NCURSES ile Yazılım Geliştirme NASIL

Yazan:
Pradeep Padala
<ppadala (at) gmail.com>

Çeviren:
Oğuz Yarımtepe
<oguzy (at) comu.edu.tr>

v1.9, 2005–06–20

Özet

Bu belge ncurses ve kardeş kütüphaneleriyle yazılım geliştirmek için "Herşey Dahil" bir kılavuz olmayı hedeflemektedir. Basit bir "Merhaba Dünya" programından başlayıp daha karmaşık yapılara doğru giden bir anlatım kullandık. Ncurses ile ilgili herhangi bir ön deneyim gerekli değildir.

Yorumlarınızı <ppadala (at) gmail.com> adresine yollayınız.

Konu Başlıkları

Bu çevirinin sürüm bilgileri:

1.0	Aralık 2005	OY
İlk çeviri		

Özgün belgenin sürüm bilgileri:

1.9	2005-06-20	ppadala
Lisans NCURSES tarafından kullanılan MIT–türü lisansa dönüştürüldü. Program da bunun altında tekrar lisanslandırıldı.		
1.8	2005-06-17	ppadala
Pekçok güncelleme yapıldı. Referanslar ve perl örnekleri eklendi. Pekçok dil bilgisi ve stil değişikliği içeriğe eklendi. NCURSES tarihçesinde değişiklik yapıldı.		
1.7.1	2002-06-25	ppadala
Kaynak koddan kurulum için README dosyası ve talimatlar eklendi.		
1.7	2002-06-25	ppadala
"Diğer biçimler" kısmı eklendi ve programda pekçok hoş görünümlü değişiklik yapıldı. Satır içi programlar çıkarıldı.		
1.6.1	2002-02-24	ppadala
Eski Değişiklik Kaydı (Changelog) kısmı çıkarıldı, programı oluşturmak için kullanılan dosyalar (makefiles) temizlendi.		
1.6	2002-02-16	ppadala
Pekçok yazım hatası düzeltildi, ACS değişkenleri eklendi		
1.5	2002-01-05	ppadala
Yapı şu anki uygun TOC (İçerik Tablosu) yapısına dönüştürüldü		
1.3.1	2001-07-26	ppadala
Paket yapımcılarıyla ilgili paragraf düzeltildi, kararlı sürüm numarası düzeltildi.		
1.3	2001-07-24	ppadala
Ana belgeye (LDP lisansı) ve programalara (GPL lisansı) lisans bilgisi eklendi, printw_ örneği düzeltildi.		
1.2	2001-06-05	ppadala
Ravi'nin daha çok giriş, menu, form ve sadece eğlenmek için ile ilgili değiklikleri uygulandı.		
1.1	2001-05-22	ppadala
"pencere hakkında bir çift söz" kısmı eklendi, scanw_ örneği eklendi.		

Yasal Açıklamalar

1. Giriş

Aptal uçbirimlerin olduğu eski zamanlarda, uçbirimler bilgisayarlardan uzakta ve onlara bağlantıları seri kablolar ile idi. Uçbirimler seri şekilde bayt verileri göndermek için ayarlanabiliyordu. Uçbirimlerin tüm bu yetenekleri (imleci yeni bir noktaya taşımak, ekranın belli bir kısmını silmek, ekranda gezinmek, durumlar arası geçişler, v.b. gibi) seri şekilde gönderilen bu baytlar ile sağlanmaktaydı. Bu denetim ardışılları genelde kaçış tuşu dizisi olarak adlandırılırdı, çünkü kaçış karakteri olan (0x1B) ile başlamaktaydı. Şimdi bile uygun bir benzetim ile benzeticiye kaçış karakterlerini göndererek uçbirim penceresinde aynı etkiyi oluşturabiliriz.

Bir satırı renkli olarak yazmak istediğinizi düşünelim. Konsolda şunu yazmayı deneyin.

```
echo -e "\033[1;37;44mRenkli"
```

Baştaki "\033[" dizgesi kaçış karakteridir. Diğer karakterler ekrana yazılabilir karakterlerdir. Sonucu mavi zemin üzerine beyaz "Renkli" olarak görüyor olmalısınız. Bu renklendirme, bu şekilde kalır, eski haline dönmek için şunu yazın:

```
echo -e "\033[0;39;49mRenkli"
```

Şimdi, bu sihirli karakterler de ne anlama geliyor? Anlaması zor mu? Bunlar farklı uçbirimler için farklı bile olabilir. Bunun için UNIX'ı tasarlayanlar **termcap** isimli bir mekanizma bulmuşlardır. Belli bir uçbirimin tüm yetenekleri ile belli bir etkiyi elde etmek için gerekli kaçış karakterlerini listeleyen bir dosyadır. Sonraki yıllarda bu **terminfo** ile yer değiştirmiştir. Çok fazla ayrıntıya inmeden, bu mekanizma uygulama programlarının terminfo veritabanını sorgulamasını ve uçbirime veya uçbirim benzeticisine gönderilecek denetim karakterlerini edinmelerini sağlar, demekle yetineceğiz.

1.1. NCURSES nedir?

Tüm bu teknik saçmalıkların ifade edilmesinin nedenini merak ediyor olabilirsiniz. Yukarıdaki senaryoda her uygulama programının terminfo'yu sorgulaması ve gerekli işlemleri yapması gerekmektedir (denetim karakterlerinin gönderilmesi v.s.). Bu karmaşıklıkla uğraşmak bir süre sonra zorlaştı ve bu da 'CURSES"ın doğmasına sebep oldu. Curses "imleç eniyileme" ("cursor optimization") anlamına gelen bir kelime oyunudur. Curses kütüphanesi ham uçbirim kodları ile çalışırken onları sarmalayıcı bir yapı ve verimli bir API (Uygulama Programlama Arayüzü) oluşturmaktadır. İmleci hareket ettirmek, pencereler oluşturmak, renkler üretmek, fare ile oynamak v.b. için işlevler sağlamaktadır. Uygulama programları alt kısımdaki uçbirim yeteneklerini düşünmek zorunda kalmazlar.

Yani NCURSES ne demektir? Asıl System V gözden geçirme 4.0 (SVr4) curses'ın bir kopyasıdır. Özgür olarak dağıtılan ve eski curses sürümleriyle tamamen uyumlu olan bir kütüphanedir. Kısacası, bir uygulamanın karakter tabanlı uçbirimlerde görüntüleri üzerinde çalışabilmeyi sağlayan işlevler kütüphanesidir. Belgenin geri kalanında, curses ve ncurses birbirleri yerine kullanılmıştır.

NCURSES hakkındaki ayrıntılı bilgi kaynak dağıtımındaki NEWS dosyasından bulunabilir. Şu anki paket Thomas Dickey <dickey (at) his.com> tarafından geliştirilmektedir. Paketi hazırlayanlarla <bug–ncurses (at) gnu.org> adresinden iletişime geçebilirsiniz.

1.2. NCURSES ile Ne Yapabiliriz?

NCURSES sadece uçbirim yetenekleri sarmalayan bir yapı oluşturmaz aynı zamanda metin kipinde hoş görüşünüşlü arayüzler oluşturmak için sağlam bir çalışma taslağı da oluşturur. Pencereler v.b. görünümleri oluşturmak için işlevler sağlar. Kardeş kütüphaneleri olan panel, menu ve form, temel curses kütüphanesine bir genişleme sağlar. Bu kütüphaneler genelde curses kütüphanesiyle beraber gelmektedir. Pekçok pencere, menu, panel ve formlar içeren bir uygulama oluşturulabilir. Pencereler üzerinde bağımsız olarak çalışılabilir, gezinilebilir ve hatta pencereler gizlenebilir.

Menüler kullanıcıya kolay şekilde komut seçimini sağlamaktadır. Formlar kolay kullanımlı veri girişi ve pencere görünümleri oluşturulmasına izin verir. Paneller ncurses'ın yeteneklerini üst üste binmiş ve yığılmış pencereler ile uğraşabilmeyi sağlar.

Tüm bunlar ncurses ile yapabileceğimiz bazı temel şeylerdir. Devam ettikçe tüm bu kütüphanelerin yeteneklerini göreceğiz.

1.3. Nereden Edinilebilir?

Tamam, artık ncurses ile ne yapabileceğinizi biliyorsunuz, başlamaya yaklaşmalısınız. NCURSES genelde yüklediğiniz dağıtım ile gelmektedir. Kütüphaneye sahipseniz veya onu kendiniz derlemek istiyorsanız okumaya devam edin.

Paketin derlenmesi

NCURSES `ftp://ftp.gnu.org/pub/gnu/ncurses/ncurses.tar.gz` adresinden veya `http://www.gnu.org/order/ftp.html` adresinde belirtilen ftp adreslerinden edinilebilir.

Yüklemek için README ve INSTALL dosyalarını dikkatlice okuyunuz. Genelde aşağıdaki adımları içermektedir.

```
tar zxvf ncurses-sürüm.tar.gz          # arşivi açın
cd ncurses-sürüm                       # oluşan dizine geçin
./configure                            # ortam değişkenlerinize göre
                                       # yapılandırın
make                                   # derleyin
su root                                # root olun
make install                           # kurun
```

RPM'den kurulum

NCURSES'ün RPM paketi `http://rpmfind.net` adresinden indirilebilir. RPM'si root olduktan sonra aşağıdaki komut kullanılarak yüklenebilir.

```
rpm -i indirilmiş–rpm–paketi
```

1.4. Belgenin Amacı/Kapsamı

Bu belge ncurses ve kardeş kütüphanelerini kullanarak yazılım geliştirmek için "Hepsi Bir Arada" bir yol gösterici olmayı hedeflemektedir. Basit bir "Hello World" programından daha karmaşık olanlara doğru yol alacağız. Ncurses ile bir ön tecrübe yaşamış olmak gerekmemektedir. Yazım şekli kurallara sıkı bağlı değildir fakat örneklerde pek çok ayrıntı sağlanmıştır.

1.5. Programlar Hakkında

Belgedeki tüm programlar sıkıştırılmış olarak
`http://www.tldp.org/HOWTO/NCURSES-«Programming-«HOWTO/ncurses_programs.tar.gz`
adresinde mevcuttur. Sıkıştırılmış olan bu dosyayı açtığınız zaman dizin yapısı şu şekilde gözükecektir.

```
ncurses
   |
   |----> JustForFun     -- sadece eğlence için yazılmış programlar
   |----> basics         -- temel programlar
   |----> demo           -- derlemeden sonra çıktı dosyaları bu dizin
   |          |          -- altına gider
```

```
   |          |----> exe -- tüm örnek programlar için exe dosyaları
   |----> forms          -- form kütüphanesiyle ilgili programlar
   |----> menus          -- menü kütüphanesiyle ilgili programlar
   |----> panels         -- panel kütüphanesiyle ilgili programlar
   |----> perl           -- örneklerin perl karşılıklar
   |                     --(Anuradha Ratnaweera tarafından sağlanmıştır)
   |----> Makefile       -- en tepedeki Makefile dosyası
   |----> README         -- en tepedeki README dosyası. Talimatları ve
   |----> COPYING        -- telif bilgisini içerir
```

Tek tek ele alırsak dizinler aşağıdaki dosyaları içerir.

```
Her bir dizindeki dosyaların açıklamaları
-----------------------------------
JustForFun
    |
    |----> hanoi.c   -- Hanoi Kulesi problemini çözücü
    |----> life.c    -- Hayat Oyunu, demo
    |----> magic.c   -- Tek Sıralı Sihirli Kare yapıcı
    |----> queens.c  -- Meşhur N-Vezir sorusunu çözücü
    |----> shuffle.c -- Öldürmek için zamanınız varsa, eğlenceli bir oyun
    |----> tt.c      -- Oldukça deneysel bir yazma öğreticisi

  basics
    |
    |----> acs_vars.c            -- ACS_ değişkenleri örneği
    |----> hello_world.c         -- Basit bir "Merhaba Dünya" örneği
    |----> init_func_example.c   -- İlklendirme işlevleri örneği
    |----> key_code.c            -- Basılan karakteri gösterir
    |----> mouse_menu.c          -- Fare ile erişilebilir menüler
    |----> other_border.c        -- box() işlevinden başka diğer çerçeve
    |                            -- işlevlerinin kullanımını gösterir
    |----> printw_example.c      -- Basit bir printw() örneği
    |----> scanw_example.c       -- Basit bir getstr() örneği
    |----> simple_attr.c         -- Bir c dosyasını yorum satırlarına
    |                            -- belli bir özellik kazandırarak gösterir
    |----> simple_color.c        -- Renkleri gösteren basit bir örnek
    |----> simple_key.c          -- Klavyenin YUKARI, AŞAĞI ok tuşlarıyla
    |                            -- erişilebilen bir menü
    |----> temp_leave.c          -- Geçici olarak curses kipini terk etmeyi
    |                            -- gösterir
    |----> win_border.c          -- Pencere ve çerçevelerin oluşturulmasını
    |                            -- gösterir
    |----> with_chgat.c          -- chgat() kullanım örneği

  forms
    |
    |----> form_attrib.c     -- Alan özelliklerinin kullanımı
    |----> form_options.c    -- Alan seçeneklerinin kullanımı
    |----> form_simple.c     -- Basit bir form örneği
    |----> form_win.c        -- Formlarla birleşik bir pencere örneği
```

```
menus
   |
   |----> menu_attrib.c      -- Menü özelliklerinin kullanımı
   |----> menu_item_data.c   -- item_name() v.b. işlevlerinın kullanımı
   |----> menu_multi_column.c    -- Çok sütunlu menüler oluşturur
   |----> menu_scroll.c      -- Menülerin gezinilebilme yetilerini gösterir
   |----> menu_simple.c      -- Ok tuşlarıyla erişilebilen basit bir menü
   |----> menu_toggle.c      -- Çok değerli menüler oluşturur ve
   |                         -- REQ_TOGGLE_ITEM'ı açıklar
   |----> menu_userptr.c     -- Kullanıcı göstericisinin kullanımı
   |----> menu_win.c         -- Menülerle pencerelerin beraber kullanımı

panels
   |
   |----> panel_browse.c     -- Sekme ile paneller üzerinde gezinme.
   |                         -- Kullanıcı göstericisinin kullanımı
   |----> panel_hide.c       -- Panelleri saklamak ve açığa çıkarmak
   |----> panel_resize.c     -- Panelleri hareket ettirmek ve yeniden
   |                         -- boyutlandırmak
   |----> panel_simple.c     -- Basit bir panel örneği

perl
   |----> 01-10.pl           -- İlk on örneğin Perl karşılığı
```

Ana dizin içerisinde temel bir Makefile dosyası bulunmaktadır. Tüm dosyaları derleyerek kullanıma hazır exe dosyalarını demo/exe dizinine koymaktadır. Dizinlerin kendilerine de girerek de derleme işlemini o dizine özgü gerçekleyebilirsiniz. Her dizinde, içerisindeki c dosyalarının amacını içeren bir README dosyası bulunmaktadır.

Her örnek için, örnekler dizininde bulunacağı yolu eklemiştim.

Eğer programlara tek tek erişmek isterseniz http://tldp.org/HOWTO/NCURSES-«Programming-«HOWTO/ncurses_programs/ adresine gidin.

Tüm programlar ncurses tarafından kullanılan lisans ile yayınlanmıştır (MIT–tarzı). Bu da size kendinize ait olduklarını söylemeniz dışında pekçok şeyi yapmanıza imkan sağlamaktadır. Yazılımınızda gerektiği takdirde kullanmaktan çekinmeyin.

1.6. Belgenin Diğer Biçimleri

Bu belge tldp.org sayfasında değişik biçimlerde de edinilebilir. Bu belgenin diğer biçimlerine bağlantılar şu şekildedir:

1.6.1. tldp.org sayfasındaki kolayca erişilebilir dosya biçimleri:

- Acrobat PDF Biçimi[B6]
- PostScript Biçimi[B7]
- Çok Sayfalı HTML biçimi[B8]
- Tek Sayfalık HTML biçimi[B9]

1.6.2. Kaynaktan Kurmak

Eğer yukarıdaki adresler hatalıysa ve sgml'yi denemek istiyorsanız okumaya devam edin.

```
    http://cvsview.tldp.org/index.cgi/LDP/howto/docbook/
        NCURSES-HOWTO/NCURSES-Programming-HOWTO.sgml
    http://cvsview.tldp.org/index.cgi/LDP/howto/docbook/
        NCURSES-HOWTO/ncurses_programs.tar.gz
adreslerindeki gzip ile sıkıştırılmış tar dosyalarını ve
kaynak dosyalarını edinin.

ncurses_programs.tar.gz'i şu şekilde açın:
tar zxvf ncurses_programs.tar.gz

Değişik dosya biçimleri oluşturmak için jade'i kullanın.
Örneğin sadece çoklu html'ler oluşturacaksanız, sadece
şunu kullanacaksınız:
jade -t sgml -i html -d docbook html için dsssl yolu \
    NCURSES-Programming-HOWTO.sgml

pdf elde etmek için önce NASIL'ın tek sayfalık html dosyasını
şu şekilde oluşturun:
jade -t sgml -i html -d docbook html için dsssl yolu -V nochunks \
    NCURSES-Programming-HOWTO.sgml > NCURSES-TEK-BUYUK-DOSYA.html
daha sonra htmldoc'u pdf elde etmek için şu şekilde kullanın
htmldoc --size universal -t pdf --firstpage p1 -f çıktı dosyası adı.pdf \
    NCURSES-TEK-BUYUK-DOSYA.html
ps için şunu kullanıyor olacaksınız
htmldoc --size universal -t ps --firstpage p1 -f çıktı dosyası adı.ps \
    NCURSES-TEK-BUYUK-DOSYA.html
```

Ayrıntılar için LDP yazar rehberine[B10] bakın. Eğer hepsi başarısız olursa bana <ppadala (at) gmail.com> adresine posta gönderin.

1.7. Katkıda Bulunanlar

Sharath <sharath_1 (at) usa.net> ve Emre Akbas'a bir kaç bölümde bana yardım ettikleri için teşekkürler. Giriş kısmı ilk olarak sharath tarafından yazıldı. Daha sonra onun ilk çalışmasından az bir bölümü tekrar yazdım. Emre, printw ve scanw bölümlerini yazmada yazdım etti.

Örnek programların Perl karşılıkları Anuradha Ratnaweera <Aratnaweera (at) virtusa.com> tarafından hazırlandı.

Sırada benim sevgili arkadaşım Ravi Parimi <parimi (at) ece.arizona.edu> var, bu proje çevrimiçi yazıldığından beri o da yer aldı. Düzenli olarak önerilerle beni topa tuttu ve tüm yazılı metni sabırla gözden geçirdi. Her bir programı da ayrıca Linux ve Solaris ortamında kontrol etti.

1.8. İstek Listesi

Bu kısım öncelik sırasına göre istek listesidir. Eğer bir istek üzerinde çalışmak istiyor veya bir öneriniz varsa bana yazın (<ppadala (at) gmail.com>).

- Form kısımlarının sonuna örnekler eklemek
- Tüm programları gösteren bir örnek hazırlamak ve her kullanıcıya tüm programları taraması imkanı sağlamak. Kullanıcılara derleme ve programın çıktısını görme imkanı vermek. Etkileşimli bir arayüz tercih edilir.

- Hata ayıklama bilgilerini eklemek. _tracef, _tracemouse konuları.
- Ncurses paketlerinin sunduğu işlevleri kullanarak termcap, terminfo'ya erişmek
- İki uçbirimde de aynı anda çalışmak.
- "Çeşitli" kısmına daha fazla eleman eklemek

2. Merhaba Dünya !!!

Curses dünyasına hoş geldiniz. Kütüphanelere dalmadan ve onların değişik özelliklerine bakmadan önce basit bir program yazarak gezegene merhaba diyelim.

2.1. NCURSES Kütüphaneleriyle Derleme

Ncurses kütüphane işlevlerini kullanabilmek için programınıza `ncurses.h` başlık dosyasını eklemelisiniz. Programlarınızı ncurses kütüphanesiyle ilintileyebilmek için `-lncurses` seçeneğiyle derleyin.

```
#include <ncurses.h>
.
.
.

derleyin ve bağlayın: gcc program dosyası -lncurses
```

Örnek 1. Merhaba Dünya !!! Programı

```
include <ncurses.h>

int main()
{
  initscr();                    /* Curses kipine giriş           */
  printw("Merhaba Dünya !!!");  /* Merhaba Dünya yazdırma        */
  refresh();                    /* Onu gerçek ekrana yazdırın    */
  getch();                      /* Kullanıcıyı bekleyin          */
  endwin();                     /* Curses kipinden çıkış         */

  return 0;
}
```

2.2. İnceleme

Yukarıdaki program ekrana "Meraha Dünya !!!" yazıp sonlanır. Bu program curses kipinin nasıl ilklendirileceğini, nasıl ekran değişikliği yapılacağını ve curses kipini nasıl sonlandırılacağını göstermektedir. Şimdi satır satır inceleyelim.

2.2.1. `initscr()` hakkında

`initscr()` uçbirimi curses kipinde ilklendirir. Bazı gerçekleştirmelerinde ekranı temizleyip boş bir ekran göstermektedir. Ekran üzerinde curses kullanarak bir değişiklik yapılacağında bu işlev ilk olarak çağrılmalıdır. Bu işlev curses sistemini ilklendirip şu anki pencere için (**`stdscr`** olarak isimlendirilen) ve bazı veri yapıları için bellekten yer ayırır. Bazı istisnai durumlarda bu işlev curses kütüphane veri yapıları için bellekten yer ayırırken yetersiz bellek yüzünden sekteye uğrayabilir.

Bu yapıldıktan sonra kendi curses ayarlarımınızı özelleştirmek için değişik ilklendirmeler yapabiliriz. Bu ayrıntılar *daha sonra* (sayfa: 13) anlatılacaktır.

2.2.2. Gizemli `refresh()`

Sonraki satır "Merhaba Dünya !!!" dizgeını ekrana yazar. Bu işlev, veriyi `stdscr` olarak isimlendirilen pencerenin o anki `(y,x)` koordinatlarına yazması dışında bildiğiniz `printf` işlevine her yönüyle benzemektedir. Şu anki koordinatlarımız `0,0` olduğundan, dizge pencerenin sol tarafından itibaren yazılır.

Bu da bizi şu sihirli bildiğiniz `refresh()`'e getirmektedir. `printw`'yi çağırdığımızda veri, hayali ve henüz ekranda güncellenmemiş bir pencereye yazılır. `printw`'nin görevi bazı bayrakları ve veri yapılarını güncelleyerek `stdscr`'ye karşılık gelen ön bellek alanına veriyi yazmaktır. Bunu ekranda göstermek için, `refresh()`'i çağırarak curses sistemine içeriği ekrana boşaltmasını söylemeliyiz.

Bunun arkadasındaki felsefe programcının hayali ekran veya penreler üzerinde birden fazla güncelleme yapması ve bu güncelemelerden sonra ekranını tazelemesidir. `refresh()` pencereyi kontrol eder ve kontrol edilen kısmı günceller. Bu da performans ve büyük bir esneklik sağlamaktadır. Fakat bazen yeni başlayanlar için sinir bozudur. Yeni başlayanların yaptığı en temel hatalardan biri `printw()` sınıfı işlevleri kullanarak güncelleme yaptıktan sonra `refresh()`'i çağırmayı unutmalarıdır. Ben bile bazen eklemeyi unutuyorum `:-)`

2.2.3. `endwin()` hakkında

Ve son olarak curses kipini sonlandırmayı unutmayın. Aksi halde uçbiriminiz program sonlandıktan sonra garip davranabilir. `endwin()`, curses alt sistemi tarafından alınan bellek bölgesini onun veri yapılarını geri verir ve uçbirimi normal çalışma kipine döndürür. Bu işlev curses kipiyle çalışmanız bittikten sonra çağrılmalıdır.

3. Kanlı Ayrıntılar

Basit bir curses programının nasıl yazılacağını şu ana kadar gördük artık ayrıntılara inelim. Ekranda gördüklerinizi özeleştirmeniz için pekçok işlev ve tam olarak kullanabileceğiniz pekçok özellik vardır.

İşte başlıyoruz...

4. İlklendirme

Artık curses sisteminin ilklendirilmesi için `initscr()` işlevinin çağrılması gerektiğini biliyoruz. Bu ilklendirmeden sonra curses oturumuzu özelleştirmemizi sağlayan işlevler vardır. Curses sisteminden uçbirimi ham kipe ayarlamasını veya renkendirmeyi ilklendirmesini veya fareyi etkin kılmasını v.b. isteyebiliriz. `initscr()`'den hemen sonra çağrılan işlevler hakkında biraz konuşalım:

4.1. `raw()` ve `cbreak()`

Normal durumda uçbirim sürücüsü satırsonu veya satırbaşı karakteri ile karşılaşana kadar kullanıcın yazdığı karakterleri ön belleğine alır. Fakat bazı programlar kullanıcı yazar yazmaz karakterlerin erişilebilir olmasını ister. Yukarıdaki iki işlev satır önbelleğini etkisizleştirir. Bu iki işlev arasındaki fark duraklatma (CTRL–Z), kesme ve çıkış (CTRL–C) için kullanılan kontrol karakterlerinin programa gönderilme şeklidir. **`raw()`** kipinde bu karakterler herhangi bir sinyal üretilmeden programa gönderilir. **`cbreak()`** kipinde ise bu karakterler herhangi bir diğer karakter gibi uçbirim sürücüsü tarafından algılanır. Kullanıcının neler yaptığı üzerinde daha fazla hakimiyet sağladığıdan kişisel olarak `raw()`'ı tercih ediyorum.

4.2. `echo()` ve `noecho()`

Bu işlevler kullanıcı tarafından yazılan karakterlerin uçbirim penceresine yansıtılmasını kontrol eder. **`noecho()`** bu yansıtmayı etkisizleştirir. Bunu yapmanızın sebebi yazılanı ekrana yansıtmada daha fazla kontrol sağlanması veya `getch()` v.b. işlevler ile kullanıcıdan girdi alınırken gereksiz karakter gösterimlerinin önüne geçmektir.

Kullanıcıyla etkileşimli pek çok yazılım ilklendirme sırasında **noecho()** işlevin çağırır ve karakterlerin ekrana yansıtılması işlemini kontrollü bir şekilde gerçeklştirir. Bu da programcıya o anki (y,x) koordinatlarını güncellemeden karakterlerin ekrana yansıtılması esnekliğini kazandırır.

4.3. keypad()

Bu benim en sevdiğim ilklendirme işlevim. F1, F2, ok tuşları, v.b gibi tuşların okunabilirliğini sağlar. Kullanıcı ile etkileşimli hemen her program ok tuşlarının bir kullanıcı arayüzünde vazgeçilmez olmasından dolayı bunu etkinleştirmektedir. **keypad(stdscr, TRUE)** ile öntanımlı olarak atanmış ekran için (stdscr) bu özelliğin etkin olmasını sağlayın. Klavye yönetimi hakkında daha fazla bilgiyi bu belgenin daha sonraki bölümlerinde öğreneceksiniz.

4.4. halfdelay()

Bu işlev çok sık kullanılmasa da, bazen oldukça yararlıdır. halfdelay(), cbreak() kipine benzer şekilde basılan karakterlerin program için edinilebilir olmasını sağlayan yarım–gecikme kipini etkinleştirir. Bununla beraber girdi için saniyenin 'X' * 10 birimi kadar bekleyip herhangi bir giriş yapılmazsa ERR geri döndürür. 'X', halfdelay() işlevine gönderilen zaman aşımı süresidir. Bu işlev kullanıcıdan bir girdi istediğiniz zaman kullanışlıdır, eğer belli zaman içerisinde bu giriş gerçeklemezse bir şeyler yapabiliriz. Muhtemel bir örnek olarak parola giriş ekranındaki zaman aşımı verilebilir.

4.5. Muhtelif İlklendirme İşlevleri

Curses davranışlarını ilklendirirken çağrılan bir kaç işlev daha vardır. Yukarıda belirtilenler kadar yoğun şekilde kullanılmazlar. Gerekli oldukları yerlerde bazıları açıklanmıştır.

