by Reha K. Gerçeker
<gerceker/at/itu.edu.tr>

About the author:

O Reha é um estudante de engenharia informática em Istanbul, na Turquia. Ela adora a liberdade que o Linux fornece como plataforma de desenvolvimento de software. Ele passa muito do seu tempo à frente do seu computador, escrevendo programas. Ele deseja tornar−se um programador inteligente. num destes dias.

Translated to English by:
Reha K. Gerçeker
<gerceker/at/itu.edu.tr>

Introdução às Ncurses

Abstract:

As Ncurses são uma biblioteca que fornecem o mapeamento para teclas de função, funções de desenho de ecrãs e a possibilidade de utilizar múltiplas janelas não sobrepostas em terminais à base de texto.

O que é que são as Ncurses?

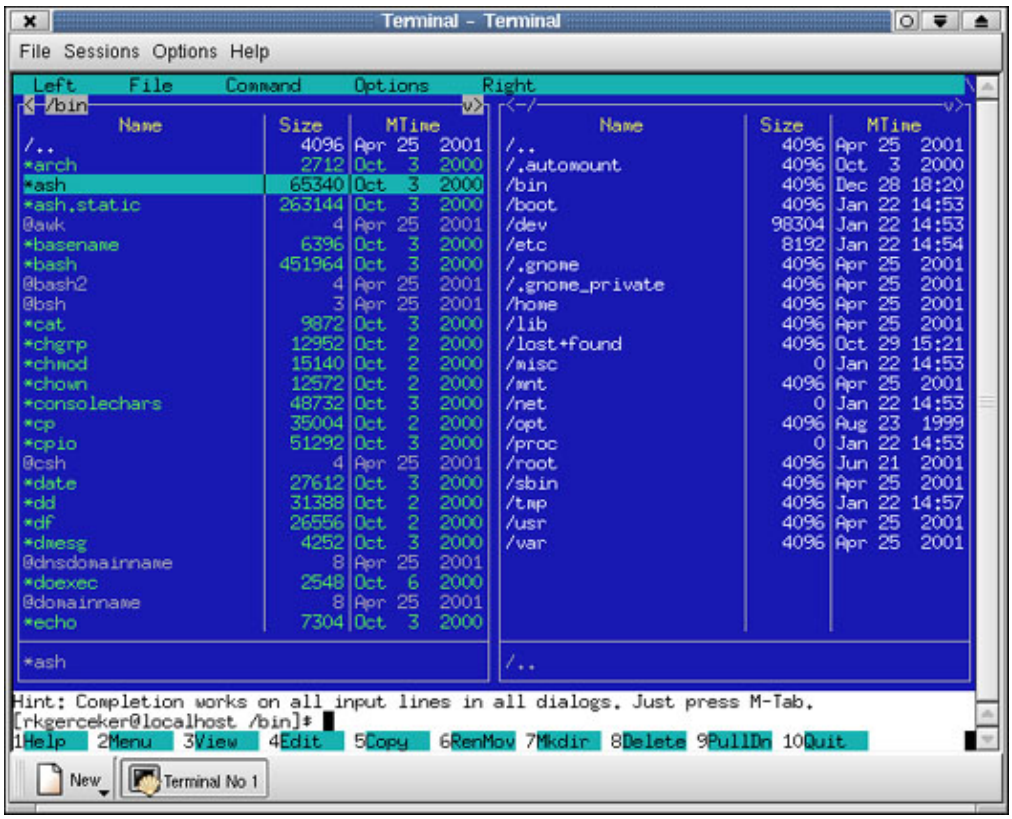
Quer que o seu programa tenha uma interface colorida nos terminais de texto? As Ncurses são uma biblioteca que lhe fornecem funcionalidades de janelas em terminais à base de texto. Coisas de que as ncurses são capazes:

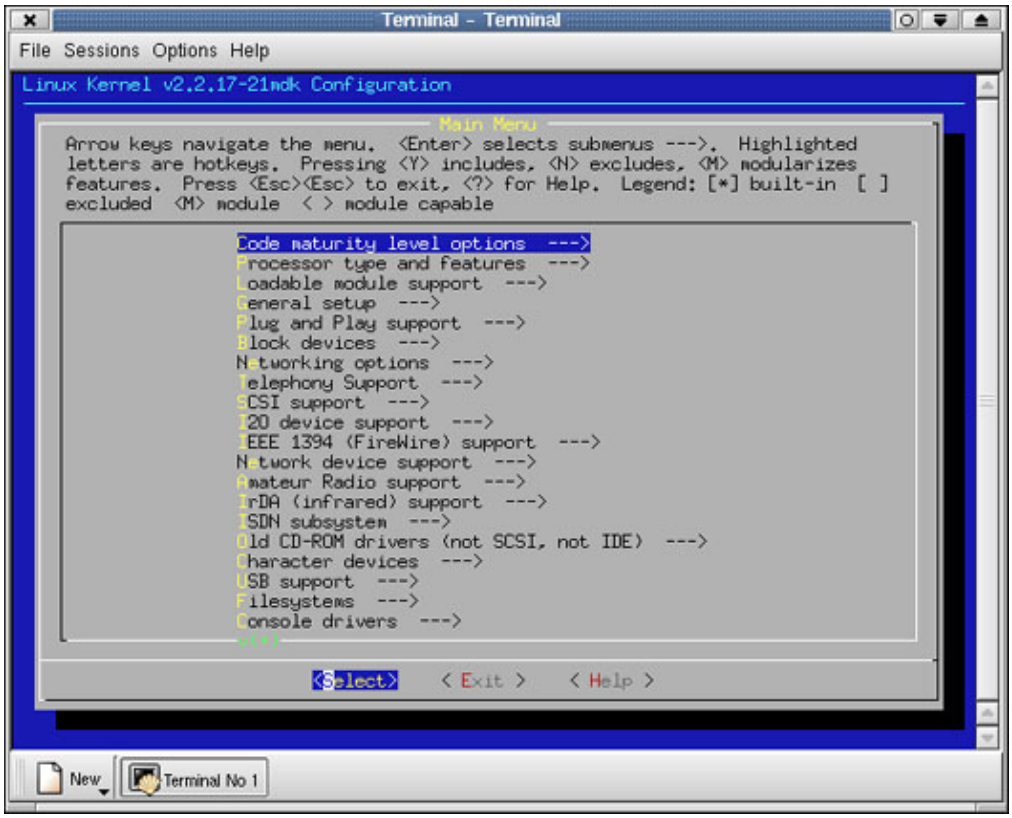
- Utilizar todo o ecrã como desejar.
- Criar e gerir janelas.
- Utilizar 8 cores diferentes.
- Dar suporte de rato aos seus programas.

- Utilizar as teclas de função do teclado.

É possível utilizar as ncurses em qualquer sistema UNIX que obedecem à norma ANSI/POSIX. À parte disto, a biblioteca é capaz de detectar propriedades do terminal a partir da base de dados do sistema e agir em conformidade, fornecendo uma interface independente do terminal. Assim as ncurses podem ser utilizadas, fiavelmente, para trabalhos que são supostos trabalhar em plataformas diferentes e em vários terminais.

O Midnight Commander é um dos exemplos escritos com as ncurses. A interface utilizada para a configuração do Kernel é escrita com as ncurses. Pode ver as suas fotografias instantâneas abaixo.





Onde fazer Download?

As Ncurses são desenvolvidas sob a GNU/Linux. Para obter a última versão, obtenha informação detalhada e outras ligações relativas às ncurses, visite www.gnu.org/software/ncurses/.

Elementos Básicos

No sentido de utilizar a biblioteca, deve incluir no seu código fonte a curses.h, e tenha a certeza de ligar o seu código à biblioteca das curses. Isto pode ser feito através do parâmetro –lcurses dado ao gcc.

É necessário conhecer a estrutura de dados fundamental, enquanto trabalha com as ncurses. Ou seja a estrutura WINDOW, como se depreende facilmente pelo nome, é usada para representar as janelas que cria. Praticamente todas as funções da biblioteca recebem como parâmetro um ponteiro do tipo WINDOW.

Os componentes comuns mais utilizados são janelas. Mesmo que não crie as suas próprias janelas o ecrã é considerado como uma só janela. Como o descritor do tipo FILE, *stdout* da biblioteca de entrada/saída padrão representa o ecrã (quando não há redirecções), as ncurses têm um ponteiro do tipo WINDOW, *stdscr* que faz o mesmo trabalho. Adicionalmente ao *stdsrc*, outro ponteiro do tipo WINDOW, com o nome de *curscr* é definido na biblioteca. O *stdsrc* representa o ecrã, o *cursrc* representa o ecrã corrente para a biblioteca. Pode perguntar "Qual é a diferença?" Continue a leitura.

No sentido de utilizar as funções ncurses e variáveis nos seus programas, tem de chamar a função *initscr*. Esta função aloca memória para as variáveis como *stdscr*, *curscr* e torna a biblioteca pronta para ser utilizada. Por outras palavras, todas as funções das ncurses devem ser precedidas de *initscr*. Do mesmo modo deve chamar a função *endwin* quando termina todo o trabalho com as ncurses. Isto liberta a memória utilizada pelas ncurses.

Depois de chamar a função *endwin* não pode utilizar as funções das ncurses a não ser que chame novamente a função *initscr*.

