İÇİNDEKİLER

Donanım Tanıma ve Yapılandırma

Konu:

Adaylar Pardus altında çalışan bilgisayar donanımını tanıma, tanıtma ve yapılandırma

Hedefler:

- PCI veri yoluna takılı donanımları bulmak
- USB aygıtlar hakkında bilgi almak
- Sabit disklerin isimlendirilmesini öğrenmek
- Çekirdek modüllerini yüklemek, kaldırmak
- Donanımla ilgili soyut çekirdek arayüzleri

Anahtar Kelimeler:

USB, lsusb, PCI, lspci, /sys, /proc, /dev, modprobe, lsmod, lspci, lsusb, sysfs, udev, hald, dbus

Donanım

Linux, navigasyon cihazından TV kutusuna, akıllı tahtaya; tabletten süper bilgisayarlara, ağ yönlendiricilere kadar çok sayıda mimari ve platformu desteklemektedir. En çok kullanılan platform Intel x86 işlemcili IBM PC ve benzerleridir.

PC donanımları yıllar içerisinde pek çok değişiklik geçirmiştir. 90'lı yıllar ve öncesinde ISA veri yolu kullanılırken artık ISA veri yollu PC üretilmemekte, yerini PCI veri yolu almıştır. PCI, işlemci, bellek ve aygıtlar arasında yüksek hızda veri transferi yaparken donanımların daha kolay tanımlanmasını sağlayan bir ID sistemine sahiptir. Bu sayede aygıt türü, üreticisi ve modeli kolaylıkla bulunabilir.

`lcpci` komutu, PCI veri yoluna bağlı aygıtların listesini almak için kullanılır. Dağıtımların kurulum programları bu verileri kullanarak doğru aygıt sürücülerini yükleyebilmektedir.

```
$ lspci -v
00:00.0 Host bridge: Intel Corporation 440BX/ZX/DX -
82443BX/ZX/DX Host bridge (rev 01)
    Subsystem: VMware Virtual Machine Chipset
    Flags: bus master, medium devsel, latency 0
```

```
    Kernel driver in use: agpgart-intel

00:01.0 PCI bridge: Intel Corporation 440BX/ZX/DX -
82443BX/ZX/DX AGP bridge (rev 01) (prog-if 00 [Normal decode])
    Flags: bus master, 66MHz, medium devsel, latency 0
    Bus: primary=00, secondary=01, subordinate=01, sec-
latency=64

00:07.0 ISA bridge: Intel Corporation 82371AB/EB/MB PIIX4 ISA
(rev 08)
    Subsystem: VMware Virtual Machine Chipset
    Flags: bus master, medium devsel, latency 0
```

Eski PS/2 klavye ve fare, seri porttan bağlı modemler ve paralel porttan takılan yazıcılar için ortak olarak USB bağlantı arayüzü geliştirilmiştir. USB, hotplug olarak canlı sisteme bağlanmayı ve bağlantıyı kesmeyi destekler.

Bir USB cihaz sisteme bağlandığında 1-127 arasında bir numara atanır. Bağlı USB cihazların listesini almak için:

```
$ lsusb
Bus 001 Device 002: ID 8087:0024 Intel Corp. Integrated Rate
Matching Hub
Bus 002 Device 002: ID 8087:0024 Intel Corp. Integrated Rate
Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 003: ID 0781:5567 SanDisk Corp. Cruzer Blade
```

Sistemde ne kadar USB aygıt tanımlı olduğunu görmek için:

```
# find /dev/bus/
/dev/bus/
/dev/bus/usb
/dev/bus/usb/002
/dev/bus/usb/002/006
/dev/bus/usb/002/005
/dev/bus/usb/002/004
```

Belirli bir aygıt için detaylı bilgi almak için:

```
# lsusb -D /dev/bus/usb/002/005
Device: ID 0951:1643 Kingston Technology DataTraveler G3 4GB
Couldn't open device, some information will be missing
Device Descriptor:
bLength 18
bDescriptorType 1
bcdUSB 2.00
bDeviceClass 0 (Defined at Interface level)
bDeviceSubClass 0
```

```
bDeviceProtocol 0
bMaxPacketSize0 64
idVendor 0x0951 Kingston Technology
idProduct 0x1643 DataTraveler G3 4GB
[...]
```

Daha fazla detay için –v seçeneği kullanılabilir.

Linux için önemli donanım bileşenlerinden biri de disklerdir. IDE (P-ATA), SATA ve SCSI olarak değişik bağlantı arayüzleri geliştirilmiştir.

IDE disk, silindir, sektör ve okuma/yazma kafasından bağımsız yazılan sıralı bloklar şeklinde kullanılır. 40 veya 80 pinlik paralel veri yolu içerir. SATA ise 7 hattan ibaret seri haberleşme kullanır. SATA diskler hotplug kullanılabildiğinden RAID-5 ve RAID 1 gibi yapılandırmalar için elverişlidir. Ayrıca 2.5'' ve 3.5'' için aynı bağlantı kablosunu kullanır. Diğer önemli özelliği ise daha hızlı olmasıdır.

Linux, P-ATA diskleri /dev/hdX şeklinde isimlendirir. İlk kablo üzerindekiler hda, hdb ve ikinci kablo üzerindekiler hdc, hdd şeklinde isimlendirilir. Bunlar da kendi içinde primary veya mantıksal bölümler içerebilir. Primary bölümler 1-4 arasında bir numara alırken, mantıksal bölümler 5-63 arasında bir numara alır. Örneğin: /dev/hda1, /dev/hdc5...

SATA diskler ise SCSI diskler gibi isimlendirilmektedir.

SCSI, disk ve teyp sürücü gibi cihazlar için uzun yıllardan beri kullanılan bir teknolojidir. Bugünlerde SCSI yerine daha hızlı olan SAS (Serial Attached SCSI) daha çok tercih edilmekte. SAS, teknoloji olarak SATA'ya çok benzemekte ve bu nedenle aynı backplane'i kullanabilmektedirler.

IDE ve SATA daha uygun maliyeti nedeniyle masaüstü bilgisayarlarda tercih edilirken SCSI ve SATA daha kararlı ve performanslı olması nedeniyle sunucu bilgisayarlarda kullanılmaktadır.

Linux altında, SCSI bağlantıları hakkındaki bilgiler `/proc/scsi` dizini altındadır.

SCSI diskler /dev altında sda, sdb, sdX şeklinde isimlendirilirken CD-ROM sürücü /dev/st0, teyp sürücü /dev/sg0 şeklinde isimlendirilir.

Çekirdek Modülleri

Linux işletim sistemi çok geniş çevre cihazı desteği sunmaktadır. Çevre cihazları, bilgisayarları tamamlayan işlemci ve bellek gibi ana donanımların dışında kalan pek çok donanımı ifade etmek için

kullanılır. Ses kartı, ekran kartı, USB arayüzler, seri ve paralel port, sabit diskler, RAID kartlar, ethernet kartlar... gibi pek çok donanım ve bu donanımları üreten pek çok üretici marka desteklenmektedir.

Çevre değişkenlerini kullanabilmek için sürücülerinin yüklenmesi gerekmektedir. Linux, yüklenebilir çekirdek modüllerine sahiptir. Bu sayede çalışan çekirdek boyutu küçük tutulurken donanımlara ait sürücüler istendiğinde ekleniyor, gerek kalmadığında kaldırılabiliyor.

Yüklenebilir çekirdek modülleri `/lib/modules` yer alır gerektiğinde aşağıdaki komut ile yüklenir:

```
# modprobe modüladı
```

Modprobe, yüklenmek istenen modülün ihtiyaç duyduğu yüklenmemiş diğer modülleri de yükler. Birbirine ihtiyaç duyan modülleri tespit etmek için `depmod` komutu kullanılır. Bu komut `/lib/modules` altında `modules.dep` dosyasını oluşturarak ilişkili modüllerin veritabanını çıkartır.

Modüller yalnızca donanımlar için kullanılmaz. Programcılar için kolaylık sağladığından ağ protokollerinin uyarlanmasından, çekirdek tabanlı bir şifreleme algoritmasına kadar pek çok destek modüller ile sağlanabilmektedir.

Şu an yüklü modülleri görmek için:

```
$ lsmod
Module                  Size  Used by
iptable_filter         12536  0
ip_tables              22042  1 iptable_filter
x_tables               19118  2 ip_tables,iptable_filter
mperf                  12453  0
[...]
```

Modülleri silmek için:

```
# modprobe –r modüladı
```

sysfs Dosya Sistemi

sysfs dosya sistemi, sisteme bağlı cihaz ve veri yollarını kullanıcı ortamında erişilebilir bir dosya sistemi olarak sunar.

Linux 2.4 sürümde çekirdek ve sistem yapılandırmalarına ulaşmanın tek yolu `/proc` dosya sistemi idi. Zamanla bu dosya sistemi, fazlaca dosya ve klasör nedeniyle karmaşık bir hal almıştır. Bu nedenle 2.6

sürüm ile birlikte çekirdek programcıları `/proc` altında yalnızca çalışan süreçlere (process) ait bilgileri tutup çekirdeğe ait diğer bilgileri `/sys` altındaki sysfs dosya sistemine taşımıştır.

```
# ls -l /sys/devices/
toplam 0
drwxr-xr-x  3 root root 0 Kas  9 16:54 breakpoint
drwxr-xr-x  3 root root 0 Kas  9 16:54 cpu
drwxr-xr-x  6 root root 0 Kas  9 16:54 LNXSYSTM:00
drwxr-xr-x 45 root root 0 Kas  9 16:54 pci0000:00
drwxr-xr-x 10 root root 0 Kas  9 16:54 platform
drwxr-xr-x 17 root root 0 Kas  9 16:54 pnp0
drwxr-xr-x  3 root root 0 Kas  9 16:54 software
drwxr-xr-x  7 root root 0 Kas  9 16:54 system
drwxr-xr-x  3 root root 0 Kas  9 16:54 tracepoint
drwxr-xr-x 15 root root 0 Kas  9 16:54 virtual
```

Udev Dosya Sistemi

Linux'un eski sürümlerinde olası tüm aygıtları /dev klasörü altında statik olarak oluşturmakta idi. Ancak modüler yapının gelmesi ile birlikte bu yaklaşım da değiştirilmiştir. Bir aygıt yalnızca yüklendiğinde ona ait kullanıcı ortamında bir dosya oluşturulmalıydı. Udev (userpace dev) bu amaçla geliştirilmiş dosya sistemidir. Beraberinde namedev isimli aygıtları adlandırmak için kullanılan kütüphane ve çekirdekte olan aygıt hareketlerini dinleyerek gerekli eylemi gerçekleştiren udevd isimli servis de gelmektedir.

Çekirdek tarafından üretilen olay mesajları (bir USB cihazın takılması veya çıkartılması gibi) kullanıcı ortamında rastgele sırada ulaşır. Udevd servisi, hotplug olayların sıraya sokulmasını da sağlar. Udevd servisi başlamadan önce olan olaylara ait mesajlar, udevd başladıktan sonra çekirdek tarafından gönderilir. Sistem başlamadan takılan cihazlar coldplug, sistem çalışırken takılan cihazlar hotplug olarak adlandırılır.

HAL ve D-Bus

HAL (Hardware Abstraction Layer), çekirdekte aygıtlarla ilgili olan bir olayın kullanıcı ortamındaki bir yazılımla ilişkilendirilebilmesi için geliştirilmiştir. Örneğin bir dijital kamera takıldığında yeni fotoğrafların otomatik indirilmesi gibi.

Programları bilgilendirmek için HAL, süreçler arası haberleşme (IPC) tekniği ile oluşturulan D-Bus'ı kullanır. Programlar D-Bus üzerinden donanım olaylarını dinler ve bilgi alırlar. Aynı olay için birden fazla programa bilgi verilebilir.

Sistem Açılışı

Konu:

Donanımın açılışından kontrolün çekirdeğe ve ordan kullanıcıya geçene kadar geçen süreci anlama

Hedefler:

- PC ve işletim sistemi açılışını anlamak
- Açılış parametrelerini ihtiyaca göre özelleştirebilmeyi öğrenmek
- Açılışa dair günlük dosyasını okuyabilmek

Anahtar Kelimeler:

MBR, Grub, /var/log/messages, dmesg, BIOS, bootloader, kernel, init, syslog, initrd

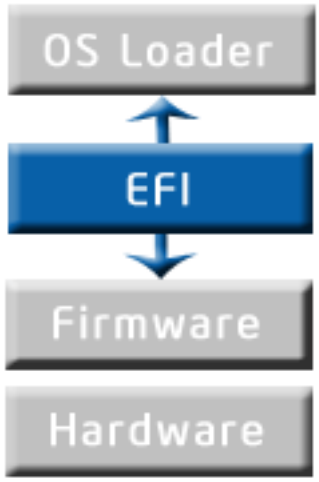
Açılış Kaydı Yükleyici (Boot Loader)

Bilgisayarınızı açtığınız zaman işlemci bildiği bir adresten kod çalıştırır. IBM uyumlu bir PC için bu adres, anakart üzerindeki flash belleğe yazılmış BIOS (Basic Input/Output System)'tur. BIOS, bilgisayarın CPU, bellek, disk gibi temel donanımları arar ve test eder. Daha sonra BIOS yapılandırmasında belirtilen sıra ile işletim sisteminin yükleneceği CD-ROM veya sabit disk gibi ortamı tespit eder.

İşletim sistemi olmadan işletim sistemini yüklemek zor bir işlemdir. O nedenle yükleme birkaç adımdan oluşmaktadır. Çünkü işletim sistemi henüz ortada olmadığı için dosya sistemi, disk yönetimi, istenen her kodun çalıştırılması söz konusu değildir. Yüklemenin her

adımında kazanılan bir yetenek ile daha fazlası yapılır ve bir sonraki adıma geçilir.

BIOS'un tespit ettiği ve işletim sistemini yüklemeye başlayacak özel kod içeren yükleme medyasının ilk 512 byte'lık kısmı "boot sector" veya "MBR (Master Boot Recor") olarak adlandırılır. Henüz ortada işletim sistemi olmadığı ve dosya sistemi tanınmadığı için diskin herhangi bir yerindeki işletim sistemini bulmak imkansızdır. O nedenle disklerin ilk 512 byte'ı sabittir ve yükleme işleminin başladığı noktadır. MBR boyutu küçük olduğu için, burada yükleme işini başlatacak ve görevi yükleme işini yapacak asıl programa devredecek küçük bir uygulama bulunmaktadır.