4.6. Bir Örnek

Bu işlevlerin kullanımını açıklayacak bir program yazalım.

Örnek 2. İlklendirme İşlevlerini Kullanım Örneği

```
#include <ncurses.h>

int main()
{ int ch;

  initscr();                    /* Curses kipine giriş                      */
  raw();                        /* Satır önbelleği etkisizleştirildi        */
  keypad(stdscr, TRUE);         /* F1, F2, v.b. etkin                       */
  noecho();                     /* getch için karakterleri ekrana yansıtma */

  printw("Koyu görmek istediğiniz bir karakteri yazınız\n");
  ch = getch();                 /* Eğer raw() çağrılmamış olsa               *
                                 * karakterin programa gönderilmesi için     *
                                 * enter'a basmamız gerekecekti              */
  if(ch == KEY_F(1))            /* Klavye tuşları etkinleştirilmeden         */
    printw("F1 tuşuna basıldı");   /* bu değeri de alamayacaktık             */
                                /* noecho() olmadan                          *
                                 * bazı çirkin gözünüşlü karakterler         *
                                 * ekranda gözükebilir                       */
  else
  { printw("Basılan tuş ");
```

```
    attron(A_BOLD);
    printw("%c", ch);
    attroff(A_BOLD);
  }
  refresh();                    /* Karakteri gerçek ekrana yazdır */
  getch();                      /* Kullanıcı girişi için bekle    */
  endwin();                     /* Curses kipinin sonu            */

  return 0;
}
```

Bu program kendisini açıklayabilmektedir. Fakat henüz açıklanmayan işlevler kullandım. **getch()** komutu kullanıcıdan bir karakter almak için kullanılır. Bildik **getchar()**'a <enter> kullanımından kaçınmak için satır önbelleğiniz etkisizleştirebilmemiz dışında aynen benzemektedir. **getch()** ve tuşların okunmasıyla ilgili *klavye tuşları yönetimi* (sayfa: 29) kısmına bakınız. `attron` ve `attroff` işlevleri sırasıyla bazı özellikleri açmak ve kapamak için kullanılır. Örnekte karakterleri koyu yazmak için kullandım. Bu işlevler daha sonra ayrıntılı olarak anlatılacaktır.

5. Pencereler Hakkında Bir Çift Söz

Bir sürü ncurses işlevine dalmadan önce pencereler hakkında bir iki şeyi açıklığa kavuşturmama müsade edin. Pencereler *bölümünde* (sayfa: 22) ayrıntılı şekilde açıklanmıştır.

Pencere, curses sistemi tarafından tanımlanan hayali bir ekrandır. Pencere, genellikle Win9X ortamında gördüğünüz çerçeveli pencereler anlamına gelmemektedir. Curses ilklendirildiği zaman **stdscr** isminde sizin 80x25 boyutlarındaki (veya çalıştığınız pencerenin boyutu) ekranınızı temsil eden varsayılan bir pencere oluşturur. Bazı dizgelerin ekrana yazılması, giriş okuma v.b. gibi basit işler yapıyorsanız bu tek pencereyi tüm amaçlarınız için güvenle kullanabiliriniz. Ayrıca pencereler oluşturabilir ve belirtilen pencere üzerinde açık şekilde çalışacak işlevler çağırabilirsiniz.

Örneğin şu işlevi çağırırsanız

```
  printw("Selam millet !!!");
  refresh();
```

Dizgeyi imlecin o anki konumunda `stdscr` üzerinde yazacaktır. Benzer şekilde `refresh()` çağrısı sadece `stdscr` üzerinde çalışır.

Artık bir *pencere* (sayfa: 22) oluşturduğunuzu söyleyebilirsiniz ve bundan sonra bildik C işlevlerini 'w' eklenmiş olarak çağırabilirsiniz.

```
  wprintw(win, "Selam millet !!!");
  wrefresh(win);
```

Belgenin geri kalanında göreceğiniz gibi işlevlerin isimlendirilmesi aynı öngörüye dahildir. Her işlev için genelde üç tane daha işlev vardır.

printw(*dizge*);
  stdscr üzerine kursorun bulunduğu konumdan itibaren yazar.

mvprintw(*y, x, dizge*);
  (y, x) konumuna gider ve dizgeyi oraya yazar.

wprintw(*win, dizge*);
  Pencere içerisinde imlecin bulunduğu yerden itibaren
  win isimli pencereye yazar

mvwprintw(*win, y, x, dizge*);
  pencereye göre (y, x) koordinatlarına git ve *dizge*yi buraya yaz

Genelde `w`'siz işlevler, karşılık gelen `w`'li işlevlere pencere değişkeni olarak `stdscr` aktarılan makrolardır.

6. Çıktı işlevleri

Sanırım artık biraz hareket görmek için bekleyemiyorsunuz. Curses işlevlerimizle olan uzun ve serüvenli yolculuğa geri dönelim. Artık curses ilklendirildi haydi onla oynayalım.

Ekrana çıktılama işlemleri için kullanabileceğiniz üç sınıf işlev bulunmaktadır.

1. `addch()` sınıfı: Tek bir karakteri öznitelliklerle yazar
2. `printw()` sınıfı: `printf()`'e benzer biçimlendirilmiş çıktı sağlar
3. `addstr()` sınıfı: Dizgelerin yazar

Bu işlevler birbirleri yerine kullanılabilir. Hangi sınıfı kullandığınız tamamen bir tarz meselesidir.

6.1. `addch()` sınıfı işlevler

Bu işlevler imlecin bulunduğu konuma tek bir karakter yazar ve imleç konumunu ilerletir. Görüntülenecek karakteri işleve verebilirsiniz, fakat genelde bir karakteri belli bir görüntüleme özelliği ile ekrana yazarlar. Görüntüleme özellikleri, belgenin *Öznitelikler* (sayfa: 19) bölümünde ayrıntılı olarak anlatılmıştır. Eğer bir karakter bir öznitelik (kalın, ters video v.b.) ile beraber kullanılırsa, curses karakteri belirtilen öznitelik ile basar.

Bir karakteri belli özniteliklerle bağdaştırmak isterseniz iki seçeneğiniz bulunmaktadır:

- İstenen öznitelik makrosunu tek bir karakter ile VEYA'lamaktır. Bu öznitelikler `ncurses.h` başlık dosyası içerisinde bulunabilir. Örneğin bir *krk* karakterini (`char` türünde) koyu ve altı çizili olarak yazmak istiyorsanız,`addch()`'ı şu şekilde çağırmalısınız:

```
addch (krk | A_BOLD | A_UNDERLINE);
```

- İkinci yöntemse **`attrset()`**, **`attron()`**, **`attroff()`** gibi işlevler kullanmaktır Bu işlevler *Öznitelikler* (sayfa: 19) bölümünde anlatılmıştır. Özetle, o anki pencerenin özniteliklerini değiştirirler. Bir kere ayarlandılar mı, etkisiz kılınana kadar pencereye yazılan karakter bu özniteliklerle ekrana yazılır.

Ek olarak **`curses`**, karakter tabanlı grafikler için bazı özel karakterler sunmaktadır. Tablolar, yatay veya dikey çizgiler, v.b. çizebilirsiniz. Tüm kullanılabilir karakterleri `ncurses.h` başlık dosyasında bulabilirsiniz. Bu dosya içerisindeki `ACS_` ile başlayan makrolara bakınız.

6.2. `mvaddch()`, `waddch()` ve `mvwaddch()`

`mvaddch()`, imleci verilen konuma hareket ettirir ve buraya yazar. Bundan dolayı şu çağrı,

```
    move(satır,sütun);
/*imleci satırıncı satırın sütununcu sütununa hareket ettirir. */
    addch(krk);
```

şununla yer değiştirilebilir:

```
    mvaddch(satır, sütun, krk);
```

`waddch()` işlevi, verilmiş bir pencere içerisine karakter eklemesi dışında **`addch()`** işlevine benzemektedir. (**`addch()`**'ın `stdscr` penceresine bir karakter eklediğini unutmayın)

Benzer şekilde **`mvwaddch()`** işlevi de verilmiş olan koordinatlarda belirtilen pencereye bir karakter eklemek için kullanılır.

Artık temel çıktı işlevi **addch()**'a yabancı değiliz. Fakat, eğer bir dizgeyi yazdırmak istiyorsak onu karakter karakter yazdırmak oldukça sinir bozucu olacaktır. Ne iyi ki, ncurses, **printf** benzeri veya **puts** benzeri işlevler sunmaktadır.

6.3. printw() sınıfı işlevler

Bu işlevler **printf()** işlevine benzemeleri yanında ekran üzerindeki herhangi bir konuma da yazabilmeyi de sağlamaktadırlar.

6.3.1. printw() ve mvprintw()

Bu iki işlev **printf()**'e benzer çalışmaktadır. **mvprintw()** imleci belirtilen konuma hareket ettirir ve buraya yazar. Eğer önce imleci hareket ettirip daha sonra **printw()** kullanarak ekrana yazmak isterseniz önce **move()**'u daha sonra da **printw()**'yi kullanın. Hoş, birinin **mvprintw()** kullanımından kaçınmasında bir mantık göremesem de değiştirme esnekliğiniz bulunmaktadır.

6.3.2. wprintw() ve mvwprintw()

Bu işlevler, parametre olarak verilen pencerelere yazmaları dışında yukarıdakilere benzemektedir.

6.3.3. vwprintw()

Bu işlev **vprintf()**'e benzemektedir. Değişken sayıda argümanla kullanılabilir.

6.3.4. Basit bir printw örneği

Örnek 3. Basit bir printw örneği

```
#include <ncurses.h>       /* stdio.h başlık dosyasını da ekler */
#include <string.h>

int main()
{
 char ileti[]="Sadece bir dizge"; /* ekranda görünecek ileti       */
 int sat, sut;                    /* ekrandaki satır sayısını ve    *
                                   * sütün sayısını tutmak için     */
 initscr();                       /* curses kipine başlama          */
 getmaxyx(stdscr, sat, sut);      /* satır ve sütun sayısını alma   */
 mvprintw(sat/2, (sut-strlen(ileti))/2, "%s", ileti);
                                  /* iletiyi ekranın ortasına yazdır */
 mvprintw(sat-2, 0, "Bu ekranda %d satır ve %d sütun var\n", sat, sut);
 printw("Ekran boyunu degistirerek (eger mümkünse)"
        " programı yeniden çalıstırın");
 refresh();
 getch();
 endwin();

 return 0;
}
```

Yukarıdaki program **printw** kullanımının ne kadar kolay olduğunu göstermektedir. Sadece koordinatları ve göstermek istediğiniz iletiyi belirtmektesiniz ve o da isteğinizi yerine getirmektedir.

Yukarıdaki program bize **getmaxyx()** isimli ncurses.h içerisinde tanımlı yeni bir makroyu tanıtmaktadır. Verilen penceredeki satır ve sütün sayısını hesaplar. **getmaxyx()** bunu kendisine verilen değişkenleri gün-

celleyerek yapar. **getmaxyx()** işlev olmadığından parametre olarak gösterici aktaramayız, sadece iki tamsayı değişken verebiliriz.

6.4. addstr() sınıfı işlevler

addstr() bir karakter dizisini verilen pencereye yazdırmak için kullanılır. Bu işlev her bir karakter için **addch()** işlevini çağırmaya benzer. Bu tüm çıktı işlevleri için geçerlidir. Bu aileden **mvaddstr(), mvwaddstr()** ve **waddstr()** gibi, curses'in isimlendirme kurallarına uyan (örn. mvaddstr(), move() ve daha sonra addstr() çağrılmasıyla aynıdır) işlevler de vardır. Bu ailenin bir başka işlevi ise ek olarak n tam sayı parametresini alan addnstr() işlevidir. Bu işlev ekrana en fazla n karakter yazdırır. Eğer n negatif ise tüm dizge yazdırılır.

6.5. Dikkat edilmesi gereken nokta

Tüm bu işlevler parametre olarak önce y daha sonra da x koordinatını almaktadır. Yeni başlayanların yaptığı en temel hata x, y şeklinde parametre göndermektir. Eğer (y,x) koordinatları üzerinde çok fazla değişiklik yapıyorsanız, ekranı her biriyle ayrı ayrı ilgilenmek için pencerelere bölün. Pencereler *Pencereler* (sayfa: 22) bölümünde açıklanmıştır.

7. Girdi işlevleri

Aslında girdi almadan ekrana yazdırmak sıkıcıdır. Kullanıcıdan girdi almamızı sağlayan işlevlere bir bakalım. Bu işlevler üç bölüme ayrılabilir;

1. getch() sınıfı: Bir karakter alır
2. scanw() sınıfı: Biçimlenmiş girdi alır
3. getstr() sınıfı: Dizge alır

7.1. getch() sınıfı işlevler

Bu işlevler uçbirimden tek bir karakter okur. Fakat gözönüne alınması gereken bir kaç ince nokta vardır. Örneğin, eğer cbreak() işlevini kullanmazsanız, curses girdiğiniz karakterleri art arda değil sadece satırsonu veya dosyasonu karakteri ile karşılaşana kadar okuyacaktır. Bundan kaçınmak için cbreak() işlevi kullanılmalıdır, böylece karakterler anında programınızdan erişilebilir olacaktır. Yaygın olarak kullanılan işlevlerden biri noecho()'dur. İsminden de anlaşılacağı üzere, bu işlev çağrıldığında (kullanıldığında) kullanıcının girdiği karakterler ekranda gösterilmez. cbreak() ve noecho() işlevleri tuş yönetimi üzerindeki en tipik örneklerdir. Bu türden işlevler *Klavye ile etkileşim* (sayfa: 29) bölümünde anlatılmıştır.

7.2. scanw() sınıfı işlevler

Bu işlevler **scanf()** işlevine benzemektedir, ek olarak ekran üzerindeki herhangi bir konumdan veri girişine olanak sağlamaktadırlar.

7.2.1. scanw() ve mvscanw()

Bu işlevlerin kullanımı taranacak satırın **wgetstr()** işlevi ile sağlandığı **sscanf()** kullanımına benzemektedir. Başka deyişle, bu işlevler **wgetstr()** işlevini çağırır (aşağıda anlatılmıştır) ve sonuç satırını taramak için kullanırlar.

7.2.2. wscanw() ve mvwscanw()

Bunlar da parametresiz olarak verilen bir pencereden okumaları dışında yukarıdaki iki işleve benzemektedir.

7.2.3. `vwscanw()`

Bu işlev **`vscanf()`**'ye benzemektedir. Bu, değişken sayıda parametre okunacağı zaman kullanılabilir.

7.3. `getstr()` sınıfı işlevler

Bu işlevler uçbirimden dizge okumak için kullanılır. Aslında bu işlev satırsonu veya satırbaşı veya dosyasonu karakteri görene kadar **`getch()`** işlevinin art arda kullanımıyla aynı işi yapmaktadır. Oluşan karakter dizisi, kullanıcı tarafından tanımlanmış bir karakter göstericisi olan *dizge* tarafından tutulur.

7.4. Bazı örnekler

Örnek 4. Basit bir `scanw` örneği

```
#include <ncurses.h>          /* ncurses.h, stdio.h da ekler */
#include <string.h>

int main()
{
 char ileti[]="Bir dizge giriniz: ";  /* ekranda gözükecek mesaj */
 char dizge[80];
 int sat,sut;                    /* ekranın satır ve sütün sayısını  *
                                  * tutmak için */
 initscr();                      /* curses kipine başlama */
 getmaxyx(stdscr,sat,sut);       /* satır ve sütün satısını elde et */
 mvprintw(sat/2, (sut-strlen(ileti))/2, "%s", ileti);
                                 /* iletiyi ekranın ortasına yaz */
 getstr(dizge);
 mvprintw(LINES - 2, 0, "Şunu girdiniz: %s", dizge);
 getch();
 endwin();

 return 0;
}
```

8. Öznitelikler

Karakterlere özel etkilerle nasıl öznitelikler kazandırılabileceğini daha önce görmüştük. Öznitelikler, özenli ayarlanmaları halinde, verilerin kolay ve anlaşılır bir şekilde basılmasını sağlarlar. Aşağıdaki program bir C dosyasını parametre olarak almakta ve yorum satırlarını koyu olarak göstermektedir. Kodu inceleyin.

Örnek 5. Basit bir öznitelik verme örneği

```
/* pager functionality by Joseph Spainhour" <spainhou (at) bellsouth.net> */
#include <ncurses.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
  int ch, prev, row, col;
  prev = EOF;
  FILE *fp;
```

```
  int y, x;

  if(argc != 2)
  {
    printf("Kullanım: %s <bir dosya adı>\n", argv[0]);
    exit(1);
  }
  fp = fopen(argv[1], "r");
  if(fp == NULL)
  {
    perror("Girdi dosyası açılamadı");
    exit(1);
  }
  initscr();                            /* Curses kipi başlangıcı    */
  getmaxyx(stdscr, row, col);           /* ekran sınırlarını bul     */
  while((ch = fgetc(fp)) != EOF)        /* dosyayı sonuna kadar oku  */
  {
    getyx(stdscr, y, x);                /* o anki imleç konumunu al  */
    if(y == (row - 1))                  /* ekranın sonunda mıyız?    */
    {
      printw("<-Bir tuşa basınız->");
                            /* kullanıcıya bir tuşa basmasını söyle */
      getch();
      clear();                          /* ekranı temizle */
      move(0, 0);                       /* ekranın başına git */
    }
    if(prev == '/' && ch == '*') /* Eğer sadece / ve * karakteri ise
                                  * koyu göstermeyi etkinleştir */
    {
      attron(A_BOLD);                   /* koyu gösterme etkin */
      getyx(stdscr, y, x);              /* imlecin o anki konumunu al */
      move(y, x - 1);                   /* bir karakter geri git */
      printw("%c%c", '/', ch);          /* esas görüntüleme burada yapılmakta */
    }
    else
      printw("%c", ch);
    refresh();
    if(prev == '*' && ch == '/')
      attroff(A_BOLD);                  /* eğer * ve / okunduysa *
                                         * koyu göstermeyi kapat */
    prev = ch;
  }
  endwin();                             /* curses kipi sonu */
  fclose(fp);
  return 0;
}
```

Tüm şu ilklendirme ve ıvır zıvırlardan korkmayın. `While` döngüsüne dikkatinizi verin. Her bir karakteri okuyarak `/*` karakter ikilisini aranmaktadır. Bir kere örüntüyü buldu mu **`attron()`** ile BOLD (koyu gösterme) özelliğini etkin hale getirmektedir. `*/` karakter ikilisi okunduğunda bu öznitelik **`attroff()`** ile kapatılmaktadır.

Yukarıdaki program bize **`getyx()`** ve **`move()`** gibi iki faydalı işlevi tanıtmaktadır. İlk işlev imlecin o anki koordinatlarını `y, x` değerlerine alır. `getyx()` bir makro olduğundan işleve gösterici aktarmak zorunda değiliz. **`move()`** işlevi imleci verilen koordinatlara hareket ettirir.

Yukarıdaki program pek de fazla bir iş yapmayan gerçekten de basit bir örnektir. Bu satırlarda birileri bir C dosyasını okuyup, onu ayrıştırıp değişik renklerde gösteren daha kullanışlı bir program yazabilir. Birisi bunu

başka dillere de genişletebilir.

8.1. Özniteliklerle ilgili ayrıntılar

Özniteliklerin ayrıntılarına inelim. **attron(), attroff(), attrset()** ve onun kardeş işlevleri **attr_get()** v.b gibi işlevler öznitelikleri açıp, özniteliğin ne olduğu bilgisini almada ve renkli bir görüntü sağlamada kullanılabilir.

attron ve attroff işlevleri bir grup özniteliği parametre olarak alabilmekte ve onları sırasıyla açıp kapamaktadır. Aşağıdaki curses.h içerisinde tanımlanmış öznitelikler bu işlevlere gönderilebilir:

```
A_NORMAL        Normal görüntü (parlaklaştırma yok)
A_STANDOUT      Uçbirimin en iyi parlaklaştırma kipi.
A_UNDERLINE     Alt çizgili
A_REVERSE       Ters video
A_BLINK         Yanıp sönme
A_DIM           Yarım parlak
A_BOLD          Daha fazla parlak ve koyu
A_PROTECT       Korumalı kip
A_INVIS         Görünmez veya boş kip
A_ALTCHARSET    İlave karakter kümesi
A_CHARTEXT      Bir karakteri açığa çıkarmak için bit maskesi
COLOR_PAIR(n)   Renk çifti numarası n
```

En sonuncusu en renkli olanıdır :-) ve *Renkler* (sayfa: 27) bölümünde renkler anlatılmıştır.

Yukarıdaki özniteliklerden pekçoğunu VEYA (|)'layarak değişik özellikler elde edebiliriz. Ters video görüntüsü ve yanıp sönme isterseniz şunu kullanabilirsiniz:

```
attron(A_REVERSE | A_BLINK);
```

8.2. attron()'a karşı attrset()

Peki öyleyse attron() ve attrset() arasındaki fark nedir? attrset, pencerenin özniteliklerini ayarlarken attron sadece kendisine verilen özniteliği etkin yapar. Bu yüzden attrset() daha önceki pencere özniteliklerinin üstüne yazar ve yeni öznitelikleri ayarlar. Benzer çekilde attroff() sadece parametre olarak verilen özniteliği etkisizleştirir. Bu bize görünüm özelliklerini rahat şekilde yönetebilme imkanı verir. Fakat bunları dikkatsizce kullanırsanız pencerenin hangi özniteliğe sahip olduğunu takip etmekte zorlanıp görüntünün bozulmasına sebep olabilirsiniz. Bu öznitelikler renkli menüler ve parlaklaştırma üzerinde çalışırken önemlidir. Bu yüzden belli bir yöntemi takip edin ve hep onu kullanın. Her zaman **attrset(A_NORMAL)** komutuna eşit olan **standend()**'i tüm öznitelikleri etkisizleştirmek ve normal görünüme dönmek için kullanabilirsiniz.

8.3. attr_get()

attr_get() işlevi o anki özniteliği ve renk çifti bilgisini alır. Her ne kadar yukarıdaki işlevler kadar sık kullanmasak da, ekranın belli kısımlarını tararken kullanışlıdır. Diyelim ki ekran üzerinde belli bir göncelleme yapmak istiyoruz ve her karakterin hangi özniteliğe sahip olduğunu bilmiyoruz. Bu işlev attrset veya attron işlevlerinden herhangi birisi ile istenen özniteliğin üretilmesi için kullanılabilir.

8.4. attr_ işlevleri

attr_set(), attr_ gibi birçok işlev vardır. Yukarıdakilere benzemekle beraber **attr_t** türünde parametre alırlar.

8.5. `wattr` işlevleri

Yukarıdaki her bir işleve karşılık gelen ve belli pencere üzerinde çalışan 'w"li işlevler de vardır. Yukarıdaki işlevler `stdscr` üzerinde çalışır.

8.6. `chgat()` işlevleri

`chgat()` işlevi, `curs_attr` kılavuz sayfalarının sonunda listelenmiştir. Aslında faydalı bir işlevdir. Belli karakterlere hareket etmeden öznitelikleri kazandırmak için kullanılabilir. İmleci hareket ettirmeden !!! demek istedim `:-)`. O anki imleç konumundan itibaren belirtilen sayıdaki öznitelik değiştirilebilir.

Karakter sayısı olarak –1 vererek satır sonuna kadar güncelleyebiliriz. O anki konumdan satır sonuna kadar öznitelikleri değiştirmek için sadece şunu kullanın:

```
chgat(-1, A_REVERSE, 0, NULL);
```

Bu işlev halihazırda ekranda olan karakterlerin özniteliklerini değiştirmek için faydalıdır. Değiştirmek istediğiniz karaktere konumlanın ve özniteliğini değiştirin.

`wchgat(), mvchgat(), wchgat()` işlevleri de belirtilen pencere üzerinde çalışmaları dışında benzerdir. `mv` işlevleri önce imleci hareket ettirip daha sonra kendilerine verilen işi yapar. Aslında `chgat`, `wchgat()` işlevinin pencere parametresi olarak `stdscr`'yi aldığı duruma bir makrodur. Pekçok "`w`'siz" işlev aslında birer makrodur.

Örnek 6. `chgat()` Kullanım Örneği

```
#include <ncurses.h>

int main(int argc, char *argv[])
{ initscr();                      /* Curses kipine başlama  */
  start_color();                  /* Renk özelliği etkin    */

  init_pair(1, COLOR_CYAN, COLOR_BLACK);
  printw("Yazmaya üsendigim oldukça uzun bir dizge ");
  mvchgat(0, 0, -1, A_BLINK, 1, NULL);
  /*
    * İlk iki parametre başlangıç konumunu
    * üçüncü parametre güncellenecek karakter sayısını belirtir.
    * -1 satır sonuna kadar demektir.
    * Dördüncü parametre karaktere kazandırmak istediğiniz özniteliktir
    * Beşincisi renk indisidir. init_pair() sırasında verilen indistir.
    * Renk istemiyorsanız 0 kullanın.
    * Altıncısı her zaman NULL değerdir.
    */
  refresh();
  getch();
  endwin();                       /* Curses kipi sonu  */
  return 0;
}
```

Bu örnek aynı zamanda curses dünyasının renk kullanımını da göstermektedir. Renkler konusu *daha sonra* (sayfa: 27) ayrıntılı olarak açıklanacaktır. Renkli görüntü istemiyorsanız 0 kullanın.

9. Pencereler

Pencereler curses kütüphanesinde en önemli konuyu oluşturmaktadır. Tüm işlevlerin gizli şekilde üzerinde çalıştıkları standart `stdscr` penceresini bu ana kadar gördünüz. Basit bir kullanıcı arayüzü tasarlamak için bile, pencereler kullanmalısınız. Pencere kullanmanızın temel sebebi, daha iyi bir tasarım için güncellenmesi gereken pencereleri değiştirerek, daha verimli bir çalışma için ekranın belli kısımlarını bağımsız olarak değiştirmektir. Son söylediğim pencereler ile ilgili en önemli noktadır. Her zaman programınızda kolay yönetilebilir ve daha iyi tasarımlara yönelmelisiniz. Eğer büyük ve karmaşık kullanıcı arayüzleri yazıyorsanız, bu yaklaşım, herhangi bir iş yapmadan önce hayati öneme sahip bir noktadır.

9.1. Temel bilgiler

Bir pencere **`newwin()`** işlevini çağırarak oluşturulabilir. Ekran üzerinde aslında birşey oluşturmaz. Pencereyi değiştirecek yapılar için bellekten yer ayırır ve pencerenin boyutu, başladığı y koordinatı, başladığı x koordinatı, v.b bilgilerle ilgili yapıları günceller. Böylece curses kullanımında pencere kavramı, ekranın diğer kısımlarından bağımsız olarak değiştirilebilen hayali bir soyutlamadır. `newwin()` işlevi, `wprintw()` ve v.b. pencere ilişkili işlevlerde kullanılabilecek WINDOW veri yapısı türünden bir gösterici döndürür. Son olarak pencere `delwin()` işlevi ile yok edilebilir. Bu işlev pencere yapısıyla bellekte ayrılmış alanı geri verecektir.

9.2. Bana bir Pencere göster !!!

Eğer bir pencere oluşturuldu ve biz de göremiyorsak bu komik bir durumdur. Eğlenceli kısım pencereyi göstermekle başlar. **`box()`** işlevi pencere etrafına bir çerçeve çizmek için kullanılır. Bu işlevleri aşağıdaki örnekte daha ayrıntılı inceleyelim.