Entre as chamadas à função *initscr* e a *endwin*, certifique−se de não enviar dados de saída para o ecrã utilizando as funções da biblioteca padrão. Caso contrário, pode obter um ecrã não pretendido e normalmente corrompido. Quando as ncurses estão activas, utilize as suas próprias funções para enviar os dados de saída para o ecrã. Antes de chamar a função *initsrc* e depois de chamar a *endwin*, pode fazer o que quiser.

Actualizando o ecrã: refrescamento

A estrutura WINDOW não somente mantém a altura, o comprimento e a posição da janela mas também assegura os conteúdos da janela. Quando escreve para uma janela os conteúdos da janela alteram−se, mas isto não quer dizer que apareçam logo no ecrã imediatamente. No sentido de ter o ecrã actualizado, quer o *refresh* ou o *wrefresh* têm de ser invocados.

Aqui está a diferença entre o *initscr* e o *curscr*. Enquanto que o *curscr* guarda os conteúdos do ecrã corrente, o *stdscr* pode ter informação diferente após as chamadas de output das ncurses. Se quiser que as últimas alterações no *stdscr* se reflictam no *curscr*, precisa de chamar o *refresh*. Por outras palavras a função *refresh* é a única que lida com o *curscr*. É recomendável que não mexa no *curscr* e o deixe para ser actualizado pela função *refresh*.

A função *refresh* tem um mecanismo de actualizar o ecrã o mais rápido possível. Quando a função é chamada, só actualiza as linhas que foram alteradas da janela. Isto salvaguarda tempo de CPU, bem como previne o programa de escrever a mesma informação novamente no ecrã. Este mecanismo é a razão pela qual as funções das ncurses e as funções da biblioteca de entrada/saída padrão podem produzir maus resultados quando utilizadas em conjunto; quando as funções das ncurses são chamadas elas alteram o valor de uma flag que diz à função *refresh* que a linha se alterou, enquanto que com as funções da biblioteca de entrada/saída padrão nada disto se passa.

As funções *refresh* e *wrefresh*, basicamente, fazem o mesmo. A função *wrefresh* recebe por parâmetro um ponteiro do tipo WINDOW e refresca somente os conteúdos dessa janela. *refresh()* é equivalente a *wrefresh(stdscr)*. Como mencionarei mais tarde, assim como a função *wrefresh* a maioria das funções das ncurses têm macros que aplicam estas modificações ao *stdscr*.

Criando Novas Janelas

Falemos agora acerca da *subwin* e da *newwin*, as funções que criam novas janelas. Ambas recebem por parâmetro a altura, o comprimento, as coordenadas do canto esquerdo superior da nova janela. Retornam um ponteiro do tipo WINDOW que representa a nova janela. Pode utilizar este novo ponteiro com a função *wrefresh* e outras que falarei mais tarde.

"Se fazem o mesmo, porquê duplicar as funções?" poderá estar a questionar−se. Você tem razão, elas são um pouco diferentes. A função *subwin* cria uma nova janela como sendo uma sub−janela de outra. Uma janela criada deste modo, herda propriedades da janela mãe. Estas propriedades podem ser mais tarde alteradas sem afectar a janela mãe.

Apesar disto, há uma coisa que une a janela mãe à janela filha. A tabela de caracteres que mantém os conteúdos de uma janela é partilhado entre as janelas mãe e filha. Por outras palavras, os caracteres na intersecção das duas janelas, pode ser alterado por qualquer uma delas. Se a janela mãe escreve para tal espaço, o conteúdo da janela filha é também alterado. O inverso também se aplica.

Ao contrário da função *subwin*, a função *newwin* cria verdadeiramente uma nova janela. Tal janela, a não ser que possua as suas próprias sub−janelas, não partilha a sua tabela de caracteres com outra janela. A vantagem de utilizar a função *subwin* é que a utilização de uma tabela de caracteres partilhada usa menos memória. Contudo, quando janelas escrevem umas por cima das outras, a utilização da função *newwin* traz as suas próprias vantagens.

Pode criar as suas próprias janelas em qualquer profundidade. Qualquer sub−janela pode ter as suas próprias sub−janelas, mas tenha em mente que a mesma tabela de caracteres é partilhada por mais do que duas janelas.

Quando não precisar mais da janela que criou, pode apagá−la utilizando a função *delwin*. Sugiro que consulte as páginas man para a lista de parâmetros destas funções.

Escreva para as Janelas, leia das Janelas

Falámos acerca do *stdscr*, do *curscr*, refrescar o ecrã e criar novas janelas. Mas então, como é que escrevemos para uma janela? Ou como é que lemos dados de uma janela?

