Ncurses'a Giriş

Yazan:
Reha K. Gerçeker
<gerceker (at) itu.edu.tr>

28 Ocak 2003

Özet

ncurses metin tabanlı uçbirimler için pencereler oluşturma ve kullanmaya, ekranı farklı renklerde boyamaya ve işlev tuşlarını kullanmaya imkan veren bir kütüphanedir.

Bu belgenin özgün sürümünü http://www.linuxfocus.org/Turkce/March2002/article233.shtml adresinde bulabilirsiniz.

Konu Başlıkları

1. Ncurses Nedir? . . . 3
1.1. Nereden Bulabilirim? . . . 4
2. Temel Bilgiler . . . 4
3. Ekran Görüntüsünün Tazelenmesi . . . 5
4. Yeni Pencerelerin Oluşturulması . . . 5
5. Pencerelere Yazma ve Pencerelerden Okuma . . . 6
6. Fiziksel ve Mantıksal İmleç . . . 6
7. Pencerelerin Temizlenmesi . . . 6
8. Renkler . . . 7
9. Pencerelerin Kutulanması . . . 8
10. İşlev Tuşları . . . 8
11. Örnek Yazılım . . . 8
12. Sonuç . . . 13
A. example.c . . . 14

Sürüm Bilgileri

v1.0

Yasal Açıklamalar

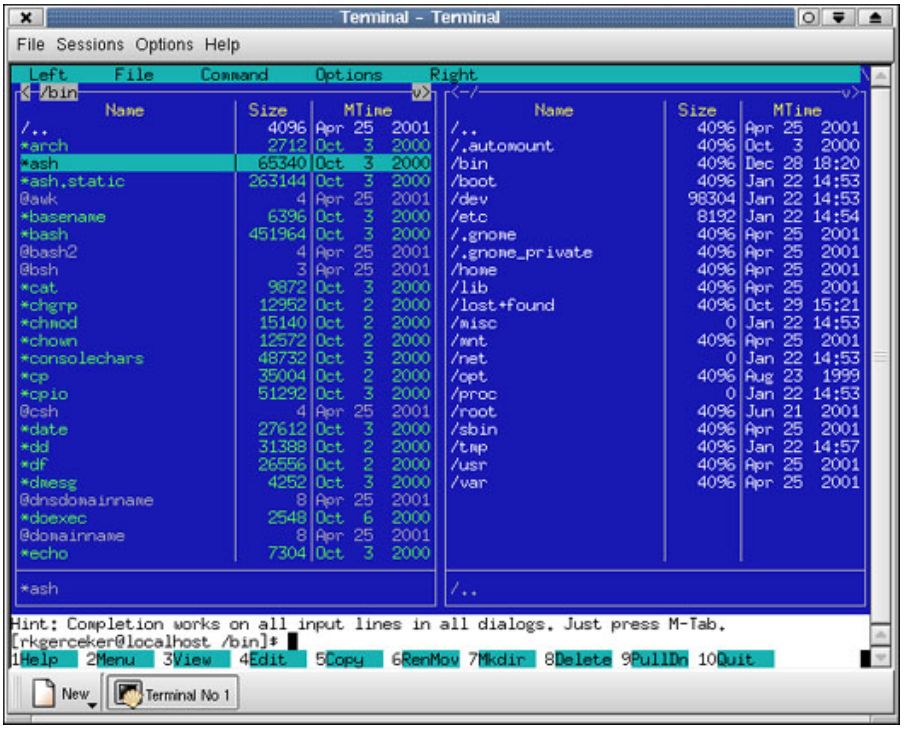
1. Ncurses Nedir?

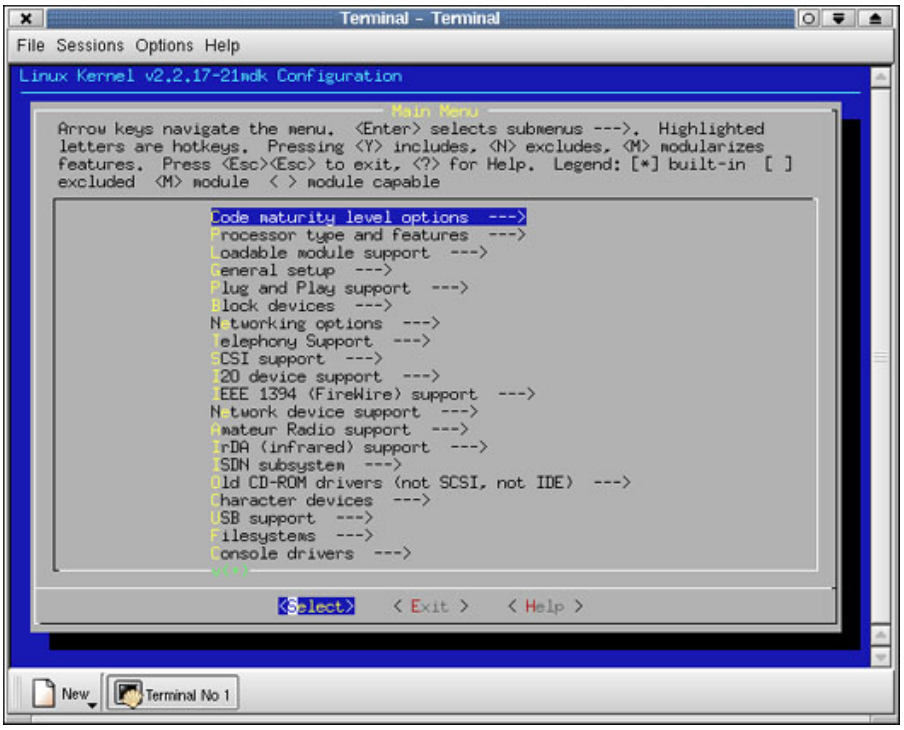
Yazdığınız bir yazılımın renkli ve pencere temelli bir arayüze sahip olmasını mı istiyorsunuz? **ncurses** metin tabanlı uçbirimler için pencereler oluşturma ve kullanmaya, ekranı farklı renklerde boyamaya ve işlev tuşlarını kullanmaya imkan veren bir kütüphanedir. **ncurses** ile yapabilecekleriniz:

