h>
void main(void)
{
      (void)initscr();
      printw("Olá LINUX");
      refresh(); // Lembre-se sempre do refresh,..
      endwin();  //não esqueça,..
}
```

FUNÇÃO(S) RELACIONADA(S):

```
  int mvprintw(int y, int x, char *fmt[, arg] ...);
```

FUNÇÃO:
scanw();

PROTOTIPO:
int scanw(char *fmt[, arg] ...);

scanw() é similar a scanf(). Ver referencia sobre scanf (C ANSI).
scanw captara uma entrada da stream de entrada do terminal.
Usa-se scanw em vez de scanf no terminal.

EXEMPLO 005:

```
#include<curses.h>
void main(void)
{
      int idade;
      (void)initscr();
      printw("Digite a sua idade: ");
      refresh();
      scanw("%d",&idade);
      printw("A sua idade é %d",  idade);
      refresh();
      endwin();  //não esqueça,..
}
```

FUNÇÃO(S) RELACIONADA(S):
int mvscanw(int y, int x, *fmt[, arg] ...);

FUNÇÃO:
newwin();
PROTOTIPO:
WINDOW *newwin(int linhas, int colunas, int lin_inicio,int cols_inicio);

Cria e retorna um ponteiro para uma nova janela com o número de linhas e de colunas.

FUNÇÃO:
delwin();
PROTOTIPO:
Int delwin(WINDOW *win);

Limpa o conteúdo da memória para onde win esta apontando. Atenção, essa função somente limpa a memória não limpa a tela onde está a janela.

FUNÇÃO:
move();
PROTOTIPO:
int move(int y,int x);

Esta função move o cursor para a linha y e coluna x, a posição (0,0) é o canto superior esquerdo.

O fragmento de código abaixo:

```
move(10,10);
printw("Olá Linux");
move(11,11,);
scanw("%d", INT);
```

tem o mesmo efeito que:

```
mvprintw(10,10,"Olá linux");
mvscanw(11,11,"%d", INT);
```

EXEMPLO 006:

```
#include<curses.h>
void main(void)
{
      (void)initscr();
      move(10,10); // move o cursor para linha 10 e coluna 10,..
      printw("Olá LINUX");
      refresh(); // Lembre-se sempre do refresh,..
      endwin();  //não esqueça,..
}
```

FUNÇÃO:
wprintw();
PROTOTIPO:
int wprintw(WINDOW *win, char *fmt[, arg] ...);

wprintw(), funciona como printw, só que a saída é redirecionada para uma janela específica apontada por win.

EXEMPLO 007:

```
#include<curses.h>
void main(void)
{
      WINDOW *janela;
      (void)initscr();
      janela = newwin(5,20,(LINES-5)/2,(COLS-10)/2);
      box(janela,'|','-');
      wprintw(janela,"| Olá LINUX |"); //não se esqueça de passar o arg.
                                        //do ponteiro.
      refresh();
      wrefresh(janela);
      delwin(janela);
      endwin();  //não esqueça,..
}
```

FUNÇÃO(S) RELACIONADA(S):

```
int mvwprintw(WINDOW *win, int y, int x, char *fmt[,arg] ...);
```

FUNÇÃO:
wscanw();

PROTOTIPO:
int wscanw(WINDOW *win, char *fmt[, arg] ...);

wscaw funciona como scanw, só que em vez de capturar uma entrada do terminal, captara uma entrada de uma janela do terminal.

EXEMPLO 008:

```
#include<curses.h>
void main(void)
{
      int idade;
      char chr;
      WINDOW *janela;
      (void)initscr();

      printw("Digite a sua idade:");
      refresh();
      scanw("%d",&idade);
      printw("A sua idade é %d",  idade);
      refresh();

      janela = newwin(5,45,(LINES-5)/2,(COLS-45)/2);
      box(janela,'|','-');
      mvwprintw(janela,0,18,"| LINUX |"); //não se esqueça de passar o arg. do ponteiro.
      mvwprintw(janela,2,2,"Agora digite a primeira letra do seu nome");
      wrefresh(janela);
      mvwscanw(janela,3,2,"%c",&chr); //não se preocupe com o 'mv' logo mais vc saberá o que é,..
      mvwprintw(janela,4,2,"Você digitou %c",chr);
      wrefresh(janela);
      delwin(janela);
      endwin();  //não esqueça,..
}
```

FUNÇÃO(S) RELACIONADA(S):
int mvwscanw(WINDOW *win, int y, int x, char *fmt[, arg] ...);

FUNÇÃO:
addch();
waddch();

PROTOTIPO:
int addch(chtype ch);
int waddch(WINDOW *win,chtype ch);

addch() coloca um caractere na stream padrão (stdscr), waddch() funciona da mesma forma só que a saída é para uma janela apontada por win. Use mvaddch() ou mvwaddch() para posicionar o caracter conforme a sua vontade.

EXEMPLO 009:

```
        #include<curses.h>
        void main(void)
        {
                WINDOW *janela;
                (void)initscr();

                box(stdscr,ACS_VLINE,ACS_HLINE);
                mvaddch(LINES-2,2,'L');
                mvaddch(LINES-2,3,'I');
                mvaddch(LINES-2,4,'N');
                mvaddch(LINES-2,5,'U');
                mvaddch(LINES-2,6,'X');
                refresh();

                janela = newwin(9,3,(LINES-9)/2,(COLS-3)/2);
                mvwaddch(janela,2,2,'P');
                mvwaddch(janela,3,2,'I');
                mvwaddch(janela,4,2,'N');
                mvwaddch(janela,5,2,'G');
                mvwaddch(janela,6,2,'U');
                mvwaddch(janela,7,2,'I');
                mvwaddch(janela,8,2,'M');
                wrefresh(janela);

                delwin(janela);
                endwin();
                printf("\n");
        }
```

FUNÇÃO(S) RELACIONADA(S):

