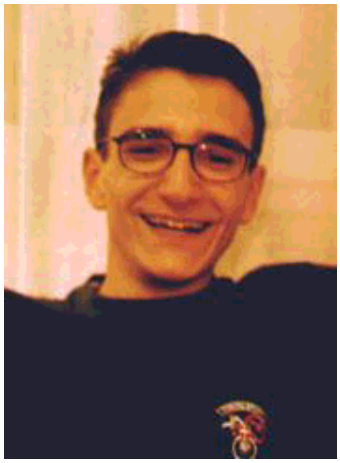
LinuxFocus article number 233
http://linuxfocus.org



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

Over de auteur:

Reha is een student computer engineering in Istanboel, Turkije. Hij houdt van de vrijheid van Linux als een omgeving voor software ontwikkeling. Veel van z'n tijd brengt door voor zijn computer, met het schrijven van programma's. Hij hoopt een slimme programmeur te worden.

Introductie tot Ncurses

Kort:

Ncuses is een bibliotheek die functie-toets mappings levert, functies voor scherm opmaak en de mogelijkheid om meerdere niet-overlappende windows te gebruiken op tekst-gebaseerde terminals.

Vertaald naar het Nederlands door:
Guus Snijders
<ghs(at)linuxfocus.org>

Wat is Ncurses?

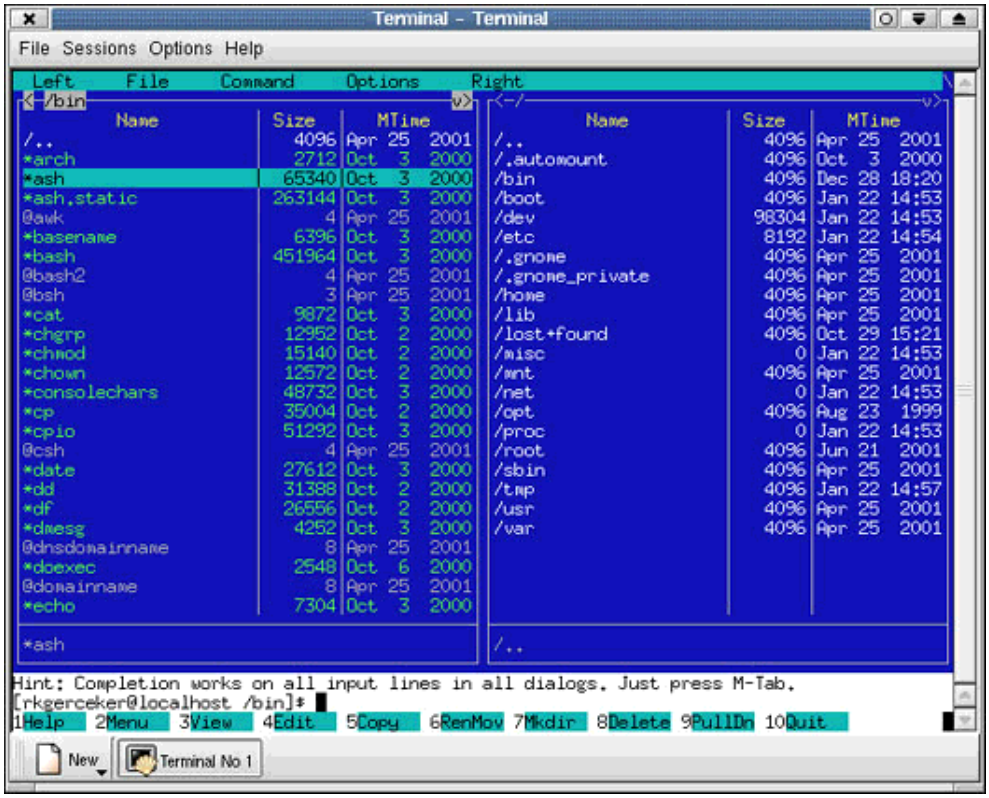
Wil je je programma's van een kleurrijke, terminal gebaseerde interface voorzien? Ncurses is een bibliotheek die window functies levert voor tekst-gebaseerde terminals. Waar Ncurses zoal voor