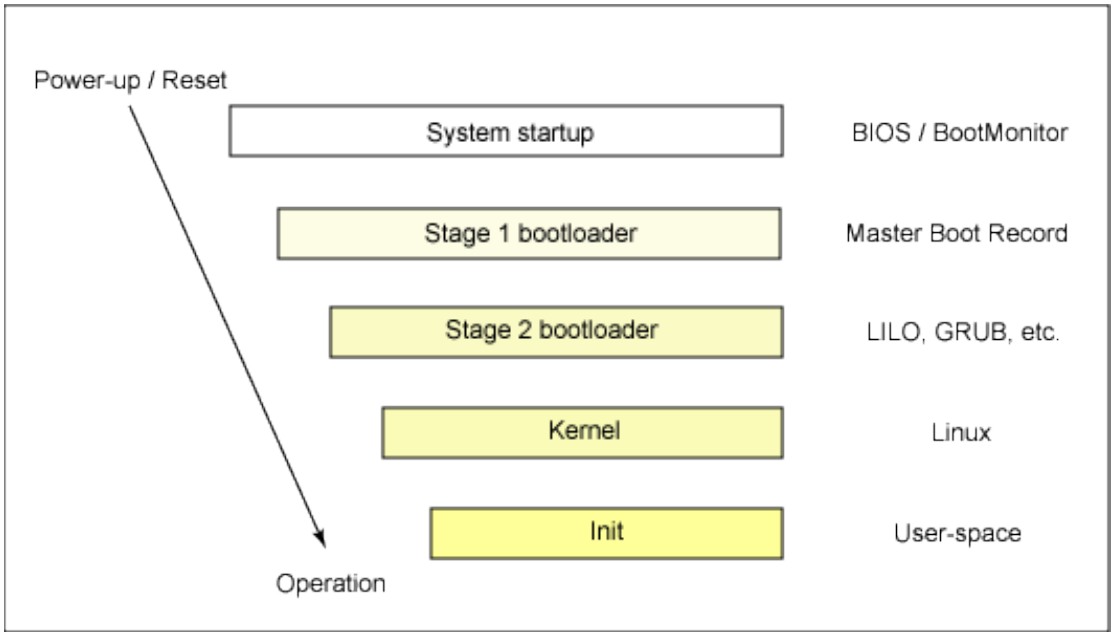


Intel, BIOS yapısının yerini alacak EFI standardını geliştirmiştir. EFI, işletim sistemi ile donanım arasında bir arayüz sunar. Bu arayüzün BIOS'a göre avantajları vardır:

- Büyük disklerden boot imkanı
- CPU bağımsız mimari
- CPU bağımsız sürücüler
- Ağ bağlantısı imkanı sunan esnek OS öncesi ortam
- Modüler yapı
- Daha fazla adreslenebilir bellek
- MBR'ye alternatif GUID Partion Table ile esnek bölümlendirme yapısı

Linux'ün GPT tabanlı diskleri okuyabilmesi için çekirdekte CONFIG_EFI_PARTITION aktif olmalıdır.

Linux altında yaygın olarak kullanılan iki açılış kaydı yükleyicisi vardır: LILO (Linux Loader) ve GRUB (Grand Unified Boot Loader). Açılış kaydı yükleyicileri temel olarak çekirdeğin yerini bulur ve başlatır. Bunun yanında açılış yöneticisi olarak çalışır ve birden fazla çekirdek yükleyip onlar arasında seçim yapılmasını da sağlar. GRUB, LILO'nun bazı dezavantajlı noktalarını kapatan daha profesyonel bir çözüm olarak daha çok tercih edilmektedir.



Çekirdek, sıkıştırılmış olarak (bzip veya gzip) saklandığından çalıştırılabilir değildir. Çekirdeğin başında minumum donanım yapılandırmasını yapacak ve takip eden sıkıştırılmış çekirdeği belleğe açacak kod parçası bulunmaktadır.

Çekirdeğin yüklenmesi esnasında GRUB tarafından belleğe yüklenmiş ilk kök dosya sistemi olacak initrd kopyalanır ve bağlanır (mount). Initrd, çekirdeğin açılış esnasında başka bir diske ihtiyaç duymadan kullanacağı bellekte yerleşmiş kök dosya sistemidir. Daha sonra gerçek dosya sistemine ulaşana kadar burayı kullanır. Yüklenen çekirdek modüler yapıyı destekler. Initrd dosya sisteminde, işletim sisteminin açılışı için gerekli modülleri içermektedir. Bu modüller yüklendikten sonra –ki bunların içinde asıl kullanılacak diskin sürücüsü de vardır- asıl dosya sistemine erişilir ve kök dosya sistemi olarak asıl disk bağlanır.

Asıl disk bağlandıktan sonra artık disk üzerindeki program ve betiklere ulaşılabilmektedir. Bu aşamada Linux, kullanıcı ortamının (user space) ilk çalışan süreci olacak init'i çalıştırır. Init, /etc/inittab yapılandırmasına göre açılış betiklerini çalıştırarak ayarlanmış çalışma seviyesinden işletim sisteminin diğer bileşenlerini çalıştırarak kullanıcı giriş ekranına kadar gelir. Bu aşamadan sonra kontrol kullanıcıya devredilir.

Grub

Grub, çekirdekten bağımsız çoklu önyükleme yapabilen küçük bir işletim sistemidir. Dosya sistemlerini tanımlar, açılış yapacak çekirdek imajlarını yükler ve yükleme işlemini yönetecek komut satırı sunar. Birden fazla işletim sistemi ve birden fazla disk arasında

seçim yapmayı sağladığı gibi kullanıcıdan aldığı parametreleri çekirdeğe geçer.

GRUB kendisini belleğe aşağıdaki adımlarda yükler:

- 1. Adım veya Birincil Önyükleyici: BIOS tarafından MBR'den okunarak yüklenir. 512 byte'dan küçüktür. Adım 1.5 veya Adım 2 açılış kaydını yükleme yeteneğine sahiptir.
- Adım 1.5 ön kayıt yükleme birincil önyükleyici tarafından belleğe gerektiğinde yüklenir. Bazı donanımlar (LBA mod harddiskte /boot bölümünün 1024'den yukarda bir silindirde yer alması gibi) yükleme yapabilmek için bu ara adıma ihtiyaç duyar.
- Adım 2 veya İkincil Önyükleyici belleğe yüklenir. Bu adım GRUB menüsünü gösterir ve komut satırı ortamını sunar. İşletim sistemi veya çekirdek seçilebilir, çekirdeğe özel parametreler gönderilebilir.
- İkincil önyükleyici, /boot/sysroot ile birlikte seçilen çekirdeği belleğe yükler ve kontrolü bu çekirdeğe devreder.

GRUB, ext2 dosya sistemlere ulaşabilir. Bu sayede LILO'da olduğu gibi /boot/grub/grub.conf ta yapılan değişiklikler için her defasında MBR'ye yükleme yapmaya gerek yok. Çünkü yükleme esnasında bu dosyaya kendisi ulaşarak yapılandırmayı okuyabilmektedir. GRUB yalnızca /boot 'un bulunduğu disk bölümü değişirse yüklemeye ihtiyaç duyar.

```
# /sbin/grub-install /dev/hda
```

GRUB komut satırı yüklemeyle ilgili birçok komutu kullanıma sunar. Bunlardan bir kısmı aşağıda verilmiştir:

boot
İşletim sistemini yükler.

displaymem
BIOS'tan alınan bilgilere göre mevcut bellek kullanımı gösterilir.

initrd /initrd-2.6.4.img
Kullanıcının başlangıç için ram disk belirtmesini sağlar.

install <stage-1> <install-disk> <stage-2> p <config-file>
GRUB'u MBR'ye yükler.
Stage-1: Birincil Önyükleyicinin bulunduğu yer. Örnek: (hd0,0)/grub/stage1

kernel /vmlinuz-2.6.4.img ro root=/dev/VolGroup00/LogVol00
İşletim sistemi için yüklenecek çekirdeği belirler. Buradan çekirdeğe parametre gönderilebilir. (Örnekteki ro gibi)

help –all komutu kullanılarak tüm komut ve parametreler hakkında bilgi alınabilir.

Bir işletim sisteminin GRUB menüsünde görünmesi için /boot/grub/grub.conf yapılandırma dosyasında olması gerekmektedir. Aşağıda hem Linux hem de MS Windows işletim sistemlerinden açılış yapabilecek örnek bir yapılandırma dosyası yer almaktadır:

```
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Pardus (2.6.4)
root (hd0,0)
kernel /boot/vmlinuz-2.6.4 ro root=LABEL=/1 rhgb quiet
initrd /boot/initrd-2.6.4.img

# section to load Windows
title Windows
rootnoverify (hd0,0)
chainloader +1
```

Ekranda GRUB menüsü görüntüğünde ‘a’ tuşuna basarak çekirdek parametreleri düzenlenebilir. Örneğin a’ya basıldığında gelen ekranda boşluk bırakıp çalışma zamanı yazılarak açılış esnasında çalışma zamanı değiştirilebilir:

```
grub append> ro root=/dev/VolGroup00/LogVol00 rhgb quiet 3
```

‘e’ harfine basarak açılıştan önce komut düzenlenebilir. Örneğin açılış diski değişmişse root komutuyla bunun gruba bildirilmesi gerektiğinde. Düzenlemeler sadece o açılış için geçerlidir, kalıcı değildir.

‘c’ harfine basarak Bash benzeri bir komut satırı alınır.

Açılış Günlüğü

Açışıl sorunlarının çözümü, araştırma imkanları sınırlı olduğundan bazı durumlarda zor olabilmektedir. Linux sistemler açılırken çekirdek ekrana bir takım mesajlar basar. Bazen bu mesajlar, açılışta kullanılan bir grafiğin arkasında kalmaktadır. ESC tuşuna basarak bu grafik kaldırılıp alttaki mesajlar görülebilir.

Çekirdek loğları açılışta özel bir tampon alana yazılır ve daha sonra ‘dmesg’ komutu ile görüntülenir. Sistem günlüğünü tutan syslogd hizmeti devreye girdikten sonra ise bu loğlar syslogd üzerinden

günlük dosyalarına yazılır. Başlangıç mesajları da dahil, bu günlükler /var/log/syslog dosyasında bulunur.

```
# tail /var/log/syslog
Dec 14 22:43:29 pardus postfix/local[9759]: 513CA17FAC1:
to=<root@pardus>, orig_to=<root>, relay=local, delay=0.62,
delays=0.32/0.27/0/0.03, dsn=2.0.0, status=sent (delivered to
command: procmail -a "$EXTENSION")
Dec 14 22:43:29 pardus postfix/qmgr[3296]: 513CA17FAC1:
removed
Dec 14 22:45:01 pardus /USR/SBiN/CRON[9817]: (root) CMD
(command -v debian-sa1 > /dev/null && debian-sa1 1 1)
Dec 14 22:45:32 pardus dbus[2423]: [system] Activating service
name='org.freedesktop.PackageKit' (using servicehelper)
Dec 14 22:45:32 pardus AptDaemon: INFO: Initializing daemon
Dec 14 22:45:32 pardus AptDaemon.PackageKit: INFO:
Initializing PackageKit compat layer
Dec 14 22:45:32 pardus dbus[2423]: [system] Successfully
activated service 'org.freedesktop.PackageKit'
Dec 14 22:50:32 pardus AptDaemon: INFO: Quitting due to
inactivity
Dec 14 22:50:32 pardus AptDaemon: INFO: Quitting was requested
Dec 14 22:55:01 pardus /USR/SBiN/CRON[9854]: (root) CMD
(command -v debian-sa1 > /dev/null && debian-sa1 1 1)
```

Çalışma Seviyeleri

Konu:

Kontrolün çekirdekten kullanıcıya geçmesi ve farklı çalışma seviyeleri için kullanıcı ortamının hazırlanması

Hedefler:

- Init programını anlamak
- Çalışma seviyesini belirleyebilmek

- Bilgisayarı kapatmak, yeniden başlatmak
- Açılış betiklerini anlamak

Anahtar Kelimeler:

/etc/inittab, shutdown, init, telinit, /etc/init.d

Init

Çekirdek açılışını tamamlayıp kullanıcı ortamına geçiş yaptığında çalışan ilk program ‘/sbin/init’ olup PID numarası daima 1’dir. Bu süreç teorik olarak hiçbir zaman ölmez ve bütün diğer süreçlerin anasıdır. ‘init’ süreci ölse de çekirdek çalışmaya devam eder.

Çekirdeğin başka bir init programı çalıştırması için açılış esnasında parametre geçilmesi gerekir. init=/sbin/sysinit

SysV Init

Linux dağıtımların pek çoğu UNIX SysV init yapısını kullanır. Bu yapıda çalışma seviyeleri vardır (run levels) ve her bir çalışma seviyesini hazır edecek betikler vardır. Böylece farklı farklı çalışma ortamları elde edilir. Örneğin sunucu sistemler genelde grafik arayüzün olmadığı bir seviyeden açılırken, disk bakımı yapacak bir sistem yöneticisi diske yazmanın en alt düzeyde olduğu bir seviyeden açılış yapar.

Çalışma seviyeleri 0-6 arası bir rakam olmak üzere 7 tanedir.

0 – Halt
1 – Tek kullanıcılı minimal açılış, bakım veya kurtarma modu
2, 3, 4, 5 – Çok kullanıcılı açılış. 3 text konsol, 5 grafik arayüz.
6 – Reboot

Sistem /etc/inittab dosyasında belirtilen initdefault değeriyle açılır. Çalışma seviyesini değiştirmek için bu değerin değiştirilmesi gerekmektedir. Pardus için öntanımlı değeri 2’dir.

```
# Level to run in
id:2:initdefault:

# Boot-time system configuration/initialization script.
si::sysinit:/etc/init.d/rcS
```

```
# What to do in single-user mode.
~:S:wait:/sbin/sulogin

# /etc/init.d executes the S and K scripts upon change
# of runlevel.
#
# Runlevel 0 is halt.
# Runlevel 1 is single-user.
# Runlevels 2-5 are multi-user.
# Runlevel 6 is reboot.

l0:0:wait:/etc/init.d/rc 0
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
l4:4:wait:/etc/init.d/rc 4
l5:5:wait:/etc/init.d/rc 5
l6:6:wait:/etc/init.d/rc 6

# What to do at the "3 finger salute".
ca::ctrlaltdel:/sbin/shutdown -t1 -h now

# Runlevel 2,3: getty on virtual consoles
# Runlevel   3: getty on terminal (ttyS0) and modem (ttyS1)
1:23:respawn:/sbin/getty tty1 VC linux
2:23:respawn:/sbin/getty tty2 VC linux
3:23:respawn:/sbin/getty tty3 VC linux
4:23:respawn:/sbin/getty tty4 VC linux
S0:3:respawn:/sbin/getty -L 9600 ttyS0 vt320
S1:3:respawn:/sbin/mgetty -x0 -D ttyS1
```

inittab dosyasında geçen bazı kelimelerin anlamları:

respawn: Sonlandığında otomatik olarak yeniden başlatılan program.
wait: Bir kere çalışır ve init bitmesi için beklemede olur.
initdefault: Öntanımlı çalışma seviyesi.
ctrlaltdel: Ctrl+Alt+Del tuş kombinasyonuna basıldığında ne yapılacak.