```
        int mvaddch(int y, int x, chtype ch);
        int mvwaddch(WINDOW *win, int y, int x, chtype ch);
```

FUNÇÃO:

```
        getch();
        wgetch();
```

PROTOTIPO:

```
        int getch(void);
        int wgetch(WINDOW *win);
```

As funções getch e wgetch são similares à função getchar() do C padrão ANSI. Você pode também usar as funções int mvget() ou int mvwgetch() para poder posicionar o cursor.

EXEMPLO 010:

```
        #include<curses.h>
        void main(void)
        {
                char ch;
                (void)initscr();
                printw("Digite um caractere ");
                refresh();
                ch = getch();
```

```
                printw("\nVocê digitou %c",ch);
                refresh();
                endwin();
        }
```

FUNÇÃO:

```
        getstr();
        wgetstr();
```

PROTOTIPO:

```
        int getstr(char *str);
        int wgetstr(WINDOW *win, char *str);
```

Essas funções são similares a gets() do C padrão ANSI, o interessante é que elas não retornam o alerta de gets(). (A função gets é perigosa e não deve ser usada). Use também mvgetstr() ou mvwgetstr().

getstr() ou wgetstr() não faz controle de limite, se o número de caracteres exceder o tamanho da string ocorrerá um erro. Provavelmente um estouro de memória.

VEJA TAMBÉM:

No linux, digite ($> man addstr).

EXEMPLO 011:

```
        #include<curses.h>
        #include<string.h>
        void main(void)
        {
                char str[31];
                WINDOW *janela;
                (void)initscr();
                border(ACS_LTEE,        //borda esquerda,..
                       ACS_RTEE,        //borda direita,..
                       ACS_BTEE,        //borda superior,..
                       ACS_TTEE,        //borda inferior,..
                       ACS_ULCORNER,    //canto da borda superior esquerdo,..
                       ACS_URCORNER,    //canto da borda superior direito,..
                       ACS_LLCORNER,    //canto da borda inferior esquerdo,..
                       ACS_LRCORNER);   //canto da borda inferior direito,..
                refresh();
                janela = newwin(5,40,(LINES-5)/2,(COLS-40)/2);
                box(janela,ACS_VLINE,ACS_HLINE);
                mvwaddstr(janela,1,2,"Digite um nome de até 30 caracteres");
                mvwaddch(janela,2,2,'>');
                wrefresh(janela);
                mvwgetstr(janela,2,3,str);
                mvwprintw(janela,3,3,"%s",strfry(str));
                wrefresh(janela);
                delwin(janela);
                endwin();
        }
```

FUNÇÃO(S) RELACIONADA(S):

```
        int mvgetstr(int y, int x, char *str);
        int mvwgetstr(WINDOW *win, int y, int x, char *str);
```

FUNÇÃO:

```
getnstr();
wgetnstr();
```

PROTOTIPO:

```
int getnstr(char *str, int n);
int wgetnstr(WINDOW *win, char *str, int n);
```

As funções getnstr() e wgetnstr() são idênticas a getstr() e wgetstr, com exceção de que só vão ler n caracteres.

EXEMPLO 012:

```
#include<curses.h>
#include<string.h>
void main(void)
{
        char str[31];
        WINDOW *janela;
        (void)initscr();
        border(ACS_LTEE,        //borda esquerda,..
               ACS_RTEE,        //borda direita,..
               ACS_TTEE,        //borda superior,..
               ACS_BTEE,        //borda inferior,..
               ACS_URCORNER,    //canto da borda superior esquerdo,..
               ACS_ULCORNER,    //canto da borda superior direito,..
               ACS_LRCORNER,    //canto da borda inferior esquerdo,..
               ACS_LLCORNER);   //canto da borda inferior direito,..

        refresh();

        janela = newwin(5,40,(LINES-5)/2,(COLS-40)/2);
        box(janela,ACS_VLINE,ACS_HLINE);
        mvwaddstr(janela,1,2,"Digite um nome de até 30 caracteres");
        mvwaddch(janela,2,2,'>');
        wrefresh(janela);
        mvwgetnstr(janela,2,3,str,30);
        mvwprintw(janela,3,3,"%s",strfry(str));
        wrefresh(janela);

        getch();
        clear();
        refresh();

        delwin(janela);
        endwin();
}
```

FUNÇÃO(S) RELACIONADA(S):

```
int mvgetnstr(int y, int x,char *str, int n);
int mvwgetnstr(WINDOW *win, int y, int x, char *str, int x);
```

FUNÇÃO:

border();
wborder();

PROTOTIPO:

```
int border(chtype ls, chtype rs, chtype ts, chtype bs,
        chtype tl, chtype tr, chtype bl, chtype br);

int wborder(WINDOW *win, chtype ls, chtype rs, chtype ts, chtype bs
        chtype tl, chtype tr, chtype bl, chtype br);
```

A função border(), wborder() desenha uma borda na stream padrão ou em uma janela apontada por win. A lista de argumento é:

ls - borda esquerda.
rs - borda direita.
ts - borda superior.
bs - borda inferior.
tl - canto da borda superior esquerdo.
tr - canto da borda superior direito.
bl - canto da borda inferior esquerdo.
br - canto da borda inferior direito.

EXEMPLO 013:

```
#include<curses.h>
void main(void)
{
        WINDOW *janela;
        (void)initscr();

        border(ACS_LTEE,        //borda esquerda,..
               ACS_RTEE,        //borda direita,..
               ACS_TTEE,        //borda superior,..
               ACS_BTEE,        //borda inferior,..
               ACS_URCORNER,    //canto da borda superior esquerdo,..
               ACS_ULCORNER,    //canto da borda superior direito,..
               ACS_LRCORNER,    //canto da borda inferior esquerdo,..
               ACS_LLCORNER);   //canto da borda inferior direito,..