Örnek 7. Pencere Çerçevesi örneği

```
#include <ncurses.h>

WINDOW *create_newwin(int height, int width, int starty, int startx);
void destroy_win(WINDOW *local_win);

int main(int argc, char *argv[])
{ WINDOW *my_win;
  int startx, starty, width, height;
  int ch;

  initscr();                    /* Curses kipini ilklendirme    */
  cbreak();                     /* Satır önbelleği etkisiz,     *
                                 * herşeyi bana gönder          */
  keypad(stdscr, TRUE);         /* Şu şeker F1'e ihtiyacım var  */

  height = 3;
  width = 10;
  starty = (LINES - height) / 2;  /* Pencerenin ortasına yazmak için */
  startx = (COLS - width) / 2;    /* hesaplama yapılıyor             */
  printw("Press F1 to exit");
  refresh();
  my_win = create_newwin(height, width, starty, startx);

  while((ch = getch()) != KEY_F(1))
  { switch(ch)
    { case KEY_LEFT:
        destroy_win(my_win);
        my_win = create_newwin(height, width, starty,--startx);
        break;
```

```
      case KEY_RIGHT:
        destroy_win(my_win);
        my_win = create_newwin(height, width, starty,++startx);
        break;
      case KEY_UP:
        destroy_win(my_win);
        my_win = create_newwin(height, width, --starty,startx);
        break;
      case KEY_DOWN:
        destroy_win(my_win);
        my_win = create_newwin(height, width, ++starty,startx);
        break;
    }
  }

  endwin();                     /* Curses kipi sonu */
  return 0;
}

WINDOW *create_newwin(int height, int width, int starty, int startx)
{ WINDOW *local_win;

  local_win = newwin(height, width, starty, startx);
  box(local_win, 0 , 0);          /* 0, 0 yatay ve düşey çizgiler    *
                                   * için öntanımlı karakteri verir  */
  wrefresh(local_win);            /* Kutuyu gösterir                 */

  return local_win;
}

void destroy_win(WINDOW *local_win)
{
  /* box(local_win, ' ', ' '); : Bu istenen pencere silme sonucunu
    * vermeyecektir. Pencerenin dört köşesi ve
    * çirkin küçük bir kısmı kalacaktır.
    */
  wborder(local_win, ' ', ' ', ' ',' ',' ',' ',' ',' ');
  /* Alınan parametreler şu şekildedir
    * 1. win: üzerinde çalışılacak pencere
    * 2. ls: pencerenin sol kenarı için kullanılacak karakter
    * 3. rs: pencerenin sağ kenarı için kullanılacak karakter
    * 4. ts: pencerenin üst kenarı için kullanılacak karakter
    * 5. bs: pencerenin alt kenarı için kullanılacak karakter
    * 6. tl: pencerenin sol üst köşesi için kullanılacak karakter
    * 7. tr: pencerenin sağ üst köşesi için kullanılacak karakter
    * 8. bl: pencerenin sol alt köşesi için kullanılacak karakter
    * 9. br: pencerenin sağ alt köşesi için kullanılacak karakter
    */
  wrefresh(local_win);
  delwin(local_win);
}
```

9.3. Örneğin açıklaması

Çığlık atmayın. Biliyorum büyük bir örnek. Fakat bazı önemli noktaları burada anlatmalıyım :–). Bu program sol, sağ, üst ve aşağı ok tuşlarıyla hareket eden dikdörtgen bir pencere oluşturur. Ardışıl olarak kullanıcının her tuşa

basışında pencereyi yıkıp yeniden oluşturur. Ekran sınırlarının ötesine gitmeyin. Bu sınırların kontrol edilmesi okuyucuya bir örnek olarak bırakılmıştır. Şimdi satır satır inceleyelim.

create_newwin() işlevi **newwin()** ile yeni bir pencere oluşturur ve etrafına box() işlevi ile bir çerçeve çizer. **destroy_win()** işlevi ilk önce ' ' karakteri ile bir çerçeve oluşturarak pencereyi ekrandan siler ve daha sonra **delwin()**'i çağırarak onunla ilişkili belleği geri verir. Kullanıcının bastığı tuşa bağlı olarak, *starty* ve *startx* değişir ve yeni bir pencere oluşturulur.

destroy_win içerisinde gördüğünüz gibi, wborder yerine box işlevini kullandım. Nedeni açıklama satırlarında yazılı (Atladınız. Biliyorum. Kodu okuyun :-)). wborder, belirtilen 4 köşe ve 4 çizgi parametresiyle pencere etrafına bir çerçeve çizer. Daha açık hale getirmek için, eğer wborder'ı aşağıdaki şekilde çağırırsanız:

```
wborder(win, '|', '|', '-', '-', '+', '+', '+', '+');
```

şöyle bir şey üretir:

```
  +-------------+
  |             |
  |             |
  |             |
  |             |
  |             |
  |             |
  +-------------+
```

9.4. Örnekteki diğer kısımlar

Yukarıdaki örnekte aynı zamanda initscr() kullanımından sonra ekran boyutlarına ilklendirilen *COLS* ve *LINES* değişkenlerini kullandığım gözükmektedir. Ekran boyutlarının bulunmasında ve yukarıdaki gibi ekranın ortasının koordinatlarının bulunmasında faydalı olabilir. **getch()** bilindiği gibi klavyeden bir tuş bilgisi alır ve basılan tuşa karşılık gelen işi yapar. Bu şekildeki switch- case kullanımı kullanıcı grafik arayüzü programlamada çok yaygındır.

9.5. Diğer Çerçeve işlevleri

Yukarıdaki program, her tuşa basımda bir pencerenin yıkılıp yenisinin oluşturulmasından dolayı oldukça verimsizdir. Bunun için diğer çerçeve çizimi ile ilgili işlevleri kullanarak daha verimli bir program yazalım.

Aşağıdaki programda **mvhline()** ve **mvvline()** kullanarak benzer etki sağlanmıştır. Bu iki işlev basittir. Belirtilen boyda ve belirtilen konumda yatay ve düşey çizgiler oluştururlar.

Örnek 8. Daha çok çerçeve işlevi

```
#include <ncurses.h>

typedef struct _win_border_struct {
  chtype  ls, rs, ts, bs,
          tl, tr, bl, br;
}WIN_BORDER;

typedef struct _WIN_struct {
  int startx, starty;
  int height, width;
  WIN_BORDER border;
```

```
}WIN;

void init_win_params(WIN *p_win);
void print_win_params(WIN *p_win);
void create_box(WIN *win, bool flag);

int main(int argc, char *argv[])
{ WIN win;
  int ch;

  initscr();                    /* curses ilklendirme           */
  start_color();                /* Renk özelliğini başlat       */
  cbreak();                     /* Satır önbelleği etkisiz,     *
                                 * herşeyi bana gönder          */
  keypad(stdscr, TRUE);         /* Şu şeker F1'e ihtiyacım var  */
  noecho();
  init_pair(1, COLOR_CYAN, COLOR_BLACK);

  /* Pencere parametrelerini ilklendir */
  init_win_params(&win);
  print_win_params(&win);

  attron(COLOR_PAIR(1));
  printw("Çıkmak için F1'e basınız");
  refresh();
  attroff(COLOR_PAIR(1));

  create_box(&win, TRUE);
  while((ch = getch()) != KEY_F(1))
  { switch(ch)
    { case KEY_LEFT:
        create_box(&win, FALSE);
        --win.startx;
        create_box(&win, TRUE);
        break;
      case KEY_RIGHT:
        create_box(&win, FALSE);
        ++win.startx;
        create_box(&win, TRUE);
        break;
      case KEY_UP:
        create_box(&win, FALSE);
        --win.starty;
        create_box(&win, TRUE);
        break;
      case KEY_DOWN:
        create_box(&win, FALSE);
        ++win.starty;
        create_box(&win, TRUE);
        break;
    }
  }
  endwin();                     /* Curses kipi sonu */
  return 0;
}
void init_win_params(WIN *p_win)
{
```

```
    p_win->height = 3;
    p_win->width = 10;
    p_win->starty = (LINES - p_win->height)/2;
    p_win->startx = (COLS - p_win->width)/2;

    p_win->border.ls = '|';
    p_win->border.rs = '|';
    p_win->border.ts = '-';
    p_win->border.bs = '-';
    p_win->border.tl = '+';
    p_win->border.tr = '+';
    p_win->border.bl = '+';
    p_win->border.br = '+';
}
void print_win_params(WIN *p_win)
{
#ifdef _DEBUG
  mvprintw(25, 0, "%d %d %d %d", p_win->startx, p_win->starty,
           p_win->width, p_win->height);
  refresh();
#endif
}
void create_box(WIN *p_win, bool flag)
{ int i, j;
  int x, y, w, h;

  x = p_win->startx;
  y = p_win->starty;
  w = p_win->width;
  h = p_win->height;

  if(flag == TRUE)
  { mvaddch(y, x, p_win->border.tl);
    mvaddch(y, x + w, p_win->border.tr);
    mvaddch(y + h, x, p_win->border.bl);
    mvaddch(y + h, x + w, p_win->border.br);
    mvhline(y, x + 1, p_win->border.ts, w - 1);
    mvhline(y + h, x + 1, p_win->border.bs, w - 1);
    mvvline(y + 1, x, p_win->border.ls, h - 1);
    mvvline(y + 1, x + w, p_win->border.rs, h - 1);

  }
  else
    for(j = y; j <= y + h; ++j)
      for(i = x; i <= x + w; ++i)
        mvaddch(j, i, ' ');

  refresh();
}
```

10. Renkler

10.1. Temel bilgiler

Hayat renkler olmadan çok donuk gözükürdü. Curses kütüphanesi renkler üzerinde işlem yapmak için hoş bir

mekanizmaya sahiptir. Basit bir program ile işlerin ayrıntılarına inelim.

Örnek 9. Basit Bir Renk Örneği

```
#include <ncurses.h>

void print_in_middle(WINDOW *win, int starty, int startx,
                     int width, char *string);
int main(int argc, char *argv[])
{ initscr();                        /* curses kipi ilklendirmesi */
  if(has_colors() == FALSE)
  { endwin();
    printf("Uçbiriminiz renkleri desteklemiyor\n");
    exit(1);
  }
  start_color();                    /* Renk başlat */
  init_pair(1, COLOR_RED, COLOR_BLACK);

  attron(COLOR_PAIR(1));
  print_in_middle(stdscr, LINES / 2, 0, 0, "Keman !!! Renkli ...");
  attroff(COLOR_PAIR(1));
  getch();
  endwin();
}
void print_in_middle(WINDOW *win, int starty, int startx,
                     int width, char *string)
{ int length, x, y;
  float temp;

  if(win == NULL)
    win = stdscr;
  getyx(win, y, x);
  if(startx != 0)
    x = startx;
  if(starty != 0)
    y = starty;
  if(width == 0)
    width = 80;

  length = strlen(string);
  temp = (width - length)/ 2;
  x = startx + (int)temp;
  mvwprintw(win, y, x, "%s", string);
  refresh();
}
```

Görüdüğünüz üzere renkleri kullanabilmeniz için öncelikle **start_color()** işlevini çağırmalısınız. Bundan sonra değişik işlevler kullanarak uçbiriminizin renk yeteneklerini kullanabilirsiniz. Bir uçbirimin renk özniteliklerinin olup olmadığını anlamak için **has_colors()** işlevini kullanabilirsiniz. Eğer FALSE dönerse, uçbirimin renk desteği yoktur.

start_color() işlevi çağrıldığı anda Curses uçbirim tarafından desteklenen tüm renkleri ilklendirir. Bu renklere **COLOR_BLACK** v.b. gibi tanımlanmış sabitlerle erişilebilir. Aslında renkleri tam olarak kullanabilmek için çiftleri tanımlamalısınız. Renkler daima çiftler halinde kullanılır. Bunun anlamı **init_pair()** işlevini kullanarak artalan ve karakter rengi için bir renk çifti belirtmeniz demektir. Bundan sonra bu renk çifti **COLOR_PAIR()** ile normal bir öznitelik olarak kullanılabilir. İlk başta karmaşık gelebilir. Fakat bu zeki yaklaşım renk çiftleri üz-

erinde rahatça işlem yapabilmemizi sağlar. Kullanımına hayran olmak için, kabuk betiklerinden etkileşimli iletiler çıkaran "dialog" uygulamasının kaynak koduna bakmalısınız. Geliştiriciler ihtiyaç olabilecek her renk için artalan ve karakter rengi türevlerini tanımlamışlar ve başlangıçta ilklendirmişlerdir. Bu da halihazırda sabit olarak tanımlanmış çiftlere erişip görünüm özelliklerini ayarlamada kolaylık sağlamaktadır.

Aşağıdaki renkler `curses.h` başlık dosyasında tanımlanmıştır. Bunları pekçok renk işlevi için parametre olarak kullanabilirsiniz.

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

10.2. Renk Tanımlamalarını Değiştirmek

`init_color()` işlevi curses kipi ilklendirildiğindeki rgb değerlerini değiştirmek için kullanılabilir. Diyelim ki kırmızı rengin yoğunluğunu biraz açmak istiyorsunuz. O zaman bu işlevi şu şekilde kullanabilirsiniz:

```
init_color(COLOR_RED, 700, 0, 0);
  /* 1. parametre          : renk adı
   * 2, 3, 4. parametreler: rgb içeriği; 0 ile 1000 arasında bir değer */
```

Eğer kullandığınız uçbirim renk tanımlamalarını değiştiremiyorsa işlev `ERR` ile geri döner. **`can_change_color()`** işlevi uçbirimin renk içeriğini değiştirme yeteneği olup olmadığını öğrenmek için kullanılabilir. RGB içeriği 0 dan 1000'e kadar ölçeklenmiştir. Kırmızı renk 1000(r), 0(g), 0(b) ile tanımlanmıştır.

10.3. Renk İçeriği

`color_content()` ve **`pair_content()`** işlevleri renk içeriğini bulmak ve renk çifti için ön ve art alan bilgisini almak için kullanılabilir.

11. Klavye ile etkileşim

11.1. Temel bilgiler

Hiçbir grafik kullanıcı arabirimi güçlü bir arayüze sahip olmadan tamamlanmış sayılmaz ve kullanıcı ile etkileşimde olabilmek için bir curses programı tuş basımına veya fare hareketlerine duyarlı olmalıdır. Önce tuşlarla ilgilenelim.

Önceki hemen her örnekte gördüğünüz gibi kullanıcıdan klavye girdisi almak çok kolaydır. Tuş basımlarını almanın en basit yolu **`getch()`** işlevidir. Satır karakterlerinin tamamı yerine (genelde satır sonu karakteri ile biter) tek bir karakter okunmasıyla ilgilenirseniz `cbreak` kipi etkin hale getirilmelidir. İşlev, ok v.b. tuşlarını kullanabilmek için tuş takımı etkin hale getirilmelidir. Ayrıntılar için *İlklendirme* (sayfa: 13) bölümüne bakınız.

`getch()` komutu basılan tuşa uygun olarak bir tam sayı döndürür. Eğer normal bir karakterse tam sayı değeri karaktere eşit olacaktır. Aksi halde `curses.h` başlık dosyasında sabit olarak tanımlanmış bir sayı ile döner. Örneğin kullanıcı F1'e basarsa geri dönen tam sayı değeri 265'tir. Bu, `curses.h` içerisinde tanımlanmış olan `KEY_F()` makrosu kullanılarak kontrol edilebilir. Bu tuşları okumayı ve taşınabilmelerini kolay ve üzerinde işlem yapılabilir hale getirmektedir.

Örneğin `getch()`'ı şu şekilde çağırırsanız,

```
  int ch;

  ch = getch();
```

`getch()` kullanıcıdan bir tuşa basmasını bekleyecek (eğer belli bir zaman aşımı tanımlamadıysanız) ve kullanıcı tuşa bastığı anda karşılık gelen tamsayıyı döndürülecektir. Daha sonra dönen değeri `curses.h` içerisinde tanımlanmış sabitlerle kıyaslayarak istediğiniz tuşlarla eşleşmeyi sağlayabilirsiniz.

Aşağıdaki kod parçası bu işi yapacaktır.

```
  if(ch == KEY_LEFT)
    printw("Sola ok tusuna basıldı\n");
```

Yukarı ve aşağı ok tuşlarıyla hareket edilebilen bir menü yazalım.

11.2. Basit bir tuş kullanım örneği

Örnek 10. Basit bir tuş kullanım örneği

```
#include <stdio.h>
#include <ncurses.h>

#define WIDTH 30
#define HEIGHT 10

int startx = 0;
int starty = 0;

char *choices[] = {
  "Choice 1",
  "Choice 2",
  "Choice 3",
  "Choice 4",
  "Exit",
};
int n_choices = sizeof(choices) / sizeof(char *);
void print_menu(WINDOW *menu_win, int highlight);

int main()
{ WINDOW *menu_win;
  int highlight = 1;
  int choice = 0;
  int c;

  initscr();
  clear();
  noecho();
  cbreak();       /* Satır ön belleği etkisiz. Herşeyi bana gönder */
  startx = (80 - WIDTH) / 2;
  starty = (24 - HEIGHT) / 2;

  menu_win = newwin(HEIGHT, WIDTH, starty, startx);
  keypad(menu_win, TRUE);
  mvprintw(0, 0, "Yukarı ve asagı için ok tuslarını kullanın, "
          "Seçmek için enterı kullanın");
  refresh();
```

```
  print_menu(menu_win, highlight);
  while(1)
  { c = wgetch(menu_win);
    switch(c)
    { case KEY_UP:
        if(highlight == 1)
          highlight = n_choices;
        else
          --highlight;
        break;
      case KEY_DOWN:
        if(highlight == n_choices)
          highlight = 1;
        else
          ++highlight;
        break;
      case 10:
        choice = highlight;
        break;
      default:
        mvprintw(24, 0,
          "Basılan karakter = %3d Muhtemelen şu şekilde de yazılabilir '%c'",
          c, c);
        refresh();
        break;
    }
    print_menu(menu_win, highlight);
    if(choice != 0) /* Sonsuz döngüden çıkmak için kullanıcı bir seçim yapar */
      break;
  }
  mvprintw(23, 0, "%d'yi %s ile seçtiniz\n",
           choice, choices[choice - 1]);
  clrtoeol();
  refresh();
  endwin();
  return 0;
}


void print_menu(WINDOW *menu_win, int highlight)
{
  int x, y, i;

  x = 2;
  y = 2;
  box(menu_win, 0, 0);
  for(i = 0; i < n_choices; ++i)
  { if(highlight == i + 1) /* O an seçileni parlaklaştır */
    { wattron(menu_win, A_REVERSE);
      mvwprintw(menu_win, y, x, "%s", choices[i]);
      wattroff(menu_win, A_REVERSE);
    }
    else
      mvwprintw(menu_win, y, x, "%s", choices[i]);
    ++y;
  }
  wrefresh(menu_win);
```

```
}
```

12. Fare ile Etkileşim

Artık tuşlar ile etkileşimi biliyorsunuz, şimdi aynısını fare ile yapalım. Genelde her kullanıcı arayüzü hem klavye hem de fare ile kullanıcının etkileşimine izin vermektedir.

12.1. Temel bilgiler

Herhangi bir şey yapmadan önce almak istediğiniz olaylar **mousemask()** ile etkin hale getirilmelidir.

```
mousemask(  mmask_t newmask,     /* Dinlemek istediğiniz olaylar */
            mmask_t *oldmask)    /* Eski olay maskesi            */
```

Yukarıdaki işleve ilk parametre dinlemek istediğiniz olaylara bir bit maskesidir. Öntanımlı olarak her olay etkisizleştirilmiştir. **ALL_MOUSE_EVENTS** bit maskesi tüm olayları almak için kullanılabilir.

Aşağıdakiler tüm olay maskeleridir.

İsim	Açıklama
BUTTON1_PRESSED	fare tuşu 1 basılı
BUTTON1_RELEASED	fare tuşu 1 serbest
BUTTON1_CLICKED	fare tuşu 1 tıklandı
BUTTON1_DOUBLE_CLICKED	fare tuşu 1 iki kere tıklandı
BUTTON1_TRIPLE_CLICKED	fare tuşu 1 üç kere tıklandı
BUTTON2_PRESSED	fare tuşu 2 basılı
BUTTON2_RELEASED	fare tuşu 2 serbest
BUTTON2_CLICKED	fare tuşu 2 tıklandı
BUTTON2_DOUBLE_CLICKED	fare tuşu 2 iki kere tıklandı
BUTTON2_TRIPLE_CLICKED	fare tuşu 2 üç kere tıklandı
BUTTON3_PRESSED	fare tuşu 3 basılı
BUTTON3_RELEASED	fare tuşu 3 serbest
BUTTON3_CLICKED	fare tuşu 3 tıklandı
BUTTON3_DOUBLE_CLICKED	fare tuşu 3 iki kere tıklandı
BUTTON3_TRIPLE_CLICKED	fare tuşu 3 üç kere tıklandı
BUTTON4_PRESSED	fare tuşu 4 basılı
BUTTON4_RELEASED	fare tuşu 4 serbest
BUTTON4_CLICKED	fare tuşu 4 tıklandı
BUTTON4_DOUBLE_CLICKED	fare tulu 4 iki kere tıklandı
BUTTON4_TRIPLE_CLICKED	fare tuşu 4 üç kere tıklandı
BUTTON_SHIFT	tuş durumu değiştiğinde shift'e basılıydı
BUTTON_CTRL	tuş durumu değiştiğinde control'e basılıydı
BUTTON_ALT	tuş durumu değiştiğinde alt'a basılıydı
ALL_MOUSE_EVENTS	tüm tuş durumu değişikliklerini bildir
REPORT_MOUSE_POSITION	fare hareketini bildir

12.2. Olayları yakalamak

Bir kere bir fare sınıfı olayları etkin hale getirildi mi, `getch()` türü işlevler her fare olayı olduğunda `KEY_MOUSE` döndürürler. Daha sonra da fare olayı **getmouse()** ile alınabilir.

Kod yaklaşık olarak şuna benzer:

```
  MEVENT event;

  ch = getch();
  if(ch == KEY_MOUSE)
    if(getmouse(&event) == OK)
      /* Olay ile ilgili bir şeyler yap */
      .
      .
```

`getmouse()` olayı kendisine verilen göstericide döndürür. Göstericinin gösterdiği değer aşağıdakileri içeren bir yapıdır:

```
typedef struct
{
  short id;         /* Pekçok aygıtı tanımlamak için ID */
  int x, y, z;      /* olay koordinatları */
  mmask_t bstate;   /* tuş durumu bitleri */
}
```

`bstate` ilgilendiğimiz temel değişkendir. Farenin tuş durumu bilgisini verir.

Aşağıdakine benzer bir kodla ne olduğunu bulabiliriz.

```
  if(event.bstate & BUTTON1_PRESSED)
    printw("Sol Tuş Basıldı");
```

12.3. Hepsini Bir Araya Getirelim

Fare ile etkileşmek için bunlar oldukça fazla. Aynı menüyü oluşturup fare etkileşimini etkin hale getirelim. İşleri basit hale getirmek için tuş etkileşimi kaldırılmıştır.

Örnek 11. Menüye fare ile erişim !!!

```
#include <ncurses.h>

#define WIDTH 30
#define HEIGHT 10

int startx = 0;
int starty = 0;

char *choices[] = {
  "Choice 1",
  "Choice 2",
  "Choice 3",
  "Choice 4",
  "Exit",
};

int n_choices = sizeof(choices) / sizeof(char *);

void print_menu(WINDOW *menu_win, int highlight);
void report_choice(int mouse_x, int mouse_y, int *p_choice);

int main()
```

```
{ int c, choice = 0;
  WINDOW *menu_win;
  MEVENT event;

  /* curses kipini etkinleştirelim */
  initscr();
  clear();
  noecho();
  cbreak();       //Satır önbelleği etkisiz. Herşeyi gönder

  /* Pencereyi ekranın ortasına koymaya çalışıyoruz */
  startx = (80 - WIDTH) / 2;
  starty = (24 - HEIGHT) / 2;

  attron(A_REVERSE);
  mvprintw(23, 1, "Çıkmak için exit'e tıklayın "
          "(Sanal konsolda iyi çalışır)");
  refresh();
  attroff(A_REVERSE);

  /* Menüyü ilk olarak oluştur*/
  menu_win = newwin(HEIGHT, WIDTH, starty, startx);
  print_menu(menu_win, 1);
  /* Tüm fare olaylarını yakala*/
  mousemask(ALL_MOUSE_EVENTS, NULL);

  while(1)
  { c = wgetch(menu_win);
    switch(c)
    { case KEY_MOUSE:
        if(getmouse( &event) == OK)
        { /* sol fare tuşuna basılınca */
          if(event.bstate  & BUTTON1_PRESSED)
          { report_choice(event.x + 1, event.y + 1,  &choice);
            if(choice == -1) //Çıkış seçildi
              goto end;
            mvprintw(22, 1, "Yapılan seçim : %d Seçilen \"%10s\"",
                     choice, choices[choice - 1]);
            refresh();
          }
        }
        print_menu(menu_win, choice);
        break;
    }
  }
end:
  endwin();
  return 0;
}

void print_menu(WINDOW *menu_win, int highlight)
{
  int x, y, i;

  x = 2;
  y = 2;
  box(menu_win, 0, 0);
```

```
  for(i = 0; i < n_choices; ++i)
  { if(highlight == i + 1)
    { wattron(menu_win, A_REVERSE);
      mvwprintw(menu_win, y, x, "%s", choices[i]);
      wattroff(menu_win, A_REVERSE);
    }
    else
      mvwprintw(menu_win, y, x, "%s", choices[i]);
    ++y;
  }
  wrefresh(menu_win);
}

/* Fare konumuna göre seçimi bildir */
void report_choice(int mouse_x, int mouse_y, int *p_choice)
{ int i,j, choice;

  i = startx + 2;
  j = starty + 3;

  for(choice = 0; choice < n_choices; ++choice)
    if(mouse_y == j + choice
       && mouse_x >= i
       && mouse_x <= i + strlen(choices[choice]))
    { if(choice == n_choices - 1)
        *p_choice = -1;
      else
        *p_choice = choice + 1;
      break;
    }
}
```

12.4. Çeşitli İşlevler

`mouse_trafo()` ve `wmouse_trafo()` işlevleri fare koordinatlarını ekran koordinatlarına dönüştürmek için kullanılabilir. curs_mouse(3X) kılavuz sayfasını ayrıntılar için inceleyiniz.

`mouseinterval` işlevi tıklamaları tanımlayabilmek için tuşa basma ile çekme arasındaki en büyük zamanı saniyenin binde biri cinsinden ayarlar. Bu işlev önceki tıklama süresini döndürür. Öntanımlı değer saniyenin beşte biridir.

13. Ekran Düzenleme

Bu bölümde ekranı verimli olarak yönetmemizi sağlayan bazı işlevlere bakacağız ve bazı güzel programlar yazacağız. Bu özellikle oyun yazımında önemlidir.

13.1. `getyx()` işlevleri

`getyx()` işlevi o anki imleç koordinatlarını bulmak için kullanılır. x ve y koordinat değerlerini kendisine verilmiş değişkenlere aktaracaktır. `getyx()` bir makro olduğundan değişkenlerin adreslerini göndermenize gerek yoktur. Şu şekilde çağrılabilir:

```
  getyx(win, y, x);
  /* win: pencere gösterici
    *   y, x: kooridatlar bu değişkenlere aktarılır
```

```
    */
```

`getparyx()` işlevi içteki pencerenin başlangıç koordinatlarını dıştakine göre döndürür. Bu bazen içteki pencereyi güncellemek gerektiğinde gereklidir. Pekçok menüden oluşan güzel bir uygulama tasarımında, menü konumlarını, ilk seçenek koordinatlarını, v.b. kaydetmek zor bir hal alır. Buna basit bir yöntem alt pencereler içerisinde menüler oluşturmak ve `getparyx()` kullanarak menülerin başlangıç koordinatlarını bulmaktır.

`getbegyx()` ve `getmaxyx()` işlevleri o anki pencerenin başlangıç ve en büyük koordinatlarını tutar. Bu işlevler yukarıdakine benzer şekilde pencere ve iç pencerelerde çalışırken faydalıdır.