Açılış esnasında ‘e’ harfine basarak komut düzenlemesi yaparak çalışma seviyesi geçici olarak değiştirilebilir. Çekirdek seçilerek komutun sonuna çalışma seviyesi rakamla yazılır ve sistem açılır. Örneğin çalışma seviyesini 3 yapmak için:

```
kernel /boot/vmlinuz-2.6.30 root=/dev/sda2 ro 3
```

Çalışma seviyesine ait bütün betikler /etc/init.d altında yer alır. Her bir çalışma seviyesi için /etc/rcX.d şeklinde bir dizin olup o seviyede çalıştırılacak yada durdurulacak betiklere ait sembolinkler linkeri içerir. Yani /etc/init.d altındaki bütün betikler çalıştırılmaz, sadece

ilgili çalışma seviyesinden linklenmiş betikler çalışır. Çalışma seviyesi dizinleri şunlardır:

rc0.d rc1.d rc2.d rc3.d rc4.d rc5.d rc6.d

rc2.d içeriği aşağıdaki gibidir:

```
$ ls /etc/rc2.d/
README            S19binfmt-support  S21acpid
S21loadcpufreq        S21ssh              S22postfix
S25bootlogs
S01decnet         S19dnet-progs      S21acpi-support  S21mysql
S21sysstat          S23openvpn      S25samba
S01motd           S19preload         S21atd           S21ntp
S21xinetd           S24cups         S26winbind
S01nvidia-kernel  S19rsyslog         S21cron
S21quotarpc           S22avahi-daemon     S24gdm3
S27plymouth
S16rpcbind        S19sudo            S21dbus          S21radvd
S22bluetooth        S24pulseaudio   S27rc.local
S17nfs-common     S19syslog-ng       S21iodined       S21rsync
S22cpufrequtils     S24saned        S27rmnologin
S19acpi-fakekey   S20apache2         S21irqbalance
S21speech-dispatcher  S22network-manager  S24scanbuttond
S27stop-readahead-fedora S89cron
```

Sembolik link isimleri S (Start) veya K (Kill) ile başlar. S ile başlayanlar bir servisi başlatırken K ile başlayanlar sonlandırır. S ile başlayanlarda betiğe ‘start’, K ile başlayanlarda ‘stop’ parametresi gönderilir. Rakamlar (01-99), bir sıralama oluşması içindir. Bazı servislerin diğerlerinden once başlaması gerekebilir. Bu durumda daha küçük rakam verilir. Örneğin:

```
S20exim4 -> ../init.d/exim4
S20fam -> ../init.d/fam
S20hylafax -> ../init.d/hylafax
S20inetd -> ../init.d/inetd
S20mailman -> ../init.d/mailman
S20makedev -> ../init.d/makedev
S20pcmcia -> ../init.d/pcmcia
S20samba -> ../init.d/samba
S20ssh -> ../init.d/ssh
```

Bir önceki ve şimdiki çalışma seviyesini görmek için:

```
# runlevel
N 2
```

Çalışma seviyesini bir seferliğine değiştirmek için init veya telinit komutu kullanılabilir:

```
# telinit 3
```

Sistemi kapatmak (halt) veya yeniden başlatmak (reboot) için telinit kullanılabilir. Ancak daha güvenli yok olarak shutdown komutu kullanılabilir. ‘halt’ ve ‘reboot’ komutları da aynı işlevi görürler. Bu iki komut da nihayetinde shutdown’u farklı parametrelerde çağıran kısaltmalardır. Aynı şekilde ‘poweroff’ komutu da yine shutdown’u kullanan farklı bir kısaltmadır.

```
# shutdown [-t sec] [-arkhncfFHP] süre [uyarı mesajı]
```

‘süre’ parametresi olarak ‘now’ denirse işlem hemen gerçekleştirilir. İstenirse ileri bir zaman verilebilir. Genel kullanıma açık sunucularda system kapamaları planlıdır ve önceden duyurulur.

Aşağıda farklı shutdown kullanımları görülmektedir.

```
# shutdown -h now

# shutdown -r now

# shutdown -h 16:30

# shutdown -r +10
```

Son kullanımdaki +10, şu andan itibaren 10 dakika sonra reboot et demektir.

Bütün komutları verebilmek için sisteme ‘root’ olarak giriş yapmak gerekmektedir. Bilgisayarı kapatmanın diğer yolu da, sisteme hiç giriş yapmadan Ctrl+Alt+Del tuş kombinasyonuna basılarak yapılır. Bunun için sistemin klavyesine fiziksel olarak erişmek gerekmektedir.

Upstart

SysV init modelinde sıralı ve senkron bir yapı vardır. Başlangıç betiklerinin belli bir sırada ve yalnızca kullanıcının çalışma seviyesi değiştirmesi durumunda çalışmasına karşın Upstart, asenkron ve olay (event) bazlıdır. SysV init modelindeki betikler yerine burada görevler (Jobs) vardır. Ubuntu tarafından desteklenen Upstart uyumluluk için SysV betikleri desteklemektedir.

Upstart, arkaplanda sürekli çalışacak hizmetler ile geçici süreyle çalışan görevleri birbirinden ayırmaktadır. Upstart’ın en önemli farkı SysV init betiklerinde olduğu gibi karmaşık ve uzun şablonlu betiklere ihtiyaç duymamasıdır. Aşağıda örnek bir betik görülmektedir:

```
description “ornek servis”
start on filesystem
```

```
stop on runlevel [06]

expect fork
respawn

exec servis -d9 -r
```

init programını control etmek için initctl komutu kullanılır:

```
# initctl list
```

Paket Yönetimi

- **Konu:**

Pardus ve Red Hat paket yönetim sistemlerini tanımak

Hedefler:

- Pardus altında paket kurmak, kaldırmak ve güncelleyebilmek
- Red Hat altında paket kurmak, kaldırmak ve güncelleyebilmek
- Kaynak kodundan program kurmak

Anahtar Kelimeler:

apt-get, dpkg, yum, rpm, rpm2cpio, ldconfig, LD_LIBRARY_PATH

Paket Yönetimi

Linux altında paket yönetimi pakete ait tüm operasyonları(kurma, güncelleme ve kaldırma) kapsar. Bir uygulama kurulumu iki türlü yapılabilmektedir.

1. Hazır paketlerden kurulum yapmak.
2. Açık kod uygulamayı kaynak kodundan derleyerek kurmak.

Paketten kurulum oldukça kolay olup tüm gereksinimler ve kurulum sonrası yapılacak

İkinci yöntem olan kaynak kodundan derleme bazı avantajları olsada birinciye göre fazla tercih edilmemektedir. Bir pakedi kaynak kodundan derlerken, derleyici

parametreleri sistemdeki işlemciye göre değiştirilerek uygulamanın daha hızlı çalışması sağlanabilir. Bazen de hazır paket ihtiyaç duyduğumuz bir parametre ile derlenmediği için kaynak koddan kurmak gerekebiliyor..Ancak kod derlemek, derleme sonrasında açılış betiklerini ayarlamak gibi işler yeni başlayanlar için zor olabilir.

Hazır paketten kurma yöntemide kullanıcı paket ve bağımlılıklarının nasıl kurulması gerektiğini bilmesine gerek kalmadan kolayca kurulum ve güncelleme yapabilir.

Pardus Paket Yönetimi

Pardus paket yönetimi olarak Debian paket yöneticisini kullanmaktadır. Debian paketleri .deb uzantısı ile ifade edilir. Paketleri yönetmek için kullanılan komutlar: dpkg, apt-get apt-cache alien ve aptitude

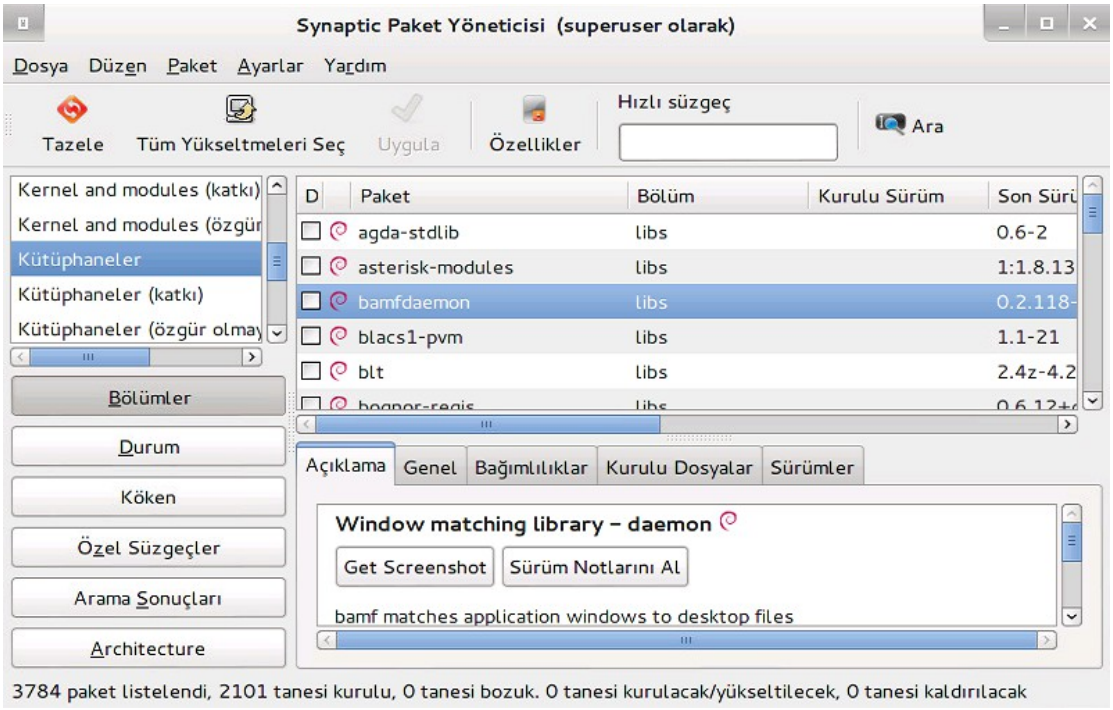
APT(Advanced Packaging Tool) paketlerin ihtiyaç duyduğu paketleri de belirleyerek otomatik olarak kuran bir paket yönetim aracıdır.

Paketlerin internet üzerinde tutulduğu yerlere depo(repository) adı verilir. Pardus'ta kullanılacak depoları /etc/apt/sources.list dosyasında tanımlanır.

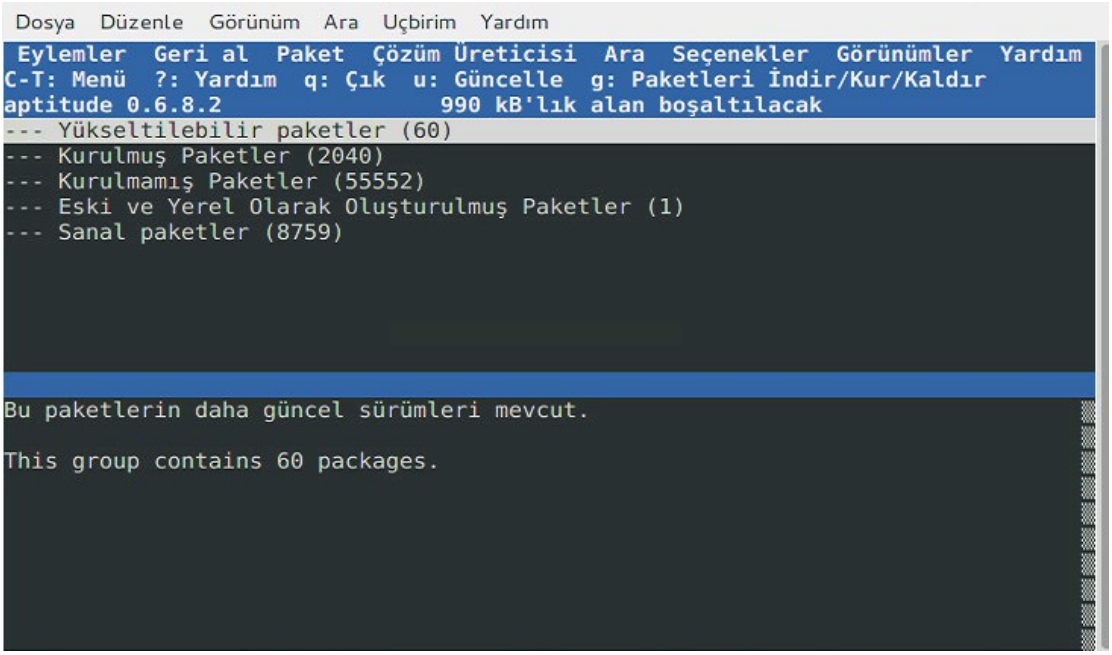
```
# cat /etc/apt/sources.list
deb [arch=amd64,i386] ftp://depo.pardus.org.tr/kurumsal/anahavuz
wheezy                   main contrib non-free
deb [arch=amd64,i386] ftp://depo.pardus.org.tr/kurumsal/anahavuz
wheezy-proposed-updates  main contrib non-free
deb [arch=amd64,i386] ftp://depo.pardus.org.tr/kurumsal/cokluortam
wheezy                   main         non-free
deb [arch=amd64,i386] ftp://depo.pardus.org.tr/kurumsal/guvenlik
wheezy/updates           main contrib non-free
deb [arch=amd64,i386] ftp://depo.pardus.org.tr/kurumsal/ozelhavuz
wheezy                   main         non-free
```

Dpkg komutu RPM dağıtımlarındaki rpm komutuna karşılık gelir. Apt-get ve apt-cache komutları da yum eşleniği denilebilir. Aptitude metin tabanlı arayüz ile paket yönetimi yapmaktadır. Alien komutu ile deb/rpm paket dönüşümü yapılabilmektedir. Bir yazılım rpm paketi var fakat deb formatında paketi yoksa alien komutu sayesinde rpm paketinden debian paketine dönüşüm yapılabilir. Tam tersi olarak deb paketinden de rpm paketi yapılabilmektedir.

Grafik arayüzünde synaptic paket yöneticisi sayesinde görsel olarak paket kurulumu yapılabilir.



Komut satırında ise menu tabanlı aptitude paket yöneticisi kullanılabilir.

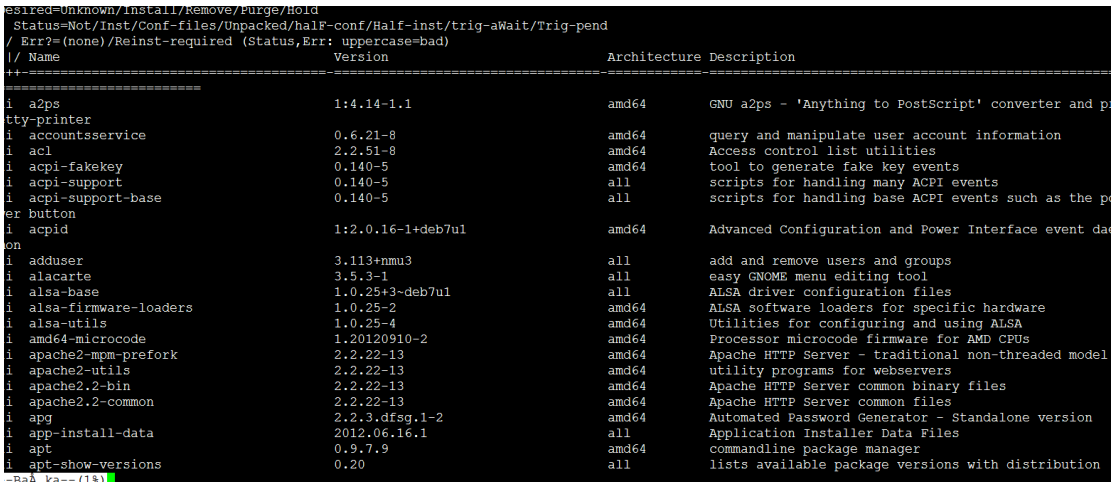


Sonraki bölümlerde kabuk üzerinde çalıştırılabilecek komutlar ile paket yönetimi anlatılacaktır.