- Ekranın her yerini isteğinize göre kullanmak.
- Pencereler oluşturmak ve onları yönetmek.
- 8 farklı renk kullanmak.
- Yazılımınıza fare desteği vermek.
- Klavyedeki işlev tuşlarını kullanmak.

ncurses kütüphanesini ANSI/POSIX uyumlu UNIX sistemlerde kullanmak mümkündür. Bunun yanında, hangisi kullanılıyor olursa olsun uçbirim özelliklerini sistemden öğrenerek uçbirimden bağımsız bir arayüz sunmaktadır. Bu açılardan yazılımcının farklı platformlar ve uçbirimler için bile güvenerek kullanabileceği bir kütüphanedir.

Midnight Commander, **ncurses** kullanılarak yazılmış yazılımlara verilecek örneklerden bir tanesidir. Ayrıca konsolda çekirdek yapılandırması için kullanılan arayüz **ncurses** ile hazırlanmıştır. Bu iki yazılıma ilişkin ekran görüntülerini aşağıda görüyorsunuz.





1.1. Nereden Bulabilirim?

ncurses GNU/Linux altında geliştirilmektedir. `http://www.gnu.org/software/ncurses/` adresinden güncel sürümünü, ayrıntılı bilgileri ve diğer gerekli adresleri öğrenebilirsiniz.

2. Temel Bilgiler

ncurses kütüphanesini kullanmak için `curses.h` başlık dosyasını kodunuza eklemeli ve kodunuzu **gcc** ile derlerken `-lcurses` seçeneğini kullanmayı unutmamalısınız.

ncurses konusuna girerken kütüphane için temel bir veri yapısından bahsetmek gerekir. `WINDOW *` türünde olan bu yapı, adından da anlaşıldığı gibi, **ncurses** ile oluşturacağınız pencereleri temsil etmek için kullanılır. Hemen hemen tüm **ncurses** işlevleri `WINDOW *` türünden bir akımı parametre olarak alırlar.

Pencereler, **ncurses**'da en çok kullanılan bileşenlerdir. Siz hiçbir pencere oluşturmasanız bile ekran tek başına bir pencere olarak kabul edilir. Standart giriş/çıkış kütüphanesinde (yönlendirmeler olmadığı sürece) çıktı ekranını `FILE *` türünde `stdout` adlı akımın temsil etmesi gibi **ncurses**'da da ekran `WINDOW *` türünde `stdscr` adlı bir akım ile temsil edilir. `stdscr` akımına ek olarak, kütüphane tarafından `curscr` adlı bir başka `WINDOW *` türünde akım daha tanımlanır. `stdscr` öntanımlı çıktı ekranı temsil ettiği halde, `curscr` o anki çıktı ekran görüntüsünü temsil eder. "*İkisinin arasında ne fark var ki?*" diye düşünüyor olabilirsiniz, okumaya devam edin.

Yazılımınızda **ncurses** işlevlerini ve değişkenlerini kullanabilmek için herşeyden önce **initscr** işlevini çağırmanız gerekir. Bu işlev `stdscr` ve `curscr` akımlarını ilklendirir (bellek ayırır) ve kütüphaneyi hazır hale getirir. Dolayısıyla her **ncurses** işlevi **initscr** işlevini takip etmek zorundadır. Benzer şekilde **ncurses** ile işinizi bitirdiğinizde **endwin** işlevini kullanarak kütüphanenin oluşturduğu akımların yokedilmesini sağlayabilirsiniz. Bunu yaptıktan sonra yeniden **initscr** çağrısı yapmadığınız sürece **ncurses** işlevlerini kullanamazsınız.

Yazılımınızda **initscr** ve **endwin** çağrıları arasında standart C kütüphanesinin giriş/çıkış işlevlerini kullanarak ekrana çıktı göndermemelisiniz. Bu durumda beklediğiniz görüntüyü elde edemeyebilirsiniz. **ncurses**

kütüphanesi etkin olduğu sürece ekran çıktısı için sadece onun işlevlerini kullanmaya dikkat edin. **initscr** çağırmadan önce ya da **endwin** çağırdıktan sonra istediğinizi yapabilirsiniz.