        janela = newwin(LINES-10,COLS-5,(LINES-(LINES-10))/2,(COLS-(COLS-
5))/2);
        wborder(janela,
               ACS_RTEE,
               ACS_LTEE,
               ACS_BTEE,
               ACS_TTEE,
               ACS_ULCORNER,
               ACS_URCORNER,
               ACS_LLCORNER,
               ACS_LRCORNER);

        refresh();
        wrefresh(janela);
        noecho();
        getch();
        delwin(janela);
        endwin();
}
```

FUNÇÃO:
start_color();

PROTOTIPO:
int start_color(void);

Para poder usar cores no terminal é necessário fazer uma chamada a start_color, geralmente isto é feito logo após a initscr(). As cores são sempre usadas em pares.(foreground, backgroud).

EXEMPLO 014:

```
#include<curses.h>
void main(void)
{
      int PAR1 = 1, PAR2 = 2;
      WINDOW *janela;

      (void)initscr();
      (void)start_color();

      init_pair(PAR1,COLOR_CYAN,COLOR_BLUE);
      init_pair(PAR2,COLOR_GREEN,COLOR_YELLOW);

      bkgd(COLOR_PAIR(PAR1));
      attrset(COLOR_PAIR(PAR1));

      printw("Não se esqueça do refresh");
      janela = newwin(5,45,(LINES-5)/2,(COLS-45)/2);
      wbkgd(janela,COLOR_PAIR(PAR2));
      wattrset(janela,COLOR_PAIR(PAR2));

      box(janela,'|','-');
      mvwprintw(janela,0,18,"| LINUX |");
      refresh();
      wrefresh(janela);
      delwin(janela);
      endwin();  //não esqueça,..
}
```

FUNÇÃO:
init_pair();

PROTOTIPO:
int int_pair(short pair, short f, short b);

A função init_pair "monta" os pares de cores. short pair é o número do par de cores pelo qual se poderá fazer referencia. Após ter feito os pares de cores, a macro COLOR_PAIR(n) pode ser chamada para setar as novas configurações de cores.
As constantes de cores estão no final desse documento.

EXEMPLO 015:
Ver exemplo número 014.

FUNÇÃO:
attrset();

PROTOTIPO:
int attrset(int attrs);

A função attrset coloca as características escolhidas no par de cores no terminal, isso faz com que todas os caracteres impressos após a chamada a essa função fiquem com a cor de fundo igual a background e a cor do caractere igual a foreground do par de cores. wattrset(WINDOW *win, int attrs) é igual a attrset só que trata as cores de uma janela.

EXEMPOLO 016:
Ver exemplo número 014.

FUNÇÃO:
attron();
wattron();

PROTOTIPO:
int attron(int attrs);
int wattron(WINDOW *win,int attrs);

A função attron é responsável por ativar as características da saída da stream, como colocar o caractere em negrito, fazer piscar, colocar underline, coisas desse tipo. wattron faz a mesma coisa que attron só que para uma janela. O argumento attrs é uma constante pré-definida.(As constantes pré-definidas serão comentadas mais a frente.)

EXEMPLO 017:

```
#include<curses.h>
void main(void)
{
      WINDOW *janela;
      (void)initscr();
      (void)start_color();

      init_pair(1,COLOR_WHITE,COLOR_BLUE);
      init_pair(2,COLOR_BLACK,COLOR_CYAN);

      attrset(COLOR_PAIR(1));
      attron(A_UNDERLINE);
      mvprintw(2,(COLS-9)/2,"Olá LINUX");
      attroff(A_UNDERLINE);

      attrset(COLOR_PAIR(2));
      attron(A_BOLD);
      mvprintw(3,(COLS-9)/2,"Olá LINUX");
      attroff(A_BOLD);
      refresh();

      janela = newwin(6,15,(LINES-6)/2,(COLS-15)/2);
      box(janela,ACS_VLINE,ACS_HLINE);

      wattrset(janela,COLOR_PAIR(2));
```

```
            wattron(janela,WA_BOLD);      //sempre coloque as opções de
formatação depois da cor,..
            mvwprintw(janela,2,3,"Olá LINUX");
            wattroff(janela,WA_BOLD);
            wattrset(janela,COLOR_PAIR(1));
            wattron(janela,WA_REVERSE);
            mvwprintw(janela,3,3,"Olá LINUX");
            wrefresh(janela);

            delwin(janela);
            endwin();
      }
```

FUNÇÃO:

```
      attroff();
      wattroff();
```

PROTOTIPO:

```
      int attroff(int attrs);
      int wattroff(WINDOW *win, int attrs);
```

A função attroff() desabilita a formatação de saída da stream, wattroff() funciona da mesma forma que attroff só que com uma janela.

EXEMPLO 018:

```
      #include<curses.h>
      void main(void)
      {
            WINDOW *janela;
            (void)initscr();

            attron(A_BLINK);
            mvprintw(0,0,"Olá LINUX");
            attroff(A_BLINK);
            mvprintw(LINES-1,COLS-9,"Olá LINUX");
            refresh();

            janela = newwin(10,30,(LINES-10)/2,(COLS-30)/2);
            box(janela,ACS_VLINE,ACS_HLINE);
            wattron(janela,A_BLINK);
            mvwprintw(janela,1,1,"Olá LINUX");
            wattroff(janela,A_BLINK);
            mvwprintw(janela,8,20,"Olá LINUX");
            wrefresh(janela);

            delwin(janela);
            endwin();
      }
```

```
FUNÇÃO:
        erase();
        werase();

PROTOTIPO:
        int erase(void);
        int werase(WINDOW *win);

        erase e werase limpa toda a tela.