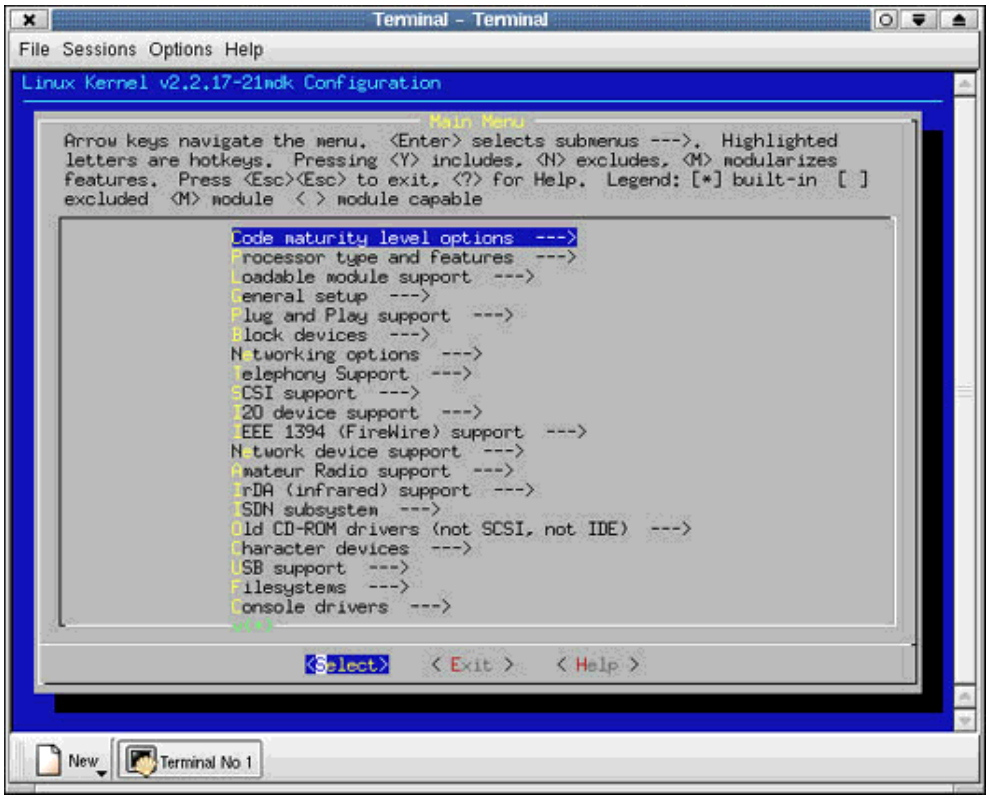
gebruikt kan worden:

- Gebruik het volledige scherm zoals je wilt.
- Creëer en beheer windows.
- Gebruik 8 verschillende kleuren.
- Geef je programma's muis ondersteuning.
- Gebruik de functie toetsen van het toetsenbord.

Ncurses kan worden ingezet op ieder ANSI/POSIX conform Unix systeem. Afgezien daarvan is de bibliotheek in staat om terminal eigenschappen te detecteren via de database van het systeem en hierop te reageren, en levert zodoende een terminal-onafhankelijke interface. Hierdoor kan ncurses gebruikt en vertrouwd worden voor ontwerpen die worden geacht te werken op verschillende platformen en verschillende terminals.

Midnight Commander is een van de voorbeelden van een programma dat is geschreven met ncurses. Ook de interface die gebruikt wordt voor de kernel configuratie op de console is geschreven met ncurses. Zie ook de schermafbeeldingen hieronder.





Waar te verkrijgen?

Ncurses wordt ontwikkeld onder GNU/Linux. Om de nieuwste release te downloaden, gedetailleerde informatie te vinden en andere gerelateerde links, bezoek www.gnu.org/software/ncurses/.

Basics

Om gebruik te maken van de library, moet je curses.h opnemen in je broncode en om er zeker van te zijn dat je code gelinkt wordt met de curses library, kun je gebruik maken van de parameter -lcurses aan gcc.

Om met ncurses te kunnen werken, is het nodig om bekend te zijn met de fundamentele data structuur. Dit is de WINDOW (venster) structuur en, zoals de naam al aangeeft, wordt het gebruikt om de windows te representeren die je creëerd. Bijna alle functies van de library gebruiken een WINDOW pointer als parameter.