3. Ekran Görüntüsünün Tazelenmesi

`WINDOW *` yapısı içinde pencerenin eni, boyu, konumu gibi bilgiler tutulduğu gibi içeriği de saklanır. **ncurses** işlevleri ile bir pencereye yazı yazıldığı zaman bu o yazının hemen ekranda görünmesi anlamına gelmez. Yazılan yazı öncelikle ilgili pencerenin içeriğinde güncellenir. Ekran görüntüsünün güncellenmesi için **refresh** ya da **wrefresh** işlevlerinın çağrılması gerekir.

`stdscr` ile `curscr` arasında yukarıda kafanıza takılmış olabilecek fark buradadır. `curscr` ekranın o anki görüntüsünü içerirken, `stdscr`'nin içeriği yapılmış yeni işlev çağrıları sonucu farklı olabilir. `stdscr` ile `curscr`'nin örtüşmesi için **refresh** işlevinin çağrılması gerekir. Diğer bir deyişle, `curscr`'nin içeriğini `stdscr`'nin içeriği ile değiştiren tek işlev **refresh** işlevidir. Sizin `curscr`'yi kurcalamamanız ve onunla ilgili işleri **refresh** işlevine havale etmeniz tavsiye edilir.

refresh işlevi ekranı güncelleme işini mümkün olduğunca hızlı yapabilmek için değişik bir mekanizmaya sahiptir. İşlev çağrıldığında görüntüsü güncellenecek olan pencerenin yalnızca değişen satırlarını günceller. Böylece değişmeyen satırları yeniden ekrana yazmak için işlemci zamanı tüketmez. **ncurses** ile standart giriş/çıkış kütüphanesi işlevlerinın birlikte kullanılmasının istenmeyen sonuçlar doğurmasının sebebi bu mekanizmadır; **ncurses** işlevleri bir pencereye yazdığı zaman o satırın değiştiğini belirten bayrağı doğrularlar, standart giriş/çıkış işlevleri ise bu işi yapmaz.

refresh ile **wrefresh** işlevleri temelde aynı işi yaparlar. **wrefresh** işlevi parametre olarak `WINDOW *` türünde bir akım alır ve bu pencerenin görüntüsünü günceller. `refresh()` ise `wrefresh(stdscr)` çağrısı yapmaya denktir. Daha sonra da bahsedeceğim gibi birçok **ncurses** işlevinin (**wrefresh** gibi) `stdscr` için yazılmış makroları vardır.

4. Yeni Pencerelerin Oluşturulması

Şimdi kendi pencerelerinizi oluşturmanızı sağlayan **subwin** ve **newwin** işlevlerindan bahsedelim. Bu iki işlev de parametre olarak oluşturmak istediğiniz pencerenin enini, boyunu, sol üst köşesinin koordinatlarını alırlar. Dönüş değerleri ise yeni pencereyi temsil eden bir `WINDOW *` türünde bir pencere akımıdır. Bu yeni akımı yukarıda bahsettiğim **wrefresh** ve daha sonra bahsedeceğim diğer işlevlerle birlikte kullanabilirsiniz.

"Madem aynı işi yapıyorlar, neden iki tane işlev var?" diye düşünebilirsiniz. Haklısınız, tam olarak aynı işi yapmıyorlar. **subwin** işlevi yeni oluşturulan pencereyi bir başka pencerenin alt penceresi olarak oluşturur. Bu şekilde oluşturulan pencere ana pencerenin o anki tüm özelliklerini miras alır. Bu özellikler daha sonra ana pencereden bağımsız olarak değiştirilebilir. Bu durum ana pencere ile alt pencereyi birbirine bağlayan bir durum değildir.

Bunun dışında ana ve alt pencereyi birbirine bağlayan bir özellik vardır: Pencere içeriğinin tutulduğu karakter dizisi ana ve alt pencereler için ortaktır. Diğer bir deyişle, ana pencere ile alt pencerenin kesiştiği bir noktadaki karakter her iki pencere tarafından da değiştirilebilir. Böyle bir kareye alt pencere yazarsa ana pencerenin, ana pencere yazarsa alt pencerenin o noktadaki karakterinin üzerine yazılır.

newwin ise tamamiyle yeni bir pencere oluşturur. Böyle bir pencere, kendi alt pencereleri olmadığı sürece karakter dizisini başka pencerelerle paylaşmaz. **subwin** işlevini kullanmanın yararı, ortak karakter dizisi kullanımı sebebiyle daha az bellek tüketilmesidir. Ancak pencerelerin birbirlerinin bilgilerinin üzerine yazmasının istenmediği durumlarda **newwin** kullanmak doğru harekettir.

Alt pencerelerinizi koşullar elverdiği sürece istediğiniz derinlikte oluşturabilirsiniz. Her alt pencerenin de kendi alt pencereleri olabilir, fakat bu durumda aynı karakter dizisini ikiden de fazla pencerenin paylaştığını unutmayın.

Oluşturduğunuz pencerelerle işiniz bittiğinde **delwin** işleviyle onları yokedebilirsiniz. İşlevlerin parametrelerini öğrenmek için man sayfalarına bakmanızı tavsiye ederim.