13.2. Ekran dökümünün alınması

Oyun yazarken bazen ekran durumunun kaydedilmesi ve aynı duruma tekrar dönülmesi gerekmektedir. `scr_dump()` işlevi kendisine parametre olarak verilen dosyaya ekranın o anki konumuyla ilgili bilgileri yazmaktadır. Daha sonra `scr_restore` işlevi ile ekran durumu geri yüklenebilir. Bu iki basit işlev değişen senaryolara sahip hızlı bir oyunda verimli bir şekilde kullanılabilir.

13.3. Pencere dökümünün alınması

Pencereleri kaydetmek ve geri yüklemek için **`putwin()`** ve **`getwin()`** işlevleri kullanılabilir. **`putwin()`**, daha sonra **`getwin()`** tarafından geri yüklenebilen pencere durumunu bir dosyaya yazar.

`copywin()` işlevi bir pencere durumunu tamamen diğer bir pencereye kopyalamak için kullanılır. Parametre olarak kaynak ve hedef pencerelerini alır, tanımlanan dikdörtgene göre dikdörtgen bölgeyi kaynaktan hedef pencereye doğru kopyalar. Son parametresi hedef pencere üzerine yazılma mı yoksa diğer durum bilgileriyle beraber bulunma mı olduğunu belirtir. Eğer parametre mantıksal olarak doğru ise, kopyalama işlemi zarar verici değildir.

14. Çeşitli Özellikler

Artık, ıslık ve zil sesleri çıkartan güzel bir program yazacak yeterli sayıda özellik biliyorsunuz. Değişik durumlarda kullanışlı olabilecek çeşitli bazı işlevler vardır. Bunların bazılarını ele alalım.

14.1. `curs_set()`

Bu işlev imleci görünmez yapmak için kullanılır. Bu işleve gönderilebilecek parametreler şunlardır:

```
0 : görünmez
1 : normal
2 : çok görünür
```

14.2. Curses Kipini Geçici Olarak Terk Etmek

Bazen ilk işlemleri gerçekleştirdiğiniz kipe (satır tamponlu kip) geçici bir süre için geri dönmek isteyebilirsiniz. Böylesi bir durumda ilk olarak tty kiplerini **`def_prog_mode()`** çağrısı ile kaydetmeniz ve curses kipini sonlandırmak için **`endwin()`** işlevini kullanmanız gerekmektedir. Bu sizi asıl tty kipine gönderecektir. İşiniz bitip tty kipine dönmek için **`reset_prog_mode()`** işlevini çağırın. Bu işlev tty kipini **`def_prog_mode()`** ile kaydedilmiş kipe dönüştürür. Daha sonra `refresh()`'i kullanın ve işte yine curses kipindesiniz. Aşağıda yapılan işlemlerin sırasını gösteren bir örnek bulunmaktadır.

Örnek 12. Geçici Olarak Curses Kipini Terk Etmek

```
#include <ncurses.h>

int main()
{
  initscr();                          /* Curses kipi başlangıcı */
  printw("Merhaba Dünya !!!\n");      /* Ekrana Merhaba Dünya yaz */
  refresh();                          /* Gerçek ekranda bunu yaz */
  def_prog_mode();                    /* tty kiplerini kaydet */
  endwin();                           /* Curses kipini geçici olarak sonlandır */
  system("/bin/sh");                  /* Kabukta ne yapmak istiyorsanız yapın */
  reset_prog_mode();                  /* Bir önceki tty kipine dönüş */
  refresh();                          /* Ekran içeriğini geri yükleyelim */
  printw("Tekrar Merhaba\n");         /* Curses'a döndük mü bakalım */
  refresh();                          /* Ekranı tampondan tazeleyelim */
  endwin();                           /* Curses kipinin sonu */

  return 0;
}
```

14.3. ACS_ değişkenleri

Eğer DOS'ta programlama yaptıysanız genişletilmiş karakter kümesi içerisindeki şu güzel görünümlü olanları biliyorsunuzdur. Sadece bazı uçbirimlerde gösterilirler. NCURSES'ün **box()** gibi işlevleri bu karakterleri kullanır. Tüm bu karakterler seçimlik karakter kümesi anlamına gelen ACS (alternative character set) ile başlar. Yukarıdaki programların bazılarında bu karakterleri kullandığımı farketmiş olabilirsiniz. Aşağıda tüm karakterleri gösteren bir örnek vardır.

Örnek 13. ACS_ değişkenleri örneği

```
#include <ncurses.h>

int main()
{
  initscr();

  printw("Sol üst köşe         "); addch(ACS_ULCORNER); printw("\n");
  printw("Sol alt köşe         "); addch(ACS_LLCORNER); printw("\n");
  printw("Sağ alt köşe         "); addch(ACS_LRCORNER); printw("\n");
  printw("Sağı dönük T         "); addch(ACS_LTEE); printw("\n");
  printw("Sola dönük T         "); addch(ACS_RTEE); printw("\n");
  printw("Yukarı dönük T       "); addch(ACS_BTEE); printw("\n");
  printw("Aşağı dönük T        "); addch(ACS_TTEE); printw("\n");
  printw("Yata çizgi           "); addch(ACS_HLINE); printw("\n");
  printw("Dikey çizgi          "); addch(ACS_VLINE); printw("\n");
  printw("Büyük artı           "); addch(ACS_PLUS); printw("\n");
  printw("Tarama satırı 1      "); addch(ACS_S1); printw("\n");
  printw("Tarama satırı 3      "); addch(ACS_S3); printw("\n");
  printw("Tarama satırı 7      "); addch(ACS_S7); printw("\n");
  printw("Tarama satırı 9      "); addch(ACS_S9); printw("\n");
  printw("Eşkenar görtgen      "); addch(ACS_DIAMOND); printw("\n");
  printw("Noktalı onay kutusu  "); addch(ACS_CKBOARD); printw("\n");
  printw("Derece sembolü       "); addch(ACS_DEGREE); printw("\n");
  printw("Artı/Eksi işareti    "); addch(ACS_PLMINUS); printw("\n");
  printw("Madde imi            "); addch(ACS_BULLET); printw("\n");
  printw("Solu gösteren ok     "); addch(ACS_LARROW); printw("\n");
  printw("Sağı gösteren ok     "); addch(ACS_RARROW); printw("\n");
```

```
  printw("Aşağı gösteren ok    "); addch(ACS_DARROW); printw("\n");
  printw("Yukarı gösteren ok   "); addch(ACS_UARROW); printw("\n");
  printw("Diyez işareti        "); addch(ACS_BOARD); printw("\n");
  printw("İçi boş kare         "); addch(ACS_LANTERN); printw("\n");
  printw("İçi dolu kare blok   "); addch(ACS_BLOCK); printw("\n");
  printw("Küçük/Eşit işareti   "); addch(ACS_LEQUAL); printw("\n");
  printw("Büyük/Eşit işareti   "); addch(ACS_GEQUAL); printw("\n");
  printw("Pi                   "); addch(ACS_PI); printw("\n");
  printw("Eşit değil           "); addch(ACS_NEQUAL); printw("\n");
  printw("UK ound sembolü      "); addch(ACS_STERLING); printw("\n");

  refresh();
  getch();
  endwin();

  return 0;
}
```

15. Diğer Kütüphaneler

Curses kütüphanesinden başka daha fazla özellik ve işlevsellik sağlayan birkaç kütüphane daha vardır. Takip eden kısımlarda curses ile dağıtılan üç standart kütüphane incelenecektir.

16. Panel Kütüphanesi

Artık curses kullanımında yetenekli hale geldiniz ve daha büyük bir şeyler yapmak istiyorsunuzdur. Üst üste çalışan ve profosyonel bir pencere görünümü sağlayan bir sürü pencere oluşturdunuz. Ne yazık ki, bir süre sonra bunları idare etmek oldukça zor olacaktır. Çoklu tazeleme ve güncellemeler sizi bir kabusa itecektir. Üst üste çalışan pencereler, pencereleri doğru sırada tazelemeyi unuttuğunuzda büyük lekeler bırakır.

Ümizsitliğe kapılmayın. Panel kütüphanesinde sunulmuş zekice bir çözüm bulunmaktadır. ncurses geliştiricilerinin dünyasında şöyle denir:

> Eğer arayüz tasarımınızda pencere yığıtınız çok derinlere iniyorsa çalışma zamanında bu pencereleri yığıttan çekmek için durmadan düzenlemeler yapmak yorucu ve üstesinden gelinmez bir hal alabilir. Böylece karşımıza panel kütüphanesi çıkar.

Eğer pekçok üst üste çalışan pencereniz varsa kullanmanız gereken panel kütüphanesidir. Seri şekilde ve doğru sırada (yukardan aşağı) `wnoutrefresh()` ve `doupdate()` bunun üstesinden gelir. Kütüphane, pencerelerin sıraları, üst üstelikleri hakkındaki bilgileri ve ekranı düzgün şekilde güncelemekle ilgili işleri yönetmektedir. O zaman ne duruyoruz? Panellere yakından bakalım.

16.1. Temel Bilgiler

Panel nesneleri, içerisinde tüm diğer panel nesnelerini barındıran bir destenin parçası olarak işleme örtük olarak alınan birer penceredir. Deste en üstteki panelin görünür olduğu diğerlerinin konumlarına göre belirli veya belirsiz olabileceği bir yığın gibi davranış gösterir. Dolayısiyle temel fikir üst üste çalışan panel yığını oluşturmak ve panel kütüphanesini kullanarak bunları düzgün göstermektir. `refresh()`'e benzeyen ve çağrıldığında panellerin düzgün gözükmesini sağlayan bir işlev vardır. Panelleri gösteren, gizleyen, hareket ettiren, boyutlarını değiştiren, v.b. işlevler vardır. Üst üste çalışma sorunu panel kütüphanesi tarafından tüm bu işlev çağrımları sırasında halledilir.

Bir panel programının genel akışı şuna benzer:

1. Panellere eklenecek pencereler oluşturulur (`newwin()` ile)
2. Görünme sırasına göre paneller oluşturulur. Görünmesi istenen sıraya göre bir yığıt oluşturulur. Yeni panel oluşturmak için `new_panel()` işlevi kullanılır.
3. Panelleri ekrana doğru görüntülenme sırasıyla yazmak için `update_panels()` işlevi kullanılır. Görüntülenmelerini sağlamak için ise `doupdate()` işlevi kullanılır.
4. Panelleri düzenlemek için `show_panel(), hide_panel(), move_panel()` v.b. işlevler kullanılır. `panel_hidden()` ve `panel_window()` gibi yardımcı işlevlerden yararlanılır. Bir panelin kullanıcı göstericisini belirtmek ve bilgi almak için `set_panel_userptr()` ve `panel_userptr()` işlevleri kullanılır.
5. Panel kullanımı bittikten sonra `del_panel()` ile panel silinir.

Bazı programlar ile konuyu açıklığa kavuşturalım. Aşağıda 3 tane üst üste çalışan panel oluşturup bunları ekranda gösteren bir program bulunmaktadır.

16.2. Panel Kütüphanesi ile derleme

Panel kütüphanesi işlevlerini kullanabilmek için `panel.h` başlık dosyası programa dahil edilmeli ve programı panel kütüphanesi ile ilintili derlemek için `-lpanel -lncurses` seçenekleri kullanılmalıdır.

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

derleme ve ilintileme: gcc program dosyası -lpanel -lncurses
```

Örnek 14. Panellerin temelleri

```
#include <panel.h>

int main()
{ WINDOW *my_wins[3];
  PANEL  *my_panels[3];
  int lines = 10, cols = 40, y = 2, x = 4, i;

  initscr();
  cbreak();
  noecho();

  /* Paneller için pencereler oluşturun */
  my_wins[0] = newwin(lines, cols, y, x);
  my_wins[1] = newwin(lines, cols, y + 1, x + 5);
  my_wins[2] = newwin(lines, cols, y + 2, x + 10);

  /*
   * Pencereler etrafında çerçeve oluşturun
   * böylece Panel etkilerini görebilirsiniz
   */
  for(i = 0; i < 3; ++i)
          box(my_wins[i], 0, 0);

  /* Her pencereyi bir panele bağlayın. Sıralama alttan üste doğru */
  my_panels[0] = new_panel(my_wins[0]);   /* 0'a  it, sıra: stdscr-0 */
```

```
  my_panels[1] = new_panel(my_wins[1]);   /* 1'e  it, sıra: stdscr-0-1 */
  my_panels[2] = new_panel(my_wins[2]);   /* 2'ye it, sıra: stdscr-0-1-2 */

  /* Yığın sıralamasını güncelle. 2. panel en üstte olacak */
  update_panels();

  /* Ekranda göster */
  doupdate();

  getch();
  endwin();
}
```

Gördüğünüz gibi yukarıdaki program açıklandığı gibi basit bir akış takip etmektedir. Pencereler `newwin()` ile oluşturulup `new_panel()` ile panellere bağlanmıştır. Bir paneli diğeri ardına bağladığımızda, panel yığını güncellenmektedir. Bunları ekrana yazdırmak için `update_panels()` ve göstermek için `doupdate()` işlevleri çağrılmıştır.

16.3. Panel Penceresinde Gezinmek

Biraz karmaşık bir örnek aşağıda verilmiştir. Bu program sekme tuşu kullanılarak gezilebilen 3 pencere oluşturmaktadır. Koda bir göz atın.

Örnek 15. Panel Penceresi Gezinti Örneği

```
#include <panel.h>

#define NLINES 10
#define NCOLS 40

void init_wins(WINDOW **wins, int n);
void win_show(WINDOW *win, char *label, int label_color);
void print_in_middle(WINDOW *win, int starty, int startx, int width,
    char *string, chtype color);

int main()
{ WINDOW *my_wins[3];
  PANEL  *my_panels[3];
  PANEL  *top;
  int ch;

  /* Curses kipini ilklendir */
  initscr();
  start_color();
  cbreak();
  noecho();
  keypad(stdscr, TRUE);

  /* Tüm renkleri ilklendir */
  init_pair(1, COLOR_RED, COLOR_BLACK);
  init_pair(2, COLOR_GREEN, COLOR_BLACK);
  init_pair(3, COLOR_BLUE, COLOR_BLACK);
  init_pair(4, COLOR_CYAN, COLOR_BLACK);

  init_wins(my_wins, 3);
```

```
  /* Her pencereyi bir panele bağla */    /* Sıralama aşağıdan yukarıya */
  my_panels[0] = new_panel(my_wins[0]);   /* 0'a  it, sıra: stdscr-0 */
  my_panels[1] = new_panel(my_wins[1]);   /* 1'e  it, sıra: stdscr-0-1 */
  my_panels[2] = new_panel(my_wins[2]);   /* 2'ye it, sıra: stdscr-0-1-2 */

  /* Kullanıcı işaretçilerini bir sonraki panele ayarla */
  set_panel_userptr(my_panels[0], my_panels[1]);
  set_panel_userptr(my_panels[1], my_panels[2]);
  set_panel_userptr(my_panels[2], my_panels[0]);

  /* Yığını güncelle. 2. panel en üstte olacak */
  update_panels();

  /* Ekranda göster */
  attron(COLOR_PAIR(4));
  mvprintw(LINES - 2, 0,
    "Pencerelerde gezinmek için tab kullanın (Çıkmak için F1)");
  attroff(COLOR_PAIR(4));
  doupdate();

  top = my_panels[2];
  while((ch = getch()) != KEY_F(1))
  { switch(ch)
    { case 9:
        top = (PANEL *)panel_userptr(top);
        top_panel(top);
        break;
    }
    update_panels();
    doupdate();
  }
  endwin();
  return 0;
}

/* Tüm pencereleri yazdır */
void init_wins(WINDOW **wins, int n)
{ int x, y, i;
  char label[80];

  y = 2;
  x = 10;
  for(i = 0; i < n; ++i)
  { wins[i] = newwin(NLINES, NCOLS, y, x);
    sprintf(label, "Pencere Numaası %d", i + 1);
    win_show(wins[i], label, i + 1);
    y += 3;
    x += 7;
  }
}

/* Pencereyi bir çerçeve ve isimle göster */
void win_show(WINDOW *win, char *label, int label_color)
{ int startx, starty, height, width;

  getbegyx(win, starty, startx);
  getmaxyx(win, height, width);
```

```
  box(win, 0, 0);
  mvwaddch(win, 2, 0, ACS_LTEE);
  mvwhline(win, 2, 1, ACS_HLINE, width - 2);
  mvwaddch(win, 2, width - 1, ACS_RTEE);

  print_in_middle(win, 1, 0, width, label, COLOR_PAIR(label_color));
}

void print_in_middle (WINDOW *win, int starty, int startx, int width,
    char *string, chtype color)
{ int length, x, y;
  float temp;

  if(win == NULL)
    win = stdscr;
  getyx(win, y, x);
  if(startx != 0)
    x = startx;
  if(starty != 0)
    y = starty;
  if(width == 0)
    width = 80;

  length = strlen(string);
  temp = (width - length)/ 2;
  x = startx + (int)temp;
  wattron(win, color);
  mvwprintw(win, y, x, "%s", string);
  wattroff(win, color);
  refresh();
}
```

16.4. Kullanıcı İşaretçilerini Kullanmak

Yukarıdaki örnekte sıralamadaki bir sonraki pencereyi bulmak için kullanıcı işaretçilerini kullandım. Bir kullanıcı göstericisi tanımlayarak herhangi bir bilgiyi panele bağlayabilir ve saklamak istediğimiz herhangi bir bilgiye işaret edebiliriz. Bu örnekte bir sonraki panel için kullandım. Kullanıcı işaretçileri **`set_panel_userptr()`** ile tanımlanır. **`panel_userptr()`** ile parametre olarak verilen panele kullanıcı göstericisi döndürülür. Bir sonraki paneli bulduktan sonra işlem `top_panel()` işlevine geçmektedir. Bu işlev verilen paneli panel yığını içerisinde en tepeye taşır.

16.5. Panelleri Hareket Ettirmek ve Boyutlandırmak

`move_panel()` işlevi paneli istenen konuma taşımak için kullanılabilir. Panelin yığın içerisindeki konumunu değiştirmez. Panele bağlı pencereler üzerinde `mvwin()` yerine `move_panel()` kullandığınızdan emin olun.

Bir paneli yeniden boyutlandırmak biraz karmaşıktır. Doğrudan bu işi yapacak bir işlev yoktur. Bir paneli yeniden boyutlandırmanın yolu istenen boyutlarda bir pencere oluşturmak ve `replace_panel()` ile boyutu değiştirilmek istenen pencereyi değiştirmektir. Eski pencereyi silmeyi unutmayın. Bir panele bağlı pencere `panel_window()` ile bulunabilir.

Aşağıdaki program bahsedilenleri basit bir şekilde göstermektedir. Her zamanki gibi pencereler üzerinde sekme tuşu ile hareket edebilirsiniz. Etkin paneli yeniden boyutlandırmak için 'r"ye taşımak için 'm"ye basın. Daha

sonra ok tuşları ile taşıma ve boyutlandırmanızı yapın, bitince enter ile sonlandırın. Bu örnek kullanıcı bilgilerini kullanarak işlemler yapmaktadır.

Örnek 16. Panel Taşıma ve Yeniden Boyutlandırma Örneği

```
#include <panel.h>

typedef struct _PANEL_DATA {
  int x, y, w, h;
  char label[80];
  int label_color;
  PANEL *next;
}PANEL_DATA;

#define NLINES 10
#define NCOLS 40

void init_wins(WINDOW **wins, int n);
void win_show(WINDOW *win, char *label, int label_color);
void print_in_middle(WINDOW *win, int starty, int startx, int width,
     char *string, chtype color);
void set_user_ptrs(PANEL **panels, int n);

int main()
{ WINDOW *my_wins[3];
  PANEL  *my_panels[3];
  PANEL_DATA  *top;
  PANEL *stack_top;
  WINDOW *temp_win, *old_win;
  int ch;
  int newx, newy, neww, newh;
  int size = FALSE, move = FALSE;

  /* Curses kipi ilklendirme */
  initscr();
  start_color();
  cbreak();
  noecho();
  keypad(stdscr, TRUE);

  /* Tüm renkleri ilklendir */
  init_pair(1, COLOR_RED, COLOR_BLACK);
  init_pair(2, COLOR_GREEN, COLOR_BLACK);
  init_pair(3, COLOR_BLUE, COLOR_BLACK);
  init_pair(4, COLOR_CYAN, COLOR_BLACK);

  init_wins(my_wins, 3);

  /* Her panele bir oencere ata */        /* Sıra aşağıdan yukarıya */
  my_panels[0] = new_panel(my_wins[0]);   /* 0'a  it, sıra: stdscr-0 */
  my_panels[1] = new_panel(my_wins[1]);   /* 1'e  it, sıra: stdscr-0-1 */
  my_panels[2] = new_panel(my_wins[2]);   /* 2'ye it, sıra: stdscr-0-1-2 */

  set_user_ptrs(my_panels, 3);
  /* Yığın sıralamasını güncelle. 2. panel en tepede olacak */
  update_panels();
```

```
  /* Ekrana göster */
  attron(COLOR_PAIR(4));
  mvprintw(LINES - 3, 0,
    "Taşımak için 'm' , yeniden boyutlandırma için 'r' kullanın");
  mvprintw(LINES - 2, 0,
    "Pencereler arasında gezinmek için tab kullanın (Çıkmak için F1)");
  attroff(COLOR_PAIR(4));
  doupdate();

  stack_top = my_panels[2];
  top = (PANEL_DATA *)panel_userptr(stack_top);
  newx = top->x;
  newy = top->y;
  neww = top->w;
  newh = top->h;
  while((ch = getch()) != KEY_F(1))
  { switch(ch)
    { case 9:         /* Tab */
        top = (PANEL_DATA *)panel_userptr(stack_top);
        top_panel(top->next);
        stack_top = top->next;
        top = (PANEL_DATA *)panel_userptr(stack_top);
        newx = top->x;
        newy = top->y;
        neww = top->w;
        newh = top->h;
        break;
      case 'r':       /* Yeniden boyutlandır */
        size = TRUE;
        attron(COLOR_PAIR(4));
        mvprintw(LINES - 4, 0,
            "Yeniden boyutlandırmaya giriş: "
            "Ok tuşları ile boyutlandırın ve "
            "bitirmek için <ENTER> kullanın");
        refresh();
        attroff(COLOR_PAIR(4));
        break;
      case 'm':       /* Move */
        attron(COLOR_PAIR(4));
        mvprintw(LINES - 4, 0,
          "Taşıma kipine girildi: "
          "Taşımak için ok tuşlarını ve "
          "bitirmek için <ENTER> kullanın");
        refresh();
        attroff(COLOR_PAIR(4));
        move = TRUE;
        break;
      case KEY_LEFT:
        if(size == TRUE)
        { --newx;
          ++neww;
        }
        if(move == TRUE)
          --newx;
        break;
      case KEY_RIGHT:
        if(size == TRUE)
```

```
        { ++newx;
          --neww;
        }
        if(move == TRUE)
          ++newx;
        break;
      case KEY_UP:
        if(size == TRUE)
        { --newy;
          ++newh;
        }
        if(move == TRUE)
          --newy;
        break;
      case KEY_DOWN:
        if(size == TRUE)
        { ++newy;
          --newh;
        }
        if(move == TRUE)
          ++newy;
        break;
      case 10:        /* Enter */
        move(LINES - 4, 0);
        clrtoeol();
        refresh();
        if(size == TRUE)
        { old_win = panel_window(stack_top);
          temp_win = newwin(newh, neww, newy, newx);
          replace_panel(stack_top, temp_win);
          win_show(temp_win, top->label, top->label_color);
          delwin(old_win);
          size = FALSE;
        }
        if(move == TRUE)
        { move_panel(stack_top, newy, newx);
          move = FALSE;
        }
        break;
    }
    attron(COLOR_PAIR(4));
    mvprintw(LINES - 3, 0,
      "Taşımak için 'm', yeniden boyutlandırmak için 'r' kullanın");
    mvprintw(LINES - 2, 0,
      "Pencereler arasında gezinti için tab kullanın "
        "(Çıkmak için F1)");
    attroff(COLOR_PAIR(4));
    refresh();
    update_panels();
    doupdate();
  }
  endwin();
  return 0;
}

/* Tüm pencereleri yazdır */
void init_wins(WINDOW **wins, int n)
```

```
{ int x, y, i;
  char label[80];

  y = 2;
  x = 10;
  for(i = 0; i < n; ++i)
  { wins[i] = newwin(NLINES, NCOLS, y, x);
    sprintf(label, "Pencere Numarası %d", i + 1);
    win_show(wins[i], label, i + 1);
    y += 3;
    x += 7;
  }
}

/* Herbir panel için PANEL_DATA veri yapısını tanımla */
void set_user_ptrs(PANEL **panels, int n)
{ PANEL_DATA *ptrs;
  WINDOW *win;
  int x, y, w, h, i;
  char temp[80];

  ptrs = (PANEL_DATA *)calloc(n, sizeof(PANEL_DATA));

  for(i = 0;i < n; ++i)
  { win = panel_window(panels[i]);
    getbegyx(win, y, x);
    getmaxyx(win, h, w);
    ptrs[i].x = x;
    ptrs[i].y = y;
    ptrs[i].w = w;
    ptrs[i].h = h;
    sprintf(temp, "Pencere Numarası %d", i + 1);
    strcpy(ptrs[i].label, temp);
    ptrs[i].label_color = i + 1;
    if(i + 1 == n)
      ptrs[i].next = panels[0];
    else
      ptrs[i].next = panels[i + 1];
    set_panel_userptr(panels[i], &ptrs[i]);
  }
}

/* Pencereyi bir çerçeve ve bir isimle göster */
void win_show(WINDOW *win, char *label, int label_color)
{ int startx, starty, height, width;

  getbegyx(win, starty, startx);
  getmaxyx(win, height, width);

  box(win, 0, 0);
  mvwaddch(win, 2, 0, ACS_LTEE);
  mvwhline(win, 2, 1, ACS_HLINE, width - 2);
  mvwaddch(win, 2, width - 1, ACS_RTEE);

  print_in_middle(win, 1, 0, width, label, COLOR_PAIR(label_color));
}
```

```
void print_in_middle(WINDOW *win, int starty, int startx, int width,
    char *string, chtype color)
{ int length, x, y;
  float temp;

  if(win == NULL)
    win = stdscr;
  getyx(win, y, x);
  if(startx != 0)
    x = startx;
  if(starty != 0)
    y = starty;
  if(width == 0)
    width = 80;

  length = strlen(string);
  temp = (width - length)/ 2;
  x = startx + (int)temp;
  wattron(win, color);
  mvwprintw(win, y, x, "%s", string);
  wattroff(win, color);
  refresh();
}
```

Ana programdaki `while` döngüsüne dikkatinizi verin. Bir kere basılan tuşun türünü buldu mu uygun davranışı göstermektedir. Eğer 'r' basıldıysa yeniden boyutlandırma başlamaktadır. Bundan sonra kullanıcı ok tuşlarına bastıkça yeni boyutları güncellenmektedir. Kullanıcı <ENTER>'a bastığı anda o anki seçim sona erer ve panel açıklanan mantıkla yeniden boyutlandırılır. Yeniden boyutlandırma kipinde program pencerenin nasıl yeniden boyutlandırıldığını göstermez. Okuyucuya yeniden boyutlandırma sırasında noktalı bir çerçeve ile yeni boyutu gösterme işi bir alıştırma olarak bırakılmıştır.

Kullanıcı 'm"e bastığında taşıma kipi başlar. Bu, yeniden boyutlandırmaktan daha kolaydır. Ok tuşlarına bastıkça yeni konum güncellenir ve <ENTER> ile `move_panel()` çağrılarak panel taşıma işlemi sonlandırılır.

Bu programda `PANEL_DATA` olarak ifade edilen kullanıcı bilgisi önemli bir rol oynamaktadır. Açıklama satırlarında da yazdığı gibi `PANEL_DATA` panel boyutu, ismi, isim rengi ve sıralamadaki sonraki panele işaretçi bilgilerini tutmaktadır.

16.6. Panelleri Gizlemek ve Göstermek

Bir panel `hide_panel()` işlevi kullanılarak gizlenebilir. Bu işlev sadece onu panel yığınından kaldırır, böylece `update_panels()` ve `doupdate()` işlevleri kullanıldığında ekrandan saklanmış olur. Gizlenmiş panel ile ilintili PANEL yapısını bozmaz. Tekrar gösterilmek istenirse `show_panel()` işlevi kullanılmalıdır.

Aşağıdaki program panellerin gizlenmesini göstermektedir. 'a' veya 'b' veya 'c' ye sırasıyla birinci, ikinci ve üçüncü pencereleri gizlemek veya göstermek için basın. Pencerenin gizlenip gizlenmediği bilgisini saklayan küçük bir hide isimli değişken ile kullanıcı verisini kullanmaktadır. Bir nedenden ötürü panelin gizlenip gizlenmediğini söyleyen **`panel_hidden()`** işlevi çalışmamaktadır. Michael Andres tarafından gönderilen bir hata raposu buradadır[B23].

Örnek 17. Panel Gizleme ve Gösterme Örneği