De meest gebruikte componenten van ncurses zijn windows. Zels als je niet je eigen windows creëerd, wordt het scherm beschouwd als een eigen window. Als FILE descriptor stdout van de standaard I/O library gebruikt ncurses de WINDOW pointer, stdscr doet hetzelfde. In aanvulling op stdscr wordt een andere WINDOW pointer, curscr genaamd, gedefiniëerd in de library. Daar stdscr het scherm representeerd, curscr representeerd het huidige scherm, zoals bekend bij de library. Je kunt je afvragen "Wat is het verschil?" Blijf lezen.

Om ncurses functies en variabelen in je programma's te gebruiken, dien je de functie initscr aan te roepen. Deze functie wijst geheugen toe aan variabelen als stdscr, curscr en maakt de library klaar voor gebruik. In andere woorden, alle ncurses functies volgen op initscr. Zo wordt ook endwin aangeroepen als je klaar bent met ncurses. Dit geeft het geheugen dat gebruikt werd voor ncurses weer vrij. Na het aanroepen van endwin kun je geen ncurses functies gebruiken, tenzij je initscr weer gebruikt.

Wees er zeker van dat er tussen de aanroepen naar initscr en endwin geen output naar het scherm wordt gestuurd met de functies van de de standaard I/O library. Anders krijg je een ongebruikelijke en vaak corrupte output op je scherm. Zolang ncurses actief is, kun je het beste zijn functies gebruiken om output naar het scherm te sturen. Voor het aanroepen van initscr of na het aanroepen van endwin kun je doen wat je wilt.

Het scherm updaten: refresh

De WINDOW structuur houdt niet alleen hoogte, breedte en positie van het window bij, maar ook de inhoud ervan. Als je schrijft naar een window word de inhoud ervan gewijzigd, maar dit betekent niet dat het onmiddelijk op het scherm verschijnt. Om het scherm te updaten dient ofwel refresh (ververs) of wrefresh gebruikt te worden.

Hier ligt het verschil tussen stdscr en curscr. Terwijl curscr de inhoud van het huidige scherm behoud, kan stdscr over andere informatie beschikken na aanroepen van de ncurses output functies. Als je de laatste veranderingen aan stdscr wilt schrijven naar curscr, gebruik je refresh. In andere woorden, refresh is de enige functie die gebruikt wordt voor curscr. Het wordt aangeraden curscr met rust te laten en het aan de refresh functie over te laten om curscr te updaten.

refresh heeft een mechanisme om het scherm zo snel mogelijk te updaten. Zodra de functie wordt gebruikt, worden alleen de veranderde regels naar het scherm geschreven. Dit scheelt CPU-tijd daar het het programma ervan weerhoud dezelfde informatie nogmaals naar het scherm te sturen. Dit mechanisme is de reden waarom ncurses functies en standaard I/O functies slechte resultaten kunnen opleveren wanneer ze samen worden gebruikt; als ncurses functies worden gebruikt, zetten ze een flag welke refresh verteld dat de regel is veranderd, terwijl er niks van dit al gebeurt als je een standaard I/O functie gebruikt.

refresh en wrefresh doen in principe hetzelfde. wrefresh neemt een WINDOW pointer als parameter en ververst de inhoud van alleen dat window. refresh() is gelijk aan wrefresh(stdscr). Zoals we later zullen zien, beschikken, net als wrefresh, de meeste ncurses functies over macro's die deze functies toepassen voor stdscr.

Nieuwe Windows creëren

Laten we het nu hebben over subwin en newwin, functies om nieuwe windows te maken. Deze beide functies nemen de hoogte, breedte en coördinaten van de linker bovenhoek van nieuwe windows als parameters. In ruil leveren ze een WINDOW pointer die het nieuwe window representeerd. Je kunt deze

pointer gebruiken met wrefresh en andere functies die later ter sprake zullen komen.

"Als ze hetzelfde doen, waarom dan dubbele functies?" vraag je je mischien af. Je hebt gelijk, ze zijn licht verschillend. subwin creëert het nieuwe window als het subwindow van een andere. Een window die op deze manier gecreëerd wordt, erft de eigenschappen van de ’ouder’ window. Deze eigenschappen kunnen later gewijzigd worden zonder de ’ouder’ window te beïnvloeden.