5. Pencerelere Yazma ve Pencerelerden Okuma

Ncurses standart akımları olan `stdscr` ve `curscr`'den, ekran görüntüsünü tazelemekten ve yeni pencereler oluşturmaktan bahsettik. Peki pencerelerin içeriğini nasıl değiştireceğiz? Pencerelere nasıl yazacağız ve pencerelerden nasıl bilgi okuyacağız?

Bunlar için standart giriş/çıkış kütüphanesinin işlevlerine çok benzeyen işlevler kullanılmaktadır. **printf** yerine **printw**, **scanf** yerine **scanw**, **putc** ya da **putchar** yerine **addch**, **getc** ya da **getchar** yerine **getch** kullanılır. Kullanımları alışıldığı gibidir, kullanım olarak sadece isimleri değişiktir. Benzer şekilde pencereye dizge yazmak için **addstr**, pencereden bir dizge okumak için **getstr** kullanılabilir. Tüm bu işlev başlarına bir '**w**' harfi eklenerek ve birinci parametre olarak `WINDOW *` türünde bir pencere akımı alarak akımı belirtilen bir başka pencereye de yazabilirler. Örneğin, `printw(...)` ile `wprintw(stdscr, ...)` aynı işi yapan iki çağrıdır, aynı `refresh()` ve `wrefresh(stdscr)` durumunda olduğu gibi.

Bu işlevlere çok ayrıntılı olarak değinmek fazla yer kaplayacaktır. Yaptıkları işleri, prototiplerini, dönüş değerlerini ve uyarıları öğrenmek için en iyi kaynak her zaman olduğu gibi man sayfalarıdır. Sözünü ettiğim her işlev için kullanmadan önce man sayfasına bir bakmanızı tavsiye ederim. Çok daha ayrıntılı bilgiler edinebilirsiniz. Ayrıca yazının son bölümünü öğretici bir örneğe ayırdım. Şu ana kadar anlattığım ve anlatacağım hemen hemen tüm işlev ve kavramları bu örnek yazılımda takip etme imkanı bulacaksınız.

6. Fiziksel ve Mantıksal İmleç

ncurses ile pencerelere yazmak ve pencerelerden okumak sözkonusu olduğu zaman ***mantıksal ve fiziksel imleç*** kavramlarına da değinmek gerekir. Fiziksel imleçten kasıt yanıp sönerek sürekli ekranda görünen bildiğimiz imleçtir ve bir tanedir. Mantıksal imleçler ise **ncurses** pencerelerine ait olan ve her pencerenin mutlaka sahip olduğu başka imleçlerdir. Birden çok pencere olabileceğine göre mantıksal imleç de birden fazla olabilir.

Mantıksal imlecin görevi pencereye bir yazı yazılacağı zaman yazma işleminin başlayacağı ya da pencereden bir bilgi okunacağı zaman bu bilginin okunacağı kareyi göstermektir. Mantıksal imleci isteğinize göre hareket ettirebilmeniz demek ekranın ya da oluşturduğunuz pencerenin istediğiniz karesine istediğiniz zaman yazı yazabilmeniz anlamına gelir. Bu, standart giriş/çıkış kütüphanesinin sağlamadığı bir avantajdır.

İmleçlerin hareket ettirilmesi işini yapan işlev **move** ya da hemen tahmin edeceğiniz üzere **wmove** işlevleridir. **move** işlevi **wmove** işlevinin `stdscr` için yazılmış bir makrosudur.

Bir de mantıksal imleç ile fiziksel imlecin koordinasyonunun sağlanması durumu sözkonusudur. Herhangi bir yazma işleminin ardından fiziksel imlecin bulunacağı konum, her pencerenin sahip olduğu `_leave` bayrağının değerine bakılarak kararlaştırılır. Eğer `_leave` doğruysa, mantıksal imleç yazma işleminden sonra fiziksel imlecin üzerine (son harfin yazıldığı kareye) getirilir. Eğer `_leave` yanlışsa, fiziksel imleç mantıksal imlecin bulunduğu kareye (ilk harfin yazıldığı kareye) geri getirilir. `_leave` bayrağını yöneten işlev **leaveok** işlevidir.

Fiziksel imlecin hareket ettirilmesini sağlayan işlev **mvcur** işlevidir. Diğer işlevlerden farklı olarak **mvcur** işlevi bir sonraki **refresh**'te değil, hemen etkin olur. Fiziksel imlecin ekranda görünmesini istemiyorsanız, **curs_set** işlevini ilgili parametreleri ile kullanabilirsiniz. Ayrıntılar için man sayfalarına bakınız.

Yukarıda bahsettiğim imleci hareket ettirme ve yazı yazma işlemlerini tek bir çağrı ile de yapabilirsiniz. Bu, işlev çağrılarını birleştiren makroların varlığı sayesinde mümkündür. Bu çağrılar da **addch, addstr, printw, getch, getstr, scanw** işlevlerinin man sayfalarında ayrıntılı olarak açıklanmıştır.