As funções utilizadas para este propósitos assemelham−se às funções correspondentes da biblioteca de entrada/saída padrão. Entre estas funções estão a *printw* em vez da *printf*, a *scanw* em vez da *scanf*, a *addch* em vez da putc ou a *putchar*, *getch* em vez da *getc* ou *getchar*. São utilizadas como normalmente, só os seus nomes são diferentes. Semelhantemente, a função *addstr* podia ser utilizada para escrever uma string para uma janela e a função *getstr* para ler uma string de uma janela. Todas estas funções com uma letra 'w' adicionada à frente do seu nome e que recebem como primeiro parâmetro um ponteiro do tipo WINDOW, fazem o seu trabalho numa janela diferente da *stdscr*. Por exemplo, *printw* e *wprintw(stdsrc, ...)* são equivalentes, tal e qual como *refresh* e *wrefresh(stdscr)*.

Seria uma história muito comprida, entrar nos detalhes destas funções. As páginas mansão a melhor fonte para aprender as suas descrições, protótipos, valores de retorno e outras notas. Sugiro que verifique as páginas man para cada função que usar. Elas oferecem informação detalhada e valiosa. A última secção deste artigo, onde apresento um programa de exemplo pode também servir de tutorial em como utilizar estas funções.

Cursores Físicos e Lógicos.

É preciso explicar os cursores físicos e lógicos depois de falar em como escrever e ler das janelas. O cursor físico é o cursor habitual que pisca no ecrã e, só existe um cursor físico. Por outro lado, os cursores lógicos pertencem às ncurses e cada janela tem um deles. Assim podem existir vários cursores lógicos.

O cursor lógico está na posição da janela onde se iniciará o processo de leitura ou escrita. Assim, sendo capaz de mover o cursor lógico significa que pode escrever em qualquer sítio do ecrã e em qualquer altura. Isto é uma vantagem das ncurses sob a biblioteca de entrada e saída padrão.

A função que move o cursor lógico é a *move* ou como pode advinhar a *wmove*. A função *move* é uma macro da *wmove*, escrita para o *stdscr*.

Outra matéria é a coordenação dos cursores lógicos e físicos. A posição do cursor físico terminará após um processo de escrita e é determinada pela flag *_leave* que existe na estrutura WINDOW. Se a flag *_leave* está activada o cursor lógico é movido para a posição do cursor físico (onde o último caracter é escrito) após o término da escrita. Se a flag *_leave* não está definida, o cursor físico regressa à posição do cursor lógico (onde o primeiro caracter é escrito) após o término da escrita. A flag *_leave* é controlada pela função *leaveok*.

A função que move o cursor físico é a *mvcur*. Ao contrário de outras, a função *mvcur* tem efeito imediato e não após o próximo refrescamento. Se quiser que o cursor físico seja invisível, então use a função *curs_set*. Verifique as páginas man para os detalhes.

Existem, também macros, que combinam as funções de escrita e movimento, descritas acima numa só chamada. São explicadas, de um modo simpático, nas mesmas páginas man que as funções *addch*, *addstr*, *printw*, *getch*, *getstr*, *scanw*, etc.

Limpando as Janelas

A escrita para janelas está feita. Mas como é que limpamos as janelas, as linhas, ou os caracteres?

Limpar nas ncurses significa preencher o quadrado, a linha ou os conteúdos da janela com espaços brancos. As funções que eu explicarei abaixo, preenchem os espaços necessários com espaços brancos e limpam, assim o ecrã.

Primeiro, falemos de funções que limpam um caracter ou uma linha. As funções *delch* e *wdelch* apagam o caracter que está sobre o cursor lógico da janela e reposicionam os caracteres que se seguem na mesma linha. A função *deleten* e *wdeleteln* apagam a linha onde está posicionada o cursor lógico e move para cima todas as linhas seguintes.

As funções *clrtoeol* e *wclrtoeol* apagam todos os caracteres na mesma linha à direita do cursor lógico. As funções *clrtobot* e *wclrtobot* primeiro chamam a função *wclrtoeol* para apagar todos os caracteres à direita da cursor lógico e depois apagam todas as linhas seguintes.

Outras, além destas, limpam todo o ecrã ou toda uma janela. Existem dois métodos para limpara todo um ecrã. O primeiro é preencher todo um espaço com espaços brancos e chamar a função *refresh* e o outro é utilizar um código de controle, incorporado de terminal. O primeiro método é mais lento que o segundo pois requer que todos espaços no ecrã sejam rescritos, enquanto que o segundo limpa o ecrã imediatamente.

A função *erase* e *werase* preenchem o vector de caracteres de uma janela com espaços brancos. No próximo refrescamento, a janela é limpa. Contudo, se a janela a ser limpa preenche todo o ecrã, não é lá muito inteligente utilizar estas funções. Elas utilizam o primeiro método descrito acima. Quando a janela a ser limpa é do tamanho do ecrã, tem vantagem utilizar as funções abaixo.