EXEMPLO 019:
        #include<curses.h>
        void main(void)
        {
                WINDOW *janela;
                (void)initscr();

                noecho();
                printw("Pressione qualquer tecla para poder limpar a tela");
                refresh();
                getch();
                erase();
                refresh();

                janela = newwin(3,55,(LINES-3)/2,(COLS-55)/2);
                box(janela,'|','=');
                mvwprintw(janela,1,2,"Pressione qualquer tecla para poder limpar
a tela");
                wrefresh(janela);
                wgetch(janela);
                werase(janela);
                wrefresh(janela);

                delwin(janela);
                endwin();
        }

FUNÇÃO(S) RELACIONADA(S):
        int clear(void);
        int wclear(WINDOW *win);
```

```
FUNÇÃO(s):
        clrtobot();
        wclrtobot();
        clrtoeol();
        wclrtoeol();

PROTOTIPO:
        int clrtobot(void);
        int wclrtobot(WINDOW *win);
        int clrtoeol(void);
        int wclrtoeol(WINDOW *win);

        As funções clrtobot e wclrtobot limpam a tela do cursor para baixo, já
as funções clrtoeol e wclrtoeol limpam somente a linha a partir da posição do
cursor para a direita.
```

EXEMPLO 020:

```
        /*>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
        / Apos pressionar qualquer tecla, a tela ficará com as
        / palavras "LIN" e "PIN".
        /<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<*/
        #include<curses.h>
        void main(void)
        {
                WINDOW *janela;
                (void)initscr();

                box(stdscr,ACS_VLINE,ACS_HLINE);
                mvaddch(LINES-2,2,'L');
                mvaddch(LINES-2,3,'I');
                mvaddch(LINES-2,4,'N');
                mvaddch(LINES-2,5,'U');
                mvaddch(LINES-2,6,'X');
                refresh();

                janela = newwin(9,3,(LINES-9)/2,(COLS-3)/2);
                mvwaddch(janela,2,2,'P');
                mvwaddch(janela,3,2,'I');
                mvwaddch(janela,4,2,'N');
                mvwaddch(janela,5,2,'G');
                mvwaddch(janela,6,2,'U');
                mvwaddch(janela,7,2,'I');
                mvwaddch(janela,8,2,'M');
                wrefresh(janela);
                noecho();
                getch();
                move(LINES-2,5);
                clrtoeol();

                wmove(janela,5,2);
                wclrtobot(janela);
                refresh();
                wrefresh(janela);

                delwin(janela);
                endwin();
                printf("\n");
        }
```

FUNÇÃO:

```
      mvwin();
```

PROTOTIPO:

```
      int mvwin(WINDOW *win,int y, int x);
```

wmove() muda a janela da posição atual para y linha x coluna. Lembre-se sempre de dar um refresh() para poder ver o efeito.

EXEMPLO 021:

```
      #include<curses.h>
      void main(void)
      {
            int idade;
            char chr;
            WINDOW *janela;
            (void)initscr();

            printw("Digite a sua idade:");
            refresh();
            scanw("%d",&idade);
            printw("A sua idade é %d",  idade);
            refresh();

            janela = newwin(5,45,(LINES-5)/2,(COLS-45)/2);
            box(janela,'|','-');

            mvwprintw(janela,0,18,"| LINUX |"); //não se esqueça de passar o
arg. do ponteiro.
            mvwprintw(janela,2,2,"Agora digite a primeira letra do seu nome");
            wrefresh(janela);

            mvwscanw(janela,3,2,"%c",&chr); //não se preocupe com o 'mv' logo
mais vc saberá o que é,..
            mvwprintw(janela,4,2,"Você digitou %c",chr);
            wrefresh(janela);

            mvprintw(20,4,"Pressione qualquer tecla");
            refresh();

            getch();
            clear();
            refresh();
            mvwin(janela,2,2);
            wrefresh(janela);


            delwin(janela);
            endwin();  //não esqueça,..
      }
```

FUNÇÃO:

```
      wresize();
```

PROTOTIPO:

```
      int wresize(WINDOW *win, int lines, int columns);
```

A função wresize redimensiona o tamanho da janela para y linhas e x colunas.

EXEMPLO 022:

```
      #include<curses.h>
      void main(void)
      {
            int x;
```

```
        WINDOW *janela;
        (void)initscr();
        noraw();
        noecho();
        janela = newwin(5,10,(LINES-5)/2,(COLS-10)/2);
        box(janela,ACS_BLOCK,ACS_BLOCK);
        mvwprintw(janela,0,1 ,"| 5X10 |");
        wrefresh(janela);
        sleep(2);
        for(x=0;x<20;x++){
              sleep(1);
              wclear(janela);
              wresize(janela,5+x,10+x);
              mvwin(janela,(LINES-(5+x))/2,(COLS-(10+x))/2);
              box(janela,ACS_BLOCK,ACS_BLOCK);
              mvwprintw(janela,0,(10+x-8)/2,"| %dX%d |",5+x,10+x);
              wrefresh(janela);
              beep();
          }
          delwin(janela);
          endwin();  //não esqueça,..
}
```

PANELS

Panel pode ser entendido como sendo uma janela com mais recursos.

Ou seja, lhe permite fazer tudo o que você fazia com uma WINDOW e algumas coisas a mais. Abordarei as funções básicas da PANEL, se você deseja saber mais, sugiro que procure uma documentação sobre o assunto na internet ou mesmo no manual(man) do linux.

Para utiliza os paneis da ncurses, você deve acrescentar o cabeçalho panel.h, que é onde os protótipos das funções da panel estão declarados. Para compilar uma código fonte que utiliza panel é bem simples, basta acrescentar a opção -lpanel na linha de comando.

Ex:

```
$>gcc codigo.c -lpanel -lcurses
```

Dê preferencia de colocar -lpanel antes de -lcuses, já que panel faz parte da ncurses.

CABEÇALHO <panel.h>

TIPO:

PANEL

O tipo PANEL assim como WINDOW é um tipo de estrutura de dados. Ele é usado na declaração das variáveis e/ou funções ponteiros e/ou constante que representaram o panel.

FUNÇÃO:

new_panel();

PROTOTIPO:

```
PANEL *new_panel(WINDOW *janela);
```

A função new_panel() é responsável por criar um novo panel.
Ela necessita com parâmetro um ponteiro valido para uma janela, e retorna um ponteira para o novo panel criado.
Seu uso é bem simples e similar a newwin().
Veja o exemplo mais adiante.

EXEMPLO:

Ver exemplo 24,..