7. Pencerelerin Temizlenmesi

Pencelere yazmak tamam. Peki pencereleri nasıl temizleyeceğiz, istediğimiz satırları ve karakterleri nasıl sileceğiz?

ncurses'da silme, silinecek karenin, satırın ya da pencerenin içeriğinin boşluklarla doldurulması demektir. Aşağıda değindiğim silme işlevlerinın yaptığı da silinecek yerleri boşlukla doldurmaktır.

Önce tek tek karakterler ya da satırlarla uğraşan işlevlere değinelim. **delch** ve **wdelch** pencerede o an mantıksal imlecin altında olan karakteri siler ve aynı satırda mantıksal imlecin sağında bulunan karakterleri birer sola kaydırır. **deleteln** ve **wdeleteln** ise mantıksal imlecin bulunduğu satırı sildikten sonra alttaki satırları birer yukarı kaydırır.

clrtoeol ve **wclrtoeol** işlevleri mantıksal imlecin bulunduğu satırda imlecin sağında kalan tüm karakterleri silerler. **clrtobot** ve **wclrtobot** işlevleri ise önce bir **wclrtoeol** çağırıp satırda mantıksal imlecin solunda kalan karakterleri, daha sonra da mantıksal imlecin bulunduğu satırın altında kalan tüm satırları silerler.

Bunların dışında bir de tüm ekranı ya da pencereyi temizlemeye yarayan işlevler vardır. Tüm ekranı silecek işlevlerin kullanabileceği iki yöntem vardır. Birincisi tüm ekranı boşluklarla doldurup **refresh** işlevini çağırmak, diğeri de uçbirime ekranı temizlemesini belirten kontrol kodunu göndermektir. Birinci yöntem ikinciden daha yavaştır çünkü tek tek her karakterin ekrana yeniden yazılmasını gerektirir. İkincisi ise tüm ekranı hemen temizler.

erase ve **werase** işlevleri bir pencerenin karakter dizisini boşluklarla dolduran işlevlerdir. Bir sonraki **refresh** çağrısında pencere temizlenmiş olacaktır. Bu işlevleri kullanmak temizlenmek istenen pencere tam ekran boyutunda ise çok mantıklı değildir. Çünkü bu işlevler yukarıda bahsedilen yöntemlerden ilkini kullanmaktadır. Temizlenecek pencere tam ekran boyutunda olduğu zaman aşağıda anlatacağım işlevleri kullanmak daha yararlıdır.

Diğer işlevlere geçmeden önce `_clear` bayrağından sözetmek gerekir. Her pencerede olan `_clear` bayrağı, eğer doğruysa bir sonraki **refresh**'te uçbirime kontrol kodunun gönderilmesini ister. Ancak **refresh** bunu yapmadan önce bunu isteyen pencerenin tam ekran boyutunda olup olmadığını (`_FULLWIN` bayrağı ile) kontrol eder. Eğer pencere tam ekran boyutundaysa, **refresh** uçbirimin ekranı temizlemesini sağlar ve sadece boşluktan farklı olan karakterleri ekrana yazar. Bu, tüm ekran boyutunda temizlemenin hızlı olmasını sağlar. Uçbirim yönteminin sadece tam ekran boyutundaki pencerelerin temizlenmesi için kullanılmasının sebebi, uçbirime kontrol kodu gönderildiği zaman tüm ekranın temizlenmesidir. Pencere tam ekran boyutunda değilse, tüm ekranın temizlenmesi istenmez. `_clear` bayrağını **clearok** işlevi kontrol eder.

clear ve **wclear** işlevleri tam ekran boyutundaki pencerelerin temizlenmesi için tercih edilir. Aslında bu işlevler bir **werase** ve bir **clearok** çağrısı yapmaya denktir. Öncelikle pencerenin karakter dizisini boşluklarla doldururlar. Daha sonra da `_clear` bayrağını doğrulayarak pencere tam ekran boyutunda ise uçbirim yoluyla, değilse de tek tek tüm karelerin üzerine boşluk karakteri yazmak yoluyla pencereyi temizlerler.

Sonuç olarak, temizlenecek pencerenin tam ekran boyutunda olduğunu biliyorsanız, **clear** yoksa **wclear** kullanın. Daha hızlı bir sonuç elde edersiniz. Fakat silinecek pencere tam ekran boyutunda değilse, **wclear** ya da **werase** kullanmanız arasında bir performans farkı yoktur.

8. Renkler

Ekran üzerinde görülen renkleri renk çiftleri olarak adlandırmak gerekir. Çünkü her karenin bir artalan rengi, bir de yazı rengi vardır. **ncurses** ile renkli yazabilmenin yolu da kendinize ait renk çiftleri oluşturmak ve yazılarınızın bu çiftler kullanılarak yazılmasını sağlamaktır.

ncurses işlevlerini kullanabilmek için **initscr** çağrısı yapmanızın gerekmesi gibi, renk işlevlerini kullanmadan önce de **start_color** işlevini çağırmanız gerekir. Bundan sonra renk çiftleri oluşturmak için kullanmanız gereken işlev **init_pair** işlevidir. **init_pair** ile bir renk çifti oluşturduğunuzda, işleve aktarmış

olduğunuz ilk parametre olan sayı ile bu renk çiftini ilişkilendirmiş olursunuz. Daha sonra yazılımınızda bu çifti kullanmak istediğinizde COLOR_PAIR makrosunu bu sayı ile çağırmanız yeterli olacaktır.