Antes de passar a outras funções, é tempo de mencionar a flag *_clear*. Existe na estrutura WINDOW e se estiver activada, pede à função *refresh* para enviar um código de controle ao terminal quando é chamada. Quando chamada, a função *refresh* verifica se a janela é do tamanho do ecrã (utilizando a flag _FULLWIN) e se for limpa o ecrã com um método incorporado de terminal. Só escreve caracteres em vez dos espaços brancos no ecrã. Isto torna a limpeza de todo o ecrã mais rápida. A razão porque o método de terminal é só utilizada para janelas que preenchem todo o ecrã é que o código de controle de terminal limpa todo o ecrã e

não somente e janela. A flag *_clear* é controlada pela função *clearok*.

As funções *clear* e *wclear* são utilizadas para limpar janelas do tamanho do ecrã. De facto, estas funções são equivalentes a chamar a função *werase* e *clearok*. Primeiro preenchem o vector de caracteres da janela com espaços brancos. Depois, definindo flag *_clear*, elas limpam o ecrã utilizando o método incorporado de terminal se a janela é do tamanho do ecrã ou refrescam todos os espaços da janela preenchendo−a com espaços brancos.

Como resultado, se souber que a janela a ser limpa ocupa todo o ecrã então utilize a função *clear* ou *wclear*. Produz o resultado mais rápido. Contudo, não existe diferença em utilizar a função *wclear* ou *werase* quando a janela não é do tamanho do ecrã.

Utilizando Cores

As cores que vê no ecrã devem ser pensadas como pares de cores. Isto porque cada quadrado tem uma cor de fundo e de frente. Escrever cores com as ncurses significa criar os seus próprios pares de cores e utilizá−los para escrever para uma janela.

Assim como o *initscr* precisa de ser chamado para iniciar as ncurses, a função *start_color* precisa de ser chamada para inicializar as cores. A função que precisa para criar o seu par de cores é a *init_pair*. Quando cria um par de cores com a função *init_pair*, este par fica associado ao número que deu à função como primeiro parâmetro. Assim, sempre que se quiser utilizar este par, refere−se a ele chamando o COLOR_PAIR com o respectivo número associado.

Para além de criar os pares de cores, precisa de ter funções que escrevam com um par de cor diferente. Isto faz−se através das funções *attron* e *wattron*. Estas funções, até que a função *attroff* ou a *wattroff* sejam chamadas, fazem com que tudo seja escrito na janela correspondente com a cor do par que escolheu.

Existem, também as funções *bkgd* e *wbkgd* que alteram o par da cor que está associado a uma janela inteira. Quando chamadas, alteram as cores de fundo e frente de todos os espaços da janela. Ou seja, no próximo refrescamento, todos os espaços da janela são rescritos com o novo par de cores.

Veja as páginas man acerca das cores disponiveis e dos detalhes das funções mencionadas aqui.

Caixas, à volta das Janelas

Você cria caixas à volta das suas janelas para criar um bom visual aso seu programa. Existe uma macro na biblioteca chamada *box* que o faz por si. Ao contrário de outras a função/macro *wbox* não existe; a *box* recebe como parâmetro um ponteiro do tipo WINDOW.

Pode encontrar facilmente os detalhes da função *box* nas páginas man. Há algo mais que deve ser mencionado. Pôr uma janela dentro de uma caixa significa, simplesmente, escrever os caracteres necessários para o vector de caracteres da janela e que corresponde aos limites de fronteiras. Se, mais tarde escrever para tais fronteiras, a caixa pode ficar corrompida. Para se prevenir disto, você cria uma janela interna, dentro da janela original com a função *subwin*, ponha a janela original numa caixa e utiliza a janela interna para escrever para a janela quando for necessário.