```
FUNÇÃO:
         panel_window();
PROTOTIPO:
         WINDOW *panel_window(PANEL *panel);

         Esta função retorna o ponteiro da janela usada pelo panel.
         A mesma janela que é passada como parâmetro para função new_panel().

EXEMPLO 23:
      #include<curses.h>
      #include<panel.h>

      void main(void)
      {
         WINDOW *win, *w;
         PANEL *pnl;

         (void) initscr();

         win = newwin(5,5,5,5); // criando a janela,..
         pnl = new_panel(win);  // criando o panel,..

         w = panel_window(pnl); // agora w aponta para o mesmo
                                // endereço que win,..
         endwin();
      }
```

```
FUNÇÃO:
         hide_panel();
PROTOTIPO:
         int hide_panel(PANEL *panel);

         hide_panel() esconde o panel, está função não libera a memória
         usada pelo panel, apenas esconde o panel, tornando-o invisível.

EXEMPLO:
         Ver exemplo 24,..
```

```
FUNÇÃO:
         show_panel();
PROTOTIPO:
         int show_panel(PANEL *panel);

         show_panel() funciona de modo contrário a hide_panel(), esta função
         além de tornar o panel visível, ainda coloca-o na frente de todos os
         paneis que já estão a mostra.

EXEMPLO:
         Ver exemplo 24,..
```

```
FUNÇÃO:
```

```
         update_panels();
PROTOTIPO:
         void update_panels();

         update_panels() é responsável por atualizar a "tela virtual".
         Imagine update_panels como o refresh, o refresh atualiza a stream
         padrão de saída, "a tela", update_panel faz isso com o panel, só que
         na "tela virtual" que é mantida pela curses, para tornar as alterações
         realmente visíveis, você deve fazer uma chamada a doupdate() que é
         responsável por atualizar a stream padrão de saída. Chamadas a
         refresh() ou wrefresh() não vão causar nenhuma alteração no panel.

EXEMPLO 24:
      #include<curses.h>
      #include<panel.h>

      void main(void)
      {
         WINDOW *janela;
         PANEL *panel;
         int x;

         (void)initscr();
         (void)start_color();
         (void)init_pair(1,COLOR_CYAN,COLOR_BLUE);

         janela = newwin(5,10,5,5);
         panel  = new_panel(janela);
         /*================================*\
            Apartir de agora, a janela será
            tratada como um panel,..
         \*================================*/
         wbkgd(janela,COLOR_PAIR(1));
         box(janela,ACS_VLINE,ACS_HLINE);
         update_panels();
         /*================================*\
            Veja que utilizei update_panels
            ao invés de wrefresh(), só que
            as alterações ainda estão na tela
            "virtual",..
         \*================================*/
         doupdate(); // Atualiza a tela,..
         getch();

         for(x=0;x<5;x++){
            hide_panel(panel);
            update_panels(); // novamente atualizando a tel. virtual,..
            doupdate();
            sleep(1),beep();
            show_panel(panel);
            update_panels(); // novamente atualizando a tel. virtual,..
            doupdate();
            sleep(1),beep();
         }
         (void)endwin();
      }
```

```
FUNÇÃO:
         bottom_panel();
PROTOTIPO:
         int bottom_panel(PANEL *panel);

         A função bottom_panel() move o panel para traz de todos os outros
         paneis que poção existir na tela.

EXEMPLO:
         Ver exemplo 25,..
```

```
FUNÇÃO:
         top_panel();
PROTOTIPO:
         int top_panel(PANEL *panel);

         A função top_panel() move o panel para frente de todos os outros
         paneis que poção existir na tela.
         Faz justamente o contrário de bottom_panel().

EXEMPLO 25:
         #include<curses.h>
         #include<panel.h>

         void main(void)
         {
            WINDOW *win1, *win2;
            PANEL *panel1, *panel2;
            int x;

            (void) initscr();
            (void) start_color();

            (void) init_pair(1,COLOR_WHITE,COLOR_BLUE); // Cor do 1° panel,..
            (void) init_pair(2,COLOR_WHITE,COLOR_RED);  // Cor do 2° panel,..

            curs_set(FALSE);
            win1 = newwin(5,30,(LINES-5)/2,(COLS-30)/2);
            win2 = newwin(5,30,(LINES-7)/2,(COLS-40)/2);

            panel1 = new_panel(win1);
            panel2 = new_panel(win2);

            wbkgd(win1,COLOR_PAIR(1));
            wbkgd(win2,COLOR_PAIR(2));

            update_panels();
            doupdate();

            for(x=0;x<5;x++){
               bottom_panel(panel1); // Movento o panel 1 para traz,..
               update_panels();      // novamente atualizando a tel. virtual,..
               doupdate();
               sleep(1),beep();
               top_panel(panel1); // Movento o panel 1 para frente.
```

```
            update_panels();   // novamente atualizando a tel. virtual,..
            doupdate();
            sleep(1),beep();
         }
         (void)endwin();
   }
```

FUNÇÃO:

```
      move_panel();
```

PROTOTIPO:

```
      int move_panel(PANEL *panel, int LinaInicial_Y, int ColudaInicial_X);
```

A função move_panel() faz o mesmo que wmove faz com uma janela, ou seja, move o panel de lugar, use sempre esta função quando quiser mover um panel de lugar e não wmove(), esta função tem mais dois parâmetros adicionais que são:
LinhaInicial_Y, ColunaInicial_X

Esses parâmetros de coordenadas são referente a posição do canto superior esquerdo do panel.

EXEMPLO 26:

```
   #include<curses.h>
   #include<panel.h>

   void main(void)
   {
      WINDOW *win;
      PANEL  *panel;

      (void) initscr();
      win = newwin(5,30,LINES-5,COLS-30);
      panel = new_panel(win);
      box(win,ACS_VLINE,ACS_HLINE);
      mvwprintw(win,2,2,"LinhaInicial_Y: %d", LINES-5);
      mvwprintw(win,3,2,"ColunaInicial_X: %d", COLS-30);

      update_panels();
      doupdate();
      getch();

      move_panel(panel,0,0); // Movendo o panel para Linha 0 e Coluna 0,..
      mvwprintw(win,2,2,"LinhaInicial_Y: 0 ");
      mvwprintw(win,3,2,"ColunaInicial_X: 0 ");
      update_panels();
      doupdate();
      getch();

      (void) endwin();
   }
```

FUNÇÃO:

```
         panel_above();
PROTOTIPO:
         PANEL *panel_above(const PANEL *pan);

         Esta função retorna um ponteiro do panel que está sobre o
         panel apontado por pan.

         Se for passado um ponteiro nulo como parâmetro (PANEL *)0,
         a função retornará o panel que está acima de todos os outros
         paneis que estão na tela.
EXEMPLO:
         Pnl = panel_above(pan);    // Pnl aponta para o panel que está em cima
                                    // do panel pan,..

         Pnl = panel_above((PANEL *)0);  // Pnl agora aponta para o panel que
                                         // esta acima de todos os outro
                                         // paineis,..
```

```
FUNÇÃO:
         panel_below();
PROTOTIPO:
         PANEL *panel_below(const PANEL *pan);

         Esta função retorna um ponteiro do panel que está abaixo do
         panel apontado por pan.

         Se for passado um ponteiro nulo como parametro (PANEL *)0,
         a função retornará o panel que está abaixo de todos os outros
         paneis que estão na tela.

         Esta função faz o contrário de panel_above().
```

```
FUNÇÃO:
         del_panel();
PROTOTIPO:
         int del_panel(PANEL *panel);

         Como o próprio nome já insinua, está função deleta o panel,
         liberando a memória alocada para o mesmo.

         Mas lembre-se, esta função apenas libera a memória, ela não
         causa nenhuma alteração na stream padrão de saída.
```

MOUSE

A biblioteca NCURSES também prove uma interface para o mouse, mas atenção, essa interface não é padrão para todas as plataformas que suportam ncurses. Se você está pensando em fazer um programa portável, sugiro que use os pré-compiladores com a macro NCURSES_MOUSE_VERSION, para ver se a ncurses em questão suporta o mouse.

Para capturar as coordenadas do mouse, assim como o seu evento, é necessário fazer uma chamada a função wgetch para que o programa fique no estado de espera, esperando um evento do usuário. Quando função wgetch captura o retorno do mouse, ela devolve um valor de pseudo-código igual a constante KEY_MOUSE.

Atualmente o mouse só funcionará no xterm, nas versões posteriores à 5.2, poderá haver algumas alterações para que também possa funcionar no shell.

Todos os protótipos de funções e macros e tipos de variáveis estão declarados no arquivo de cabeçalho curses.h [ncurses.h].

Tentarei explicar de uma forma simples e bem direta cada função relacionada ao uso do mouse. O exemplo mostrado no final desse tópico não fará o uso de todas as funções, fica a seu cargo as implementações necessária para o uso das mesmas.

A tabela de contantes eu copiei do manual curs_mouse, por isso não está traduzido. Em curs_mouse você também encontra uma pequena referência sobre mouse_trafo, wmouse_trafo, sugiro que você mesmo procure as informações necessária para o uso das mesmas.

TIPOS DE VARIÁVEIS:

```
typedef unsigned long mmask_t;

typedef struct
{
   short id;           // ID para destinguir o dispositivo,..
   int x;              // Posição da coordenada x do mouse,..
   int y;              // Posição da coordenada y do mouse,..
   int z;              // Posição z do mouse. Sem muita utilização por
                       // enquanto,..
   mmask_t bstate;     // Evento disparado pelo mouse,..
} MEVENT;
```

FUNÇÃO:
mousemask();
PROTOTIPO:

```
mmask_t mousemask(mmask_t newmask, mmask_t *oldmask);
```

A função mousemask é responsável por habilitar o retorno dos eventos disparados pelo mouse. O retorno é uma "mascara" para especificar quais tipos de eventos podem ser retornados.

Se o argumentos oldmask não for um ponteiro mmask_t nulo, pode ser usado para se saber qual foi o penúltimo evento disparado.

FUNÇÃO:

getmouse();

PROTOTIPO:

```
int getmouse(MEVENT *event);
```

A função getmouse é responsável por retorna para a estrutura MEVENT as propriedades do mouse, como as coordenadas o id e o evento disparado.

O uso dessa função é tão simples quanto usar getch().

EXEMPLO 27:

```
#include<curses.h>
void main(void)
{
   MEVENT mevent;
   mmask_t mask;
   int key;

   (void) initscr();
   (void) keypad(stdscr,TRUE);
   (void) mousemask(mask,(mmask_t *)NULL);

   key = getch();

   if(key == KEY_MOUSE){
      (void) getmouse(&mevent);
      printw("x: %d y: %d",mevent.x,mevent.y); // Imprimindo as
                                               // coordenadas do mouse,..
      refresh();
   }

   (void) endwin();
}
```

FUNÇÃO:

ungetmouse();

PROTOTIPO:

```
int ungetmouse(MEVENT *event);
```

A função ungetmouse faz o contrário de getmouse, ela coloca um evento na fila de eventos do mouse.

EXEMPLO 28:

```
#include<curses.h>
void main(void)
{
   MEVENT mevent;
   mmask_t mask;

   (void) initscr();
```

```
         (void) mousemask(mask,(mmask_t *)NULL);
         mevent.x = 10;
         mevent.y = 11;

         (void) ungetmouse(&mevent);  // Colocando o evento na fila,..
         (void) getmouse(&mevent);    // Retirando o evento da fila,..
         printw("x: %d y: %d",mevent.x,mevent.y);
         refresh();
         sleep(2),beep();
         (void) endwin();
      }
```

```
FUNÇÃO:
      wenclose();
PROTOTIPO:
      bool wenclose(WINDOW *win, int y, int x);

      A finalidade dessa função é verificar se as coordenadas x e y
      estão dentro do "quadrante" da janela apontada por *win. Ela
      é ultio quando você quer verificar se uma determinada posição
      do par de coordenadas pode ser usada pelo seu programa.
      Retorna TRUE se sim, e FALSE e não.

EXEMPLO 29:
   #include<curses.h>
   void main(void)
   {
      int x=1, y;
      mmask_t mask;

      (void) initscr();
      (void) mousemask(mask,(mmask_t *)NULL);

      while(x!=1000){ // FLAG,..
         clear();
         mvprintw(1,1,"Digite x = 1000 para sair.");
         mvprintw(2,3,"Qual o valor de x:");
         mvprintw(3,3,"Qual o valor de y:");
         refresh();
         mvscanw(2,21,"%d",&x);
         if(x==1000) continue;   // * Não faça uso indiscriminado disso,..
         mvscanw(3,21,"%d",&y);
         if(wenclose(stdscr,y,x))
            printw("A coordenada é valida!!");
         else
            printw("A coordenada é invalida!!");
         refresh();
         sleep(2);
      }
      (void) endwin();
   }
```

```
FUNÇÃO:
      mouseinterval();
PROTOTIPO:
      int mouseinterval(int erval);

      Esta função específica o intervalo máximo em milisegundos entre
      o pressionamento e a liberação do botão do mouse.
      Por padrão o valor é 200 milisegundos ( 1/5 de segundo).
```

CONSTANTES DO MOUNSE:

Nome	Descrição
BUTTON1_PRESSED	mouse button 1 down
BUTTON1_RELEASED	mouse button 1 up
BUTTON1_CLICKED	mouse button 1 clicked
BUTTON1_DOUBLE_CLICKED	mouse button 1 double clicked
BUTTON1_TRIPLE_CLICKED	mouse button 1 triple clicked
BUTTON2_PRESSED	mouse button 2 down
BUTTON2_RELEASED	mouse button 2 up
BUTTON2_CLICKED	mouse button 2 clicked
BUTTON2_DOUBLE_CLICKED	mouse button 2 double clicked
BUTTON2_TRIPLE_CLICKED	mouse button 2 triple clicked
BUTTON3_PRESSED	mouse button 3 down
BUTTON3_RELEASED	mouse button 3 up
BUTTON3_CLICKED	mouse button 3 clicked
BUTTON3_DOUBLE_CLICKED	mouse button 3 double clicked
BUTTON3_TRIPLE_CLICKED	mouse button 3 triple clicked
BUTTON4_PRESSED	mouse button 4 down
BUTTON4_RELEASED	mouse button 4 up
BUTTON4_CLICKED	mouse button 4 clicked
BUTTON4_DOUBLE_CLICKED	mouse button 4 double clicked
BUTTON4_TRIPLE_CLICKED	mouse button 4 triple clicked
BUTTON_SHIFT	shift was down during button state change
BUTTON_CTRL	control was down during button state change
BUTTON_ALT	alt was down during button state change
ALL_MOUSE_EVENTS	report all button state changes
REPORT_MOUSE_POSITION	report mouse movement

EXEMPLO GERAL:

```
#include<curses.h>

void main(void)
{
   MEVENT mevent;
   mmask_t mt;
   int key=0;

   (void) initscr();
   (void) start_color();
   (void) keypad(stdscr,TRUE);
   (void) mousemask(mt, (mmask_t *)NULL);
   curs_set(0);
   noecho();
   init_pair(1,COLOR_CYAN,COLOR_BLUE);
```