```
#include <panel.h>

typedef struct _PANEL_DATA {
```

```
  int hide;       /* eğer panel gizlenmişse TRUE */
}PANEL_DATA;

#define NLINES 10
#define NCOLS 40

void init_wins(WINDOW **wins, int n);
void win_show(WINDOW *win, char *label, int label_color);
void print_in_middle(WINDOW *win, int starty, int startx, int width,
     char *string, chtype color);

int main()
{ WINDOW *my_wins[3];
  PANEL  *my_panels[3];
  PANEL_DATA panel_datas[3];
  PANEL_DATA *temp;
  int ch;

  /* Curses kipi ilklendir */
  initscr();
  start_color();
  cbreak();
  noecho();
  keypad(stdscr, TRUE);

  /* Tüm renkleri ilklendir */
  init_pair(1, COLOR_RED, COLOR_BLACK);
  init_pair(2, COLOR_GREEN, COLOR_BLACK);
  init_pair(3, COLOR_BLUE, COLOR_BLACK);
  init_pair(4, COLOR_CYAN, COLOR_BLACK);

  init_wins(my_wins, 3);

  /* Her panele bir pencere bağla */      /* Sıralama aşağıdan yukarıya */
  my_panels[0] = new_panel(my_wins[0]);   /* 0'a  it, sıra: stdscr-0 */
  my_panels[1] = new_panel(my_wins[1]);   /* 1'e  it, sıra: stdscr-0-1 */
  my_panels[2] = new_panel(my_wins[2]);   /* 2'ye it, sıra: stdscr-0-1-2 */

  /* Hiçbir şeyin gizlenmediğiniz söyleyerek panel bilgisini ilklendir */
  panel_datas[0].hide = FALSE;
  panel_datas[1].hide = FALSE;
  panel_datas[2].hide = FALSE;

  set_panel_userptr(my_panels[0], &panel_datas[0]);
  set_panel_userptr(my_panels[1], &panel_datas[1]);
  set_panel_userptr(my_panels[2], &panel_datas[2]);

  /* Yığın sıralamasını güncelle. 2. panel en üstte olacak */
  update_panels();

  /* Ekranda göster */
  attron(COLOR_PAIR(4));
  mvprintw(LINES - 3, 0, "Pencereleri şu tuşlarla gizleyin veya gösterin"
      " 'a'(ilk pencere)  'b'(ikinci pencere)  'c'(üçüncü pencere)");
  mvprintw(LINES - 2, 0, "Çıkmak için F1");

  attroff(COLOR_PAIR(4));
```

```
  doupdate();

  while((ch = getch()) != KEY_F(1))
  { switch(ch)
    { case 'a':
        temp = (PANEL_DATA *)panel_userptr(my_panels[0]);
        if(temp->hide == FALSE)
        { hide_panel(my_panels[0]);
          temp->hide = TRUE;
        }
        else
        { show_panel(my_panels[0]);
          temp->hide = FALSE;
        }
        break;
      case 'b':
        temp = (PANEL_DATA *)panel_userptr(my_panels[1]);
        if(temp->hide == FALSE)
        { hide_panel(my_panels[1]);
          temp->hide = TRUE;
        }
        else
        { show_panel(my_panels[1]);
          temp->hide = FALSE;
        }
        break;
      case 'c':
        temp = (PANEL_DATA *)panel_userptr(my_panels[2]);
        if(temp->hide == FALSE)
        { hide_panel(my_panels[2]);
          temp->hide = TRUE;
        }
        else
        { show_panel(my_panels[2]);
          temp->hide = FALSE;
        }
        break;
    }
    update_panels();
    doupdate();
  }
  endwin();
  return 0;
}

/* Tüm pencereleri ekrana yazdır */
void init_wins(WINDOW **wins, int n)
{ int x, y, i;
  char label[80];

  y = 2;
  x = 10;
  for(i = 0; i < n; ++i)
  { wins[i] = newwin(NLINES, NCOLS, y, x);
    sprintf(label, "Window Number %d", i + 1);
    win_show(wins[i], label, i + 1);
    y += 3;
```

```
    x += 7;
  }
}

/* Penceleri bir çerçeve ve isimle göster */
void win_show(WINDOW *win, char *label, int label_color)
{ int startx, starty, height, width;

  getbegyx(win, starty, startx);
  getmaxyx(win, height, width);

  box(win, 0, 0);
  mvwaddch(win, 2, 0, ACS_LTEE);
  mvwhline(win, 2, 1, ACS_HLINE, width - 2);
  mvwaddch(win, 2, width - 1, ACS_RTEE);

  print_in_middle(win, 1, 0, width, label, COLOR_PAIR(label_color));
}

void print_in_middle(WINDOW *win, int starty, int startx, int width,
     char *string, chtype color)
{ int length, x, y;
  float temp;

  if(win == NULL)
          win = stdscr;
  getyx(win, y, x);
  if(startx != 0)
          x = startx;
  if(starty != 0)
          y = starty;
  if(width == 0)
          width = 80;

  length = strlen(string);
  temp = (width - length)/ 2;
  x = startx + (int)temp;
  wattron(win, color);
  mvwprintw(win, y, x, "%s", string);
  wattroff(win, color);
  refresh();
}
```

16.7. `panel_above()` ve `panel_below()`

`panel_above()` ve **`panel_below()`** işlevleri bir panelin altındaki ve üstündeki panelleri bulmak için kullanılabilir. Eğer bu işlevlere gönderilen parametreler NULL ise, o zaman sırasıyla en alttaki ve en üstteki panellere bir işaretçi döndürürler.

17. Menü Kütüphanesi

Menü kütüphanesi curses kütüphanesinin temel özelliklerine menüler oluşturmanızı sağlayan hoş özellik kazandırır. Menüler oluşturmak için bir takım işlevler sunar. Fakat güzel bir görünüm için renkler v.b. özelliklerle özelleştirilmeleri gerekir. Ayrıntılara bir bakalım.

Bir menü verilmiş olan öğe kümesinden kullanıcının bazı alt kümeleri seçmesini sağlayan ekran görüntüsüdür. Basitçe söylersek, içerisinden bir veya daha fazla öğe seçilebilen öğeler kümesidir. Bazı okuyucular birden fazla öğe seçilebilme özelliğinin olduğunun farkında olmayabilir. Menü kütüphanesi kullanıcının birden fazla seçim yapmasını sağlayabilen menüler yazma özelliği sunmaktadır. Bu, daha sonraki kısımda ele alınacaktır. Şimdi bazı ilkeleri öğrenmenin zamanı.

17.1. Temel Bilgiler

Menüleri oluşturmak için önce öğeler oluşturulur ve sonra da ekranda menü oluşturulur. Bundan sonra, kullanıcının tüm işlemleri menü programının dolap beygiri olan `menu_driver()` işlevi tarafından zekice gerçeklenir.

Bir menü programının genel akışı şu şekildedir.

1. Curses kipi ilklendirilir
2. `new_item()` kullanarak öğeler oluşturulur. Öğeler için isim ve tanımlama belirtebilirsiniz.
3. `new_menu()` ile kendisine ilintilenecek öğeler belirtilerek menü oluşturulur.
4. Menü `menu_post()` ile ekrana yazdırılır ve ekran tazelenir.
5. Bir döngü ile kullanıcı isteklerini işlenir ve `menu_driver` ile menü üzerinde gerekli güncellemeler yapılır.
6. `menu_unpost()` ile ekrandan menü görüntüsü silinir.
7. `free_menu()` ile menü tarafından kullanılan bellek alanı serbest bırakılır.
8. Menü içerisindeki öğelere ayrılan bellek alanı `free_item()` ile serbest bırakılır.
9. Curses kipi sonlandırılır.

Şimdi, yukarı ve aşağı ok tuşları kullanarak seçilen öğeyi güncelleyen basit bir program görelim.

17.2. Menü Kütüphanesi ile derleme

Menü kütüphanesini kullanmak için `menu.h` başlık dosyası programa eklenmeli ve programı menü kütüphanesiyle ilintileyerek derlemek için de `-lmenu -lncurses` seçenekleri kullanılmalıdır.

```
#include <menu.h>
.
.
.

derleme ve bağlama: gcc program file -lmenu -lncurses
```

Örnek 18. Menülerin Temelleri

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

#define ARRAY_SIZE(a) (sizeof(a) / sizeof(a[0]))
#define CTRLD   4

char *choices[] = {
  "Choice 1",
  "Choice 2",
  "Choice 3",
  "Choice 4",
  "Exit",
};
```

```
int main()
{ ITEM **my_items;
  int c;
  MENU *my_menu;
  int n_choices, i;
  ITEM *cur_item;


  initscr();
  cbreak();
  noecho();
  keypad(stdscr, TRUE);

  n_choices = ARRAY_SIZE(choices);
  my_items = (ITEM **)calloc(n_choices + 1, sizeof(ITEM *));

  for(i = 0; i < n_choices; ++i)
          my_items[i] = new_item(choices[i], choices[i]);
  my_items[n_choices] = (ITEM *)NULL;

  my_menu = new_menu((ITEM **)my_items);
  mvprintw(LINES - 2, 0, "F1 to Exit");
  post_menu(my_menu);
  refresh();

  while((c = getch()) != KEY_F(1))
  { switch(c)
    { case KEY_DOWN:
        menu_driver(my_menu, REQ_DOWN_ITEM);
        break;
      case KEY_UP:
        menu_driver(my_menu, REQ_UP_ITEM);
        break;
    }
  }

  free_item(my_items[0]);
  free_item(my_items[1]);
  free_menu(my_menu);
  endwin();
}
```

Bu program menü kütüphanesi kullanılarak bir menü oluşturmanın temel adımlarını göstermektedir. Öncelikle `new_item()` ile öğeleri oluşturup onları daha sonra `new_menu()` ile menüye ekleyerek menüyü ilklendirdik. Menüyü ekrana yazdırıp ekranı tazeledikten sonra ana işlem döngüsü başlamaktadır. Kullanıcı girdisini alarak buna karşılık gelen işlemi gerçekleştirmektedir. `menu_driver()` işlevi menü sisteminin dolap beygiridir. Bu işlevin ikinci parametresi menü ile ilgili ne yapılacağını söyler. Parametreye göre, `menu_driver()` karşılık gelen görevi yapmaktadır. Parametre değeri, bir menü gezinti isteği, bir ascii karakter veya bir tuş ile ilişkili `KEY_MOUSE` özel anahtarı olabilir.

`menu_driver` aşağıdaki istekleri kabul eder.

`REQ_LEFT_ITEM`	`Bir öğenin soluna hareket et`
`REQ_RIGHT_ITEM`	`Bir öğenin sağına hareket et`
`REQ_UP_ITEM`	`Bir öğenin üstüne hareket et`
`REQ_DOWN_ITEM`	`Bir öğenin altına hareket et`

```
REQ_SCR_ULINE       Bir satır yukarı
REQ_SCR_DLINE       Bir satır aşağı
REQ_SCR_DPAGE       Bir sayfa aşağı
REQ_SCR_UPAGE       Bir sayfa yukarı
REQ_FIRST_ITEM      İlk öğeye hareket et
REQ_LAST_ITEM       Son öğeye hareket et
REQ_NEXT_ITEM       Sonraki öğeye hareket et
REQ_PREV_ITEM       Önceki öğeye hareket et
REQ_TOGGLE_ITEM     Bir öğeyi seç/seçimi kaldır
REQ_CLEAR_PATTERN   Menü örüntü belleğini sil
REQ_BACK_PATTERN    Örüntü belleğinden bir önceki karakteri sil
REQ_NEXT_MATCH      Örüntü eşine uygun bir sonraki öğeye git
REQ_PREV_MATCH      Örüntü eşine uygun bir önceki öğeye git
```

Seçeneklerin sayısından dolayı bunalmış hissetmeyin. Herbirini tek tek göreceğiz. Bu listedeki ilgi alanımız REQ_UP_ITEM ve REQ_DOWN_ITEM öğeleridir. Bu iki seçenek menu_driver işlevine gönderildiklerinde, o anki öğeyi sırasıyla bir öğe yukarı veya aşağı günceller.

17.3. menu_driver: Menü sisteminin dolap beygiri

Yukarıdaki örnekte gördüğünüz gibi menu_driver menüyü güncellemede önemli bir rol oynamaktadır. Aldığı çeşitli seçenekleri ve neler yaptığını anlamak oldukça önemlidir. Yukarıda açıklandığı üzere menu_driver()'ın ikinci parametresi bir menü gezinti isteği, yazılabilir bir karakter veya KEY_MOUSE anahtarı olabilir. Değişik gezinti isteklerini parçalara ayırıp inceleyelim.

REQ_LEFT_ITEM ve REQ_RIGHT_ITEM
: Bir menü bir öğe için birden fazla sütun ile gösterilebilir. **menu_format()** işlevi kullanılarak bu gerçeklenebilir. Çok sütunlu bir menü gösterildiği zaman yukarıdaki istekler menü sürücüsünün o anki seçimin sağa ya da sola hareket etmesini sağlar.

REQ_UP_ITEM ve REQ_DOWN_ITEM
: Yukarıdaki örnekte bu seçenekleri görmüştünüz. Bu istekler verildiğinde menu_driver'ın o anki seçimi yukarı veya aşağı kaydırmasını sağlar.

REQ_SCR_* seçenekleri
: REQ_SCR_ULINE, REQ_SCR_DLINE, REQ_SCR_DPAGE ve REQ_SCR_UPAGE seçenekleri kaydırma ile ilgili seçeneklerdir. Eğer menüdeki tüm öğeler alt pencere menüsünde gösterilemiyorsa o zaman menü kaydırılabilirdir. Bu istekler menu_driver'a verilerek sırasıyla kaydırmanın yukarı, aşağı, bir sayfa aşağı veya yukarı yapılması sağlanabilir.

REQ_FIRST_ITEM, REQ_LAST_ITEM, REQ_NEXT_ITEM ve REQ_PREV_ITEM
: Bu istekler kendilerini açıklar niteliktedir (Sırayla ilk, son, sonraki, önceki).

>REQ_TOGGLE_ITEM
: Bu istek verildiğinde o anki seçim değiştirilir. Bu seçenek çok değerli menülerde kullanılmalıdır. Bu isteği kullanabilmek için O_ONEVALUE değeri etkisizleştirilmelidir. Bu seçenek set_menu_opts() ile etkin veya etkisiz yapılabilir.

Örüntü İstekleri
: Her menünün kullanıcın girdiği karakterlere en yakın olan ascii karakter karşılığını bulmak için bir örüntü önbelleği bulunmaktadır. menu_driver'a ascii karakterler verildiği zaman örüntü önbelleğine alınır. Aynı zamanda listedeki örüntüye en yakın olan eşleşmeyi de bulmaya çalışır ve o anki seçimi o

öğeye hareket ettirir. REQ_CLEAR_PATTERN isteği örüntü belleğini siler. REQ_BACK_PATTERN isteği örüntü belleğindeki bir önceki karakteri siler. Örüntünün birden fazla öğe ile eşlemesi halinde eşleşen öğeler REQ_NEXT_MATCH ve REQ_PREV_MATCH ile seçim üzerinden sonraki veya öncekine hareket sağlanabilir.

Fare İstekleri

KEY_MOUSE istekleri durumunda farenin konumuna göre bir davranış sergilenir. Gösterilecek davranış kılavuz sayfalarında açıklanmıştır:

> Eğer ikinci parametre KEY_MOUSE özel karakteri ise bununla ilişkili fare olayı, yukarıda önceden tanımlanmış isteklere çevrilir. Sadece kullanıcı penceresindeki (örn. menü görüntü alanı veya dekor penceresi) tıklamalar işlenir. Menünün görüntülenebilir alanı üzerine tıklanırsa REQ_SCR_ULINE üretilir, eğer çift tıklarsanız REQ_SCR_UPAGE üretilir ve eğer üç kere tıklarsanız REQ_FIRST_ITEM üretilir. Eğer menü görüntüleme alanı aşağısına tıklanırsa REQ_SCR_DLINE üretilir, eğer iki kere tıklanırsa REQ_SCR_DPAGE üretilir ve eğer üç kere tıklanırsa REQ_LAST_ITEM üretilir. Eğer menü görüntüleme alanının içerisinde bir öğeye tıklanırsa menü imleci bu öğeye konumlanır.

Yukarıdaki isteklerin her biri takip eden satırlarda değişik örneklerle gerekli oldukça açıklanacaktır.

17.4. Menü Pencereleri

Oluşturulan her bir menü bir pencere ve bir alt menü ile ilişiklendirilir. Menü penceresi menü ile ilişiklendirilmiş herhangi bir başlığı veya çerçeveyi gösterir. Menü alt penceresi seçimi mümkün olan menü öğelerini gösterir. Fakat basit örnekte herhangi bir pencere veya alt pencere belirtmedik. Bir pencere belirtilmediği zaman, temel pencere olarak stdscr alınır ve daha sonra menü sistemi öğelerin gösterimi için pencere büyüklüğünü ayarlar. Daha sonra öğeler hesaplanmış alt menüde gösterilir. Öyleyse şimdi bu pencerelerle oynayıp bir çerçeve ve başlıkla bir menü gösterelim.

Örnek 19. Menü Pencereleri Kullanım örneği

```
#include <menu.h>

#define ARRAY_SIZE(a) (sizeof(a) / sizeof(a[0]))
#define CTRLD   4

char *choices[] = {
  "Choice 1",
  "Choice 2",
  "Choice 3",
  "Choice 4",
  "Exit",
  (char *)NULL,
};
void print_in_middle(WINDOW *win, int starty, int startx, int width,
     char *string, chtype color);

int main()
{ ITEM **my_items;
  int c;
```

```
MENU *my_menu;
WINDOW *my_menu_win;
int n_choices, i;

/* Curses kipini ilklendir */
initscr();
start_color();
cbreak();
noecho();
keypad(stdscr, TRUE);
init_pair(1, COLOR_RED, COLOR_BLACK);

/* Öğeleri oluştur */
n_choices = ARRAY_SIZE(choices);
my_items = (ITEM **)calloc(n_choices, sizeof(ITEM *));
for(i = 0; i < n_choices; ++i)
  my_items[i] = new_item(choices[i], choices[i]);

/* Menüyü oluştur */
my_menu = new_menu((ITEM **)my_items);

/* Menü ile ilişiklendirilecek pencereyi oluştur */
my_menu_win = newwin(10, 40, 4, 4);
keypad(my_menu_win, TRUE);

/* Ana pencereyi ve alt pencereleri ayarla */
set_menu_win(my_menu, my_menu_win);
set_menu_sub(my_menu, derwin(my_menu_win, 6, 38, 3, 1));

/* Menü göstericisini " * " olarak ayarla*/
set_menu_mark(my_menu, " * ");

/* Ana pencere etrafında bir çerçeve çiz ve bir başlık yaz */
box(my_menu_win, 0, 0);
print_in_middle(my_menu_win, 1, 0, 40, "My Menu", COLOR_PAIR(1));
mvwaddch(my_menu_win, 2, 0, ACS_LTEE);
mvwhline(my_menu_win, 2, 1, ACS_HLINE, 38);
mvwaddch(my_menu_win, 2, 39, ACS_RTEE);
mvprintw(LINES - 2, 0, "F1 to exit");
refresh();

/* Menüyü ekrana yaz */
post_menu(my_menu);
wrefresh(my_menu_win);

while((c = wgetch(my_menu_win)) != KEY_F(1))
{ switch(c)
  { case KEY_DOWN:
      menu_driver(my_menu, REQ_DOWN_ITEM);
      break;
    case KEY_UP:
      menu_driver(my_menu, REQ_UP_ITEM);
      break;
  }
  wrefresh(my_menu_win);
}
```

```
  /* Menüyü ekrandan sil ve tahsis edilen belleği geri ver */
  unpost_menu(my_menu);
  free_menu(my_menu);
  for(i = 0; i < n_choices; ++i)
    free_item(my_items[i]);
  endwin();
}

void print_in_middle(WINDOW *win, int starty, int startx, int width,
     char *string, chtype color)
{ int length, x, y;
  float temp;

  if(win == NULL)
    win = stdscr;
  getyx(win, y, x);
  if(startx != 0)
    x = startx;
  if(starty != 0)
    y = starty;
  if(width == 0)
    width = 80;

  length = strlen(string);
  temp = (width - length)/ 2;
  x = startx + (int)temp;
  wattron(win, color);
  mvwprintw(win, y, x, "%s", string);
  wattroff(win, color);
  refresh();
}
```

Bu örnek bir başlık, çerçeve ve öğelerle başlığı ayıran hoş bir çizgiye sahip bir menü oluşturur. Gördüğünüz gibi, bir pencereyi bir menüye ilişiklendirmek için `set_menu_win()` işlevi kullanılmak zorundadır. Böylece alt pencereyi de bağlamış oluruz. Bu, öğeleri alt pencerede gösterir. Seçilen öğenin sol tarafında gözüken işaretleme dizgesini `set_menu_mark()` ile ayarlayabilirsiniz.

17.5. Kaydırılabilen Menüler

Eğer pencere için oluşan alt pencere tüm öğeler için yeterince büyük değilse o zaman menü kaydırılabilir olur. O anki listedeki son öğede olduğunuzda eğer `REQ_DOWN_ITEM` isteğini gönderirseniz `REQ_SCR_DLINE`'e çevrilir ve menü bir adım aşağı kayar. El ile de `REQ_SCR_` isteklerini kaydırma işlemi için verebilirsiniz. Nasıl yapılabileceğine bakalım:

Örnek 20. Kaydırılabilir Menü örneği

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

#define ARRAY_SIZE(a) (sizeof(a) / sizeof(a[0]))
#define CTRLD   4

char *choices[] = {
  "Choice 1",
  "Choice 2",
  "Choice 3",
```

```
  "Choice 4",
  "Choice 5",
  "Choice 6",
  "Choice 7",
  "Choice 8",
  "Choice 9",
  "Choice 10",
  "Exit",
  (char *)NULL,
};
void print_in_middle(WINDOW *win, int starty, int startx, int width,
     char *string, chtype color);

int main()
{ ITEM **my_items;
  int c;
  MENU *my_menu;
  WINDOW *my_menu_win;
  int n_choices, i;

  /* Curses kipini ilklendir */
  initscr();
  start_color();
  cbreak();
  noecho();
  keypad(stdscr, TRUE);
  init_pair(1, COLOR_RED, COLOR_BLACK);
  init_pair(2, COLOR_CYAN, COLOR_BLACK);

  /* Öğeleri oluştur */
  n_choices = ARRAY_SIZE(choices);
  my_items = (ITEM **)calloc(n_choices, sizeof(ITEM *));
  for(i = 0; i < n_choices; ++i)
    my_items[i] = new_item(choices[i], choices[i]);

  /* Menüyü oluştur */
  my_menu = new_menu((ITEM **)my_items);

  /* Menü ile ilişiklendirilecek pencereyi oluştur */
  my_menu_win = newwin(10, 40, 4, 4);
  keypad(my_menu_win, TRUE);

  /* Ana ve alt pencereyi ayarla*/
  set_menu_win(my_menu, my_menu_win);
  set_menu_sub(my_menu, derwin(my_menu_win, 6, 38, 3, 1));
  set_menu_format(my_menu, 5, 1);

  /* Menü göstericisini" * " olarak ayarla*/
  set_menu_mark(my_menu, " * ");

  /* Ana pencere etrafında bir çerçeve çiz ve başlığı yaz */
  box(my_menu_win, 0, 0);
  print_in_middle(my_menu_win, 1, 0, 40, "My Menu", COLOR_PAIR(1));
  mvwaddch(my_menu_win, 2, 0, ACS_LTEE);
  mvwhline(my_menu_win, 2, 1, ACS_HLINE, 38);
  mvwaddch(my_menu_win, 2, 39, ACS_RTEE);
```

```
  /* Menüyü ekrana yaz */
  post_menu(my_menu);
  wrefresh(my_menu_win);

  attron(COLOR_PAIR(2));
  mvprintw(LINES - 2, 0,
    "Sayfayı aşağı yukarı kaydırmak için PageUp ve PageDown kullanın.");
  mvprintw(LINES - 1, 0,
    "Hareket etmek için Ok tuşları kullanın (Çıkmak için F1)");
  attroff(COLOR_PAIR(2));
  refresh();

  while((c = wgetch(my_menu_win)) != KEY_F(1))
  { switch(c)
    { case KEY_DOWN:
        menu_driver(my_menu, REQ_DOWN_ITEM);
        break;
      case KEY_UP:
        menu_driver(my_menu, REQ_UP_ITEM);
        break;
      case KEY_NPAGE:
        menu_driver(my_menu, REQ_SCR_DPAGE);
        break;
      case KEY_PPAGE:
        menu_driver(my_menu, REQ_SCR_UPAGE);
        break;
    }
    wrefresh(my_menu_win);
  }

  /* Menüyü ekrandan sil ve ayrılan bellek alanını geri ver */
  unpost_menu(my_menu);
  free_menu(my_menu);
  for(i = 0; i < n_choices; ++i)
    free_item(my_items[i]);
  endwin();
}

void print_in_middle(WINDOW *win, int starty, int startx, int width,
     char *string, chtype color)
{ int length, x, y;
  float temp;

  if(win == NULL)
    win = stdscr;
  getyx(win, y, x);
  if(startx != 0)
    x = startx;
  if(starty != 0)
    y = starty;
  if(width == 0)
    width = 80;

  length = strlen(string);
  temp = (width - length)/ 2;
  x = startx + (int)temp;
  wattron(win, color);
```

```
  mvwprintw(win, y, x, "%s", string);
  wattroff(win, color);
  refresh();
}
```

Bu program kendisini açıklar niteliktedir. Bu örnekte seçenek sayısı, alt menümüzün 6 tane gösterebilme kapasitesi olan öğe sayısından 10'a çıkarılmıştır. Bu mesaj menü sistemine açık bir şekilde `set_menu_format()` ile haber verilmelidir. Burada, tek bir sayfa için gösterilmesini istediğimiz satır ve sütun sayısını belirttik. Satır değişkenlerinde, eğer alt menü yüksekliğinden küçükse, herhangi bir sayıdaki öğeyi gösterilmesi için tanımlayabiliriz. Eğer kullanıcı tarafından basılan tuş PAGE UP veya PAGE DOWN ise, `menu_driver()` tarafından verilen isteklerden dolayı (`REQ_SCR_DPAGE` ve `REQ_SCR_UPAGE`) menü bir sayfa kayar.

17.6. Çok Sütunlu Menüler

Yukarıdaki örnekte `set_menu_format()` işlevini nasıl kullanacağınızı gördünüz. Sütun değişkeninin (3. değişken) ne yaptığından bahsetmedim. Aslında, eğer alt menünüz yeterince genişse bir satırda birden fazla öğe gösterimini seçebilirsiniz. Bu, sütun değişkenlerinde tanımlanabilir. İşleri daha basit hale getirmek için, aşağıdaki örnek öğeler için açıklamaları göstermemektedir.

Örnek 21. Çok Sütunlu Menü Örneği

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

#define ARRAY_SIZE(a) (sizeof(a) / sizeof(a[0]))
#define CTRLD   4

char *choices[] = {
  "Choice 1",  "Choice 2",  "Choice 3",  "Choice 4",  "Choice 5",
  "Choice 6",  "Choice 7",  "Choice 8",  "Choice 9",  "Choice 10",
  "Choice 11", "Choice 12", "Choice 13", "Choice 14", "Choice 15",
  "Choice 16", "Choice 17", "Choice 18", "Choice 19", "Choice 20",
  "Exit",
  (char *)NULL,
};

int main()
{ ITEM **my_items;
  int c;
  MENU *my_menu;
  WINDOW *my_menu_win;
  int n_choices, i;

  /* Curses kipini ilklendir */
  initscr();
  start_color();
  cbreak();
  noecho();
  keypad(stdscr, TRUE);
  init_pair(1, COLOR_RED, COLOR_BLACK);
  init_pair(2, COLOR_CYAN, COLOR_BLACK);

  /* Öğeleri oluştur */
  n_choices = ARRAY_SIZE(choices);
  my_items = (ITEM **)calloc(n_choices, sizeof(ITEM *));
```

```
  for(i = 0; i < n_choices; ++i)
    my_items[i] = new_item(choices[i], choices[i]);

  /* Menüyü oluştur */
  my_menu = new_menu((ITEM **)my_items);

  /* Menü seçeneğini açıklamayı göstermeyecek şekilde ayarla */
  menu_opts_off(my_menu, O_SHOWDESC);

  /* Menü ile ilişiklendirilecek pencereyi oluştur */
  my_menu_win = newwin(10, 70, 4, 4);
  keypad(my_menu_win, TRUE);

  /* Ana ve alt pencereyi oluştur */
  set_menu_win(my_menu, my_menu_win);
  set_menu_sub(my_menu, derwin(my_menu_win, 6, 68, 3, 1));
  set_menu_format(my_menu, 5, 3);
  set_menu_mark(my_menu, " * ");

  /* Ana pencere etrafında bir çerçeve çizer ve bir başlık yazar */
  box(my_menu_win, 0, 0);

  attron(COLOR_PAIR(2));
  mvprintw(LINES - 3, 0,
    "Menüyü kaydırmak için PageUp ve PageDown kullanın");
  mvprintw(LINES - 2, 0,
    "Hareket etmek için ok tuşlarını kullanın (Çıkmak için F1)");
  attroff(COLOR_PAIR(2));
  refresh();

  /* Ekranı yazdır */
  post_menu(my_menu);
  wrefresh(my_menu_win);

  while((c = wgetch(my_menu_win)) != KEY_F(1))
  { switch(c)
    { case KEY_DOWN:
        menu_driver(my_menu, REQ_DOWN_ITEM);
        break;
      case KEY_UP:
        menu_driver(my_menu, REQ_UP_ITEM);
        break;
      case KEY_LEFT:
        menu_driver(my_menu, REQ_LEFT_ITEM);
        break;
      case KEY_RIGHT:
        menu_driver(my_menu, REQ_RIGHT_ITEM);
        break;
      case KEY_NPAGE:
        menu_driver(my_menu, REQ_SCR_DPAGE);
        break;
      case KEY_PPAGE:
        menu_driver(my_menu, REQ_SCR_UPAGE);
        break;
    }
    wrefresh(my_menu_win);
  }
```

```
  /* Ekrandan sil ve tahsis edilen belleği geri ver */
  unpost_menu(my_menu);
  free_menu(my_menu);
  for(i = 0; i < n_choices; ++i)
    free_item(my_items[i]);
  endwin();
}
```

`set_menu_format()` işlev çağrısına bakın. Olması gereken sütun sayısını 3 olarak tanımlar ve bu yüzden de her satırda 3 öğe gözükür. `menu_opts_off()` ile de menü açıklamalarını kapatmıştık. Menü seçeneklerini değiştirmek için `set_menu_opts(), menu_opts_on()` ve `menu_opts()` gibi birkaç işlev vardır. Aşağıdaki menü seçenekleri belirtilebilir.