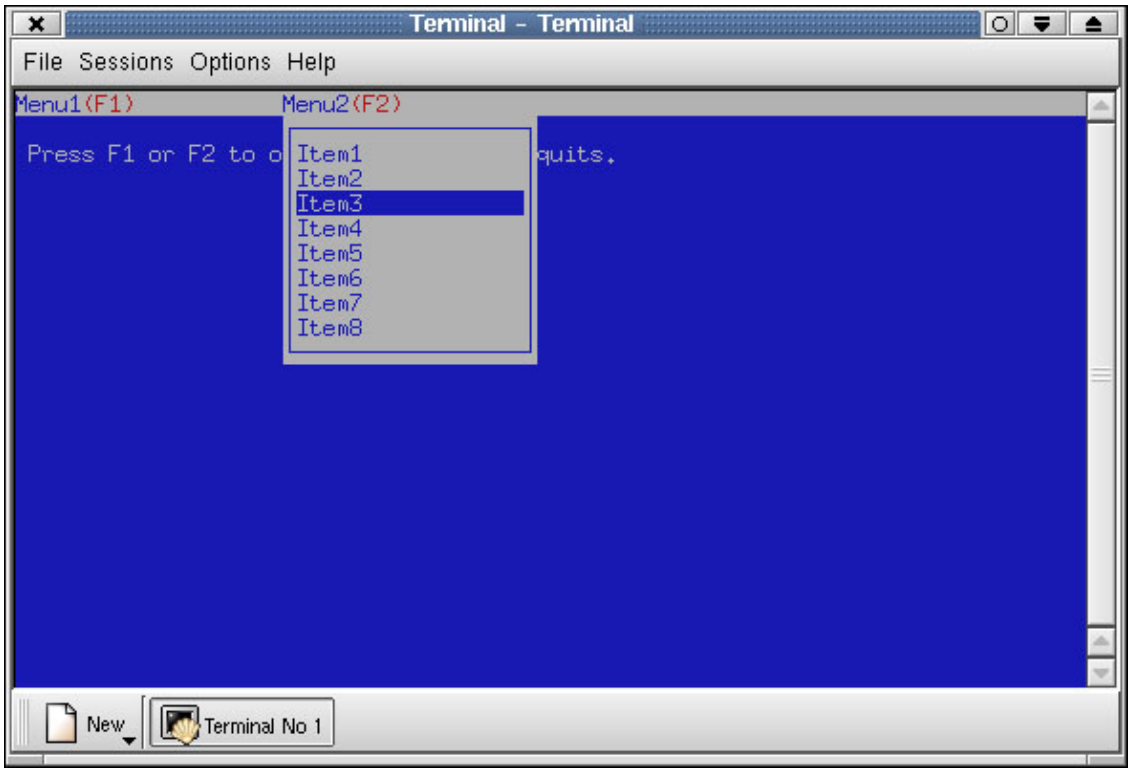
Teclas de Função

No sentido de poder utilizar as teclas de função, a flag *_use_keypad* deve ser activada na janela de onde está a obter os dados. *keypad* é a função que define o valor de *_use_keypad*. Quando define o valor de *_use_keypad*, pode obter dados do teclado com as funções de entrada de dados, normalmente.

Neste caso, se por exemplo, utilizar a função *getch* para obter dos dados deve ter cuidado para guardar os dados numa variável do tipo int em vez de uma variável do tipo char. Isto é assim porque os valores numéricos das teclas de função são superiores aqueles que uma variável do tipo char pode suportar. Não precisa de saber estes valores numéricos das teclas de função, utilize em vez dos números os nomes definidos na biblioteca. Estes nomes estão listados na página man da função *getch*.

Um exemplo

Vamos agora analisar um programa simples e simpático. Neste programa, criam−se menus utilizando as ncurses e a selecção de uma opção do menu é demonstrada. Um aspecto interessante deste programa é a utilização das janelas ncurses para gerar o efeito de menu. Pode ver a sua fotografia abaixo.



O programa começa com ficheiro header incluídos, como é habitual. Depois definimos as constantes que são os valores ASCII das teclas enter e escape.

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

#define ENTER 10
#define ESCAPE 27
```

A função abaixo é a primeira a ser chamada quando o programa corre. Primeiro chama a função *initscr* para inicializar as ncurses e a função *start_color* para tornar o uso das cores possíveis. Os pares de cores que serão utilizados no programa são definidos logo de seguida. A função *curs_set(0)* torna o cursor físico invisível. A função *noecho* impede que os dados introduzidos através do teclado sejam apresentados no ecrã. Pode também utilizar a função *noecho* para controlar a entrada de dados através do teclado e apresentar somente as partes que deseja apresentar. A função *echo* deve ser chamada quando é necessário eliminar o efeito de *noecho*. A função abaixo chama por fim a função *keypad* para activar as teclas de função quando obtendo dados de entrada a partir do *stdscr*. Isto é necessário visto utilizarmos as teclas F1, F2 e os cursores mais tarde no programa.

```
void init_curses()
{
    initscr();
    start_color();
    init_pair(1,COLOR_WHITE,COLOR_BLUE);
    init_pair(2,COLOR_BLUE,COLOR_WHITE);
    init_pair(3,COLOR_RED,COLOR_WHITE);
    curs_set(0);
    noecho();
    keypad(stdscr,TRUE);
}
```

A próxima função cria a barra de menu que aparece no topo do ecrã. Pode verificar a função *main* abaixo e verificar que a barra de menu que aparece como uma só linha no topo do ecrã é de facto definido como uma só linha de uma sub−janela do *stdscr*. A função abaixo recebe como parâmetro o ponteiro para essa janela, altera em primeiro lugar a cor de fundo e depois escreve o nome dos menus. Utilizamos a função *waddstr* para escrever os nomes dos menus, outra função poderia ter sido utilizada. Preste atenção às chamadas da função *wattron* que são usadas para escrever com um par de cor diferente (número 3) em vez de utilizar o par com a cor por omissão (número 2). Lembre−se que o par número 2 foi definido como sendo o par por omissão na primeira linha através da função *wbkgd*. A função *wattroff* é chamada quando queremos alterar o par com a cor por omissão.