Renk çiftleri oluşturmanın dışında yazılarınızın bu çiftlerle yazılmasını sağlamanız gerekir. Bunu yapan işlevler **attron** ve **wattron** işlevleridır. Bu işlevler, **attroff** veya **wattroff** çağrıları yapılana kadar ilgili pencereye yazılacak her yazının belirtilen renk çifti ile yapılmasını sağlarlar.

Bir de pencerelerin genel artalan ve yazı renklerini değiştiren **bkgd** ile **wbkgd** işlevleri vardır. Bu işlevler çağrıldıkları zaman ilgili pencerenin renk çiftini istenen renk çifti olarak değiştirirler ve bundan sonra pencerenin tüm karelerinin artalan rengi ile tüm yazıların rengi otomatik olarak değişir.

Kullanabileceğiniz renkler ve adı geçen işlevlerin kullanımları ile ilgili ayrıntılar için man sayfalarına bakınız.

9. Pencerelerin Kutulanması

Güzel bir görüntü oluşturmak için pencerelerinizi kutu içine alabilirsiniz. Kütüphane içerisinde bunu sizin için yapacak **box** adlı bir makro vardır. Ancak diğer işlevlerden farklı olarak **wbox** mevcut değildir; **box** makrosu zaten bir pencere akımını parametre olarak alır.

Makronun kolay kullanımları ile ilgili ayrıntıları man sayfasından öğrenebilirsiniz. Burada başka bir noktaya dikkat etmeniz gerekmektedir. Bu makroyla pencerelerinizi kutu içine almanızın tek anlamı, o pencerenin karakter dizisinde pencereyi sınırlayan karelere kutuyu oluşturacak karakterlerin yazılmasıdır. Eğer siz bu karakterlerin üzerine yazarsınız kutunun görüntüsünü bozabilirsiniz. Bunun için, asıl pencerenin içinde **subwin** ile yeni bir pencere oluşturup dıştaki pencereyi kutu içine almanız ve yazı yazmak için içteki pencereyi kullanmanız güvenilir bir yöntem olabilir.

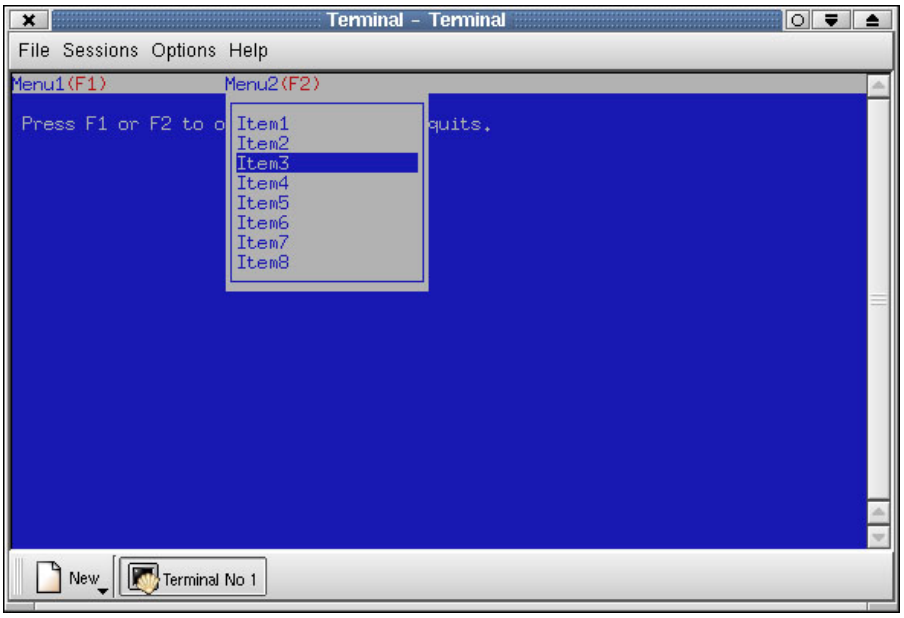
10. İşlev Tuşları

Klavyedeki işlev tuşlarını kullanabilmeniz için öncelikle klavyeden bilgi okuyacağınız pencerede _use_keypad bayrağının değerinin doğrulanması gerekir. Bunu yapan işlev **keypad** işlevidir. Bu işleve çağrı yaparak işlev tuşlarını kullanılır hale getirdikten sonra normal girdi işlevleri ile klavyeden giriş alabilirsiniz.

Ancak bu durumda, örneğin **getch** ile aldığınız bir bilgiyi, **char** türünde değil de **int** türünde bir değişkenin içerisinde tutmaya dikkat edin. Çünkü işlev tuşlarının sayısal değerleri bir **char** türü değişken içinde saklayabileceğiniz değerlerden büyüktür. İşlev tuşlarının sayısal değerlerini bilmenize gerek yoktur, bu sayısal değerler yerine kullanabileceğiniz isimleri **getch** işlevinin man sayfasında bulabilirsiniz.