Paket sorgulama

dpkg -l veya --list paremetresi ile sistemde kurulu tüm paketler listelenir.

 # dpkg –l



dpkg –S parametresi ile bir dosyanın hangi paket tarafından kurulduğu öğrenilir.

 # dpkg -S /bin/ls
coreutils: /bin/ls

Yukarıdaki çıktıya göre /bin/ls dosyası coreutils isimli paket ile birlikte gelmektedir.
dpkg –L komutu ile bir paketin kurduğu tüm dosya ve komutlar listelenir.

 # dpkg -L vim
/.
/usr
/usr/bin
/usr/bin/vim.basic
/usr/share
/usr/share/doc
/usr/share/bug
/usr/share/bug/vim
/usr/share/bug/vim/script
/usr/share/bug/vim/presubj
/usr/share/lintian
/usr/share/lintian/overrides
/usr/share/lintian/overrides/vim
/usr/share/doc/vim

Bir paketin ihtiyaç duyduğu tüm bağımlı paketleri listelemek için apt-cache depends komutu kullanılır.

```
# apt-cache depends vim
vim
  Depends: vim-common
  Depends: vim-runtime
  Depends: libacl1
  Depends: libc6
  Depends: libgpm2
  Depends: libselinux1
  Depends: libtinfo5
  Suggests: <ctags>
    exuberant-ctags:i386
    exuberant-ctags
  Suggests: vim-doc
  Suggests: vim-scripts
  Conflicts: vim:i386
```

apt-cache search komutu ile tanımlı paket depolarında bulunan paket listesinde araması yapılabilir. Aşağıdaki örnekte içinde vim geçen tüm paketler listelenmiştir.

```
# apt-cache search vim
apvlv - PDF viewer with Vim-like behaviour
bicyclerepair - A refactoring tool for python
cernlib-base - CERNLIB data analysis suite - common files
vim-migemo - VIM plugin for C/Migemo
...
```

Paket Kurma

Pardus paket depoları /etc/apt/sources.list dosyasında tanımlanır.

```
# cat /etc/apt/sources.list
deb [arch=amd64,i386] ftp://depo.pardus.org.tr/kurumsal/anahavuz
wheezy                  main contrib non-free
deb [arch=amd64,i386] ftp://depo.pardus.org.tr/kurumsal/anahavuz
wheezy-proposed-updates  main contrib non-free
deb [arch=amd64,i386] ftp://depo.pardus.org.tr/kurumsal/cokluortam
wheezy                  main         non-free
deb [arch=amd64,i386] ftp://depo.pardus.org.tr/kurumsal/guvenlik
wheezy/updates          main contrib non-free
deb [arch=amd64,i386] ftp://depo.pardus.org.tr/kurumsal/ozelhavuz
wheezy                  main         non-free
```

apt-get install parametresi veya dkg --install ile Aşağıdaki komutlarla paket kurulabilir.

apt-get install paketadı
dpkg –i paketadı.deb veya dpkg –install paketadı.deb

```
# apt-get install zsh
Reading package lists... Done
Building dependency tree
Reading state information... Done
..
```

Paket Silme

dpkg –r veya apt-get remove komutu ile yapılandırma dosyaları hariç tüm dosyalar silinir. Herşeyi silmek için de apt-get purge parametresi kullanılır.

```
# apt-get remove zsh
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer
required:
  libdbi1 libevtlog0 libmongo-client0 libnet1 libsyslog-ng-3.3.5
Use 'apt-get autoremove' to remove them.
The following packages will be REMOVED:
  zsh
0 upgraded, 0 newly installed, 1 to remove and 40 not upgraded.
After this operation, 11,7 MB disk space will be freed.
Do you want to continue [Y/n]? y
(Reading database ... 184455 files and directories currently
installed.)
Removing zsh ...
Processing triggers for menu ...
Processing triggers for man-db ...
```

```
# apt-get purge zsh
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer
required:
  libdbi1 libevtlog0 libmongo-client0 libnet1 libsyslog-ng-3.3.5
Use 'apt-get autoremove' to remove them.
The following packages will be REMOVED:
  zsh*
0 upgraded, 0 newly installed, 1 to remove and 40 not upgraded.
After this operation, 0 B of additional disk space will be used.
```

```
Do you want to continue [Y/n]? y
(Reading database ... 183293 files and directories currently
installed.)
Removing zsh ...
Purging configuration files for zsh ...
Processing triggers for menu ...
```

Paket Güncelleme

apt-get update komutu ile gerçek manada bir paket güncellemesi yapılmaz. Sadece paketin indeks dosyaları ve kaynakları güncellenir. apt-get upgrade ile de yeni sürümler kurulur.

```
# apt-get update
Get:1 ftp://depo.pardus.org.tr wheezy Release.gpg [1.672 B]
Get:2 http://download.opensuse.org ./ Release.gpg [189 B]
Get:3 ftp://depo.pardus.org.tr wheezy-proposed-updates Release.gpg
[836 B]
Hit http://download.opensuse.org ./ Release
Hit ftp://depo.pardus.org.tr wheezy Release.gpg
..
```

Red Hat Paket Yönetimi

Pardus, Red Hat paket yönetim sistemlerinin hiçbirini kullanmaz ancak Red Hat ve klonu işletim sistemlerinin yaygın kullanılması Pardus sistemlere destek veren mühendislerin bu popüler paket yönetim sistemini de bilmesini gerekli kılmaktadır.

RPM

RPM, Red Hat'in klasik paket yönetim aracıdır. Paket kurmak, kaldırmak ve güncellemek için kullanılır.

Temel rpm komutları aşağıdaki tabloda yer almaktadır.

İşlem	Kısa Parametre	Uzun Parametre
Güncelleme/Kurma	-U	--upgrade
Kurma	-i	--install
Silme	-e	--erase
Sorgulama	-q	--query
Doğrulama	-V	--verify
İmza Kontrolü	K	--checksig
Tazelem,sadece kurulu ise güncelle	-F	--freshen
Veritabanını ilkle	Yok	--initdb
Veritabanını tekrar	Yok	--rebuilddb

oluştur		

```
# rpm -ivh xsnow-1.41-1.i386.rpm
xsnow ##################################################
```

Paket zaten kurulu ise aşağıdaki gibi bir hata mesajı alınır:

```
# rpm -ivh xsnow-1.41-1.i386.rpm
package xsnow-1.41-1 is already installed
```

Bu durumda paketin önce silinmesi gerekir. Paket silmek için:

```
# rpm -e xsnow
error: removing these packages would break dependencies:
/usr/X11R6/bin/xsnow is needed by x-amusements-1.0-1
```

Bir pakete ait yazılımlar bir başka paket tarafından kullanılıyorsa rpm doğrudan silinmesine izin vermez. Öncelikle silinecek pakete ihtiyaç duyan paketin silinmesi gerekir. Eğer ne yaptığımızdan eminsek rpm'i aşağıdaki gibi --force ile zorlayabiliriz:

```
# rpm -ivh --force xsnow-1.41-1.i386.rpm
xsnow ##################################################
```

Mevcut paketin yeni sürümü çıkmış ise aşağıdaki komut ile mevcut paket güncellenir:

```
# rpm -Uvh xsnow-1.42-1.i386.rpm
xsnow ##################################################
```

Eğer paket kurulu değilse, güncelleme anlamındaki -U, yükleme anlamındaki -i gibi çalışır.

Bir paketin kurulu olup olmadığını sorgulamak için:

```
# rpm -q xsnow
xsnow-1.41-1
```

Paket içerisinden çıkan dosyaları listelemek için:

```
# rpm -ql xsnow
/etc/X11/applnk/Games/xsnow.desktop
/usr/X11R6/bin/xsnow
```

Kurulu tüm paketleri görmek için:

```
# rpm -qa | less
filesystem-2.4.0-1.el5.centos
termcap-5.5-1.20060701.1
glibc-2.5-12
glib2-2.12.3-2.fc6
```

...

-V parametresi ile orijinal kurulumdan sonra paketin dosyalarında değişiklik var mı diye kontrol edilebilir. Sisteme bir saldırgan girdiğinde hangi programları vs değiştirdiğini anlamada oldukça faydalı olmaktadır.

rpm2cpio

RPM olarak paketlenmiş bir dosyayı cpio arşiv formatına çevirir. Genel olarak RPM paketten istenen bir dosyayı çıkartmak için kullanılır.

```
# rpm2cpio logrotate-1.0-1.i386.rpm > blah.cpio
# file blah.cpio
blah.cpio: ASCII cpio archive (SVR4 with CRC)
#
```

rpm2cpio, tıpkı cpio gibi Stdout'a yazdığından çıktının bir dosyaya yönlendirilmesi gerekmektedir.

Diğer yöntem:

```
# cat logrotate-1.0-1.i386.rpm | rpm2cpio > blah.cpio
```

Bu aşamadan sonra cpio üzerindeki tüm işlemler yapılabilir.

Paket içeriğini görüntülemek için:

```
# rpm2cpio logrotate-1.0-1.i386.rpm  | cpio -t
usr/man/man8/logrotate.8
usr/sbin/logrotate
14 blocks
```

Paketin içerisinden bir dosya çıkartmak için:

```
# rpm2cpio logrotate-1.0-1.i386.rpm |cpio -ivd
usr/man/man8/logrotate.8
usr/man/man8/logrotate.8
14 blocks
```

Yum Yapılandırma

Yum yapılandırma dosyası /etc/yum.conf dosyasıdır. Öntanımlı dosya aşağıdaki gibidir:

```
[main]
```

```
cachedir=/var/cache/yum
keepcache=0
debuglevel=2
logfile=/var/log/yum.log
exactarch=1
obsoletes=1
gpgcheck=1
plugins=1
installonly_limit=2
```

cachedir:
Yum'un önbellek ve dosya veritabanını tutacağı dizin

keepcache:
1 veya 0 olabilir.

logfile:
Yum'un günlüklerini yazacağı dosya

exactarch:
1 veya 0 olabilir. Yalnızca bilgisayar mimarisine (i386, i686, x64 gibi) ait
paketleri yükleme kararı

gpgcheck:
1 veya 0 olabilir. Paketin sayısal imzasını kontrol etme kararı

Yum depo alanları yum.conf'ta [repository] bölümü açılarak tanımlanabilir. Ancak genelde /etc/yum.repos.d dizini altındaki .repo dosyalarında tanımlanır. /etc/yum.conf altında aşağıdaki gibi bildirimle depo tanımlarının (.repo dosyaları) olduğu dizin belirtilir.

reposdir=/etc/yum.repos.d/

Depo bölümleri en az aşağıdaki iki değeri içermelidir:

```
[repository]
name=repository_name
baseurl=repository_url
```

name:
Depo adı

baseurl:
Depo'nun tam yolu

Baseurl aşağıdaki biçimdedir:

baseurl=http://path/to/repo/rel/$releasever/server/$basearch/os/

Depoyu etkin veya devre dışı yapmak için enabled tanımı yapılır. 1 veya 0 olur. 1 olması durumunda paket yüklemelerinde kaynak olarak kullanılacaktır.

enabled=1

Örnek:

```
[red-hat-enterprise-linux-sfs]
name = Red Hat Enterprise Linux Scalable FS
baseurl = https://cdn.redhat.com/content/dist/rhel/entitlement-
6/releases/$releasever/$basearch/scalablefilesystem/os
enabled = 1
gpgcheck = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
sslverify = 1
sslcacert = /etc/rhsm/ca/redhat-uep.pem
sslclientkey = /etc/pki/entitlement/key.pem
sslclientcert = /etc/pki/entitlement/11300387955690106.pem
```

Yum yapılandırmalarında kullanabileceğiniz değişkenler vardır:

$releasever
: Sürüm numarası

$arch
: Mimari

yumdownloader

Yum-utils paketi gelen bu programcık ihtiyaç duyulan rpm paketi indirmek için kullanılır. Normalde Yum güncel olan paketi indirmez. Ancak bu pakete ihtiyacınız varsa yumdownloader kullanabilirsiniz.

Yum dosyayı indirmiyor:

```
$ yum --downloadonly install openssh-server
Loading "downloadonly" plugin
Setting up Install Process
Setting up repositories
Reading repository metadata in from local files
Parsing package install arguments
Nothing to do
```

Yumdownloader her durumda indiriyor:

```
$ yumdownloader openssh-server
openssh-server-4.3p2-19.f 100% |=========================| 252
kB    00:01
```

Kaynak kod paketini indirmek için:

```
$ yumdownloader --source zsh
...
Enabling fedora-source repository
fedora-source/metalink                          | 1.7 kB    00:00
zsh-4.3.10-5.fc13.src.rpm                       | 2.6 MB    00:30
```

Yum

Yum, Red Hat'in paket yöneticisi olup paketler hakkında sorgular yapılmasına, paketlerin depolardan çekilmesine, yüklenmeleri ve kaldırılmalarına ve bütün sistemin güncellenmesine imkan tanır.

Yum aynı zamanda paket bağımlılıklarını otomatik tespit ederek kaldırılan yada yüklenen paketin ihtiyaç duyduğu diğer paketleri de yükler.

Yum, kendi paket deponuzu kurmanızı da sağlar.
Yum'u kullanmak için super kullanıcı olmanız gerekmektedir.

Güncel Paket Sorgulama

Sisteminizdeki hangi paketin güncellemesini çıktığını öğrenmek için:

```
~]# yum check-update
Loaded plugins: product-id, refresh-packagekit, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
PackageKit.x86_64                    0.5.8-2.el6                    rhel
PackageKit-glib.x86_64               0.5.8-2.el6                    rhel
PackageKit-yum.x86_64                0.5.8-2.el6                    rhel
PackageKit-yum-plugin.x86_64         0.5.8-2.el6                    rhel
glibc.x86_64                         2.11.90-20.el6                 rhel
glibc-common.x86_64                  2.10.90-22                     rhel
kernel.x86_64                        2.6.31-14.el6                  rhel
kernel-firmware.noarch               2.6.31-14.el6                  rhel
rpm.x86_64                           4.7.1-5.el6                    rhel
rpm-libs.x86_64                      4.7.1-5.el6                    rhel
rpm-python.x86_64                    4.7.1-5.el6                    rhel
udev.x86_64                          147-2.15.el6                   rhel
yum.noarch                           3.2.24-4.el6                   rhel
```

Paket Güncelleme

Tek bir paketi güncellemek için:

```
~]# yum update udev
Loaded plugins: product-id, refresh-packagekit, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
Setting up Update Process
Resolving Dependencies
--> Running transaction check
---> Package udev.x86_64 0:147-2.15.el6 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package       Arch            Version                 Repository      Size
================================================================================
Updating:
 udev          x86_64          147-2.15.el6            rhel           337 k

Transaction Summary
================================================================================
Install       0 Package(s)
Upgrade       1 Package(s)

Total download size: 337 k
Is this ok [y/N]:
```

Paket Arama

Bütün RPM paketlerinin isminde, tanımında ve özet bilgisinde anahtar kelime kullanılarak arama yapılabilir. Birden fazla anahtar kelime belirtilebilir.