```
void draw_menubar(WINDOW *menubar)
{
    wbkgd(menubar,COLOR_PAIR(2));
    waddstr(menubar,"Menu1");
    wattron(menubar,COLOR_PAIR(3));
    waddstr(menubar,"(F1)");
    wattroff(menubar,COLOR_PAIR(3));
    wmove(menubar,0,20);
    waddstr(menubar,"Menu2");
    wattron(menubar,COLOR_PAIR(3));
    waddstr(menubar,"(F2)");
    wattroff(menubar,COLOR_PAIR(3));
}
```

A próxima função desenha os menus quando as teclas F1 e F2 são premidas. Para criar o efeito de menu uma nova janela com a mesma cor branca que a barra de menu é criada por cima da janela azul que faz de fundo. Não queremos que esta nova janela sobreponha os caracteres escritos na parte de fundo. Devem permanecer lá até que o menu seja fechado. Esta é a razão pela qual a janela de menu não pode ser criada como uma sub−janela do *stdscr*. Como verá abaixo, os itens[0] da janela são criados com a função *newwin* e os outros 8 itens são criados como sub−janelas dos itens[0]. Aqui os itens[0] são utilizados para desenhar uma caixa à volta do menu e os outros itens de janelas são utilizados para mostrar o item seleccionado no menu, bem como para não sobrepor os caracteres à volta da caixa do menu. Para fazer com que um item de menu pareça seleccionado , basta tornar a sua cor de fundo diferente do resto dos itens. É o que é feito ma terceira linha a contar do topo; a cor de fundo do primeiro item é tornada diferente das outras, assim quando o menu de pop up aparece é o primeiro item que é seleccionado.

```
WINDOW **draw_menu(int start_col)
{
    int i;
    WINDOW **items;
    items=(WINDOW **)malloc(9*sizeof(WINDOW *));

    items[0]=newwin(10,19,1,start_col);
    wbkgd(items[0],COLOR_PAIR(2));
    box(items[0],ACS_VLINE,ACS_HLINE);
    items[1]=subwin(items[0],1,17,2,start_col+1);
    items[2]=subwin(items[0],1,17,3,start_col+1);
    items[3]=subwin(items[0],1,17,4,start_col+1);
    items[4]=subwin(items[0],1,17,5,start_col+1);
    items[5]=subwin(items[0],1,17,6,start_col+1);
    items[6]=subwin(items[0],1,17,7,start_col+1);
    items[7]=subwin(items[0],1,17,8,start_col+1);
    items[8]=subwin(items[0],1,17,9,start_col+1);
    for (i=1;i<9;i++)
        wprintw(items[i],"Item%d",i);
    wbkgd(items[1],COLOR_PAIR(1));
    wrefresh(items[0]);
    return items;
}
```

A próxima função apaga, simplesmente, a janela de menu criada pela função acima. Apaga primeiro os itens da janela com a função *delwin* e depois liberta a memória dos ponteiros do tipo item.

```
void delete_menu(WINDOW **items,int count)
{
    int i;
    for (i=0;i<count;i++)
        delwin(items[i]);
    free(items);
}
```

A função *scroll_menu* permite–nos fazer scroll entre e dentro dos menus. Lê as teclas primidas no teclado com a função *getch*. Se os cursores baixo, cima são premidos então os itens abaixo ou em cima são seleccionados. Isto é feito, como se recorda, tornando a cor de fundo do item seleccionado diferente do resto. Se os cursores direita, esquerda são premidos então o menu aberto é fechado e aberto outro. Se a tecla enter for pressionada, então o item seleccionado é retornado. Se a tecla ESC é premida, os menus são fechados sem que seja seleccionado algum item. A função ignora outras teclas de entrada. Nesta função o *getch* é capaz de ler as teclas de cursor a partir do teclado. Permita–me que o lembre que isto é possível visto que a primeira função *init_curses* chama *keypad(stdsrc, TRUE)* e os valor de retorno da função *getch* é mantido numa variável inteira, em vez de numa variável do tipo char, pois os seus valores são para além dos valores suportados por uma variável do tipo char.