`O_ONEVALUE`
: Bu menü için sadece tek öğe seçilebilir.

`O_SHOWDESC`
: Menü ekrana yazıldığıda öğe açıklamalarını göster.

`O_ROWMAJOR`
: Menüyü satır-düzensel sırada göster.

`O_IGNORECASE`
: Örüntü eşlemede büyük küçük harfe duyarsız ol.

`O_SHOWMATCH`
: Örüntü eşleme sırasında imleci öğe isminin içerisine konumlandır.

`O_NONCYCLIC`
: İmleci menünün sonundan başına (veya tersi) aktarma.

Öntanımlı olarak tüm seçenekler etkindir. Belli özellikleri `menu_opts_on()` ve `menu_opts_off()` işlevleri ile açıp kapatabilirsiniz. Aynı zamanda doğrudan, bazı özellikleri `set_menu_opts()` ile belirtebilirsiniz. Bu işleve parametre yukarıdaki sabitlerle VEYA'lanmış bir değer olmalıdır. `menu_opts()` işlevi menünün halihazırdaki seçeneklerini bulmak için kullanılabilir.

17.7. Çok Değerli Menüler

`O_ONEVALUE` seçeneğini kapattığımızda ne olur diye merak ediyor olabilirsiniz. Bunun anlamı birden çok öğe seçebilirsiniz demektir. Bu da bizi `REQ_TOGGLE_ITEM` isteğine getirmektedir. Bir örnekle görelim:

Örnek 22. Çok Değerli Menü Örneği

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

#define ARRAY_SIZE(a) (sizeof(a) / sizeof(a[0]))
#define CTRLD   4

char *choices[] = {
  "Choice 1",
  "Choice 2",
  "Choice 3",
```

```
  "Choice 4",
  "Choice 5",
  "Choice 6",
  "Choice 7",
  "Exit",
};

int main()
{ ITEM **my_items;
  int c;
  MENU *my_menu;
  int n_choices, i;
  ITEM *cur_item;

  /* Curses kipini ilklendir */
  initscr();
  cbreak();
  noecho();
  keypad(stdscr, TRUE);

  /* Öğeleri ilklendir */
  n_choices = ARRAY_SIZE(choices);
  my_items = (ITEM **)calloc(n_choices + 1, sizeof(ITEM *));
  for(i = 0; i < n_choices; ++i)
    my_items[i] = new_item(choices[i], choices[i]);
  my_items[n_choices] = (ITEM *)NULL;

  my_menu = new_menu((ITEM **)my_items);

  /* Menüyü çok değerli yap */
  menu_opts_off(my_menu, O_ONEVALUE);

  mvprintw(LINES - 3, 0,
    "<SPACE>'i seçmek veya seçileni iptal için kullanın.");
  mvprintw(LINES - 2, 0,
    "<ENTER> 'ı o anki seçileni görmek için kullanın "
    "(Çıkmak için F1)");
  post_menu(my_menu);
  refresh();

  while((c = getch()) != KEY_F(1))
  { switch(c)
    { case KEY_DOWN:
        menu_driver(my_menu, REQ_DOWN_ITEM);
        break;
      case KEY_UP:
        menu_driver(my_menu, REQ_UP_ITEM);
        break;
      case ' ':
        menu_driver(my_menu, REQ_TOGGLE_ITEM);
        break;
      case 10:        /* Enter */
      { char temp[200];
        ITEM **items;

        items = menu_items(my_menu);
        temp[0] = '\0';
```

```
        for(i = 0; i < item_count(my_menu); ++i)
          if(item_value(items[i]) == TRUE)
          { strcat(temp, item_name(items[i]));
            strcat(temp, " ");
          }
        move(20, 0);
        clrtoeol();
        mvprintw(20, 0, temp);
        refresh();
      }
      break;
    }
  }

  free_item(my_items[0]);
  free_item(my_items[1]);
  free_menu(my_menu);
  endwin();
}
```

Vay canına, bir sürü yeni işlev daha. Teker teker her birini ele alalım. İlk olarak `REQ_TOGGLE_ITEM` ile başlayalım. Çok değerli bir menüde kullanıcı birden fazla öğe seçebilmeli veya seçimi iptal edebilmelidir. `REQ_TOGGLE_ITEM` isteği o anki seçimi değiştirir (etkinleştirme veya iptal etme). Bu durumda space tuşuna basıldığında `REQ_TOGGLE_ITEM` isteği `menu_driver`'a sonucu gerçekleştirmek için gönderilir.

İşte şimdi kullanıcı <ENTER>'a bastığında onun seçtiği öğeleri görürüz. Önce menü ile ilişiklendirilmiş öğeleri `menu_items()` işlevini kullanarak bulduk. Daha sonra öğenin seçili mi değil mi olduğunu anlamak için öğeler üzerinde gezindik. Eğer öğe seçiliyse `item_value()` işlevi `TRUE` döndürür. `item_count()` işlevi menüdeki öğe sayısını döndürür. Ayrıca öğe açıklamalarını `item_description()` ile alabilirsiniz.

17.8. Menü Seçenekleri

Evet, şimdi artık menünüzde pekçok özellikler ile birtakım farklılıklar oluşturmak için can atıyorsunuzdur. Biliyorum. Renk istiyorsunuz!!! Metin kipinde çalışan şu dos oyunları[B24] gibi hoş menüler oluşturmak istiyorsunuz. `set_menu_fore()` ve `set_menu_back()` işlevleri seçili öğenin görünüm özelliklerini değiştirmek için kullanılır. İsimleri yanıltıcıdır. Hiçbir işe yaramayacak olan menünün ön ve arka fon renklerini değiştirmezler.

`set_menu_grey()` işlevi menüdeki seçilemez öğelerinin görünüm özelliklerini değiştirmek için kullanılır. Bu da bizi bir öğe için oldukça ilginç ve tekil olan `O_SELECTABLE`'a getirmektedir. Bunu `item_opts_off()` işlevi ile kapatabiliriz ve bundan sonra öğe seçilemez hale gelir. Bu durum şu hoş pencere menülerindeki gri öğe durumuna benzemektedir. Bütün bunları pratiğe bir örnekle dökelim.

Örnek 23. Menü Seçenekleri Örneği

```
#include <menu.h>

#define ARRAY_SIZE(a) (sizeof(a) / sizeof(a[0]))
#define CTRLD   4

char *choices[] = {
  "Choice 1",
  "Choice 2",
  "Choice 3",
  "Choice 4",
  "Choice 5",
```

```
  "Choice 6",
  "Choice 7",
  "Exit",
};

int main()
{ ITEM **my_items;
  int c;
  MENU *my_menu;
  int n_choices, i;
  ITEM *cur_item;

  /* Curses kipini ilklendir */
  initscr();
  start_color();
  cbreak();
  noecho();
  keypad(stdscr, TRUE);
  init_pair(1, COLOR_RED, COLOR_BLACK);
  init_pair(2, COLOR_GREEN, COLOR_BLACK);
  init_pair(3, COLOR_MAGENTA, COLOR_BLACK);

  /* Öğeleri ilklendir */
  n_choices = ARRAY_SIZE(choices);
  my_items = (ITEM **)calloc(n_choices + 1, sizeof(ITEM *));
  for(i = 0; i < n_choices; ++i)
    my_items[i] = new_item(choices[i], choices[i]);
  my_items[n_choices] = (ITEM *)NULL;
  item_opts_off(my_items[3], O_SELECTABLE);
  item_opts_off(my_items[6], O_SELECTABLE);

  /* Menüyü oluştur */
  my_menu = new_menu((ITEM **)my_items);

  /* Menünün ön ve arka fon rengini ayarla */
  set_menu_fore(my_menu, COLOR_PAIR(1) | A_REVERSE);
  set_menu_back(my_menu, COLOR_PAIR(2));
  set_menu_grey(my_menu, COLOR_PAIR(3));

  /* Menüyü ekrana yaz */
  mvprintw(LINES - 3, 0,
    "Seçilen öğeyi görmek için <ENTER>'a basınd");
  mvprintw(LINES - 2, 0,
    "Hareket etmek için Yukarı ve Aşağı tuşlarını kullanın "
    "(Çıkmak için F1)");
  post_menu(my_menu);
  refresh();

  while((c = getch()) != KEY_F(1))
  { switch(c)
    { case KEY_DOWN:
        menu_driver(my_menu, REQ_DOWN_ITEM);
        break;
      case KEY_UP:
        menu_driver(my_menu, REQ_UP_ITEM);
        break;
      case 10: /* Enter */
```

```
        move(20, 0);
        clrtoeol();
        mvprintw(20, 0, "Seçilen öğe : %s",
                 item_name(current_item(my_menu)));
        pos_menu_cursor(my_menu);
        break;
    }
  }
  unpost_menu(my_menu);
  for(i = 0; i < n_choices; ++i)
    free_item(my_items[i]);
  free_menu(my_menu);
  endwin();
}
```

17.9. Faydalı Kullanıcı Göstericisi

Menüdeki her öğeye bir kullanıcı göstericisi atayabiliriz. Panellerdeki kullanıcı göstericisi ile aynı şekilde çalışmaktadır. Menü sistemi tarafından kullanılmazlar. Bunlar içerisine istediğiniz herhangi bir şeyi koyabilirsiniz. Ben genelde menü seçeneği seçildiğinde çalışacak işlevi (seçilen ve muhtemelen kullanıcının <ENTER>'ladığı) tutmak için kullanırım.

Örnek 24. Menü Kullanıcı Göstericilerinin Kullanımı

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

#define ARRAY_SIZE(a) (sizeof(a) / sizeof(a[0]))
#define CTRLD   4

char *choices[] = {
  "Choice 1",
  "Choice 2",
  "Choice 3",
  "Choice 4",
  "Choice 5",
  "Choice 6",
  "Choice 7",
  "Exit",
};
void func(char *name);

int main()
{ ITEM **my_items;
  int c;
  MENU *my_menu;
  int n_choices, i;
  ITEM *cur_item;

  /* Curses Kipi İlklendir */
  initscr();
  start_color();
  cbreak();
  noecho();
  keypad(stdscr, TRUE);
  init_pair(1, COLOR_RED, COLOR_BLACK);
```

```
  init_pair(2, COLOR_GREEN, COLOR_BLACK);
  init_pair(3, COLOR_MAGENTA, COLOR_BLACK);

  /* Öğeleri İlklendir */
  n_choices = ARRAY_SIZE(choices);
  my_items = (ITEM **)calloc(n_choices + 1, sizeof(ITEM *));
  for(i = 0; i < n_choices; ++i)
  { my_items[i] = new_item(choices[i], choices[i]);
    /* Set the user pointer */
    set_item_userptr(my_items[i], func);
  }
  my_items[n_choices] = (ITEM *)NULL;

  /* Menüyü oluştur */
  my_menu = new_menu((ITEM **)my_items);

  /* Menüyü ekrana yaz */
  mvprintw(LINES - 3, 0,
    "Seçili öğeyi görmek için <ENTER>'a basın");
  mvprintw(LINES - 2, 0,
    "Hareket etmek için Yukarı ve Aşağı ok tuşlarını kullanın "
    "(Çıkmak için F1)");
  post_menu(my_menu);
  refresh();

  while((c = getch()) != KEY_F(1))
  { switch(c)
    { case KEY_DOWN:
        menu_driver(my_menu, REQ_DOWN_ITEM);
        break;
      case KEY_UP:
        menu_driver(my_menu, REQ_UP_ITEM);
        break;
      case 10: /* Enter */
      { ITEM *cur;
        void (*p)(char *);

        cur = current_item(my_menu);
        p = item_userptr(cur);
        p((char *)item_name(cur));
        pos_menu_cursor(my_menu);
        break;
      }
      break;
    }
  }
  unpost_menu(my_menu);
  for(i = 0; i < n_choices; ++i)
    free_item(my_items[i]);
  free_menu(my_menu);
  endwin();
}

void func(char *name)
{ move(20, 0);
  clrtoeol();
  mvprintw(20, 0, "Seçili öge: %s", name);
```

```
}
```

18. Form Kütüphanesi

Aslında, eğer web sayfalarında şu kullanıcıdan girdi alan ve bir takım işler yapan formları gördüyseniz, bunların metin kipinde nasıl oluşturulacağını merak ediyor olabilirsiniz. Bu etkileyici formların yalın ncurses ortamında yazılması biraz zordur. Form kütüphanesi formların kolay şekilde oluşturulması ve üzerinde çalışılması için temel bir çalışma taslağı oluşturur. Doğrulama, alanların çalışma zamanında genişlemesi v.b. gibi pekçok özellikleri (işlevleri) vardır. Hepsini görelim.

Bir form alanların bir araya gelmesinden oluşur; her bir alan bir etiket (durağan metin) veya veri girişi alanı olabilir. Forms, formların birden fazla sayfaya bölünmesini sağlayan bir kütüphanedir.

18.1. Temel Bilgiler

Formlar menülerle aynı şekilde oluşturulur. Önce form ile ilgili alanlar `new_field()` ile oluşturulur. Alanlar için bir takım güzel özellikler ile gösterilmelerini sağlayan seçenekler tanımlayabilirsiniz: alandan imleci çekmeden önce alan içerisine yazılanı kontrol etmek v.b.. Daha sonra alanlar formlara ilintilenir. Bundan sonra form görüntülenmek için ve girdi almak için hazırdır. `menu_driver()` satırlarına benzer şekilde form için sürücü, `form_driver()` ile değiştirilir. `form_driver`'a bir takım istekler göndererek belli alanlara odaklanabiliriz, alanın sonuna konumlanabiliriz, v.b.. Kullanıcının alanlara veri girmesinden ve doğrulamanın yapılmasından sonra form ekrandan silinip ayrılan bellek alanları geri verilebilir.

Bir form programının genel akışı şu şekildedir:

1. Curses kipini ilklendirilir
2. `new_field()` kullanılarak alanlar oluşturulur. Alanın form üzerindeki konumu, yükseklik ve genişliğini tanımlayabilirsiniz.
3. Formlar kendilerine ilintilenecek alanlar ile birlikte `new_form()` kullanarak oluşturulur.
4. `form_post()` ile ekrana yazılır ve ekran tazelenir.
5. Bir döngü ile kullanıcıdan gelen istekleri işlenir ve `form_driver` ile gerekli güncelleştirmeler yapılır.
6. Form ekrandan `form_unpost()` ile silinir.
7. `free_form()` ile tahsis edilen bellek alanını geri verilir.
8. Alanlar için ayrılan bellek alanını `free_item()` ile geri verilir.
9. Curses kipi sonlandırılır.

Gördüğünüz gibi form kütüphanesi ile çalışmak menüler üzerinde işlemler yapmaya benzemektedir. Aşağıdaki örnekler form işlemenin değişik durumlarını gösterecektir. Basit bir örnekle yolculuğumuza başlayalım.

18.2. Form Kütüphanesi ile derleme

Form kütüphanesi işlevlerini kullanmak için `form.h` başlık dosyasını programınıza eklemeniz ve kütüphane ile ilintileyerek derlemek için `-lform -lncurses` seçeneklerini kullanmanız gerekmektedir.

```
#include <form.h>
.
.
.

derleme ve ilintileme: gcc program dosyası -lform -lncurses
```

Örnek 25. Formların Temelleri

```
#include <form.h>

int main()
{ FIELD *field[3];
  FORM  *my_form;
  int ch;

  /* Curses kipini ilklendir*/
  initscr();
  cbreak();
  noecho();
  keypad(stdscr, TRUE);

  /* Alanları ilklendir */
  field[0] = new_field(1, 10, 4, 18, 0, 0);
  field[1] = new_field(1, 10, 6, 18, 0, 0);
  field[2] = NULL;

  /* alan seçeneklerini ayarla*/
  set_field_back(field[0], A_UNDERLINE);  /* Seçenek için bir satır yazın  */
  field_opts_off(field[0], O_AUTOSKIP);   /* Eğer alan dolduyda            */
                                          /* alan içerisinde ilerleme      */
  set_field_back(field[1], A_UNDERLINE);
  field_opts_off(field[1], O_AUTOSKIP);

  /* Form2u oluştur ve ekrana yaz */
  my_form = new_form(field);
  post_form(my_form);
  refresh();

  mvprintw(4, 10, "Değer 1:");
  mvprintw(6, 10, "Değer 2:");
  refresh();

  /* Kullanıcı isteklerinde hareket edin */
  while((ch = getch()) != KEY_F(1))
  { switch(ch)
    { case KEY_DOWN:
        /* Sonraki alana gidin */
        form_driver(my_form, REQ_NEXT_FIELD);
        /* O anki önbelleğin sonuna gidin */
        /* Son karakterde döngüden çık */
        form_driver(my_form, REQ_END_LINE);
        break;
      case KEY_UP:
        /* Bir önceki alana git */
        form_driver(my_form, REQ_PREV_FIELD);
        form_driver(my_form, REQ_END_LINE);
        break;
      default:
        /* Eğer normal bir karakterse */
        /* Ekrana yazılır             */
        form_driver(my_form, ch);
```

```
        break;
    }
  }

  /* Formu ekrandan sil ve ayrılan bellek alanını geri ver */
  unpost_form(my_form);
  free_form(my_form);
  free_field(field[0]);
  free_field(field[1]);

  endwin();
  return 0;
}
```

Yukarıdaki örnek oldukça açık şekildedir. **new_field()** ile iki alan oluşturur. new_field() parametre olarak yükseklik, genişlik, y başlangıç koordinatı, x başlangıç koordinatı, ekranda gözükmeyen satır sayısı ve ilave çalışma önbelleği sayısını almaktadır. Beşinci parametre, ekranda gözükmeyen satır sayısı alanın ne kadarının gözükeceğini belirtir. Eğer sıfırsa, tüm alan gösterilir, aksi halde kullanıcı alanın görüntülenmeyen alanlarına taştığında form kaydırılabilir olacaktır. Form kütüphanesi kullanıcının girdiği verilerin saklanması için her alan için bir önbellek tahsis eder. new_field()'in son parametresini kullanarak ilave ek bellekler tahsis etmesini sağlayabiliriz. İstediğiniz herhangi bir amaç için kullanılabilir.

Alanları oluşturduktan sonra, set_field_back() ile artalan özellikleri bir alt çizgi olarak ayarlanır. AUTOSKIP seçeneği field_opts_off() kullanılarak kapatılır. Eğer bu özellik açıksa, odak eğer üzerinde işlem yapılan alan tamamen dolduysa bir sonraki alana kayacaktır.

Alanlar forma ilintilendikten sonra ekrana yazılır. İşte şu while döngüsü içerisinde de kullanıcı girdileri form_driver'daki uygun karşılıkları ile işlenir. form_driver()'a yapılacak olan isteklerin ayrıntıları daha sonra açıklanacaktır.

18.3. Alanlar ile Oynamak

Her bir form alanı birden fazla özellik ile ilintilenmiştir. İstenilen özelliği elde etmek ve eğlenmek için değiştirilebilirler. Daha ne bekliyoruz?

18.3.1. Alan Konumunun ve Boyutununun Alınması

Alan oluşturulması sırasında verdiğimiz parametreler field_info() ile okunabilir. Kendisine verilen parametrelere yükseklik, genişlik, y başlama koordinatı, x başlama koordinatı, görünmeyen satır sayısı ve ilave önbellek miktarını geri döndürür. new_field() bu işlevin bir çeşit ters işlevidir.

işlev
```
int
field_info (FIELD *alan,          /* döndürülecek alan */
            int    *yukseklik,    /* alan yüksekliği */
            int    *genislik,     /* alan genişliği */
            int    *ust,          /* üst kenar */
            int    *sol,          /* sol kenar */
            int    *gorunmeyen,   /* görünmeyen satır sayısı */
            int    *tampon_boyu); /* tampon boyu */
```

18.3.2. Alanı taşımak

Alanın konumu farklı bir konuma move_field() ile taşınabilir.

```
int                                                                    işlev
move_field (FIELD *alan, /* değiştirilecek alan */
            int     ust,  /* yeni üst kenar */
            int     sol); /* yeni sol kenar */
```

Alışıldığı gibi, değişen konum bilgileri `field_infor()` ile alınabilir.

18.3.3. Alan Hizalama

Alanlar için hizalama `set_field_just()` işlevi ile yapılabilir.

```
int                                                                    işlev
set_field_just (FIELD *alan, /* değiştirilecek alan */
                int     kip); /* ayarlanacak kip */
```

```
int                                                                    işlev
field_just (FIELD *alan); /* alan için hizalama kipi */
```

Bu işlevler tarafından kullanılan ve geri döndürülen hizalama kipleri: `NO_JUSTIFICATION`, `JUSTIFY_RIGHT`, `JUSTIFY_LEFT`, or `JUSTIFY_CENTER`

18.3.4. Alan Görüntüleme Öznitelikleri

Gördüğünüz gibi, yukarıdaki örnekte alan için görünüm öznitelikleri `set_field_fore()` ve `setfield_back()` ile ayarlanabilir. Bu işlevler alanların ön ve artalan özelliklerini ayarlar. Aynı zamanda alanın doldurulmamış kısımları için bir tampon karakteri tanımlayabilirsiniz. Tampon karakteri `set_field_pad()` çağrısıyla tanımlanır. Varsayılan tampon karakteri boşluktur. `field_fore(), field_back, field_pad()` işlevleri ön, artalan özniteliklerini ve tampon karakterini sorgulamak için kullanılabilir. Aşağıdaki liste işlevlerin kullanımını vermektedir.

```
int                                                                    işlev
set_field_fore (FIELD *alan,     /* değiştirilecek alan */
                chtype  öznitelik); /* ayarlanacak öznitelik */
```

```
chtype                                                                 işlev
field_fore (FIELD *alan); /* sorgulanacak alan */
                          /* önalan özniteliğini döndürür */
```

```
int                                                                    işlev
set_field_back (FIELD *alan,     /* değiştirilecek alan */
                chtype  öznitelik); /* ayarlanacak öznitelik */
```

```
chtype                                                                 işlev
field_back (FIELD *alan); /* sorgulanacak alan */
                          /* artalan özniteliğini döndürür */
```

```
int                                                                    işlev
set_field_pad (FIELD *alan,      /* değiştirilecek alan */
               int     dolgu_krk); /* adımlama karakteri */
```

```
chtype                                                          işlev
field_pad (FIELD *alan); /* sorgulanacak alan */
                         /* adımlama karakterini döndürür */
```

Her ne kadar yukarıdaki işlevler oldukça basit gözükse de `set_field_fore()` ile renkleri kullanmak başlangıçta korkutucu olabilir. Öncelikle alanın ön ve artalan özniteliklerini açıklamama izin verin. Önalan özniteliği karakter ile ilişiklendirilmiştir. Bunun anlamı bir karakterin `set_field_fore()` ile ayarlanmış öznitelik ile gösterilmesidir. Artalan özniteliği alanın arka fonunu doldurmak için kullanılan özelliktir, bir karakter olmak zorunda değildir. Peki ya renkler? Renkler çiftler halinde tanımlandığından renklendirilmiş alanları göstermenin en iyi yolu nedir? İşte renk görüntü özelliklerini netleştiren bir özellik:

Örnek 26. Form Öznitelikleri Örneği

```
#include <form.h>

int main()
{ FIELD *field[3];
  FORM  *my_form;
  int ch;

  /* Curses kipini ilklendir */
  initscr();
  start_color();
  cbreak();
  noecho();
  keypad(stdscr, TRUE);

  /* Bazı renk çiftlerini ilklendir */
  init_pair(1, COLOR_WHITE, COLOR_BLUE);
  init_pair(2, COLOR_WHITE, COLOR_BLUE);

  /* alanları ilklendir */
  field[0] = new_field(1, 10, 4, 18, 0, 0);
  field[1] = new_field(1, 10, 6, 18, 0, 0);
  field[2] = NULL;

  /* Alan özelliklerini ayarla */
  /* Alanların arka fonunu mavi yap */
  set_field_fore(field[0], COLOR_PAIR(1));
  /* ön fonun beyaz ya */
  set_field_back(field[0], COLOR_PAIR(2));
  /* karakterler beyaz olarak gösterilir         */
  /* bu alan dolduğunda */
  field_opts_off(field[0], O_AUTOSKIP);
  /* sonraki alana geçme           */
  set_field_back(field[1], A_UNDERLINE);
  field_opts_off(field[1], O_AUTOSKIP);

  /* Alanı oluştur ve ekrana yaz */
  my_form = new_form(field);
  post_form(my_form);
  refresh();

  /* Renklendirilmiş alana odaklan */
  set_current_field(my_form, field[0]);
  mvprintw(4, 10, "Değer 1:");
```

```
  mvprintw(6, 10, "Değer 2:");
  mvprintw(LINES - 2, 0,
      "Alanlar arasında geçiş için YUKARI ve AŞAĞI ok tuşlarını kullanın");
  refresh();

  /* Kullanıcı isteklerini almak için döngü */
  while((ch = getch()) != KEY_F(1))
  { switch(ch)
    { case KEY_DOWN:
        /* Sonraki alana git */
        form_driver(my_form, REQ_NEXT_FIELD);
        /* O anki önbelleğin sonuna git */
        /* Son karakterde döngüden çık */
        form_driver(my_form, REQ_END_LINE);
        break;
      case KEY_UP:
        /* Önceki alana git */
        form_driver(my_form, REQ_PREV_FIELD);
        form_driver(my_form, REQ_END_LINE);
        break;
      default:
        /* Eğer bu normal bir karakterse */
        /* ekrana yazılır                */
        form_driver(my_form, ch);
        break;
    }
  }

  /* Formu ekrandan sil ve ayrılan belleği geri ver */
  unpost_form(my_form);
  free_form(my_form);
  free_field(field[0]);
  free_field(field[1]);

  endwin();
  return 0;
}
```

Renk çiftleriyle oynayarak ön ve artalan özniteliklerini anlamaya çalışın. Genelde sadece `set_field_back()` ile artalanı ayarlarım. Curses tek tek renk özniteliklerinin doğrudan ayarlanmasını izin vermemektedir.