Arama, hangi paketi kuracağınızdan emin olmadığınız ancak neyle ilgili olduğunu tahmin ettiğinizde kullanılabilirsiniz.

```
~]# yum search meld kompare
Loaded plugins: product-id, refresh-packagekit, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
============================== Matched: kompare ==============================
kdesdk.x86_64 : The KDE Software Development Kit (SDK)
Warning: No matches found for: meld
```

Paket Listeleme

Paket veya paket grupları hakkında bilgi almak için list kullanılır. Arama yaparken ifadeler kullanılabilir.

Arama, hangi paketi kuracağınızdan emin olmadığınız ancak neyle ilgili olduğunu tahmin ettiğinizde kullanılabilirsiniz.

```
$ yum list kernel
Installed Packages
kernel.x86_64            2.6.18-238.12.1.el5      installed
Available Packages
kernel.x86_64             2.6.18-371.1.2.el5      updates
```

yum list all
: Bütün yüklü veya kullanıma hazır paketleri listeler

yum list installed
: Sadece yüklenmiş paketleri gösterir

yum grouplist
: Paket gruplarını listeler

yum repolist
: Depo ID'si, ismi ve sunulan paket sayısını listeler

Paket Bilgisi

Bir veya daha fazla paket hakkında bilgi almak için info kullanılır.

```
~]# yum info abrt
Loaded plugins: product-id, refresh-packagekit, subscription-manager
Updating Red Hat repositories.
INFO:rhsm-app.repolib:repos updated: 0
Installed Packages
Name        : abrt
Arch        : x86_64
Version     : 1.0.7
Release     : 5.el6
Size        : 578 k
Repo        : installed
From repo   : rhel
Summary     : Automatic bug detection and reporting tool
URL         : https://fedorahosted.org/abrt/
License     : GPLv2+
Description: abrt is a tool to help users to detect defects in applications
           : and to create a bug report with all informations needed by
           : maintainer to fix it. It uses plugin system to extend its
           : functionality.
```

Paket Kurmak

Bir veya daha fazla paketi kurmak için install kullanılır.

~]# yum install sqlite2.i586
~]# yum install audacious-plugins-*
Paket adını bilmiyorsanız yüklemek istediğiniz programın tam yolunu vererek de kurulum yapabilirsiniz.

~]# yum install /usr/sbin/named

Paket Grupları

Aynı amaca yönelik paketler aynı Grup ID'si altında gruplanmışlardır. Böylece tek komutla birbiriyle ilgili bütün paketler kurulabilir.

Bütün grupların listesini çıkartmak için:
~]# yum grouplist –v

Bir paket grubunu yüklemek için:
~]# yum groupinstall kde-desktop

Aynı işi yapan install komutu:
~]# yum install @kde-desktop

Geçmiş Hareketler

history parametresi yum ile yapılan hareketleri görmenizi sağlar.

En son hareketleri listelemek için:
 # yum history list

Bütün hareketleri görmek için:
 # yum history list all

Verilen aralıktaki hareketleri görmek için:
 # yum history list start_id..end_id

Yeni bir geçmiş başlatmak için:
 # yum history new

```
~]# yum history list 1..5
Loaded plugins: product-id, refresh-packagekit, subscription-manager
ID     | Login user               | Date and time    | Action(s)      | Altered
-------------------------------------------------------------------------------
     5 | Jaromir ... <jhradilek>  | 2011-07-29 15:33 | Install        |    1
     4 | Jaromir ... <jhradilek>  | 2011-07-21 15:10 | Install        |    1
     3 | Jaromir ... <jhradilek>  | 2011-07-16 15:27 | I, U           |   73
     2 | System <unset>           | 2011-07-16 15:19 | Update         |    1
     1 | System <unset>           | 2011-07-16 14:38 | Install        | 1106
history list
```

ID: Tek bir hareketi tanımlayan tamsayı değer
Login user: Hareketi başlatan kullanıcı
Actions: Hangi hareket yapıldı
Altered: Hareketin kaç paketi etkilediği

Kaynak Kodundan Paket Derleme

Linux altında programları derlemek için gcc kullanılır. Programlar insanların anlayacağı dille yazıldığından makinanın (programları yorumlayan işlemcidir) anlayacağı dile çevrilmesi lazım. Bu işi derleyiciler yapar ve bu işleme program derleme denir.

Kurulacak bir programın öncelikle indirilmesi gerekir. Programlar genelde geliştiricilerin web sayfalarından veya SourceForge ve Freshmeat gibi programlara ev sahipliği yapan genel amaçlı yerlerden elde edilebilir.

http://www.sourceforge.net

http://www.freshmeat.net

Programlar, dağıtımının daha kolay yapılabilmesi için kodları birleştirilmiş ve sıkıştırılmış olarak gelirler. Tar komutu birleştirmek, gzip komutu ise sıkıştırmak için kullanılır. Programcının bu komutlarla birleştirip dağıttığı paketin ters işlemlerle açılması gerekir. .tar.gz uzantılı bi kaynak kod aşağıdaki gibi açılır:

```
# tar -zxf qsheff-II-2.1-r3.tar.gz
```

Programı açmadan önce içinde ne var görmek isteyebilirsiniz:

```
# tar -ztf qsheff-II-2.1-r3.tar.gz | less
qsheff-II-2.1-r3/README
qsheff-II-2.1-r3/src/exec.h
qsheff-II-2.1-r3/src/qsheff-config.h.in
qsheff-II-2.1-r3/src/main.c
```

tar komutuna verilen 'z' parametresi gzip ile sıkıştırılmış dosyayı açmak içindir. 'x' ise tar ile birleştirilmiş dosyayı parçalamak içindir. Bazı programlar gzip yerine daha çok sıkıştırma sağlayan bzip2 ile sıkıştırılır. Böyle bir dosyayı açmak için 'z' yerine 'j' kullanılır:

```
# tar -jxf qsheff-II-2.1-r3.tar.bz2
```

GNU autotools ile oluşturulmuş yazılımlar aşağıdaki sırayla kurulur:

```
# ./configure
# make
# make install
```

Bunlara 'make config' veya 'make test' gibi ilave kontroller de eklendiği oluyor. Ancak genel kurulm bu şekildedir. Configure betiği genel ve her programa özel olmak üzere parametreler alabilir. Örneğin --prefix parametresi genel bir parametre olup programın nereye kurulacağını gösterir. Parametreler hakkında bilgi almak için kaynak kodun olduğu dizinde aşağdaki komutu verin:

```
# ./configure –help
Installation directories:
  --prefix=PREFIX         install architecture-independent files in
PREFIX
                          [/usr/local]
  --exec-prefix=EPREFIX   install architecture-dependent files in
EPREFIX
                          [PREFIX]
Optional Features:
  --enable-debug                  Enable debug messages
  --disable-local-users           Disable the filters for local users
  --enable-syslog                 Enable syslog messages
  --enable-backup                 Enable backup
```

make programı derler, make install ise dosya sistemine yerleştirir.

Kütüphaneler

Linux programlar ya dinamik yada statik derlenirler. Statik olarak derlenen programlar, her ne fonksiyona ihtiyaç duyarsalar kendi içerisinde onu barındırırlar. Yani tam ve bağımsız olarak çalışabilecek şekildedirler. Dinamik olarak derlenen programlar, çalışması için harici kütüphanelere ihtiyaç duyarlar. Kütüphaneler, programcıların sık kullanılan fonksiyonları topladıkları kodlardır.

ldd komutu bir programın statik mi yoksa dinamik mi olduğunu; dinamikse hangi kütüphanelere ihtiyaç duyduğunu belirler.

```
# ldd /bin/ln
      linux-vdso.so.1 (0x00007fffae5ff000)
      libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
(0x00007f52a6ccb000)
      /lib64/ld-linux-x86-64.so.2 (0x00007f52a7096000)
```

LD Yükleyici

Dinamik derlenmiş programlar bazı fonksiyonları harici kütüphanelerden kullanır; kendi içinde bulundurmaz. Dinamik bir program çalıştırıldığında ihtyaç duyduğu kütüphaneleri yükleyen programa “dinamik yükleyici” denir. Bu program ld-linux programıdır:

```
# ls -l /lib/ld-linux.so.2
lrwxrwxrwx 1 root root 25 Ağu 29 15:16 /lib/ld-linux.so.2 -> i386-
linux-gnu/ld-2.17.so
```

Dinamik yükleyici kütüphaneleri bulmak için iki dosyayı kullanır: /etc/ld.so.conf ve /etc/ld.so.cache dosyaları.

```
# cat /etc/ld.so.conf
include /etc/ld.so.conf.d/*.conf
```

Yukarıdaki dosyada sistem arayacağı paylaşımlı kütüphane dizinlerinin listesinin /etc/ld.so.conf.d dizininde .conf ile biten

dosyalarda tutulduğu ifade edilmektedir. Dizinin içeriği şu şekilde dosyalardan oluşmaktadır.

```
# ls /etc/ld.so.conf.d
i486-linux-gnu.conf  libc.conf  x86_64-linux-gnu.conf  zz_i386-biarch-compat.conf
```

```
# cat  /etc/ld.so.conf.d/x86_64-linux-gnu.conf
# Multiarch support
/lib/x86_64-linux-gnu
/usr/lib/x86_64-linux-gnu
```

Bu dosyanın içerisine kütüphaneler için bakılması gerekli dizinler yazılır.

Linux ld.so.conf dosyası ve dizinlerinedeğil, onun derlenmiş hali olan ld.so.cache'e bakar. ld.so.conf dosyalarında bir değişiklik yapıldığında her zaman aşağıdaki komutla cache dosyası oluşturulmalıdır:

```
# ldconfig
```

Dinamik yükleyicinin gördüğü tüm kütüphaneler aşaıdaki komutla görüntülenebilir:

```
# ldconfig -p
1292 kitaplık, `/etc/ld.so.cache' arabelleğinde bulundu
	libzvbi.so.0 (libc6,x86-64) => /usr/lib/x86_64-linux-gnu/libzvbi.so.0
	libzvbi-chains.so.0 (libc6,x86-64) => /usr/lib/x86_64-linux-gnu/libzvbi-chains.so.0
	libzbar.so.0 (libc6,x86-64) => /usr/lib/libzbar.so.0
	libz.so.1 (libc6,x86-64) => /lib/x86_64-linux-gnu/libz.so.1
.....
```

Eğer dinamik yükleyicinin öncelikle istenen bir dizindeki kütüphaneleri kontrol etmesi istenirse LD_LIBRARY_PATH çevre değişkeni kullanılabilir.

```
# export LD_LIBRARY_PATH="/usr/lib/dizin1:/usr/local/lib/dizin2"
```

GNU ve UNIX Komutları

Konu:

Kabuk, komut satırı ve temel Linux komutları

Hedefler:

- Kabuk hakkında bilgi sahibi olmak
- Çevre değişkenlerini görüntülemek, atamak, kullanmak
- Komut satırında çalışabilmek
- Komut çalıştırmak, komutları birbirine bağlamak
- Temel Linux komutlarını tanımak

Anahtar Kelimeler:

Bash, export, env, echo, cd, pwd, ls, sed, awk, xargs, tr, find, cat, wc

Kabuk

Kullanıcıların işletim sistemi ile etkileşebileceği iki ana ortam mevcuttur. Bunlardan birisi grafik masaüstü, diğeri de komut satırıdır. Komut satırını sunan ve komutları yorumlayıp çalıştıran kabuk programıdır.

En yaygın kullanılan kabuk programı Bash olmakla birlikte ksh, csh, zsh gibi farklı kabuklar da bulunmaktadır.

Sistemde kullanılabilecek tüm kabuklar /etc/shells dosyasında tanımlanmaktadır.

Sisteme giriş yaparken her kullanıcı için tanımlı bir kabuk çalışır. Kabuk programı kullanıcıya bir komut satırı (prompt da denir) sunar ve komut girmesi için bekler. Bash bu komutu yorumlayıp gerekli programları çalıştırır ve sonlanmalarını bekler. İşlemler bittikten sonra kontrol yeniden kabuğa geçer ve kabuk yeniden komut girişi için bekler. Kabuktan çıkış yapılana kadar bu döngü devam eder.

Kullanılan kabuğu görüntülemek için aşağıdaki komut kullanılır.

```
$ echo $SHELL
/bin/bash
```

Kabuğu değiştirmek için chsh komutu kullanılabilr:

```
$ chsh -s /bin/sh
Changing shell for user1.
Password:
Shell changed.
```

Yeni kabuğun aktif hale geçmesi için sistemden çıkıp yeniden giriş yapmak gerekir. Eğer kullanılan kabuk yalnızca mevcut oturum için değiştirilmek istenirse doğrudan kabuk çalıştırılır:

```
$ bash
user1@pardus $
```

Bu durumda sisteme bir sonraki girişte yine eski kabuk çalışacaktır.

Yardım Alma

Linux, konsol altında text tabanlı yardım alınabilecek ‘man (manual)’ olarak bilinen bir sistem sunmaktadır. Programlar, biçimlendirilmiş manual yardım dosyaları ile birlikte dağıtılmaktadır. ‘man’ komutu bu sayfaları biçimli bir şekilde okumaya ve kullanıcıya göstermeye yarar. Örneğin aşağıdaki man komutuyla man’ın kendisi hakkında yardım alınabilir:

```
$ man man
```

Man sayfası okunurken ok tuşları kullanılarak aşağı ve yukarı ilerlenebilir. ‘/’ işaretine basıldığında yardım sayfası içinde arama yapılabilir. Bir sonraki arama sonucu için ‘n’, bir ekran aşağı kaydırmak için SPACE, bir ekran yukarı çıkmak için ‘b’ ve yardım sayfasından çıkmak için ‘q’ tuşuna basılır.

Aynı konu adı iki farklı anlam içerebildiğinden (komut olan read ile system çağrısı olan read gibi) yardım sayfaları aşağıdaki gibi bölümlendirilmiştir:

Bölüm No	Açıklama
1	Komutlar (sistem yöneticileri için)
2	Çekirdek sistem çağrıları (çekirdek programcıları için)
3	Kütüphane fonksiyonları (kullanıcı ortamı programcıları için)
4	Aygıt dosyaları
5	Dosya biçimleri

Komut Geçmişi

Kullanıcının verdiği komutlar ev dizini altındaki .bash_history dosyasında tutulur. ‘history’ komutu bu dosyayı görüntüler. Yakın bir zamanda kullandığınız uzunca bir komutu bu şekilde tekrar geri çağırabilirsiniz.