```
int scroll_menu(WINDOW **items,int count,int menu_start_col)
{
    int key;
    int selected=0;
    while (1) {
        key=getch();
        if (key==KEY_DOWN || key==KEY_UP) {
            wbkgd(items[selected+1],COLOR_PAIR(2));
            wnoutrefresh(items[selected+1]);
            if (key==KEY_DOWN) {
                selected=(selected+1) % count;
            } else {
                selected=(selected+count-1) % count;
            }
            wbkgd(items[selected+1],COLOR_PAIR(1));
```

```
            wnoutrefresh(items[selected+1]);
            doupdate();
        } else if (key==KEY_LEFT || key==KEY_RIGHT) {
            delete_menu(items,count+1);
            touchwin(stdscr);
            refresh();
            items=draw_menu(20-menu_start_col);
            return scroll_menu(items,8,20-menu_start_col);
        } else if (key==ESCAPE) {
            return -1;
        } else if (key==ENTER) {
            return selected;
        }
    }
}
```

Por último há a função *main*. Utiliza todas as funções que descrevi em cima e que fazem com que o programa trabalhe devidamente. Lê também as teclas premidas com a função *getch* e se a tecla F1 ou a F2 é premida, desenha o janela de menu correspondente com a função *draw_menu*. Após isto chama a função *scroll_menu* e deixa o utilizador fazer uma selecção a partir dos menus. Após o retorno da função *scroll_menu* apaga as janelas de menu e imprime o item seleccionado na barra de mensagem.

Devia mencionar a função *touchwin*. Se a função *refresh* fosse chamada sem a *touchwin* após os menus serem fechados, o último menu aberto teria ficado no ecrã. Isto é assim porque as funções de menu não alteram a *stdsrc* de nenhum modo, quando a função *refresh* é chamada. Não escreve por cima de nenhum caracter da *stdsrc* pois assume que a janela não se alterou. A função *touchwin* define todas as flags na estrutura da WINDOW o que diz para refrescar todas as linhas da janela que se alteraram e assim, até ao próximo refrescamento toda a janela tem de ser rescrita mesmo que os conteúdos da janela não se tenham alterado. A informação escrita na *stdsrc* permanece lá após o fecho dos menus visto que estes não escrevem por cima do *stdsrc* mas por cima de novas janelas.

```
int main()
{
    int key;
    WINDOW *menubar,*messagebar;

    init_curses();

    bkgd(COLOR_PAIR(1));
    menubar=subwin(stdscr,1,80,0,0);
    messagebar=subwin(stdscr,1,79,23,1);
    draw_menubar(menubar);
    move(2,1);
    printw("Press F1 or F2 to open the menus. ");
    printw("ESC quits.");
    refresh();

    do {
        int selected_item;
        WINDOW **menu_items;
        key=getch();
        werase(messagebar);
        wrefresh(messagebar);
        if (key==KEY_F(1)) {
            menu_items=draw_menu(0);
            selected_item=scroll_menu(menu_items,8,0);
            delete_menu(menu_items,9);
            if (selected_item<0)
                wprintw(messagebar,"You haven't selected any item.");
            else
                wprintw(messagebar,
```

```
                  "You have selected menu item %d.",selected_item+1);
            touchwin(stdscr);
            refresh();
        } else if (key==KEY_F(2)) {
            menu_items=draw_menu(20);
            selected_item=scroll_menu(menu_items,8,20);
            delete_menu(menu_items,9);
            if (selected_item<0)
                wprintw(messagebar,"You haven't selected any item.");
            else
                wprintw(messagebar,
                  "You have selected menu item %d.",selected_item+1);
            touchwin(stdscr);
            refresh();
        }
    } while (key!=ESCAPE);

    delwin(menubar);
    delwin(messagebar);
    endwin();
    return 0;
}
```

Se copiar o código para um ficheiro chamado *example.c* e remover todas as minhas explicações, pode compilar o código com

```
gcc -Wall example.c -o example -lcurses
```

e teste o programa. Pode também fazer download do código abaixo na secção das referências.

Conclusão

Falei acerca das bases das ncurses que são suficientes para criar uma boa interface para o seu programa. Contudo, as capacidades da biblioteca não estão limitadas ao que aqui expliquei. Descobrirá imensas outras coisas nas páginas man, que tantas vezes lhe pedi para ler e, compreenderá que a informação apresentada aqui é só uma introdução.

Referências

- O programa de exemplo: example.c
- O website das ncurses:www.gnu.org/software/ncurses/

Webpages maintained by the LinuxFocus Editor team © Reha K. Gerçeker "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org	Translation information: tr −−> −− : Reha K. Gerçeker <gerceker/at/itu.edu.tr> tr −−> en: Reha K. Gerçeker <gerceker/at/itu.edu.tr> en −−> pt: Bruno Sousa <bruno/at/linuxfocus.org>

2005−01−10, generated by lfparser_pdf version 2.51