```
while(key!=27){
   key = wgetch(stdscr);
   clear();
   if(key == KEY_MOUSE){
      getmouse(&mevent);
      box(stdscr,ACS_BLOCK,ACS_BLOCK);
      if(mevent.bstate == BUTTON1_PRESSED)
         mvprintw(4,2,"BUTTON1_PRESSED");
      else if(mevent.bstate == BUTTON1_RELEASED)
         mvprintw(4,2,"BUTTON1_RELEASED");
      else if(mevent.bstate == BUTTON1_CLICKED)
         mvprintw(4,2,"BUTTON1_CLICKED");
      else if(mevent.bstate == BUTTON1_DOUBLE_CLICKED)
         mvprintw(4,2,"BUTTON1_DOUBLE_CLICKED");
      else if(mevent.bstate == BUTTON1_DOUBLE_CLICKED)
         mvprintw(4,2,"BUTTON1_DOUBLE_CLICKED");
      else if(mevent.bstate == BUTTON1_TRIPLE_CLICKED)
         mvprintw(4,2,"BUTTON1_TRIPLE_CLICKED");
      else if(mevent.bstate == BUTTON2_PRESSED)
         mvprintw(4,2,"BUTTON2_PRESSED");
      else if(mevent.bstate == BUTTON2_RELEASED)
         mvprintw(4,2,"BUTTON2_RELEASED");
      else if(mevent.bstate == BUTTON2_CLICKED)
         mvprintw(4,2,"BUTTON2_CLICKED");
      else if(mevent.bstate == BUTTON2_DOUBLE_CLICKED)
         mvprintw(4,2,"BUTTON2_DOUBLE_CLICKED");
      else if(mevent.bstate == BUTTON2_TRIPLE_CLICKED)
         mvprintw(4,2,"BUTTON2_TRIPLE_CLICKED");
      else if(mevent.bstate == BUTTON3_PRESSED)
         mvprintw(4,2,"BUTTON3_PRESSED");
      else if(mevent.bstate == BUTTON3_RELEASED)
         mvprintw(4,2,"BUTTON3_RELEASED");
      else if(mevent.bstate == BUTTON3_CLICKED)
         mvprintw(4,2,"BUTTON3_CLICKED");
      else if(mevent.bstate == BUTTON3_DOUBLE_CLICKED)
         mvprintw(4,2,"BUTTON3_DOUBLE_CLICKED");
      else if(mevent.bstate == BUTTON3_TRIPLE_CLICKED)
         mvprintw(4,2,"BUTTON3_TRIPLE_CLICKED");
      else if(mevent.bstate == BUTTON_SHIFT)
         mvprintw(4,2,"BUTTON_SHIFT");
      else if(mevent.bstate == BUTTON_CTRL)
         mvprintw(4,2,"BUTTON_CTRL");
      else if(mevent.bstate == BUTTON_ALT)
         mvprintw(4,2,"BUTTON_ALT");

      mvprintw(2,2,"Posição X: %d",mevent.x);
      mvprintw(3,2,"Posição Y: %d",mevent.y);

      attron(COLOR_PAIR(1));
      mvprintw(LINES/2,(COLS-8)/2, "[ Sair ]");
      attroff(COLOR_PAIR(1));

      if(mevent.bstate == BUTTON2_PRESSED && (mevent.x>((COLS-8)/2) ||
         mevent.x<(COLS-8)/2 + 8) && mevent.y==LINES/2){

         attron(A_REVERSE);
```

```
            mvprintw(LINES/2,(COLS-8)/2 , "[ Sair ]");
            attroff(A_REVERSE);
          }
          else if(mevent.bstate == BUTTON1_CLICKED && (mevent.x>((COLS-8)/2) ||
                  mevent.x<(COLS-8)/2 + 8) && mevent.y==LINES/2)

           key = 27;
       }
       refresh();
    }
    (void) endwin();
}
```

CONSTANTES:

LINES - constante que contem o número de linha da tela do terminal.
COLS - constante que contem o número de colunas da tela do terminal.

CONSTANTES DE COR:

```
COLOR_BLACK     = 0
COLOR_RED       = 1
COLOR_GREEN     = 2
COLOR_YELLOW    = 3
COLOR_BLUE      = 4
COLOR_MAGENTA   = 5
COLOR_CYAN      = 6
COLOR_WHITE     = 7
```

CONSTANTES DE SIMBOLOS:

Alguns significados de constantes estão em inglês porque a sua tradução o deixaria um tanto sem sentido.

```
ACS_ULCORNER    - Canto superior esquerdo.
ACS_LLCORNER    - Canto inferior esquerdo.
ACS_URCORNER    - Canto superior direito.
ACS_LRCORNER    - Canto inferior direito.
ACS_LTEE        - Meta ponto esquerdo.
ACS_RTEE        - Meta ponto direito.
ACS_BTEE        - Meta ponto inferior.
ACS_TTEE        - Meta ponto superior.
ACS_HLINE       - Linha horizontal.
ACS_VLINE       - Linha Vertical.
ACS_PLUS        - Grande sinal de mais(+).
ACS_S1          - Linha esquadrinha 1.
ACS_S9          - Linha esquadrinha 9.
ACS_DIAMOND     - Diamante.
ACS_CKBOARD     - Checker board.
```

```
ACS_DEGREE      - Degree.
ACS_PLMINUS     - Mais/menos.
ACS_BULLET      - É um ponto ( . ).
ACS_LARROW      - Seta para esquerda.
ACS_RARROW      - Seta para direita.
ACS_DARROW      - Seta para baixo.
ACS_UARROW      - Seta para cima.
ACS_BOARD       - Bordas de um quadrado.
ACS_LANTERN     - Simbolo de uma lanterna.
ACS_BLOCK       - Bloco quadrado solido.
ACS_S3          - Scan line 3.
ACS_S7          - Scan line 7.
ACS_LEQIAL      - Menor igual.
ACS_GEQUAL      - Maior igual.
ACS_PI          - Pi.
ACS_NEQUAL      - Diferente.
ACS_STERLING    - UK pound sign.
```

CONSTANTES DE FORMATAÇÃO PARA SAÍDA DE TELA:

```
                A_ATTRIBUTES
WA_NORMAL       A_NORMAL
WA_STANDOUT     A_STANDOUT
WA_UNDERLINE    A_UNDERLINE
WA_REVERSE      A_REVERSE
WA_BLINK        A_BLINK
WA_DIM          A_DIM
WA_BOLD         A_BOLD
WA_ALTCHARSET   A_ALTCHARSET
WA_INVIS        A_INVIS
WA_PROTECT      A_PROTECT
WA_HORIZONTAL   A_HORIZONTAL
WA_LEFT         A_LEFT
WA_LOW          A_LOW
WA_RIGHT        A_RIGHT
WA_TOP          A_TOP
WA_VERTICAL     A_VERTICAL
```

Bibliografia:

Paginas do manual do linux. A maioria das informações de que você precisa pode ser encontrada ai, basta arregaçar as mangas e começar a ler.
Para ter acesso a primeira parte do manual digite:

```
$>man ncurses
```

Apoio:

Diêgo Rodrigues de Melo
http://ucg.mobilebr.com/