Afgezien hiervan, is er een ding dat de ouder en de ’kind’ window bij elkaar houdt. De karakter array die de inhoud van een window bijhoudt, wordt gedeeld door ouder en ’kind’ window. In andere woorden, karakters op de intersectie van de twee windows, kunnen door beiden veranderd worden. Als de ouder naar zo’n vierkant schrijft, wordt de kinds’ inhoud ook gewijzigd. Andersom geldt dit ook.

In tegenstelling tot subwin, creëert newwin een volledig nieuw window. Zo’n window deelt, tenzij het z’n eigen subwindows heeft, zijn karakter array niet met een ander window. Het voordeel van subwin is dat het gebruik van een gedeelde karakter array minder geheugen gebruikt. Echter, wanneer windows elkaar beginnen te overschrijven, bied newwin zijn eigen voordelen.

Je kunt je eigen subwindows op iedere diepte aanbrengen. Ieder subwindow kan zijn eigen subwindows hebben, maar hou in gedachten dat dezelfde karakter array wordt gedeeld door meer dan twee windows.

Wanneer je klaar bent met een window die hebt aangemaakt, kun je deze verwijderen met de functie delwin. Ik stel voor om de man page te raadplegen voor de parameter lijsten van deze functies.

Schrijven naar Windows, lezen van Windows

We hebben reeds gesproken over stdscr, curscr, het verversen van het scherm en het aanmaken van nieuwe windows. Maar hoe schrijven we naar een window? Of hoe lezen we data van een window?

De functies die voor deze doeleinden worden gebruikt, doen sterk denken aan hun tegenhangers van de standaard I/O library. Onderdeel van deze functies zijn printw in plaats van printf, scanw in plaats van scanf, addch in plaats van putc of putchar, getch in plaats van getchar. Ze worden op dezelfde manier gebruikt, alleen de namen verschillen. Zo kan addstr gebruikt worden om een string te lezen van een window. Al deze functies met een letter ’w’ aan het begin van naam toegevoegd en een WINDOW pointer als eerste parameter, doen hun werk op een ander window dan stdscr. Bijvoorbeeld, printw(...) en wprintw(stdscr, ...) zijn gelijk, net als refresh() en wrefresh(stdscr).

Het zou een lang verhaal worden als we op de details van al deze functies zouden ingaan. De man pages zijn de beste bronnen om de beschrijvingen, prototypes en return waardes te leren. Ik stel voor om de man page te checken voor iedere functie die je gebruikt. Ze bieden gedetailleerde en waardevolle informatie. De laatste sectie van dit artikel, waar we een voorbeeld programma zullen bespreken, kan ook dienen als een tutorial over het gebruik van de functies.

Fysieke en Logische Cursors

Het is nodig om fysieke en logische cursors te bespreken na een bespreking over het schrijven naar en het lezen van windows. Wat er bedoeld wordt meteen fysieke cursor is de meestal knipperende cursor op het scherm en er is slechts een fysieke cursor. Aan de andere kant, de logische cursors horen bij ncurses windows en ieder window heeft er een. Daardoor kunnen er meerdere logische cursors zijn.

De logische cursor is aan het begin van het vierkant van de window waar het lees- of schrijfproces begint. Daardoor betekend de mogelijkheid om een logische cursor rond te bewegen dat je naar ieder stuk van het scherm of window op ieder moment kan schrijven. Dit is een voordeel van ncurses over de standaard I/O library.

De functie die de logische cursor verplaatst is ofwel move, of, zoals je misschien al hebt geraden, wmove. move is een macro van wmove, geschreven voor de stdscr.

Een andere probleem is de co&oml;rdinatie van fysieke en logische cursors. De positie waar de fysieke cursor zal eindigen na een schrijfproces wordt bepaald door de _leave flag, welke bestaat in de window structure. Als _leave is gezet, wordt de logische cursor verplaatst naar de positie van de fysieke cursor (waar het laatste karakter wordt geschreven) als het schrijven is voltooid. Als _leave niet is gezet, wordt de fysieke cursor weer naar de positie van de logische cursor verplaatst (waar het eerste karakter is geschreven) als het schrijven is voltooid. De _leave flag wordt gecontroleerd door de leaveok functie.