18.3.5. Alan Seçenek Bitleri

Bunların yanında ayrıca form işlemenin pekçok yönünü kontrol etmenizi sağlayan geniş bir alan seçenekleri bit kümesi de bulunmaktadır. Şu işlevlerle bunları değiştirebilirsiniz:

```
int                                                              işlev
set_field_opts (FIELD *alan,     /* değiştirilecek alan */
                int     öznitelik); /* ayarlanacak öznitelik */
```

```
int                                                              işlev
field_opts_on (FIELD *alan,     /* değiştirilecek alan */
               int     öznitelik); /* etkinleştirilecek öznitelik */
```

```
int                                                                    işlev
field_opts_off (FIELD *alan,     /* değiştirilecek alan */
                int      öznitelik); /* etkisizleştirilecek öznitelik */
```

```
int                                                                    işlev
field_opts (FIELD *alan); /* sorgulanacak alan */
```

`set_field_opts()` işlevi bir alanın özelliklerini değiştirmek için doğrudan kullanılabilir veya `field_opts_on()` ve `field_opts_off()` ile bazı özellikleri seçime göre etkinleştirebilir veya kapatabilirsiniz. Herhangi bir zamanda bir alanın özelliklerini `field_opts()` ile sorgulayabilirsiniz. Aşağıda kullanılabilir özellikler listesi bulunmaktadır. Öntanımlı olarak, tüm özellikler etkindir.

O_VISIBLE
: Alanın ekranda görünüp görünmediğini kontrol eder. Ana alanın özelliğine bağlı olarak form işleme sırasında alanları saklamak ya da göstermek için kullanılır.

O_ACTIVE
: Form işleme sırasında alanın etkin olup olmadığını kontrol eder (örn. form gezinti tuşları tarafından ziyaret edilimiş mi?). Kullanıcı tarafından değil ama form uygulamasının kendisi tarafından değiştirilebilen önbellek değerleri ile türetilmiş alanlar veya etiketler yapmak için kullanılabilir.

O_PUBLIC
: Alana veri girişi sırasında verinin gösterilip gösterilmediğini kontrol eder. Eğer bu seçenek bir alan için kapatılırsa kütüphane, alan içerisindeki verileri alıp düzenleyecek, fakat veriler görülebilir olmayacak ve görünmesi gereken imleç de ilerlemeyecektir. Parola alanlarını tanımlamak için `O_PUBLIC` bitini kapatabilirsiniz.

O_EDIT
: Bir alan verisinin değiştirilip değiştirilemeyeceğini kontrol eder. Bu özellik kapalı olduğunda, `REQ_PREV_CHOICE` ve `REQ_NEXT_CHOICE` dışındaki tüm düzenleme istekleri sekteye uğrayacaktır. Bu sadece okunabilir alanlarda yardım iletileri için faydalı olabilir.

O_WRAP
: Çok satırlı alanlarda satırın sarmalanmasını sağlar. Normal çalışma şeklinde (boşluklarla ayrılmış) bir kelimenin herhangi bir karakteri o anki satırın sonuna geldiğinde, kelimenin tamamı bir sonraki satıra (yeni bir satır olduğunu varsayıyoruz) olduğu gibi taşınır. Bu seçenek etkisizleştirildiğinde satırsonu ile beraber kelime parçalanacaktır.

O_BLANK
: Alanların boşaltılmasını kontrol eder. Bu seçenek etkin olduğunda, alanın ilk konumuna bir karakter girmek tüm alanı siler (hali hazırda var olan karakterler dışında).

O_AUTOSKIP
: Alanın kendisi dolduğunda otomatik olarak sonraki alana atlamayı kontrol eder. Normalinde, form kullanıcısı alana alabileceğinden fazlasını yazmaya kalkarsa yazılanlar sonraki alana geçer. Bu seçenek etkisiz olduğunda kullanıcı imleci alanın sonunda ilerlemeden duracaktır. Bu özellik çalışma zamanında boyutları değişen alanlar için ihmal edilir.

O_NULLOK
: Boş alanlara doğrulama yapılıp yapılmadığını kontrol eder. Normal olarak, bu yapılmaz; kullanıcı bir alanı boş bırakıp çıkabilir. Eğer bu seçenek bir özellik için etkisizse, alandan çıkarken doğrulama yapılır.

O_PASSOK

Alandaki odaklanmadan her ayrılmada doğrulama yapılacak mı yoksa sadece alan değiştiğinde mi bir doğrulama yapılacak diye kontrol eder. Normalinde sonuncusu doğrudur. O_PASSOK'u ayarlamak form işlemeniz sırasında eğer alanın doğrulama işlevi değişiyorsa faydalı olabilir.

O_STATIC
Alanın ilk boyutunda sabit olup olmadığını kontrol eder. Bu seçeneği etkisizleştirdiğinizde, alan çalışma zamanında değişken bir hal alır ve yazılan girdiye göre genişleyebilir bir özellik kazanır.

Bir alanın özelliği o alan seçildiği anda değiştirilemez. Yine de, henüz seçilmemiş fakat ekranda gözüken alanların seçenekleri değiştirilebilir.

Seçenek değerleri bit maskeleridir ve açık şekilde, mantıksal VEYA kullanarak oluşturulabilir. O_AUTOSKIP değerinin etkisizleştirilmesini daha önce görmüştünüz. Aşağıdaki örnek birkaç seçeneğin daha kullanımını açıklamaktadır. Diğer seçenekler gerekli görüldüğü yerde açıklanmıştır.

Örnek 27. Alan Özelliklerini Kullanım Örneği

```
#include <form.h>

#define STARTX 15
#define STARTY 4
#define WIDTH 25

#define N_FIELDS 3

int main()
{ FIELD *field[N_FIELDS];
  FORM  *my_form;
  int ch, i;

  /* Curses kipini ilklendir */
  initscr();
  cbreak();
  noecho();
  keypad(stdscr, TRUE);

  /* Alanları ilklendir */
  for(i = 0; i < N_FIELDS - 1; ++i)
    field[i] = new_field(1, WIDTH, STARTY + i * 2, STARTX, 0, 0);
  field[N_FIELDS - 1] = NULL;

  /* Alan özelliklerini ayarla */
  set_field_back(field[1], A_UNDERLINE);  /* Seçenek için bir satır yaz  */

  field_opts_off(field[0], O_ACTIVE); /* Bu alan durağan bir etiket */
  field_opts_off(field[1], O_PUBLIC); /* Bu alan bir parola alanı gibi*/
  field_opts_off(field[1], O_AUTOSKIP); /* Son karakter girildikten sonra */
                                        /* aynı alana girişi önlemek için */

  /* Alanı oluştur ve ekrana yaz */
  my_form = new_form(field);
  post_form(my_form);
  refresh();

  set_field_just(field[0], JUSTIFY_CENTER); /* Ortalama ayarı */
  set_field_buffer(field[0], 0, "Bu durağan bir alandır");
```

```
                                              /* Alanı ilklendir  */
  mvprintw(STARTY, STARTX - 10, "Alan 1:");
  mvprintw(STARTY + 2, STARTX - 10, "Alan 2:");
  refresh();

  /* Kullanıcıdan elen istekler için dögü */
  while((ch = getch()) != KEY_F(1))
  { switch(ch)
    { case KEY_DOWN:
        /* Sonraki alana git */
        form_driver(my_form, REQ_NEXT_FIELD);
        /* O anki önbelleğin sonuna git */
        /* Son karakterde döngüden çık */
        form_driver(my_form, REQ_END_LINE);
        break;
      case KEY_UP:
        /* Önceki alana git */
        form_driver(my_form, REQ_PREV_FIELD);
        form_driver(my_form, REQ_END_LINE);
        break;
      default:
        /* Eğer bu normal bir karakterse */
        /* Ekrana yazılır                */
        form_driver(my_form, ch);
        break;
    }
  }

  /* Formu ekrandan sil ve tahsis edilen belleği geri ver */
  unpost_form(my_form);
  free_form(my_form);
  free_field(field[0]);
  free_field(field[1]);

  endwin();
  return 0;
}
```

Bo örnek, işe yaramaz olsa da, seçeneklerin kullanımını göstermektedir. Eğer düzgün kullanılırsa, bir form içerisindeki bilgileri çok verimli şekilde gösterebilirler. `O_PUBLIC` olmayan ikinci alan yazdığınız karakterleri göstermez.

18.3.6. Alan Durumu

Alan durumu alan üzerinde düzenleme yapılıp yapılmadığını söyler. Başlangıçta `FALSE` değerindedir. Kullanıcı bir takım veriler girdiğinde ve veri belleğini değiştirdiğinde `TRUE` değerini alır. Dolayısıyla bir alanın durum bilgisi, alan verisi üzerinde değişiklik yapılıp yapılmadığının kontrolü için kullanılabilir. Aşağıdaki işlevler bu türden işlemlere yardım edebilir:

```
int                                                              işlev
set_field_status (FIELD *alan,    /* değiştirilecek alan */
                  int     durum); /* atanacak durum */
```

```
int                                                                    işlev
field_status (FIELD *alan); /* sorgulanacak alan */
                            /* alanın durumu ile döner */
```

Alanın durumunu alandan ayrılırken kontrol etmek, veri doğrulama halen devam ettiğinden ve veri önbelleği henüz değişmemiş olabileceğinden daha sağlıklıdır. Dönen doğru değeri almak için, alanın ayrılma ile ilgili kısmına veya alan veya formun ilklendirme veya sonlandırılma kısımlarında veya hemen form sürücüsü tarafından `REQ_VALIDATION` isteği işlendikten sonra `field_status()` çağrılmalıdır.

18.3.7. Alan Kullanıcı Göstericisi

Her alan yapısı kullanıcı tarafından değişik amaçlar için kullanılabilen bir gösterici içerir. Form kütüphanesi tarafından kullanılmaz ama kullanıcı tarafından herhangi bir amaç için kullanılabilir. Sıradaki işlevler kullanıcı göstericilerini ayarlar ve onları eşler.

```
int                                                                    işlev
set_field_userptr (FIELD *alan,      /* değiştirilecek alan */
                   char   *kull_gost); /* ilişiklendirilecek kullanıcı
                                          göstericisi */
```

```
int                                                                    işlev
field_userptr (FIELD *alan); /* sorgulanacak alan */
                             /* kullanıcı göstericisini döndürür */
```

18.3.8. Değişken Boydaki Alanlar

Eğer çalışma zamanında değişken genişlikte bir alan istiyorsanız bu, tüm özellikleri ile kullanmanız gereken bir özelliktir. Bu, kullanıcının, asıl alan boyundan daha büyük değerler girmesini ve alanın büyümesini sağlar. Alan, yerleştirilme konumuna göre yeni girilen veriler ile bütünleşik olabilmek için yatay veya dikey kaydırılabilir olacaktır.

Bir alanın dinamik olarak büyümesini sağlamak için `O_STATIC` seçeneği etkisiz olmalıdır. Bu da,

```
field_opts_off(field_pointer, O_STATIC);
```

ile yapılabilir. Fakat bir alanın sonsuz boyda uzamasına izin vermek pek de tavsiye edilmez. Alanın genişleyebileceği en fazla büyüklüğü tanımlayabilirsiniz.

```
int                                                                    işlev
set_max_field (FIELD *alan,      /* değiştirilecek alan */
               char    azm_buy); /* alan için olası azami büyüme miktarı */
```

Dinamik olarak büyüyebilen bir alandan bilgiler şu şekilde alınabilir:

```
int                                                                    işlev
dynamic_field_info (FIELD *alan,      /* değiştirilecek alan */
                    int    *sat_say,  /* sığdırılacak satır sayısı */
                    int    *sut_say,  /* sığdırılacak sütun sayısı */
                    int    *azm_buy); /* sığdırılacak azami büyüme miktarı */
```

`field_inf` bildik şekilde çalışmasına rağmen, çalışma zamanında genişleyen alanların özellikleri ile ilgili bilgileri bu işlevi kullanarak almanız tavsiye edilir.

new_field kütüphane yordamını hatırlayın; bir birim yüksekliğinde ve tek satırlık bir alan oluşturuluyordu. Bir birimden fazla yüksekliğe sahip alanlar çok satırlı alanlar olarak tanımlanacaklardır.

O_STATIC özelliği etkisizleştirilmiş tek satırlık bir alan tek bir sabit satır içerecektir, fakat sütun sayısı kullanıcının ilk alanın taşıyabileceğinden fazlasını girmesi durumunda artabilir. Görüntülenen alan sayısı sabit kalacak ve ilave veriler kaydırılabilir olacaktır.

Çok satırlı ve O_STATIC özelliği etksizleştrilmiş bir alan ise (çalışma zamanında genişleyebilen) sabit sayıda sütun içerecek, fakat satır sayısı alanın ilk tutabileceği değerden daha fazlası girilmesi durumunda artacaktır. Görüntülenen satır sayısı sabit olacak ve ilave veriler kaydırılabilir olacaktır.

Yukarıdaki iki paragraf çalışma zamanında genişleyen alanların davranış özellikleriyle ilgili oldukça açıklayıcıdır. Diğer form alanlarının davranış özellikleri aşağıda açıklanmıştır:

1. O_AUTOSKIP seçeneği eğer O_STATIC etkisiz ve alan için tanımlanmış azami büyüyebilme miktarı tanımlı değilse gözardı edilecektir. Şu anda, kullanıcı alanın en sonuna gelip bir karakter yazdığında O_AUTOSKIP otomatik şekilde REQ_NEXT_FIELD form sürücüsü isteğini üretmektedir. En fazla büyüyebilme miktarı tanımlanmamış genişleyebilir bir alanda son karakter için bir konum yoktur. Eğer en fazla genişleme oranı tanımlanmışsa, O_AUTOSKIP seçeneği eğer alan en büyük değerine kadar genişlediyse normal çekilde çalışacaktır.

2. O_STATIC özelliği eğer etkisizse alan içerisinde hizalama gözardı edilecektir. Şu anda, set_field_just tek satıra JUSTIFY_LEFT, JUSTIFY_RIGHT, JUSTIFY_CENTER özellikleri kazandırmak için kullanılabilir. Genişleyebilir tek satırlık alan yatay olarak genişleyip kaydırılabilir ve hizalanabilen çok daha fazla veri içerebilir. field_just'ten geri dönen değer değişmemiş olacaktır.

3. Aşırı yüklenmiş form sürücü isteği REQ_NEW_LINE, O_NL_OVERLOAD form seçeneğinde O_STATIC değerinin etkisiz olmasına ve alan için en fazla genişleme miktarı tanımlanmamasına bakmaksızın aynı şekilde çalışır. Şu anda eğer O_NL_OVERLOAD etkinse, REQ_NEW_LINE eğer alanın son satırında çağrılırsa gizli olarak REQ_NEXT_FIELD üretir. Eğer bir alan sınırlara bağlı kalmadan genişleyebiliyorsa, son bir satır yoktur, REQ_NEW_LINE hiç bir zaman gizli olarak REQ_NEXT_FIELD üretmez. Eğer en fazla büyüme miktarı belirtilmişse ve O_NL_OVERLOAD form özelliği etkinse, eğer alan büyüyebileceği en büyük büyüklüğe gelmiş ve son satırda ise REQ_NEW_LINE sadece gizli olarak REQ_NEXT_FIELD üretecektir.

4. dup_field çağrısı bilindiği gibi çalışır; alanı o anki önbellek boyunu ve alanın içeriği ile ikileştirecektir. Belirtilmiş en fazla büyüme miktarı da ikileştirilecektir.

5. link_field bilindiği gibi çalışacaktır; O an bağlanan alan ile birlikte tüm alan özelliklerini ve paylaşılan önbelleği ikileştirecektir. Eğer O_STATIC alan seçeneği alanı paylaşımlı önbellek tarafından ardışıl olarak değiştirilirse, önbelleğin sahip olduğu veri miktarından fazlasının alana girilmesi durumunda nasıl davranış göstereceği o anki alanın özellik ayarına bağlıdır.

6. field_info bilindiği gibi çalışacaktır; *sat_say* değişkeni new_field'a yapılan asıl çağrının değerini tutacaktır. Kullanıcı o anki önbellek boyutunu sorgulamak için dynamic_field_info'yu kullanmalıdır.

Yukarıdakilerin bazıları sadece form sürücüsünden sonra açıklanınca anlam kazanmaktadır. Sonraki bir kaç kısımda buna bakıyor olacağız.

18.4. Form Pencereleri

Form penceresi mantığı menü pencerelerinkine oldukça benzemektedir. Her bir form bir ana pencere ve alt pencere ile ilişiklendirilir. Ana pencere başlık, etiket, çerçeve veya kullanıcı her ne istiyorsa gösterir. Alt pencere

ise tüm alanları içerir ve onları konumlarına göre gösterir. Bu durum hoş görüntüler üzerinde esnek şekilde değişiklik yapma imkanı verir.

Menü pencerelerine çok benzediğinden oldukça açıklamalı bir örnek sunuyorum. İşlevler aynıdır ve aynı şekilde çalışır.

Örnek 28. Form Pencereleri Örneği

```
#include <form.h>

void print_in_middle(WINDOW *win, int starty, int startx, int width,
     char *string, chtype color);

int main()
{
  FIELD *field[3];
  FORM  *my_form;
  WINDOW *my_form_win;
  int ch, rows, cols;

  /* Curses kipi ilklendir */
  initscr();
  start_color();
  cbreak();
  noecho();
  keypad(stdscr, TRUE);

  /* Renk çiftlerini ilklendir */
  init_pair(1, COLOR_RED, COLOR_BLACK);

  /* Alanları ilklendir */
  field[0] = new_field(1, 10, 6, 1, 0, 0);
  field[1] = new_field(1, 10, 8, 1, 0, 0);
  field[2] = NULL;

  /* alan seçeneklerini ayarla */
  set_field_back(field[0], A_UNDERLINE);
  field_opts_off(field[0], O_AUTOSKIP); /* Bu alan dolduğunda */
                                        /* sonraki alana geçme */
  set_field_back(field[1], A_UNDERLINE);
  field_opts_off(field[1], O_AUTOSKIP);

  /* Formu oluştur ve ekrana yaz */
  my_form = new_form(field);

  /* Form için gerekli alanı hesapla */
  scale_form(my_form, &rows, &cols);

  /* Form ile ilişiklendirilecek pencereyi oluştur */
  my_form_win = newwin(rows + 4, cols + 4, 4, 4);
  keypad(my_form_win, TRUE);

  /* Ana pencereyi ve alt pencereyi ata */
  set_form_win(my_form, my_form_win);
  set_form_sub(my_form, derwin(my_form_win, rows, cols, 2, 2));

  /* Ana pencere etrafında bir çerçeve çiz ve başlık yaz */
```

```
  box(my_form_win, 0, 0);
  print_in_middle(my_form_win, 1, 0, cols + 4,
    "Benim Formum", COLOR_PAIR(1));

  post_form(my_form);
  wrefresh(my_form_win);

  mvprintw(LINES - 2, 0,
    "YUKARI ve AŞAĞI ok tuşlarıyla alanlar arasında geçiş yapın");
  refresh();

  /* Kullanıcı istekleri için döngü */
  while((ch = wgetch(my_form_win)) != KEY_F(1))
  { switch(ch)
    { case KEY_DOWN:
        /* sonraki alana git */
        form_driver(my_form, REQ_NEXT_FIELD);
        /* O anki önbelleğin sonuna git */
        /* Son karakterde döngüden çık */
        form_driver(my_form, REQ_END_LINE);
        break;
      case KEY_UP:
        /* Önceki alana git */
        form_driver(my_form, REQ_PREV_FIELD);
        form_driver(my_form, REQ_END_LINE);
        break;
      default:
        /* Eğer bu normak bir karakterse */
        /* Ekrana yazılır                */
        form_driver(my_form, ch);
        break;
    }
  }

  /* Ekrandan sil ve ayrılan bellek alanını geri ver */
  unpost_form(my_form);
  free_form(my_form);
  free_field(field[0]);
  free_field(field[1]);

  endwin();
  return 0;
}

void print_in_middle(WINDOW *win, int starty, int startx, int width,
     char *string, chtype color)
{ int length, x, y;
  float temp;

  if(win == NULL)
    win = stdscr;
  getyx(win, y, x);
  if(startx != 0)
    x = startx;
  if(starty != 0)
    y = starty;
  if(width == 0)
```

```
    width = 80;

  length = strlen(string);
  temp = (width - length)/ 2;
  x = startx + (int)temp;
  wattron(win, color);
  mvwprintw(win, y, x, "%s", string);
  wattroff(win, color);
  refresh();
}
```

18.5. Alan Doğrulama

Varsayılan olarak bir alan kullanıcıdan herhangi bir girdiyi kabul eder. Alana bir doğrulama özelliği ilintilemek mümkündür. Böylece kullanıcının alanı terk ederken alan içerisindeki veriler onaylanması gereken şekille eşleşmezse o zaman işlem sekteye uğrayacaktır. Bazı doğrulama türleri karakter seviyesinde alana her bir karakter girilirken gerçeklenmektedir.

Alana doğrulama özelliği aşağıdaki işlevlerle ilintilenebilir.

```
int                                                                    işlev
set_field_type (FIELD    *alan,        /* değiştirilecek alan */
                FIELDTYPE *alanozelligi, /* ilintilenecek özellik */
                ...);                 /* ilave parametreler */
```

Bir kere ayarlandıktan sonra bir alan için doğrulama türü şu şekilde sorgulanabilir:

```
FIELDTYPE *                                                            işlev
field_type (FIELD *alan); /* sorgulanacak alan */
```

Form sürücüsü bir alan içerisindeki veriyi sadece son kullanıcı tarafından girildiğinde doğrular.

- Uygulama programının kendisi `set_field_buffer` çağırarak alan değerini değiştirdiğinde ise doğrulama gerçeklenmez.
- Birbirine bağlı alanlar dolaylı şekilde değiştirilir: Bağlı oldukları alanın değeri değiştirilir.

Aşağıdakiler önceden tanımlanmış doğrulama türleridir. Kendiniz de yorucu ve biraz ustalık isteyen bir iş olmasına rağmen doğrulama türü tanımlayabilirsiniz.

TYPE_ALPHA

Bu alan türü alfabetik verileri kabul eder; boşluk, sayı, özel karakter olmayan (karakter girişi tarafından bu kontrol edilir). Şu şekilde ayarlanır:

```
int                                                                    işlev
set_field_type (FIELD *alan,     /* değiştirilecek alan */
                TYPE_ALPHA,      /* ilişiklendirilecek tür */
                int      genişlik); /* alanın en fazla genişliği */
```

genişlik parametreleri veri için an az genişlik değerini ayarlar. Kullanıcının alandan veri girişini terk edebilmesi için en az bu ayarlanan değer genişliğinde veri girmesi gerekir. Tipik olarak bunu alan genişliği ile aynı ayarlamak isteyeceksiniz; eğer alan genişliğinden daha büyük olursa alan doğrulama sekteye uğrayacaktır. Değer olarak sıfır girilmesi ise alanın tamamlanmasını isteğe bağlı yapar.

TYPE_ALNUM

Bu alan türü alfabetik veriler ve sayılar kabul eder; boşluk ve özel karakterler (karakter girişi sırasında kontrol edilir) olmaz. Şu şekilde ayarlanır:

```
int                                                                    işlev
set_field_type (FIELD *alan,     /* değiştirilecek alan */
                TYPE_ALNUM,      /* ilişiklendirilecek tür */
                int      genişlik); /* alanın en fazla genişliği */
```

genişlik parametreleri verinin en az genişliğini ayarlar. TYPE_ALPHA ile bunu alanın genişliği ile aynı yapmak isteyeceksiniz; eğer alanın genişliğinden daha büyük olursa doğrulama sekteye uğrar. Değeri sıfır olarak ayarlanırsa alanın tamamlanması isteğe bağlı hale gelir.

TYPE_ENUM

Bu tür bir ayarlama alan değerlerinin belli dizgelerden biri (örneğin, iki harfli posta kodu) olmasını zorunlu kılar. Şu şekilde ayarlanır:

```
int                                                                    işlev
set_field_type (FIELD *alan,       /* değiştirilecek alan */
                TYPE_ENUM,         /* ilişiklendirilecek tür */
                char   **değerlistesi, /* olası değerlerin listesi */
                int      buyukluk,   /* büyük küçük harf duyarlı mı? */
                int      tekillik);  /* tekil olarak belirtilmeli mi? */
```

değerlistesi parametreleri NULL ile sonlanmış geçerli dizgeler listesini işaret etmelidir. *buyukluk* değişkeni dizge ile karşılaştırmayı büyük küçük harfe duyarlı hale getirir.

Kullanıcı TYPE_ENUM alanını terk ettiğinde, doğrulama işlemi önbellekteki veriyi geçerli bir girişe tamamlamaya çalışır. Tam bir dizge seçeneği girildiyse, elbette ki geçerlidir. Fakat aynı zamanda bir dizgenin öneklerini girmek te mümkündür, bu durumda kalanı tamamlanacaktır.

Öntanımlı olarak eğer böylesi bir önek girer ve bununla eşleşen birden fazla dizge listesi bulunursa, ilk eşlenen değere tamamlanır. Fakat eğer *tekillik* değeri mantıksal doğru ise, bu eşlenenin tekil olması zorunluluğunu getirir.

REQ_NEXT_CHOICE ve REQ_PREV_CHOICE girdi istekleri bu alanlar ile faydalı şekilde kullanılabilir.

TYPE_INTEGER

Bu alan türü bir tamsayı kabul eder. Şu şekilde ayarlanır:

```
int                                                                    işlev
set_field_type (FIELD *alan,     /* değiştirilecek alan */
                TYPE_INTEGER,    /* ilişiklendirilecek tür */
                char     adimlama, /* sıfırlarla doldurulacak yer sayısı */
                int      vmin,     /* geçerli aralık */
                int      vmax);    /* geçerli aralık */
```

Geçerli karakterler önlerinde eksi işareti olabilen sayılardır. Menzil kontrolü çıkış anında gerçeklenir. Eğer aralığın en büyük değeri en az değerden küçük veya eşitse aralık gözardı edilir.