11. Örnek Yazılım

Burada inceleyeceğimiz örnek yazılım basit ama öğretici bir yazılımdır. Bu yazılımda **ncurses** kullanılarak menüler oluşturulmuş ve menüdeki seçeneklerden bir tanesinin seçilmesi basit bir biçimde uygulanmıştır. Bu yazılımda önemli olan **ncurses**'in pencere bileşenlerini menü etkisi yaratabilecek şekilde kullanmaktır. Yazılımın ekran görüntüsünü aşağıda görüyorsunuz:



Yazılımda her zaman olduğu gibi önce kullanılan kütüphanelerin başlık dosyaları yazılır. Bir de yazılım içerisinde kullanacağımız `<enter>` ve `<escape>` tuşlarının ASCII karşılıkları yerine kullanılacak sabitler tanımlanır.

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

#define ENTER 10
#define ESCAPE 27
```

Aşağıdaki `init_curses` işlevi yazılımın ilk çalıştığı zaman çağırdığı işlevdir. Önce **`initscr`** çağrısı yaparak **`ncurses`** çalıştırılır, daha sonra **`start_color`** ile renkler kullanılır hale getirilir ve ardından da yazılım boyunca kullanılacak renk çiftleri tanımlanır. **`curs_set(0)`** çağrısı fiziksel imlecin ekran üzerinde görünmemesini sağlarken, **`noecho`** çağrısı da kullanıcının klavyeden girdiği karakterlerin ekrana yazılmasını engeller. **`noecho`** işlevini girdileri kontrol ederek almak ve sadece kabul edilen karakterleri ekrana aktarmak gibi bir amaç için de kullanabilirsiniz. **`noecho`** işlevinin etkisini kaldırmak için gerektiği zaman **`echo`** çağrısı yapılır. Aşağıdaki işlev son olarak **`keypad`** çağrısı yaparak `stdscr`'den alınacak girdilerde işlev tuşlarının da kullanılabilmesini sağlar. F1 ve F2 ile ok tuşlarını kullabilmek için bu çağrı gereklidir.

```
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);
}
```

Bundan sonraki işlev üzerinde menü isimleri bulunan satırı yazan işlevdir. Biraz aşağılara inerek yazılımda yalnızca bir satır olarak görülen menü çubuğunun, `main` işlevi içerisinde `stdscr`'nin alt penceresi olarak tanımlanmış tek satırlık bir pencere olduğunu görebilirsiniz. İşte aşağıdaki işlev o pencereyi parametre olarak alıp önce artalan rengini değiştirir, sonra da menü isimlerini yazar. Menü isimlerini burada **`waddstr`** ile yazıyoruz. Bu tercihin belirli bir sebebi yok, istenilen işlev kullanılabilirdi. İlk satırdaki **`wbkgd`** çağrısı ile pencerenin renk

çifti olarak tanımlanan 2 numaralı çiftin dışında bir renk çifti (3) ile yazmak istediğimizde **wattron** işlevini kullandığımıza dikkat edin. Yeniden 2 numaralı çift ile yazmak için ise **wattroff** kullanıyoruz.

```
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));
}
```

Sıradaki işlev F1 ya da F2 tuşlarına basıldığı zaman menülerin açılmasını sağlayan işlevdir. Menü açılma etkisi oluşturmak için artalanı oluşturan pencere (mavi renkli) üzerine menü çubuğu ile aynı renkte (beyaz) yeni bir pencere açılmaktadır. Ancak bu yeni pencere açıldığı zaman altta kalan pencerede yazılmış olan yazıların silinmesini istemiyoruz. Menü kapandığı zaman bu yazıların yeniden görünmesini istiyoruz. Bu sebeple yeni oluşturulan pencerenin `stdscr`'nin bir alt penceresi olarak oluşturulması doğru değildir. Aşağıda da görüldüğü gibi *items[0]* adlı pencere **newwin** işleviyle oluşturulmuştur. Diğer 8 tane *items* penceresi ise *items[0]*'ın alt pencereleri olarak oluşturulmuştur. Burada *items[0]* menüyü sınırlayan çerçeveyi çizmek için kullanılırken diğer pencereler ise hem çerçeveyi oluşturan karakterlerin üzerine yazmamak hem de menüde seçili olan elemanı göstermek için kullanılacaktır. Seçili olan elemanı göstermek için karşılık gelen *items* penceresinin artalan rengini diğerlerinden farklı yapmak yeterlidir. Sondan iki önceki satırda da bu yapılmaktadır ve menü ilk açıldığında birinci sıradaki eleman seçili olarak görünmektedir.

```
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;
}
```

Sıradaki işlev yukarıdaki işlev tarafından oluşturulan menü pencerelerini yokeden işlevdir. Bunun için önce pencereler tek tek **delwin** ile yokedilip daha sonra da *items* akım göstericisi için alınan bellek geri verilmektedir.