De functie die de fysieke cursor verplaatst is mvcur. In tegenstelling tot anderen, heeft mvcur onmiddelijk effect, in plaats van bij de volgende refresh. Als je de fysieke cursor onzichtbaar wil maken, gebruik dan de functie curs_set. Check de man pages voor details.

Er bestaan ook macros die de verplaats en schrijf functies, zoals hierboven beschreven, samenvatten in een call. Deze worden goed uitgelegd in de man pages over addch, addstr, printw, getch, getstr, scanw, etc.

Windows opschonen

het schrijven naar windows is klaar. Maar hoe kunnen we windows, regels of karakters wissen?

Opschonen (clearing) in ncurses, betekend het vierkant, de regel of de inhoud van het window met witruimtes vlakken te vullen. Functies die hieronder beschreven worden vullen de benodigde vlakken met witruimtes en schonen zo het scherm op.

Laten we het eerst hebben over functies voor het opschonen van een karakter of een regel. De functies delch en wdelch verwijderen het karakter onder de logische cursor van het window en verschuiven de volgende karakters op de zelfde regel naar links. deleteln en wdeleteln verwijderen de regel van de logische cursor en schuiven alle volgende regels omhoog.

De functies clrtoeol en wcrltoeol verwijderen alle karakters op dezelfde regel, rechts van de logische cursor. clrtobot en wclrtobo roepen eerst wcrltoeol om alle karakters rechts van de logische cursor te verwijderen en daarna alle volgende regels.

Behalve deze, zijn er ook functies die het hele scherm of window opschonen. Er bestaan twee methoden om een heel scherm op te schonen. De eerste is door alle vlakken te vullen met witte spaties en refresh aan te roepen en de andere is om de ingebouwde terminal control code te gebruiken. De eerste methode is langzamer dan de tweede, daar het vereist dat alle vlakken op het scherm herschreven worden, waar de tweede het hele scherm onmiddelijk opschoond.

erase en werase vullen de karakter array van een window met witruimtes. Bij de volgende refresh zal het window opgeschoond worden. Echter, als het window om opgeschoond te worden het hele scherm vult, is het niet slim om deze functies te gebruiken. Ze maken gebruik van de eerste methode die hierboven beschreven is. Als het window de hele breedte van het scherm vult, is het voordelig om de onderstaande functies te gebruiken.

Alvorens verdet te gaan met andere functies, is het eerst tijd om de _clear flag te noemen. Het bestaat in de WINDOW structuur en, indien gezet, vraagt deze refresh om de control code van de terminal zodra hij wordt aangeroepen. Wanneer gebruikt, checkt refresh of het window scherm-breed is (met de _FULLWIN flag) en indien zo, schoont hij het scherm op met de ingebouwde terminal methode. Het schrijft alleen karakters anders dan witruimtes naar het scherm. Dit maakt het opschonen van het scherm sneller. De reden dat de terminal methode alleen voor scherm vullende windows wordt gebruikt, is dat de terminal control code het hele scherm opschoond, en niet alleen het window. De _clear flag wordt gebruikt met de functie clearok.

De functies clear en wclear worden gebruikt om windows op te schonen die schermbreed zijn. In feite staan deze functies gelijk aan het gebruiken van werase en clearok. Eerst vullen ze de window’s karakter array met witruimtes. Dan, door de _clear flag te zetten, schonen ze het scherm op met behulp van de ingebouwde terminal methode als het window schermbreed is, anders verversen ze alle vlakken van de window door ze te vullen met witruimtes.

Als resultaat, als je weet dat het window om op te schonen, full screen is, gebruik je clear of wclear. Het produceert sneller het resultaat. Echter, er is geen verschil om wclear of werase te gebruiken als het window niet full screen is.

Kleuren gebruiken

De kleuren die je op het scherm ziet, kun je beschouwen als kleur paren. Dat is om dat ieder vierkant een achtergrond en een voorgrond kleur heeft. Om in kleur te schrijven met ncurses moet je je eigen kleuren paren creëren en deze paren gebruiken om naar een window te schrijven.

Net zoals initscr moet worden aangeroepen om ncurses te starten, moet start_color aangeroepen worden om kleuren te initiëren. De functie om je kleur paren te creëren is init_pair. Als je een kleuren paar maakt met init_pair, wordt dit paar geaccosiëerd met het nummer dat je de functie als eerste parameter meegaf. Dan, als je dit paar wilt gebruiken, refereer je er aan door COLOR_PAIR aan te roepen met dat geaccosiëerde nummer.