Eğer değer aralık kontrolünü geçerse, *adimlama* parametresindeki değere göre uygun sıfırlarla doldurma gerçeklenir.

TYPE_INTEGER değer önbelleği C kütüphanesindeki atoi(3) işlevi ile uygun şekilde ifade edilebilir.

TYPE_NUMERIC

Bu alan türü 10 tabanında bir sayı kabul eder. Şu şekilde ayarlanır:

işlev
```
int
set_field_type (FIELD *alan,     /* değiştirilecek alan */
                TYPE_NUMERIC,    /* ilişiklendirilecek tür */
                int    hassasiyet, /* noktadan sonraki hassasiyet */
                int    vmin,     /* geçerli aralık */
                int    vmax);    /* geçerli aralık */
```

Geçerli karakterle önlerinde eksi işareti de olabilen sayılardır. Muhtemelen bir de ondalık belirteci bir nokta vardır. Menzil kontrolü çıkışta kontrol edilir. Eğer aralığın en büyük değeri en küçük değerden küçük veya ona eşitse aralık gözardı edilir.

Eğer değer, aralık kontrolünü geçerse *hassasiyet* kısmındaki değere göre uygun olarak sağ taraf sıfırlarla doldurulur.

Bir TYPE_NUMERIC değeri C kütüphanesindeki atof(3) işlevi ile uygun şekilde ifade edilebilir.

TYPE_REGEXP

Bu alan düzenli ifadeler şeklinde yazılmış verileri kabul eder. Şu şekilde ayarlanır:

işlev
```
int
set_field_type (FIELD *alan,   /* değiştirilecek alan */
                TYPE_REGEXP,   /* ilişiklendirilecek tür */
                char   *duzifd); /* eşleşecek ifade */
```

Düzensiz ifade sözdizimi regcomp(3) işlevindeki ile aynıdır. Düzensiz ifadeyi kontrol etmek için kontrol, çıkış anında gerçeklenir.

18.6. form_driver: Form sisteminin dolap beygiri

Menü sisteminde olduğu gibi, form_driver() form sisteminde çok önemli bir rol oynamaktadır. Form sistemine yapılan tüm istekler form_driver()'dan süzülmek zorundadır.

işlev
```
int
form_driver (FORM *form,  /* değiştirilecek form */
             int    istek); /* form istek kodu */
```

Yukarıdaki örneklerin bazılarında da gördüğünüz gibi, bir döngü kurup kullanıcıdan gelen istekleri kontrol edip bunun bir veri mi yoksa kullanıcı isteği mi olduğuna karar vermeniz gerekir. Form istekleri daha sonra işlemi gerçeklemesi için form_driver()'a gönderilir.

İstekler kabaca şu alt sınıflara ayrılabilir. Değişik isteklerin kulanımları aşağıda açıklanmıştır:

18.6.1. Sayfada Gezinti İstekleri

Bu istekler form içerisinde sayfa bazında hareketlere sebep olur, form ekranının kayması sağlanır. Bir form pek çok sayfadan oluşmuş olabilir. Eğer çok sayıda alandan ve mantıksal bölümden oluşmuş büyük bir formunuz varsa onu sayfalara bölebilirsiniz. set_new_page() belirtilen alanda yeni bir sayfa oluşturur.

```
int                                                                    işlev
set_new_page (FIELD *alan,          /* değiştirilecek alan */
              bool   sayfa_kesme); /* sayfa kesmesi yapılacak mı? */
```

Aşağıdaki istekler farklı sayfalara hareket etmenizi sağlar.

- REQ_NEXT_PAGE Sonraki form sayfasına hareket
- REQ_PREV_PAGE Önceki form sayfasına hareket
- REQ_FIRST_PAGE İlk form sayfasına hareket
- REQ_LAST_PAGE Son form sayfasına hareket

Bu istekler döngüsel şekilde liste üzerinde hareket eder; yani REQ_NEXT_PAGE son sayfadan sonra ilk sayfaya gider, REQ_PREV_PAGE ilk sayfadan sonra son sayfaya gider.

18.6.2. Alan İçi Gezinti İstekleri

Aynı sayfadaki alanlarda gezinti isteklerinde bunlar kullanılır:

- REQ_NEXT_FIELD Sonraki alana git
- REQ_PREV_FIELD Önceki alana git
- REQ_FIRST_FIELD İlk alana git
- REQ_LAST_FIELD Son alana git
- REQ_SNEXT_FIELD Sıralanmış sonraki alana git
- REQ_SPREV_FIELD Sıralanmış önceki alana git
- REQ_SFIRST_FIELD Sıralanmış ilk alana git
- REQ_SLAST_FIELD Sıralanmış son alana git
- REQ_LEFT_FIELD Alanın soluna git
- REQ_RIGHT_FIELD Alanın sağına git
- REQ_UP_FIELD Alanın yukarısına git
- REQ_DOWN_FIELD Alanın aşağısına git

Bu istekler bir sayfadaki alan listesi üzerinde döngüsel şekilde işlenir; yani REQ_NEXT_FIELD son alandan sonra ilk alana döner, REQ_PREV_FIELD ilk alandan sonra son alana döner. Bunlar için (aynı zamanda REQ_FIRST_FIELD ve REQ_LAST_FIELD) alanların sırası form dizisindeki alan göstericisinin sırasıdır (new_form() veya set_form_fields() ile oluşturulmuştur).

Alanları ekranda görüntülendikleri şekilde soldan sağa ve yukarıdan aşağı gezmek de mümkündür. Bunu gerçeklemek için sıralanmış hareket listesinden ikinci guruptakileri kullanın.

Son olarak da alanlar arasında görsel yukarı, aşağı, sağ, sol yönlerini kullanarak gezinmek mümkündür. Bunu gerçeklemek için, üçüncü guruptaki dört isteği kullanın. Bir formun bu tür işler için başlangıcının sol üst köşesi olduğunu unutmayın.

Örneğin, çok satırlı bir B alanı, tek satırlı A ve C alanlarının aynı satırda A, B, C şeklinde sıralandığını düşünün. A alanından verilmiş REQ_MOVE_RIGHT isteği A, B ve C nin aynı satırı paylaşması halinde B alanına geçmeyi sağlayacaktır. Aksi halde B ve C alanlarını atlayacaktır.

18.6.3. Alan İçi Dahili Gezinti İstekleri

Bu istekler o an düzenlenilen alan içerisinde imlecin hareketini sağlar.

- REQ_NEXT_CHAR Bir sonraki karaktere git

- REQ_PREV_CHAR Bir önceki karaktere git
- REQ_NEXT_LINE Bir sonraki satıra git
- REQ_PREV_LINE Bir önceki satıra git
- REQ_NEXT_WORD Bir sonraki kelimeye git
- REQ_PREV_WORD Bir önceki kelimeye git
- REQ_BEG_FIELD Alanın başına git
- REQ_END_FIELD Alanın sonuna git
- REQ_BEG_LINE Satır başına git
- REQ_END_LINE Satır sonuna git
- REQ_LEFT_CHAR Alan içerisinde sola git
- REQ_RIGHT_CHAR Alan içerisinde sağa git
- REQ_UP_CHAR Alan içerisinde yukarı git
- REQ_DOWN_CHAR Alan içerisinde aşağı git

Her kelime bir önceki bir sonraki kelimeler arasında boşluk karakteriyle ayrılır. Satır veya alanın başına ve sonuna gitmeye yarayan komutlar kendi menzilleri içerisindeki ilk veya son dolgu karakteri olmayan karakteri arar.

18.6.4. Kaydırma İstekleri

Çalışma zamanında büyüyebilen ve açık şekilde ekranda görülmeyen satırlardan oluşmuş alanlar kaydırılabilir. Tek satırlık alanlar yatay olarak kaydırılır, çok satırlık alanlar dikey olarak kaydırılır. Pek çok kaydırma işlemi alan içi hareketlerle ve düzenleme ile başlatılır (kütüphane imleci görünür kılmak için alan içerisinde kaydırma işlemi gerçekler). Açık şekilde kaydırma işlemlerine istekte bulunmak aşağıdaki şekilde mümkündür:

- REQ_SCR_FLINE Bir satırın ilerisine dikey olarak kaydırma
- REQ_SCR_BLINE Bir satırın gerisine dikey olarak kaydırma
- REQ_SCR_FPAGE Bir sayfanın ilerisine dikey kaydırma
- REQ_SCR_BPAGE Bir sayfanın gerisine dikey olarak kaydırma
- REQ_SCR_FHPAGE Bir sayfanın yarısına ileri yönde dikey kaydırma
- REQ_SCR_BHPAGE Bir sayfanın yarısına geri yönde dikey kaydırma
- REQ_SCR_FCHAR Yatay olarak bir karakter ilerisine kaydırma
- REQ_SCR_BCHAR Yatay olarak bir karakter gerisine kaydırma
- REQ_SCR_HFLINE Yatay olarak bir alan ilerisine kaydırma
- REQ_SCR_HBLINE Yatay olarak bir alan gerisine kaydırma
- REQ_SCR_HFHALF Yatay olarak yarım alan ilerisine kaydırma
- REQ_SCR_HBHALF Yatay olarak yarım alan gerisine kaydırma

Kaydırma işlemleri için, bir alanın sayfası görünür kısmının yüksekliğidir.

18.6.5. İstekleri Düzenlemek

Form sürücüsüne bir ASCII karakter gönderdiğiniz zaman, alan veri belleğine bir karakter eklemek için bir istek olarak algılanır. Bunun bir araya ekleme mi yoksa üzerine yazma mı olduğu alanın düzenleme kipine bağlıdır (varsayılan araya eklemedir).

Aşağıdaki istekler alanı düzenlemeyi ve düzenleme kipini değiştirmeyi sağlar:

- REQ_INS_MODE Araya ekleme kipini ayarla
- REQ_OVL_MODE Üzerine yazma kipini ayarla

- REQ_NEW_LINE Yeni satır isteği (açıklama için aşağıya bakınız)
- REQ_INS_CHAR Karakterin olduğu konuma boşluk yerleştir
- REQ_INS_LINE Karakterin olduğu konuma boş satır ekle
- REQ_DEL_CHAR İmlecin olduğu konumdaki karakteri sil
- REQ_DEL_PREV İmlecin olduğu konumdan önceki kelimeyi sil
- REQ_DEL_LINE İmlecin olduğu satırı sil
- REQ_DEL_WORD İmlecin olduğu konumdaki kelimeyi sil
- REQ_CLR_EOL Satırı sonuna kadar temizle
- REQ_CLR_EOF Alanı sonuna kadar temizle
- REQ_CLR_FIELD Alanın tüm içeriğini temizle

REQ_NEW_LINE ve REQ_DEL_PREV istekleri kısmen bazı form seçenek çiftleri tarafından kontrol edilen karmaşık isteklerdir. Özel durumlar imlecin alanın başında veya alanın sonunda olması durumunda etkin olur.

Önce REQ_NEW_LINE isteğini gözönüne alalım:

Araya ekleme kipinde REQ_NEW_LINE, halihazırdaki imlecin olduğu konumdan satırı keser, kalanı yeni bir satır olarak ekler ve imleci o satırın başına konumlandırır (bunu alan önbelleğine yeni satır eklemek olarak düşünebilirsiniz).

Üzerine yazma kipinde REQ_NEW_LINE, imlecin o an bulunduğu konumdan itibaren satırı satır sonuna kadar siler. Daha sonra imleç yeni satırın başına konumlanır.

Yine de alanın başında veya sonunda REQ_NEW_LINE aslında, REQ_NEXT_FIELD gerçekler. Bu özel durumun gerçekleşmesi için O_NL_OVERLOAD özelliği kapalı olmalıdır.

Şimdi de REQ_DEL_PREV isteğini ele alalım:

REQ_DEL_PREV'in normal davranışı bir önceki karakteri silmek şeklindedir. Eğer araya ekleme kipi açık, imleç satır başındaysa ve o satırdaki metin bir öncekine sığıyorsa, o anki satırın içeriğini bir öcnekinin sonuna ekler ve o anki satırı siler (bunu alan önbellek alanından bir satırın silinmesi olarak düşünebilirsiniz).

Aslında REQ_DEL_PREV alanın başında REQ_PREV_FIELD gibi davranır.

Eğer O_BS_OVERLOAD kapalıysa, bu özel davranış biçimi etkisizdir ve form sürücüsü sadece E_REQUEST_DENIED üretir.

18.6.6. Emir İstekleri

Eğer alanınızın türü emir türünde ve verilmiş bir türün sonraki ve önceki değerlerini almakla ilişkilendirilmiş işlevler içeriyorsa, alan ön belleğinde bu değeri eşleyecek istekler vardır:

- REQ_NEXT_CHOICE O anki değerden sonrakini önbelleğe yerleştir.
- REQ_PREV_CHOICE O anki değerin öncekisini önbelleğe yerleştir.

Yerleşik alan türleri için sadece TYPE_ENUM'un yerleşik önceki ve sonraki işlevleri vardır. Kendinize ait bir alan türü tanımlarsanız, kendi sıralama işlevlerinizi ilişiklendirebilirsiniz (Özel Doğrulama Türleri'ne bakınız — Ç.N.: belgede bulunamadı).

18.6.7. Uygulama Komutları

KEY_MAX'tan büyük ve sabit MAX_COMMAND'den küçük veya eşit imleç değerleri üzerindeki form istekleri tamsayı olarak gösterilir. Bu sınır değerleri içerisindeki bir değer form_driver() tarafından ihmal edilir. Dolayısıyla bu, uygulama tarafından herhangi bir amaç için kullanılabilir. Uygulamaya özgü bir davranış olarak ele alınıp buna özgü bir davranış sergilenebilir.

19. Araçlar ve Küçük Uygulama Kütüphaneleri

Şimdiye kadar ncurses kütüphanesinin yeteklerini ve kardeş kütüphanelerini tanıdınız, ekranınızı göz doldurucu şekilde değiştirecek bir proje için heyecanla beklemektesinizdir. Fakat acele etmeyin. Karmaşık grafik arayüzleri tasarlamak ve onlar üzerinde çalışmak ek kütüphanelerle bile yalın ncurses içerisinde oldukça zor olmaktadır. Halihazırda kendi küçük uygulamalarınızı yazarken kullanabileceğiniz araçlar ve küçük uygulama kütüphaneleri bulunmaktadır. Bazılarını kullanabilirsiniz, kodlardan ilham almaya bakın, bazılarını genişletip kullanın.

19.1. CDK Curses Geliştirme Kiti (Curses Development Kit)

Yazarının sözleri şöyle:

> CDK, Curses Geliştirme Kiti anlamına gelir ve tam ekran curses program geliştirmesinin hızlı bir şekilde olmasını sağlayan 21 tane gereç içerir.

Uygulama kümesi, programlarınızda doğrudan kullanabileceğiniz gereçler (widget) içermektedir. Oldukça güzel yazılmıştır ve belgelendirmesi de oldukça iyidir. Yeni başlayanlar için örnekler dizinindeki örnekler başlamak için iyi bir yer olabilir. CDK `http://invisible-«island.net/cdk/` adresinden indirilebilir. Yüklemek için README dosyasındaki adımları takip ediniz.

19.1.1. Gereç Listesi

Aşağıda CDK ile gelen gereçlerin listesi ve açıklamaları bulunmaktadır.

`Alphalist`
: Aranan kelimeyle ilgili ilk bir kaç karakterinin yazılmasıyla arama listesinin daraltılmasını sağlayabilme özelliği sayesinde kullanıcının bir listeden kelime seçmesini sağlar.

`Buttonbox`
: Çok butonlu bir küçük programcık oluşturur.

`Calendar`
: Küçük basit bir takvim programcığı oluşturur.

`Dialog`
: Kullanıcıya bir ileti ile seçim hakkı verir, kullanıcı da sunulan düğmelerden bir tanesini seçer.

`Entry`
: Kullanıcının değişik türlerde bilgi girmesini sağlar.

`File Selector`
: CDK tabanlı küçük uygulamalardan oluşan bir dosya seçme uygulamasıdır. Bu örnek, CDK gereç kütüphanesini kullanarak daha karmaşık uygulamaların nasıl oluşturulacağını gösterir.

`Graph`
: Bir grafik çizer.

`Histogram`
: İstatistik grafiği çizer.

`Item List`
: Kullanıcının ufak bir alandaki pek çok seçenek içerisinden birisini seçmesini sağlayan ekrana çıkagelen bir alan oluşturur. Haftanın günleri veya ay adları için çok kullanışlıdır.

`Label`

Ekrana çıkagelen bir kutu içerisinde iletileri gösterir veya etiket ekranın bir parçası olarak düşünebilir.

`Marquee`
Ekranda kayan bir ileti gösterir.

`Matrix`
Karmaşık bir matriksi pek çok seçeneklerle oluşturur.

`Menu`
Tıklayınca aşağı doğru açılan menu arayüzünü oluşturur.

`Multiple Line Entry`
Çok satırlı alan girişi. Uzun alanlar için oldukça kullanışlıdır (açıklama alanı gibi).

`Radio List`
Radyo düğmesi oluşturur.

`Scale`
Sayısal bir derecelendirme oluşturur. Kullanıcıdan sayısal bir değer alınmasını fakat bunun belli bir değerler alanı içerisinde sınırlı kalmasını sağlar.

`Scrolling List`
Kayan bir menü/liste oluşturur.

`Scrolling Window`
Kaydırılabilir bir günlük kayıt dosyası göstericisi oluşturur. Çalışırken içerisine bilgi girişi yapılabilir. Bir işlemin gelişimini göstermek için güzel bir küçük uygulamadır (konsol penceresine benzer).

`Selection List`
Çok seçenekli bir seçim listesi oluşturur.

`Slider`
Derecelendirme uygulamasına benzer, yan tarafta duran görsel bir çubukta sayısal değeri yansıtır.

`Template`
Karakterlerin bulunduğu konuma duyarlı bir giriş alanı oluşturur. Önceden biçimlenmiş tarih ve telefon numaraları gibi alanlar için kullanılır.

`Viewer`
Bu bir dosya/bilgi göstericisidir. Çok fazla bilgiyi göstermek istediğinizde çok faydalıdır.

Son sürümde bazı küçük uygulamalar Thomas Dickey tarafından değiştirilmiştir.

19.1.2. Bazı Çekici Özellikler

Kullanışlı küçük gereçlerle hayatınızı daha kolay bir hale getirmesinin yanında CDK, hizalanmış ve çok renkli dizgeleri yazdırılma gibi can sıkıcı bir çok sorunu da zekice çözer. CDK işlevlerine dizge içerisine gömülmüş biçimlendirme etiketleri gönderilebilir. Örneğin;

eğer dizge şu şekildeyse,

```
"</B/1>Bu satır sarı arka fon ve mavi ön fon rengine sahip olmalıdır.<!1>"
```

ve bu parametre olarak `newCDKLabel()` işlevine verilirse, satırı sarı artalanlı ve mavi karakterli olarak yazar. Dizgeyi hizelemek, içerisine özel çizim karakterleri eklemek, v.b. için de başka özel etiketler mevcuttur. Lütfen ayrıntılar için cdk_display(3X) kılavuz sayfasına bakınız. Kılavuz sayfası kullanışı hoş örneklerle açıklamaktadır.

19.1.3. Sonuç

Herşeyi hesaba katarsak, CDK gereçler içeren iyi yazılmış ve eğer uygun şekilde kullanılırsa karmaşık bir grafik arayüzün gelişiminin güçlü bir taslağını oluşturabilen bir pakettir.

19.2. `dialog` hakkında

Uzun uzun yıllar önce, 1994 Eylülünde, çok az insanın Linux'u bildiği zamanlarda Jeff Tranter, Linux Journal'daki bir söyleşiden yola çıkarak makalesini[B30] yazdı. Makalesine şu sözlerle başlamıştır:

> Linux, Unix işletim sistemi temellidir, fakat çok sayıda kendine has ve kullanışlı çekirdek özellikleri ve uygulama programları sunmasıyla Unix altında edinilebilenden ötelere gitmiştir. Bilinen küçük bir tanesi "dialog" ismindeki profesonel görünümlü etkileşimli kutuları kabuk betikleriyle hazırlamaya yarayan uygulamasıdır. Bu makale giriş seviyesinde dialog uygulamasına bir ders niteliği taşımakta ve nerede ve nasıl kullanılabileceğini örneklerle göstermektedir.

Açıklamasını sürdürdüğü sıralarda, dialog gerçekten de kolay şekilde profosyonel görünümlü etkileşimli kutular oluşturmak için kullanılabilecek bir uygulamaydı. Pek çok değişik etkileşimli kutular, menüler, işaretlenebilir listeler v.b. oluşturmaktaydı. Genelde öntanımlı olarak yükleniyordu. Değilse bile Thomas Dickey[B31]'in sayfasından indirebilirsiniz.

Yukarıda bahsedilen makale, uygulamaları ve onunla yapılabilecekler ilgili çok iyi bir ön bilgi vermektedir. Kılavuz sayfaları daha ayrıntılı bilgiye sahiptir. Pek çok değişik durumda kullanılabilir. Bir tanesi linux çekirdeğini metin kipinde yapılandırmadır. Linux çekirdeği, dialog uygulamasının ihtiyaçlarına göre uygun hale getirilmiş bir sürümünü kullanmaktadır.

Dialog ilk başlarda kabuk betikleri ile beraber kullanılmak için tasarlandı. Eğer onun özelliklerini bir C programında kullanmak isterseniz o zaman `libdialog`'u kullanın. Bununla ilgili belgelendirme biraz dağınıktır. En belirgin yardım alınabilecek nokta kütüphane ile beraber gelen `dialog.h` başlık dosyasıdır. Orada ve burada bazı kotarımları, isteğinizi almak için gerçeklemeniz gerekebilir. Kaynak kolay şekilde özelleştirilebilirdir. Kodunu değiştirerek onu pekçok değişik durumda kullandım.

19.3. Perl Curses Modullleri: `CURSES::FORM` ve `CURSES::WIDGETS`

Curses'ın perl modülleri `Curses::Form` ve `Curses::Widgets` Perl içerisinden curses kipine erişimi sağlar. Curses ve temel Perl yüklü ise bu modülleri CPAN Modül Sayfası[B32]ndan edinebilirsiniz. Curses kategorisi içerisindeki üç zipli dosyayı indirin. Bir kere yükledikten sonra bu modülleri perl betikleri içerisinden diğer modüller gibi kullanabilirsiniz. Perl modülleri ile ilgili ayrıntılı bilgi için `perlmod` kılavuz sayfasına bakın. Yukarıdaki modüller iyi bir belgelendirme ve denemeniz için bir takım demo betiklerle beraber gelmektedir. Gereçler çok gelişmemiş olmasına rağmen bu modüller Perl içerisinden curses'a iyi bir şekilde erişimi sağlamaktadır.

Örneklerimin bazıları Anuradha Ratnaweera tarafından Perl hallerine dönüştürüldü ve **`perl`** dizini içerisinden alınabilir.

Daha fazla ayrıntı için `Curses(3)` , `Curses::Form(3)` ve `Curses::Widgets(3)` kılavuz sayfalarına bakınız. Bu sayfalar sadece yukarıdaki modüller yüklendikten sonra yüklenir.

20. Sadece Eğlence İçin !!!

Bu kısım eğlence olsun diye yazılmış bir kaç uygulamadan oluşmaktadır. Bunlar en iyi programlama tekniklerini ya da ncurses kütüphanesinin en iyi kullanım şekillerini göstermemektedir. Yeni başlayanlara birer fikir vermesi ve bu alana daha çok program eklemeleri için oluşturulmuşlardır. Eğer curses içerisinde bir iki hoş, basit program yazdıysanız ve onların da burada yer almasını istiyorsanız bana (<ppadala (at) gmail.com>) yazın.

20.1. Hayat Oyunu

Hayat Oyunu matematikteki bir mucize. Paul Callahan[B33]'ın sözleriyle

> Hayat Oyunu (veya sadece Hayat) anlaşıldığı manada oyun değildir. Herhangi bir oynayıcı, kazanan ya da kaybeden yoktur. Parçalar bir kere başlangıç konumuna yerleştirildi mi kurallar sonra olacak herşeyi sonradan belirler. Herşeye rağmen, Hayat süprizlerle doludur! Pekçok durumda başlangıç konumuna dönmek (veya biçimine) imkansızdır veya gelecekte ne olacağını görmek. Keşfetmenin tek yolu oyunun kurallarını takip etmektir.

Bu program basit bir ters U örüntüsü ile başlar ve hayatın nasıl mükemmel çalıştığını gösterir. Program içerisinde geliştirilecek pekçok oda vardır. Kullanıcıların kendi örüntülerini girmelerini veya bir dosyadan girdi almayı sağlayabilirsiniz. Pekçok değişiklikle oyunun kurallarını değiştirebilirsiniz. google[B34]'da hayat oyunu hakkında ilginç sonuçlar için aramalar yapın.

Dosya Yolu: `JustForFun/life.c`

20.2. Sihirli Kare

Sihirli Kare, başka bir matematiksel hayret veren sonuçlardan biridir. Anlaşılması çok kolay fakat yapması oldukça zordur. Sihirli karede her satır ve sütundaki sayıların toplamı eşittir. Köşegensel toplam dahi eşittir. Daha pekçok özel niteliklere sahiptir.

Bu program tek sırada sayılardan oluşan basit bir sihirli kare oluşturur.

Dosya Yolu: `JustForFun/magic.c`

20.3. Hanoi Kuleleri

Meşhur Hanoi kulesi çözücüsü. Oyunun amacı en tepedeki askıdan sondakine kadar ortadakini geçici bir askı gibi kullanarak sıralamaktır. Herhangi bir zamanda büyük diski diğerinin üzerine yerleştirmeme hilesine dayanır.

Dosya Yolu: `JustForFun/hanoi.c`

20.4. Vezir Yerleştirme

Meşhur N–Vezir sorusudur. N*N'lik bir satranç tahtasında hiç birisi birbirini tehdit etmeyecek şekilde N tane veziri yerleştirme amacına dayanır.

Basit bir geri dönüş tekniği kullanarak problemi çözer.

Dosya Yolu: `JustForFun/queens.c`

20.5. Shuffle

Eğer öldürecek zamanınız varsa eğlenceli bir oyundur.

Dosya Yolu: `JustForFun/shuffle.c`

20.6. Onparmak Eğitmeni

Basit bir yazma yardımcısı. İhtiyaçtan daha çok kolay kullanımı için tasarladım. Eğer parmaklarınızı nasıl koyacağınızı biliyorsanız fakat pratik eksiğiniz varsa bu yardımcı olabilir.

Dosya Yolu: `JustForFun/tt.c`

21. Kaynakça

- NCURSES kılavuz sayfaları
- NCURSES SSS http://invisible-«island.net/ncurses/ncurses.faq.html
- Eric Raymond ve Zeyd M. Ben–Halim tarafından http://invisible-«island.net/ncurses/ncurses-«intro.html adresinde bulunan NCURSES ile program yazma kılavuzu. Biraz eskidir. O belgeden ilham aldım, zaten bu NASIL belgesinin yapısı da asıl belgeyi takip etmektedir.

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.

(B6) http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-«formats/pdf/NCURSES-«Programming-«HOWTO.pdf

(B7) http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-«formats/ps/NCURSES-«Programming-«HOWTO.ps.gz

(B8) http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-«formats/html/NCURSES-«Programming-«HOWTO-«html.tar.gz

(B9) http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-«formats/html_single/NCURSES-«Programming-«HOWTO.html

(B10) http://www.tldp.org/LDP/LDP-«Author-«Guide/

(B23) http://www.geocrawler.com/archives/3/344/1999/9/0/2643549/

(B24) http://www.jersey.net/~debinjoe/games/

(B30) http://www2.linuxjournal.com/lj-«issues/issue5/2807.html

(B31) http://invisible-«island.net/dialog/

(B32) http://www.cpan.org/modules/01modules.index.html

(B33) http://www.math.com/students/wonders/life/life.html

(B34) http://www.google.com

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

8 Şubat 2007