Daha once verilmiş bir komutu geri getirmenin en kolay tuşu yukarı ve aşağı ok tuşuyla komutlar üzerinde dolaşmaktır. Diğer bir yöntem ise Ctrl-R ile geriye doğru komutlar üzerinde arama yapmaktır. Komutla alakalı birkaç harf yazınca ilgili bütün geçmiş komutları bulacaktır. Birden fazla alternative sonuç çıktığında tekrar tekrar Ctrl-R yapılarak istenen komut bulunana kadar dolaşılabilir.

‘history’ komutu ‘grep’ ile birlikte kullanılarak da aranılan komut bulunabilir.

```
$ history | grep mysqladmin
```

Komut Satırı

Kabuk programın kullanıcıya sunduğu iki komut satırı vardır: Esas komut satır, yardımcı komut satır. Bunlar PS1 ve PS2 değişkenlerinde tutulur.

```
[user1@pardus ~]$ echo $PS1
[\u@\h \W]\$
[user1@pardus ~]$ echo $PS2
>
[user1@pardus ~]$
```

\u kullanıcı (user), \h host, \W ise çalışma dizini anlamındadır. Böyle bir prompt user1 kullanıcısında $ ile biterken root kullanıcısında # ile biter.

$ ve # komut girişi için beklenildiğini gösteren simgelerdir. Yetkili kullanıcı sisteme giriş yapmış ise #, normal kullanıcı giriş yapmış ise $ görecektir.

Yardımcı prompt tamamlanmamış komut satırları için çıkar. Örneğin:

```
$ ls | \
> grep conf*
httpd.conf
```

\ simgesi komut satırının bitmediğini ve bir alt satırdan devam ettiğini gösterir. Genelde bir satıra sığmayacak kadar uzun olan komutlar için veya komutların güzel görünmesi için kullanılır. Bu durumda bir sonraki satırda yardımcı prompt (>) çıkarak komutun devam ettiğini ve hala girdi yapılması gerektiğini bildirir.

```
$ echo Pardus'un sayfasi
>
```

Yukarıdaki komutta yardımcı prompt çıkmasının nedeni tek tırnak (‘) işaretinin korumasız kullanımıdır. Kabuk tırnak açtığınıza kanaat getirdiğinden tırnağı kapatana kadar sizden komut girdisi isteyecektir. Komutun devam etmesi gerektiğini göstermek için yardımcı prompt sunmuştur. Tırnak burada ayraç olarak kullanılmıştır. Ayraç olarak kullanılan tırnağın \ katakteri ile korunması gerekir:

```
$ echo Pardus\'un sayfasi
Pardus'un sayfasi
```

Eğer tırnak açmış iseniz bu durumda korumaya gerek yoktur. Çünkü \ karakteri olmadığı sürece kabuk tırnakları bu şekilde yorumlar.

```
$ echo Kullanicilar: '
> user1 user8 user15'
Kullanicilar:
user1 user8 user15
```

Komut satırından girilen komut eğer özel anlamı olan fonksiyonlardan biri değilse diskte bulunarak çalıştırılır. Örneğin:

```
$ ls
dosya.txt  sirala.sh
```

Eğer komutun özel anlamı var ise Bash tarafından yorumlanarak gereği bash tarafından yerine getirilir. Aşağıdaki ifadeler Bash için özel anlamlıdır. Bunlar özel anlamlı olup kabuğu yetenekli kılan özel fonksiyonlar sunar. Örneğin for ve while ile döngü kurulabilir. if-then-else-fi yapısı ile şarta bağlı işlem yapılabilir.

```
! case do done elif else esac fi for function if in select
then until while { } time [[ ]]
```

Çevre Değişkenleri

Çevre değişkenleri, programlama dillerindeki değişkenler (variable) gibi belirli bilgileri taşıyan özel stringlerdir. Uygulamalar, bu çevre değişkenlerinin değerlerine göre davranışlarını değiştirebilir.

Çevre değişkenlerine, değişken adının başına $ eklenerek ulaşılır.

env komutu ile o anda tanımlı tüm çevre değişkenleri listelenir.

```
root@pardus:~# env
TERM=xterm
SHELL=/bin/bash
XDG_SESSION_COOKIE=9076065b0c479abada74f8f6524155ec-
1386782093.179827-446063787
SSH_CLIENT=10.41.255.17 55148 22
SSH_TTY=/dev/pts/0
USER=root
MAIL=/var/mail/root
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/
bin
PWD=/root
LANG=tr_TR.UTF8
SHLVL=1
HOME=/root
LOGNAME=root
SSH_CONNECTION=10.41.255.17 55148 172.16.45.219 22
DISPLAY=localhost:10.0
_=/usr/bin/env
```

Bir değişken sadece içinde bulunulan kabuk içinde kullanılacaksa şu şekilde tanımlanır:

```
$ VAR1=15
$ echo $VAR1
15
```

Eğer değişken global olacaksa açılan alt süreçlerde de bu değerin geçerli olması isteniyorsa export komutu kullanılır.

```
$ export VAR1=15
```

Bir betikten başka bir betik çağrıldığında yeni bir kabuk ortamına girilir. Başka kullanıcıların içinde bulunduğu kabukların her biri de ayrı bir ortam sayılır.

Tanımlı olan bir çevre değişkenini silmek için ise unset komutu kullanılır. Bu komut parametre olarak değişken adını –başında $ işareti olmadan- alır.

```
$ echo $A
5
$  unset A
$  echo $A
```

Sık kullanılan bazı çevre değişkenleri aşağıdaki gibidir:

PATH: Verilen komutların hangi dizinlerde aranacağını belirler. Bu değişkende kolon ayracı ‘:’ ile ayrılmış birden fazla dizin yolu tanımlanabilir.

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
```

HOME: Kulanıcının ev dizinini gösterir.

```
$ echo $HOME
/home/user1
```

SHELL: Kullanıcının kabuğunun tam yolunu gösterir.

```
$ echo $SHELL
/bin/bash
```

LANG: Sisteminde tanımlı dil ayarlarını gösterir.

```
$ echo $LANG
tr_TR.UTF8
```

EDITOR: Aksi belirtilmedikçe dosya işlemlerinde kullanılacak öntanımlı dosya düzenleyiciyi belirler. Örneğin crontab –e ile zamanlanmış görevler açılırken bu değişkene bakılır.

PS1: Kullanıcının birincil komut satırını şekillendirir.

PS2: Kullanıcının ikincil komut satırını şekillendirir.

Program Çalıştırma

Linux altında bir programı, komutu veya betiği çalıştırabilmek için programa ait dosyanın çalıştırma yetkisinin olması gerekmektedir. Yetkiler 'ls -l' komutuyla görülebilir.

```
$ ls -l /bin/cat
-rwxr-xr-x 1 root root 51856 Oca 26  2013 /bin/cat
```

Bir komut eğer çalışma yolunda(PATH) ise herhangi bir dizinden çalıştırılabilir. PATH kabukta komut tam yol verilmeden yazıldığında komutun sıra ile hangi dizinlerde aranacağını belirleyen özel bir çevre değişkenidir. Sistem komutları çalışma PATH'ine yerleştirildiği için tam yolunu vermeden çalıştırılabilir. Ancak kendi programlarımız veya betikler genelde standart sistem PATH'inde olmadığı için tam yolu verilerek çalıştırılmalı.

```
$ ls
```

ls komutu sistemin çalışma yolunda olduğundan her yerden çalıştırılabilmektedir.

```
$ /home/ahmet/scripts/script2.sh
$ ./script2.sh
```

Yukarıdaki kullanımlardan ilkinde çalıştırılacak betiğin tam yolu verilmiştir. Tam yolu verilmezse sistem tarafından betiğin bulunması mümkün değildir. İkinci kullanımda ise çalıştırılmak istenen betik bulunulan dizindedir. '.' sembolü bulunulan dizini simgelemektedir.

```
$ ../script2.sh
```

Yukarıdaki kullanımda ise '..' ile bir üst dizin simgelenmektedir. Siz bir alt dizinde yer alıyor ve bir üst dizindeki betiği çalıştırmak istiyorsanız bu şekilde kullanılabilir.

Sistem yalnızca standart yerlerdeki programlardan haberdardır. Sistemin standart PATH'ini görmek için:

```
$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Bu komutun cevap olarak döndüğü dizinlerdeki bütün çalıştırılabilir programlar sistemin herhangi bir yerinden başlatılabilir.

Eğer bir dosya veya dizine erişmek için dosya/dizin adı / kısmından başlayarak yazılırsa buna tam yol denir. Bunun dışındaki tüm dosya yollarına(. .. ./ vb) göreceli yol adı verilir.

Çıkışı Yönlendirme

Bashgelişmiş bir kabuk programı olup pek çok işleri kolaylaştırıcı özellikler sunmaktadır. Bash kabuk altında bir komutun çıktıları bir dosyaya yönlendirilebilir. Bu sayede komut çıktıları kalıcı olarak diske yazılıp daha sonra üzerlerinde çalışılabilir.

Linux'de 3 adet öntanımlı dosya tanımlayıcısı (file descriptor) vardır:

- Standart Girdi (stdin) : Çalışan programın dosya vb kaynakları açmadan veri okumak için kullanacağı kaynağı belirtir. Ön tanımlı olarak stdin klavyeden veri alır. Stdin dosya numarası 0'dır.
- Standart Çıktı (stdout) : Çalışan programın çıktılarını göndereceği hedefi belirtir. Ön tanımlı olarak çıktı terminale basılır. Stdout dosya tanımlama numarası 1'dir.
- Standart Hata (stderr) : Çalışan programın hata çıktılarını göndereceği hedefi belirtir. Ön tanımlı olarak hata terminale basılır. Hata dosya tanımlama numarası 2'dir.

Stdin yönlendirmesi < karakteri ile yapılır.

```
$ sort < /etc/passwd
```

Stdout yönlendirilmesi > veya >> karakteri ile yapılır.

```
$ echo 2048 > /proc/sys/fs/file-max
```

Stderr yönlendirmesi için stdout'dan farklı olarak 2 nolu standart hata numarası belirtilerek 2> şeklinde yapılır.

```
$ touch dosya.txt 2> /tmp/err.log
```

Hem STDOUT hem de STDERR'yi aynı dosyaya yönlendirmek için:

```
$ cat dosya.txt > /tmp/err.log 2>&1
```

Örnekler:

```
$ ps -U ahmet > procs.txt
$ cat procs.txt
  PID TTY          TIME CMD
14481 ?        00:00:00 sshd
14482 pts/1    00:00:00 bash
14565 pts/1    00:00:00 ps
```

Ps komutu çalışan süreçleri gösterir. Bu komutun çıktısı görüldüğü gibi ekrana değil procs.txt dosyasına yazılmıştır.

```
$ echo "Dosya sonu" >> procs.txt
$ cat procs.txt
  PID TTY          TIME CMD
14481 ?        00:00:00 sshd
14482 pts/1    00:00:00 bash
14565 pts/1    00:00:00 ps
Dosya sonu
```

Yukarıda > yerine >> karakterleri kullanılmıştır. Bu karakter mevcut dosyanın sonuna ekleme yapar. Dosyanın eski içeriği silinmez, onun hemen ardından eklenir. Tek > karakteri ise daha önceki içeriği temizler ve dosyanın içerisinde yalnızca son çalışan komutun çıktısı yer alır.

“> /dev/null 2>&1” : Stdoutu /dev/null’a yönlendirir. Stderr’yi de stdout’a yönlendirir. Stdout /dev/null’a yönlendirildiği için stderr ve stdout /dev/null’a yönlendirilir.

“2>&1 > /dev/null”: Stderr, stdout’a yönlendirilir. Sonra sadece stdout /dev/null’a yönlendirilir.

“1>&2 > /dev/null”: stdout, stderr’ye yönlendirilir. Sonra stdout /dev/null’a yönlendirilir. Stderr’a dokunulmaz.

Ardışık Komutlar - Pipe

Bash altında bir komutun çıktısı bir dosyaya yönlendirilebildiği gibi bir başka komuta gönderilebilir. Pipe denen bu işlem şu biçimdedir:

```
# komut1 | komut2 | komut3 ...
```

Burada komut2 ve komut3 ‘ün girdileri standart girişten alan programlar olması gerekir. Örneğin “grep” komutu girişten verilen metin içerisinde istenen kelimeleri arar ve bulursa ekrana yazar.

Yalnızca ls komutu verildiğinde çıktı şu şekilde:

```
$ ls
a.txt  beni.oku  procs.txt  program  sirala.sh  surecler.txt
yedek.tar.gz
```

| ile beraber ls ve grep birleştirildiğinde:

```
$ ls | grep s
procs.txt
sirala.sh
surecler.txt
```

ls komutunun çıktısı ekrana basılmadı fakat grep komutuna gönderildi. ‘grep’ komutu ise kendine gelen içerikte “s” harfini aradı ve bulduğunu ekrana bastı.

Bu komutlar sırayla çalıştırılır. Yani önce ls çalıştırılır, sonra çıktısı grep’e verilerek grep çalıştırılır.

Bağlı Komutlar

Birbirine bağlı iki komut && karakterleri kullanılarak çalıştırılır. Örneğin “make” komutu başarılı olursa “make install” komutu çalıştırılacaksa:

```
# make && make install
```

Şeklinde iki komut bağlanır. Eğer make işlemi hata verirse “make install” yapılmaz.

Temel Komutlar

Linux altında kullanılabilecek yüzlerce komut ve yardımcı araç vardır. Aşağıda bunlardan Linux’u yönetmek için gerekli olanlar anlatılmıştır.

ls

Dizin içeriğini (dosya ve alt dizinleri) listeler.

ls komutunun pek çok parametresi vardır. Yardım sayfaları (man ls) kullanılarak detaylı bilgi alınabilir.

-a

. ile başlayan dosyaları da listeler.

-A

. (içinde olunan dizin) ve .. (bir üst dizin) hariç tüm dosyaları listeler.

-F

Dosyaları birbirinden ayırt etmek için dizinlerin sonuna (‘/’), çalıştırılabilir dosyaların sonuna (‘*’), sembolik linklerin sonuna (‘@’), soket dosyaların sonuna (‘=’) ve FIFO dosyaların sonuna (‘|’) ekler.

```
# ls -F
dosya.txt*  sirala.sh*
```

-R

Alt dizinleri de özyinelemeli olarak gösterir.

-S

İsim sıralamasından önce boyut sıralaması (en büyük dosya önce) yapar.

-i

'l' ile birlikte her bir dosya için seri numarası olarak kullanılabilecek inode numarasını gösterir.

-l

Detaylı biçimde dosyaları listeler.

-p

Dizin olan dosyaların sonuna “/” karakteri ekler.

```
# ls -p
dizin1/  dosya.txt  sirala.sh
```

-n

-l ile kullanıldığında kullanıcı adı yerine kullanıcı ID'si görüntülenir.

-r

Ters sırada listeler.

```
# ls -l
toplam 8
drwxr-xr-x 2 root root 4096 Ara 13 20:56 dizin1
-rwxr-xr-x 1 root root   18 Ara 11 19:19 dosya.txt
-rwxr-xr-- 1 root root    0 Ara 11 20:10 sirala.sh
```

Detaylı gösterimde sol başta dosya türünü belirten bir karakter yer alır. Bu karakterlerin anlamları şöyledir:

```
b     Blok dosya.
c     Karakter dosya.
d     Dizin.
l     Sembolik bağlantı.
s     Soket haberleşme dosyası.
p     FIFO, pipe.
-     Normal dosya.
```

pwd

İçinde bulunulan dizinin yolunu verir. Bazı kabuklarda bu komut dahili komut olarak da bulunmaktadır.