Afgezien van het creëren van kleuren paren heb je ook de benodigde functies voor een ander kleuren paar. Dit wordt gedaan door de functies attron en wattron. Deze functies, totdat attraff of wattroff worden gebruikt, zorgen ervoor dat alles wordt geschreven in de corresponderende windows in het

kleuren paar dat je hebt geselecteerd.

Ook bestaan de fucnties bkgd en wbkgd die het kleuren paar geaccosiëerd met het het hele window veranderen. Wanneer gebruikt, veranderen ze de achtergrond en voorgrond kleuren van alle vlakken van de window. Dat betekent dat, tijdens de volgende refresh, ieder vlak van het window wordt herschreven met een nieuw kleuren paar.

Zie de man pages voor de beschikbare kleuren en details van de hier genoemde functies.

Boxen rond je Windows

Je kunt boxen rond je windows creëren om je programma er goed uit te laten zien. Er is een macro in de library, genaamd box, die dit voor je doet. In tegenstelling tot andere, bestaat er geen wbox; box neemt een WINDOW pointer als argument.

De eenvoudige details van box zijn te vinden in de man pages. Er is nog iets anders dat genoemd zou moeten worden. Een window in een box plaatsen betekend simpelweg de nodige karakterste plaatsen in de karakter array van het window dat correspondeerd met de grens vlakken. Als je die vlakken later om de een of andere reden wilt gebruiken, raakt de box corrupt. Om dit te voorkomen, creëer je een sub window binnen het originele window met subwin, plaats je het originele window in een box, en gebruik je het binnenste window om naar te schrijven, indien nodig.

Functie Toetsen

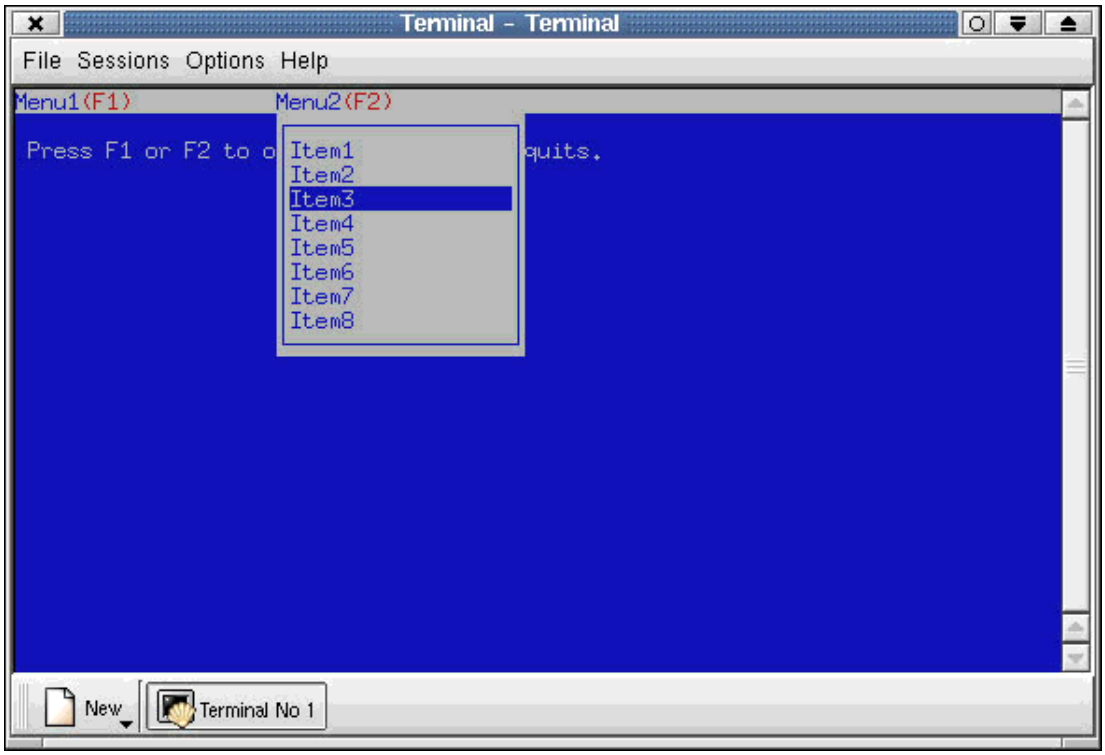
Om in staat te zijn de functie toetsen te gebruiken, moet de _use_keypad flag gezet zijn in het window waar je je input van krijgt. keypad is de functie die de waarde van _use_keypad zet. Als je _use_keypad zet, krijg je input van het toetsenbord zoals gewoonlijk met de input functies.