```
void delete_menu(WINDOW **items, int count)
{
  int i;

  for (i = 0; i < count; i++)
    delwin(items[i]);

  free(items);
}
```

scroll_menu işlevi menüler arasında ve menü içinde gezmeyi sağlayan işlevdir. Sonsuz bir döngü içinde **getch** işlevi ile klavyeden girilen tuşları okumaktadır. Kullanıcı aşağı yukarı ok tuşlarına basarsa menüde aşağıdaki ya da yukarıdaki eleman seçili olarak gösterilmektedir. Bu, yukarıda da bahsettiğim gibi, seçili olan elemanın bulunduğu pencerenin artalan rengini diğerlerinden farklı yaparak sağlanmaktadır. Kullanıcı eğer sağ ve sol ok tuşlarına basarsa içinde bulunulan menü kapatılmakta ve diğer menü açılmaktadır. Kullanıcı <Enter> tuşuna basarsa seçilmiş olan elemanın değeri geri döndürülür. <Escape> tuşu ise herhangi bir elemanı seçmeden menüleri kapatmaya yarar. İşlev diğer tuşları dikkate almamaktadır. Bu işlevde **getch** ile klavyeden ok tuşlarının okunabilmesi için iki hatırlatma yapmakta fayda var. Öncelikle en baştaki **init_curses** işlevinde keypad(stdscr, TRUE) çağrısı yapılarak stdscr'den işlev tuşlarının okunabilmesi sağlanmıştı. Buna ek olarak bir de **getch** ile okunan değer **int** türünde bir değişkende saklanmaktadır, çünkü ok tuşları ve diğer işlev tuşlarının sayısal değerleri **char** türünde bir değişkenin tutabileceğinden büyüktür.

```
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;
  }
}
```

Son olarak sırada `main` işlevi var. `main` işlevi yukarıda anlatılan işlevleri kullanarak yazılımın çalışmasını sağlamaktadır. Bu işlev de **`getch`** ile basılan tuş değerleri okumakta ve eğer F1 ya da F2 tuşlarına basılmışsa `draw_menu` ile karşılık gelen menüyü çizdirmektedir. Menü çizildikten sonra `scroll_menu` işlevi ile menülerden bir eleman seçilmesi sağlanmaktadır. `scroll_menu` işlevi geri döndükten sonra da önce menü pencereleri silinmekte ve ardından seçilen eleman *messagebar* penceresine yazdırılmaktadır.

Burada **`touchwin`** işlevine değinmekte fayda var. Menüler kapandıktan sonra **`touchwin`** çağrılmadan **`refresh`** yapılsaydı en son açılan menü ekranda durmaya devam ederdi. Bunun sebebi menüler oluşturulduğu zaman `stdscr` üzerinde hiçbir değişiklik yapılmaması ve **`refresh`** çağrıldığı zaman `stdscr` hiç değişmemiş olarak görüldüğü için yeniden yazılmamasıdır. **`touchwin`** işlevi parametre olarak aldığı pencerenin tüm satırlarındaki satırın değiştiğini belirten bayrakları doğrulayarak bir sonraki **`refresh`**'te pencerenin, hiç değişmemiş olsa bile, en baştan çizilmesini sağlar. Menüler oluşturulduğu zaman `stdscr`'ye hiç dokunulmamış olması sayesinde, menüler kapandığı zaman `stdscr`'de yazan bilgi kaybolmamıştır.

```
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);
      wprintw(messagebar, "You have selected menu item %d.", selected_item + 1);

      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;
}
```

Aralara yazdığım açıklamaları çıkarıp kaynak kodunu `example.c` adlı dosyaya kaydettiğinizi varsayarak,

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

komutu ile kodu derleyebilir ve yazılımı test edebilirsiniz.

12. Sonuç

ncurses kullanarak yazılımınız için güzel bir arayüz oluşturabilmeniz için gereken temel konulara değindim. Ancak bu kütüphaneyle yapabilecekleriniz burada anlatılanlarla sınırlı değildir. Sık sık bakmanızı tavsiye ettiğim man sayfalarında kütüphane ile ilgili başka yönler de keşfedecek ve bu anlattıklarımın yalnızca giriş düzeyinde bilgiler olduğunu göreceksiniz.

A. example.c

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

#define ENTER 10
#define ESCAPE 27
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);
}
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));
}
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;
}
void delete_menu(WINDOW **items,int count)
{
        int i;
```

```
        for (i=0;i<count;i++)
                delwin(items[i]);
        free(items);
}
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;
                }
        }
}
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;
}
```

Notlar

a) Belge içinde dipnotlar ve dış bağlantılar varsa, bunlarla ilgili bilgiler bulundukları sayfanın sonunda dipnot olarak verilmeyip, hepsi toplu olarak burada listelenmiş olacaktır.

b) Konsol görüntüsünü temsil eden sarı zeminli alanlarda metin genişliğine sığmayan satırların sığmayan kısmı ¬ karakteri kullanılarak bir alt satıra indirilmiştir. Sarı zeminli alanlarda ¬ karakteri ile başlayan satırlar bir önceki satırın devamı olarak ele alınmalıdır.

Bu dosya (ncurses.pdf), belgenin XML biçiminin TEXLive ve belgeler-xsl paketlerindeki araçlar kullanılarak PDF biçimine dönüştürülmesiyle elde edilmiştir.

9 Şubat 2007