-L

Çalışılan dizini Mantıksal(logical) olarak gösterir.

-P

Bütün sembolik bağlantıları çözerek fiziksel çalışma dizinini göster.

```
# pwd
/root/test


# ln -s / /home/user1/temp /tmp
# cd /tmp/temp
# ls
# pwd -L
/tmp/temp

# pwd -P
/home/user1/temp
```

cd

Komut satırında dizinler arasındaki geçişler cd komutuyla yapılır.

```
$ cd /
$ pwd
/
```

cd komutu dizin değiştirmek için kullanılır. Bu komut sayesinde komut satırında iken dosya sisteminde dolaşabiliriz. Parametre olarak hedef yolu alır.

Aşağıdaki yollar mutlak yollardır:

```
/etc
/etc/X11
/sbin
/usr/local/sbin
```

Mutlak yollar / simgesi ile başlar. Bu şekilde başlamayan yollar ise göreceli (relative) yollardır.

```
$ cd /usr
$ cd local/bin
$ pwd
/usr/local/bin
```

İkinci cd komutu göreceli bir yol için verilmiştir. Göreceli yollar kökten itibaren değil, bulunulan dizinden itibaren işleme konur.
Yollar, bir üst dizin anlamında .. içerebilir.

```
$ pwd
/usr/local/bin
$ cd ..
$ pwd
/usr/local
$ cd ../local/bin
$ pwd
/usr/local/bin
```

cd komutu parametre verilmeden çalıştırıldığında kullanıcının ev dizinine gider. ~ ifadesinden sonra kullanıcı adı verildiğinde kullanıcının ev dizinine gider.

```
# cd ~ahmet
# pwd
/home/ahmet
```

cp

Dosya ve dizin kopyalar. Bir dosyayı başka bir dosya olarak kopyalayacağı gibi birkaç dosyayı bir dizine de kopyalayabilir.

-i

Eğer hedef dosya mevcut ise kullanıcıyı uyarır. Eğer kullanıcı ‘y’ tuşuna basarsa kopyalama gerçekleştirilir. ‘n’ tuşuna basarsa kopyalama iptal edilir.

-f

Eğer hedef dosya mevcut ise kullanıcıyı uyarmadan üzerine yazar.

-p

Dosya özelliklerini korur. Korunan dosya özellikleri: değiştirilme zamanı, erişim zamanı, dosya bayrakları, dosya erişim modu, kullanıcı ID'si ve grup ID'si.

-r

Dizini alt dizinlerle birlikte kopyalar.

-v

Ekrana ne yaptığına dair bilgi basar.

```
$ cp -pv *.txt arsiv/
`11-01-2013.txt' -> `arsiv/11-01-2013.txt'
`11-02-2013.txt' -> `arsiv/11-02-2013.txt'
`ozet.txt' -> `arsiv/ozet.txt'

$ cp –pr dizin1 dizin2
```

find

Dosya ve dizin aramak için kullanılır.

Kullanımı:

```
# find dizin seçenekler
```

Dizin adı belirtilmezse bulunulan dizin ifade edilir.

Seçenekler:

-name isim: Aranılacak dosyanın ismi.

-iname isim: Aranılacak dosya ismi büyük/küçük harf duyarsız

-type tip: Aranan dosyanın tipini belirler. Tip aşağıdaki değerlerden biri olabilir.

f: Normal Dosya

d: Dizin

b: Blok dosyası

c: Karakter dosyası

l: Sembolik bağlantı

-cnewer dosya: Verilen tarihten daha sonra değiştirilen dosyalar.

-amin n: n dakika önce erişilen dosyalar.

-cmin n: Dosya durumu n dakika önce değişenler.

-atime n: n*24 saat önce erişimiş dosyalar.

-mtime n: n gün içinde işlem gören dosyalar.

-ctime n: n gün içinde değiştirilen dosyalar.

Rakam olarak verilen parametrelerde:

+n n ‘den büyük

-n n’den küçük

n tam olarak n kadar

-print: Bulunan dosyaların ekranda görüntülenmesini sağlar.

-exec: Çıkan arama sonuçları üzerinde komut çalıştırmayı sağlar. {} \; parametrenin bittiğini ifade eder.

Örnekler:

Tüm disk üzerinde httpd.conf dosyasını aramak için:

```
# find / -name httpd.conf
```

/usr/local altında içinde http geçen dosyaları bulmak için:

```
# find /usr/local -name “*http*”
```

/tmp dizinde son erişim tarihi 30 günden fazla olan dosyaları görüntülemek için.

```
# find /tmp –atime +30d
/tmp/install.XErPCgBn/cap_mkdb
/tmp/install.XErPCgBn/cat
/tmp/install.XErPCgBn/chflags
```

/tmp dizinde 60 günden fazla süredir değiştirilmeyen dosyaları silmek için:

```
# find /tmp -type f -ctime  +60  -exec rm {} \;
```

Arama işleminin derinliğini kısıtlamak için –maxdepth parametresi kullanılır. Verilen pozitif tamsayı kadar alt klasörlerde de arama yapar.

```
# find / -maxdepth 3 -name passwd
./usr/bin/passwd
./etc/pam.d/passwd
./etc/passwd
```

mkdir

Dizin oluşturmak için kullanılır.

-p

Oluşturulacak dizinin üst dizinleri yoksa onları da oluşturur.

```
$ mkdir arsiv
$ mkdir –p 2013/02/17/18/00
```

mv

Bir dosyayı veya dizini başka bir dosya veya dizin olarak taşır. Veya birkaç dosya veya dizini başka bir dizine taşır.

-i

Hedef dosya varsa kullanıcıya sorar. Eğer kullanıcı ‘y’ veya ‘Y’ tuşuna basarsa üzerine yazar. ‘n’ veya ‘N’ tuşuna basarsa yazmaktan vazgeçer.

--reply={yes,no,query}

Etkileşimli durumda sorulara otomatik olarak belirtilen cevap verilir.

-f

Hedef dosya var ise kullanıcıya sormadan üzerine yazar.

-u

Yalnızca kaynak dosya hedef dosyadan daha yeni ise veya hedef dosya erişelemez ise taşı.

```
$ mv kaynak hedef
```

rm

Dosya veya dizin siler.

-f

Silerken sormaz, varolmayan bir dosya için bilgi vermez.

-i

Silmeden önce sorar.

-R -r

Dizinlerin ve alt dizinlerin içeriklerini ardarda siler.

-v

Yapılan işlem hakkında daha ayrıntılı bilgi verir.

```
$ rm –f beni.oku
$ rm –fr dizin1
```

```
$ rm -v ciro.txt
removed `ciro.txt'
```

rmdir

Boş dizin silmek için kullanılır. Dizin içinde dosya varsa bu komut dizini silmez. Bunun yerine rm –fr komutu verilmelidir.

```
$ rmdir mydir
```

-p parametresi iç içe dizinleri sırasıyla siler.

```
$ rmdir –p a/b/c
```

Yukarıdaki komut şu üç işlemi yapar:

rmdir a/b/c, rmdir a/b, rmdir a

touch

Eğer belirtilen dosya mevcut ise dosya erişim ve değiştirilme zamanlarını günceller. Eğer belirtilen dosya mevcut değilse dosyayı oluşturur.

-a

Erişim zamanını günceller.

-c

Dosya mevcut değilse dosyayı oluşturma.

-m

Değiştirilme zamanını değiştir.

-t

Erişim ve değiştirilme zamanını belirtildiği şekilde değiştir:
[[CC]YY]MMDDhhmm[.ss]

CC Yılın ilk iki hanesi

YY Yılın son iki hanesi

MM Yılın ayı. 1’den 12’ye.

DD Ayın günleri. 1’den 31’e.

hh Günün saati. 0’dan 23’e.

mm Saatin dakikaları. 0’dan 59’a.

SS Dakikanın saniyeleri. 0'dan 59'a.

```
$ ls -l sirala.sh
-rw-r--r--  1 ahmet  ahmet  44 Ara 10 20:12 sirala.sh
$ touch sirala.sh
$ ls -l sirala.sh
-rwxr-xr-- 1 root root 44 Ara 11 20:10 sirala.sh
```

18 Ocak 2008 12:05:09 tarihli bir dosya oluşturmak için:

```
$ touch -t 200801181205.09 deneme
$ ls -al deneme
-rw-r--r-- 1 ahmet ahmet 44 Oca 18  2008 deneme
```

cat

Dosyaları birleştirmek ve standart çıktıya basmak için kullanılır.

```
$ cat 1.txt
Ben 1.dosyayim
$ cat 2.txt
Ben 2.dosyayim
```

```
$ cat 1.txt 2.txt
Ben 1.dosyayim
Ben 2.dosyayim
```

Dosyalar komut satırında belirtildiği sırada okunur. Eğer dosya olarak eksi (-) karakteri kullanılırsa standart girişten okur (yani klavyeden yazmanızı bekler). Eğer dosya olarak UNIX domain soket verilmiş ise sokete bağlanır ve EOF(Dosya sonu, End Of File) görene kadar soketten okur.

-b parametresi boş olmayan satırları –n parametresi ise tüm satırları numaralandırır.

```
$ cat -b beni.oku
      1 Ben bir dosyayim.
      2 Bende toplam 3 satir vardir.

      3 Iste bu da son satir.
$ cat -n beni.oku
      1 Ben bir dosyayim.
```

```
2 Bende toplam 3 satir vardir.
3
4 Iste bu da son satir.
```

wc

Kelime, satır, karakter ve byte sayar. Girdiyi standart girişten veya parametre olarak verilen dosyadan alır.

-c

Karakter sayısını verir.

-l

Satır sayısını verir.

-m

Karakter sayısını verir.

-w

Kelime sayısını sayar.

```
$ cat beni.oku
Ben bir dosyayim.
Bende toplam 3 satir vardir.

Iste bu da son satir.
$ cat beni.oku | wc –l
      4
$ wc –w beni.oku
      13
$ cat beni.oku | wc -c
      70
```

head

Verilen dosyanın ilk satırlarını görüntüler. –n ile ilk kaç satırın görüntülenmesi gerektiği belirtilir. Eğer satır sayısı verilmez ise ön tanımlı olarak ilk 10 satırı gösterir.

```
$ head –n 2 beni.oku
Ben bir dosyayim.
Bende toplam 3 satir vardir.
```

-c ile dosya başından kaç byte okuyacağı belirtilebilir.

```
$ head -c 3 beni.oku
Ben
```

Eğer birden fazla dosya verilirse veriliş sırasına göre dosyaların ilk satırları gösterilir ve her dosyanın satırlarının başına '==> XXX <==' biçiminde ayraç koyar.

```
$ head -n 1 1.txt beni.oku
==> 1.txt <==
Ben 1.dosyayim
==> beni.oku <==
Ben bir dosyayim.
```

tail

Dosyanın son kısımlarını gösterir. –c ile byte sayısı, -n ile satır sayısı verilebilir. Eğer dosya verilmez ise standart girişten okur.

```
$ cat beni.oku
Ben sevimli bir dosyayim.
Bende toplam 3 satir vardir.

Iste bu da son satir.
```

```
$ tail -c 7 beni.oku
satir.
$ tail -n 1 beni.oku
Iste bu da son satir.
```

Eğer –f parametresi verilirse tail komutu EOF (dosya sonu) karakterini okuyunca durmaz. Yeni veriler için bekler ve veri hazır olur olmaz okuyup ekrana basar. Genellikle sürekli akan bir log dosyasını görüntülemek için kullanılır:

```
# tail –f /var/log/messages
```

-r parametresi çıktıyı ters sırada basar.

```
$ tail -r beni.oku
Iste bu da son satir.

Bende toplam 3 satir vardir.
Ben sevimli bir dosyayim.
```

more

Dosyanın içeriğini sayfa sayfa görmek için kullanılır. Eğer dosya uzunluğu bir sayfadan fazla ise ekrana sadece bir sayfalık kısmını gösterir. Enter tuşu ile bir satır ilerlenir, boşluk(space) tuşu ile de bir sayfa ilerlenir.

```
# more /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
..
--Başka--(73%)
```

less

more komutunun daha gelişmişidir. More komutu ile geriye doğru gidelemezken less ile gidilebilmektedir.

```
$ less /etc/passwd
```

nl

Satır başlarına satır numarasını ekler.

```
$ cat a.txt
linux
101
pardus
```

```
business
academy
linux
101
```

```
$ nl a.txt
     1  linux
     2  101
     3  pardus
     4  business
     5  academy
     6  linux
     7  101
```

cut

Standart girişten okuduğu satırlar içindeki istenen sütunları gösterir. Sütun seçme işlemi bir ayraca göre yapılabileceği gibi sabit boyda da yapılabilir.

-c

Karakter olarak seçilecek sütunları belirler.

-d

Ayraç belirlemek için kullanılır.

-f

Ayraç göre sütunlara ayrıldığında görüntülenmek istenen sütunları seçer.

Aşağıda /etc/password dosyasının bir kısmı görülmektedir. Görüldüğü gibi bilgiler ‘:’ karakteri ile birbirinden ayrılmış. –d ile ayraç olarak ‘:’ belirtildiğinde soldan sağa kullanıcı ismi 1. sütun, parola 2.sütun, UID 3. sütun, GID 4.sütun, açıklama 5.sütun, ev dizini 6.sütun ve kabuk 7.sütun olarak yerleşir. Örneğin kullanıcı adı ve kabuk bilgilerini ekrana yazmak için:

```
$ cat /etc/passwd
....
sshd:x:117:65534::/var/run/sshd:/usr/sbin/nologin
postfix:x:118:123::/var/spool/postfix:/bin/false
ahmet:x:1001:1001::/data/ahmet:/bin/sh
pardus:x:1004:1006::/home/pardus:/bin/sh
```

```
$ cut -d : -f1,7 /etc/passwd
....
sshd:/usr/sbin/nologin
postfix:/bin/false
ahmet:/bin/sh
pardus:/bin/sh
```

tr

Tekrar eden karakterleri değiştirmek, silmek veya sıkıştırmak için kullanılır. STDIN'den okur, STDOUT'a yazar.

```
$ tr [seçenek] SET1 [SET2]
```

Hem SET1, hem de SET2 belirtilmişse SET1'deki her bir karakter karşılık gelen SET2'deki karakterle değiştirilir. Örnekler:

1. Küçük harfleri büyük harflerle değiştirmek:

```
$ tr abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ
pardus
PARDUS
```

Aynı iş şu şekilde de yapılabilir:

```
$ tr a-z A-Z
pardus
PARDUS
```

2. Süslü parantezleri normal parantezlere dönüştürme:

```
$ tr '{}' '()' < girişdosyası > çıkışdosyası
```

3. Boşlukları TAB karakterine dönüştürme:

```
$ tr [:space:] '\t'
Pardus Linux
Pardus	Linux
```

4. Tekrar eden karakterleri –s ile sıkıştırma:

Birden fazla boşluğu tek boşluğa indirme:

```
$ echo "GNU      Linux" | tr -s [:space:] ' '
GNU Linux
```

5. Belirtilen karakterleri –d ile slime:

```
$ echo "1 GNU 2 Pardus 3 Linux" | tr -d [:digit:]
 GNU  Pardus  Linux
```

expand, unexpand

Bu komut, giriş metnindeki sekmeleri(TAB) boşluklarla değiştirir.

```
$ expand dosya.txt > cikti.txt
```

unexpand tersine olarak boşluk karakterlerini sekmelere dönüştürür.

```
$ unexpand out
```

fmt

Paragraf metnini yeniden biçimlendirir.