In dit geval, als je getch gebruikt om data te verkrijgen bijvoorbeeld, zou je voorzichtig moeten zijn en de data in een int variabele te plaatsen, liever dan een char variabele. Dit is omdat de nummerieke waarden van functie toetsen groter zijn dan een char variabele kan opslaan. Je hoeft deze nummerieke waarden niet te weten maar in plaats daarvan kun je de namen ervan in de library gebruiken. Deze namen worden opgesomd in de man page van getch.

Een Voorbeeld

Nu gaan we een aardig en eenvoudig programma analyseren. In dit programma worden menus gecreërd met ncurses en de selectie van een optie van een menu wordt gedemonstreerd. Een interessant aspect van dit programma is het gebruik van ncurses windows om een menu effect te verkrijgen. Hieronder kun je afbeelding zien:

see a snapshot below:



Het programma begint met de ’included header’ bestanden, als altijd. Daarna definiëren we de constanten die de ASCII waardes zijn van de enter en escape toetsen.

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

#define ENTER 10
#define ESCAPE 27
```

De functie hieronder wordt als eerste aangeroepen als het programma draait. Het roept eerst initscr aan om curses te initialiseren en daarna start_color om het gebruik van kleuren mogelijk te maken. Kleurparen die door het hele programma worden gebruikt, worden later gedefini&eml;erd. De aanroep curs_set(0) maakt de fysieke cursor onzichtbaar. noecho stopt de weergave van de invoer van het toetsenbord. Je kunt de noecho functie ook gebruiken om de invoer van het toetsenbord onder controle te houden en alleen de delen die je wilt gebruiken weer te geven. De echo functie zou gebruikt moeten worden als het effect van noecho ongedaan moet worden gemaakt. De functie hieronder roept tenslotte keypad aan om de functie toetsen bruikbaar te maken, zodra er invoer van stdscr komt. Dit nodig omdat we F1, F2 en cursor toetsen later in het programma gebruiken.

```
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);
}
```

De volgende functie creëert de menubalk die bovenin het scherm verschijnd. Je kunt de main functie hieronder controleren, en zien dat de menubalk die als een enkele lijn bovenin het scherm verschijnd in feite is gefiniëerd als een een-regel subwindow van stdscr. De functie eronder neemt de pointer naar dat window als parameter, veranderd eerst de achtergrond kleur en schrijft dan de menu namen. we gebruiken waddstr om de menu namen te schrijven, er had ook een andere functie gebruikt kunnen worden. Let op de wattron calls die gebruikt worden om met een ander kleurenpaar (nummer 3) te schrijven, in plaats van het standaard kleurenpaar (nummer 2). Onthoud dat paar nummer 2 als standaard was aangegeven op de eerste regel door wbkgd. wattroff wordt gebruikt als we willen switchen naar het standaard kleuren paar.

```
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));
}
```

De volgende functie tekent de menus waneer F1 of F2 wordt ingedrukt. Om het menu effect te krijgen wordt een nieuwe window met dezelfde witte kleur als de menubalk gecreëerd over het blauwe window dat de achtergrond opmaakt. We willen niet dat dit nieuwe window eerder geschreven karakters op de achtergrond overschrijft. Ze zouden daar moeten blijven nadat het menu wordt gesloten. Dit is waarom het menu window niet kan worden gecreëerd als een subwindow van stdscr. Zoals je hieronder kunt zien, de window items[0] is gecreëerd met de functie newwin en de andere 8 items windows worden gecreëerd als als een subwindow van items[0]. Hier wordt items[0] gebruikt om een box om het menu te tekenen en de andere items windows worden gebruikt om een geselecteerd item in het menu weer te geven en ook om de karakters van de box rond het menu niet te overschrijven. Om een item geselecteerd weer te geven, is het voldoende om de achtergrond kleur te veranderen ten opzichte van de rest van de items. Dit wordt gedaan door de derde regel van onderen; de achtergrond kleur van het eerste item wordt anders gemaakt dan de anderen en dus, als het menu te voorschijn komt, wordt het eerste item geselecteerd.

```
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;
}
```

De volgende functie verwijderd simpel het menu window dat werd gecreëerd door de functie hierboven. Eerst verwijderd het de items windows met delwin en dan schoont het het geheugen op dat werd gebruikt voor de items pointer.

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

De scoll_menu functie laat ons scrollen tussen en in menus. Het leest de toetsen die worden ingedrukt op het toetsenbord met getch. Als de omhoog of omlaag cursor toetsen worden ingedrukt, zal het item boven of onder worden geselecteerd. Dit wordt, zoals je je zult herinneren, gedaan de achtergrond kleur van het geselecteerde item anders te maken dan de anderen. Als de linker of rechter cursor toets wordt ingedrukt, wordt het geopende menu gesloten en de andere geopend. Als de enter toets wordt ingedrukt, wordt het geselecteerde item geretourneerd. Als ESC wordt ingedrukt, worden de menus gesloten zonder een item te selecteren. De functie negeert andere invoer toetsen. In deze functie is getch in staat om de cursor toetsen van het toetsenbord te lezen. Laat me je helpen herinneren dat dit mogelijk is daar de eerste functie init_curses keypad(stdscr,TRUE) gebruikte en de retour waarde van getch wordt opgeslagen in een int variabele, liever dan een char variabele, daar de waardes van de functie toetsen te groot zijn voor een char variabele.

```
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;
        }
    }
}
```

Tenslotte is er nog de main functie. Deze gebruikt alle functies die ik heb geschreven en hierboven heb besproken, om het programma juist te laten werken. Ook leest het de ingedrukte toetsen met getch, en als F1 of F2 word ingedrukt, tekent het het corresponderende menu met draw_window. Daarna roept het scroll_menu aan en laat de gebruiker een selectie maken uit de menus. Nadat scroll_menu terugkeert, verwijderd het de menu windows en drukt het het geselecteerde item af op de bericht balk.

Ik zou de functie touchwin nog moeten noemen. Als refresh direct zou worden aangeroepen zonder touchwin nadat de menus werden gesloten, zou het laatste geopende menu op scherm blijven. Dit is omdat de menu functies stdscr in het geheel niet wijzigen en als refresh wordt aangeroepen herschrijft het geen enkel karakter van stdscr daar deze aanneemt dat de window niet veranderd is. touchwin zet alle flags in de WINDOW structuur om refresh te vertellen dat alle regels van het window gewijzigd zijn en dus zal bij de volgende refresh het hele window herschreven worden, zelfs als de inhoud van het window niet veranderd is. De informatie die geschreven wordt op stdscr blijft daar nadat de menus sluiten daar de menus niet over stdscr schrijven maar in plaats daarvan worden gecreëerd als nieuwe windows.

```
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;
}
```

Als de code kopieëert naar een bestand met de naam example.c en mijn beschrijvingen verwijderd, kun je de code compileren met

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

en het programma testen. Je kunt ook de code downloaden in het hoofdstuk referenties.

Conclusie

We hebben gesproken over de basics van ncurses, voldoende om een goede interface te ontwikkelen voor je programma. Echter, de library is niet beperkt tot wat hier besproken is. Je zult vele andere dingen ontdekken in de man pages, die ik je vaak heb gevraagd te lezen, en je zult de begrijpen dat de informatie die hier wordt gepresenteerd alleen een introductie is.

Referenties

- Het voorbeeld programma: example.c
- The ncurses website: www.gnu.org/software/ncurses/
-

Site onderhouden door het LinuxFocus editors team © Reha K. Gerçeker "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org	Vertaling info: tr --> -- : Reha K. Gerçeker <gerceker/at/itu.edu.tr> tr --> en: Reha K. Gerçeker <gerceker/at/itu.edu.tr> en --> nl: Guus Snijders <ghs(at)linuxfocus.org>

2005-01-14, generated by lfparser_pdf version 2.51