Bir satırda ön tanımlı olarak en fazla 75 karakter olacak şekilde biçimlendirme yapar.

```
$ fmt myfile.txt
```

-w: Satır genişliği

```
$ fmt –w 80 myfile.txt
```

-s: Uzun satırları böler fakat kısaları uzunlara eklemez.

```
$ fmt –s myfile.txt
```

-u: Boşluklar tekdüze olsun. Kelimeler arası tek, cümleler arası iki boşluk.

```
$ fmt –u myfile.txt
```

pr

Baskı için çıktıyı biçimlendirir, sayfalandırır. Satır veya sütün sayısını düzenleyen, kenar boşluklarını ayarlayan, satırları numaralandıran, sayfa başlığı ekleyen seçeneklere sahiptir.

Çıktıyı çift boşlukla biçimlendirme:

```
$ pr –d myfile.txt
```

Kenar boşluğu 5, satır boyu 65 olan sayfalar halinde çıktı üretmek için:

```
$ pr -o 5 --width=65 myfile.txt | more
```

od

cat veya diğer komutlarla bir girdi görüntülendiğinde bazı zamanlar çıkışta anlamsız karakterler ve hizalama bozulmaları görür. Bunun sebebi okunabilir bir anlamı olmayan kontrol karakterleridir. Bunların ne olduğu cat benzeri komutlarla anlaşılmaz. Bu karakterlerin ne olduğunu görmek için od (Octal Dump) komutu kullanılır.

Od komutu öntanımlı olarak octal gösterime göre çıktı üretir. –A parametresi kullanılarak çıktı sayı sistemi o (octal, 8 tabanında), d (decimal, 10'luk tabanda), x (hexadecimal, 16'lık tabanda) belirtilebilinir. Od normal çalışmasında satır başlarına girdinin kaçıncı byte'ında olduğunu gösteren offset koyar. –n ile offset gösterimi iptal edilir. –t ile çıktı biçimlendirilebilir.

```
$ od text2
0000000 004471 066160 066565 031412 061011 067141 067141 005141
0000020 030061 060411 070160 062554 000012
0000031
```

Aşağıdaki çıktıda offset bulunmaktadır ve kontrol karakterleri ters bölü (\) gösterimi ile yazılmıştır.

```
$ od -A d -t c text2
0000000 9  \t   p   l   u   m  \n   3  \t   b   a   n   a   n   a  \n
0000016 1   0  \t   a   p   p   l   e  \n
0000025
```

Aşağıdaki örnekte offset olmadığı gibi –t ile bazı kontrol karakterlerin yerine özel isimleri çıktıya konulmuştur. (nl: new line, ht: horizontal tab gibi.)

```
$ od -A n -t a text2
   9  ht   p   l   u   m  nl   3  ht   b   a   n   a   n   a  nl
   1   0  ht   a   p   p   l   e  nl
```

Çıktı genişliğini (offset) sınırlandırmak için –w kullanılır.

```
$ od -w1 -c -Ad input
0000000   1
0000001  \n
0000002   2
0000003  \n
0000004   3
0000005  \n
0000006   4
0000007  \n
0000008   5
0000009  \n
```

paste

İki dosyayı satır satır veya sütun sütun birleştirmek için kullanılır.

```
$ cat 1.txt
GNU
Linux
Debian
Mint
Ubuntu
```

Hiçbir parametre verilmezse paste komutu cat komutu gibi davranır.

```
$ paste 1.txt
GNU
Linux
Debian
Mint
Ubuntu
```

Bütün satırları bir dosyada birleştirmek için:

```
$ paste -s 1.txt
GNU Linux   Debian   MintUbuntu
```

Birleştirmeyi bir ayraçla yapmak için:

```
$ paste -d: -s 1.txt
GNU:Linux:Debian:Mint:Ubuntu
```

Üç sütunlu bir çıktıda birleştirmek için:

```
$ paste - - - < 1.txt
GNU Linux   Debian
MintUbuntu
$ paste - - -d: < 1.txt
GNU:Linux
Debian:Mint
Ubuntu:
```

İki dosyayı karşılıklı (sütun sütuna) birleştirmek için:

```
$ cat 2.txt
1 A
2 B
3 C
4 D
$ paste 1.txt 2.txt
GNU 1 A
Linux   2 B
Debian  3 C
```

```
Mint4 D
Ubuntu
```

İki dosyayı satır satır birleştirmek için \n ayracı kullanılır:

```
$ paste -d'\n' 1.txt 2.txt
GNU
1 A
Linux
2 B
Debian
3 C
Mint
4 D
Ubuntu
```

join

Paste gibi iki dosyayı birleştirmekle birlikte biraz daha ileri bir komut olan join, iki dosyadaki sütunları eşleştirerek birleştirir. Hiç bir parametre verilmediğinde aşağıdaki gibi bir sonuç ortaya çıkar:

```
$ cat A.txt
1 A
2 B
3 C
$ cat B.txt
1 Ali
2 Yusuf
3 Selim
$ join A.txt B.txt
1 A Ali
2 B Yusuf
3 C Selim
```

Alan seçme

Dosyalardaki ortak alanlar kullanılarak birleşme yapılabilir.

Aşağıdaki örnekte 1.dosyanın 1.sütunu ile 2.dosyanın 1.sütunu örtüşecek şekilde dosyalar birleştirilir.

```
$ cat 1.txt
06 ANKARA 1
21 DIYARBAKIR 2
34 ISTANBUL 3
61 TRABZON 4
```

```
$ cat 2.txt
06 CANKAYA
21 KULP
34 USKUDAR
61 AKCAABAT

$ join -1 1 -2 1 1.txt 2.txt
06 ANKARA 1 CANKAYA
21 DIYARBAKIR 2 KULP
34 ISTANBUL 3 USKUDAR
61 TRABZON 4 AKCAABAT
```

Görüldüğü gibi plaka numaraları ortak alan olarak kullanılarak illerle ilçeler eşleştirilmiş oldu.

Ayraç Belirtme

Yukarıdaki örneklerde alanlar (diğer deyişle sütunlar) birbirinden boşluk ile ayrılmıştır. Boşluk ve TAB öntanımlı ayraç olduğundan join komutu sütunları bu ikisinden hangisi varsa ona göre ayırır ve numaralandırır. Eğer sütunlar başka bir ayraçla ayrılmışsa (virgül, noktalı virgül, iki nokta üst üste gibi) –t parametresi kullanılarak ayraç belirtilebilir.

```
$ join -t, -1 3 -2 1 A.txt B.txt
1,Ali,A,A
2,Yusuf,B,B
3,Selim,C,C
```

Harf Duyarlılığı

Birleştirme için eşleştirme yapılırken normalde küçük harf ile büyük harf eşleşmez. Örneğin bir dosyadaki A ile diğerindeki a karakteri eşleşmez. Büyük-küçük harf duyarlılığını iptal etmek için –i parametresi kullanılır.

Çıktı Biçimlendirme

join normalde dosyalardaki sıraya göre satırları ve sütunları sıralar. Sadece sıralama kriteri olan ortak sütunu başa alır. Alanları istenilen sırada çıktıya yazmak için –o parametresi kullanılır.

Aşağıdaki örnekte 1.dosyanın 1.sütunu ve 2.dosyanın 1.sütunu sırasıyla çıktı oluşur.

```
$ join -o 1.1 2.1 -1 2 -2 2 A.txt B.txt
Ali 1
Yusuf 2
```

split

split komutu dosyayı iki veya daha fazla parçaya bölmek için kullanılır.
-l parametresi ile satır sayısına göre –b parametresi ile byte miktarına göre parçalanır.

```
$ split split.zip
$ ls
split.zip  xab  xad  xaf  xah  xaj  xal  xan
$ wc -l *
   40947 split.zip
    1000 xaa
    1000 xab
    1000 xac
```

Görüldüğü gibi öntanımlı olarak 1000 satırlı dosyalara böler. –l parametresi ile bölünmüş dosyaların satır sayısı değiştirilebilir. Veya –b parametresi ile dosya boyutu verilebilir.

```
$ split -l 3 a.txt dosya
$ ls dosya*
dosyaaa  dosyaab  dosyaac
$ cat dosyaaa
linux
101
Pardus
$ cat dosyaab
business
academy
Linux
$ cat dosyaac
101
```

sed

Metin ve stringler üzerinde işlem yapan bir programdır. Genel olarak programların çıktılarını düzenlemek veya pek çok dosyayı otomatik olarak düzenlemek için kullanılır. Metin üzerinden tek geçişte istenen kriterlere uygun değişiklikleri yapabildiğinden oldukça etkin ve hızlı bir araçtır.

```
$ sed [seçenekler] dosya […]
```

Değiştirme

eski dosyasındaki gun metnini bulup gece ile değiştirir, sonucu yeni dosyasına yazar:

```
$ sed 's/gun/gece/' <eski >yeni
$ echo "iyi gunler" | sed 's/gun/gece/g'
iyi geceler
```

Sed komutunun 4 parçası var:
s değiştirme(substitute) komutu
/../../ Ayraçlar
gun Aranacak düzenli ifade
gece Uygulanacak, yerleştirilecek ifade

Öntanımlı olarak her satırdaki ilk bulduğunu değiştirir.

```
$ cat numbers.txt
bir iki uc, bir iki uc
dort uc iki bir
bir yuz
$ sed 's/bir/BIR/' < numbers.txt
BIR iki uc, BIR iki uc
dort uc iki BIR
BIR yuz
```

Global değiştirme için:

```
$ sed 's/bir/BIR/g' < numbers.txt
```

Bu durumda ilk gördüğünü değil bütün bulduklarını değiştirir.

Birden çok eşleşme olsa da bunlardan seçtiğimiz bir tanesi için işlem yapabiliriz. "\(" ve "\)" ile patternleri belirleyebilir, \1 ve \2 şeklinde eşleşen patterni seçip kullanabiliriz. Bu durumda sadece seçilenler ekrana basılacaktır, diğer çıktılar silinecektir.

```
$ sed 's/\([a-zA-Z]*\) \([a-zA-Z]*\) /\1 /' <eski > yeni
```

Veya ifadenin sonunda /2 şeklinde de kullanılabilir:

```
$ sed 's/[a-zA-Z]* //2' <eski >yeni
```

\2 ve /2 karıştırılmamalı. İlki ifade içinde, ikincisi ise ifade sonundaki kullanım şeklidir.

Ayraç (delimiter) ed, more ve vi'da olduğu gibi slash (/) 'dır. Bir dosya yolu verilirken yolun içerdiği slash'ler korunmalı:

```
$ sed 's/\/usr\/local\/bin/\/common\/bin/' <eski >yeni
```

Ayraç olarak `_ , : , |` gibi semboller kullanılabilir.

Bulunan bir örüntünün sağına soluna birşeyler eklemek için:

```
$ sed 's/abc/(abc)/' <eski >yeni
```

Burda değiştirmek istediğimiz şey tam belli. Değilse:

```
$ sed 's/[a-z]*/(&)/' <eski >yeni
```

İlk kısımda bulunacak pattern var, & ise bulunan her ne ise onu gösterir. Bu simgeyi birden fazla kullanabiliriz:

```
$ echo "123 abc" | sed 's/[0-9]*/& &/'
123 123 abc
```

123 eşleştiği için iki kere tekrar edildi, abc için herhangi birşey yapılmadı.

Örnekler:

foo veya bar desenlerini içeren satırları silmek için:

```
$ cat del.txt
satir 1
satir 2 foo 56
satir 3
satir 4 bar 34
$ sed -r '/foo|bar/ d' del.txt
satir 1
satir 3
```

Satır başındaki boşlukarı (SPACE ve Tab) silmek için:

```
$ cat space.txt
 satir 1
    satir 4
satir   3
$ sed 's/^[ \t]*//' space.txt
satir 1
satir 4
satir   3
```

Sed komutu –i parametresiyle değişiklikleri tandart çıktıya yazmak yerine dosya içerisinde gerçekleştirilir. Yukarıdaki değişiklikler ekran çıktısında olmakla birlikte orjinal dosyada olmamıştır.

awk

Awk, güçlü bir desen arama aracıdır. Sed gibi benzerlerinden ayıran özellik ise matematik işlemler, şartlı ifadeler, değişkenler ve dosya G/Ç işlemleri gibi script dillerinde olan temel işlevlere sahip olmasıdır. Genelde yapılandırma dosyalarını okumak veya komut çıktılarını işlemek için kullanılır. Dosyalarla işlem yapmak için –f parametresi kullanılır. Çağrılma şekli:

```
$ awk –f awk_script dosya
$ awk desen {aksiyon} dosya
```

Eğer desen belirtilmezse bütün satırlar eşleşir. Sonda verilen dosya parametresi yerine metin akımı üzerinde de işlem yapabilir.

Belirtilen dosyadaki 2 ve 3 numaralı sütunu ekrana basmak için (sütunlar $2 ve $3 ile belirtildi):

```
$ awk < dosya '{ print $2, $3 }'
```

Bash ile biten satırları göstermek için:

```
$ awk ‘/bash$/’ /etc/passwd
```

Awk ön tanımlı olarak boşluk ve Tab’a göre sütunlara ayırır. Bunu değiştirmek için –F parametresi kullanılır.

Bash ile biten satırlardaki birinci kolonu (kullanıcı adı) gösterir. Komuttaki $1, birinci kolonu gösteriyor. /etc/passwd dosyasında sütunlar : ile ayrıldığından ayraç olarak –F ile tanımlanmıştır.

```
$ awk < /etc/passwd -F: '{ print $6 }'root
mysql
cyrus
asterisk
spamfilter
user1
```

awk ile matematik işlemler yapılabilir. Aşağıdaki örnek Fahrenheit’ten Celsius’a sıcaklık çevrimi yapar:

```
$ awk '{ print ($1-32)*(5/9) }'
```

sort

sort komutu dosyadaki satırları sıralamak için kullanılır.

-n: sayısal(numeric) sıralama yapar.
-f: Büyük/küçük harf ayrımı olmadan sıralar.
-r: sıralama sonucunu tersler listeler.

```
$ cat a.txt
Linux
101
pardus
business
academy
```

```
$ sort a.txt
101
Academy
business
Linux
pardus
$ sort -n a.txt
Academy
business
Linux
pardus
101
```

uniq

uniq komutu birden fazla olan aynı satırları tek satıra düşürmek için kullanılır. Genellikle sort komutu ile sıralama yapıldıktan sonra aynı çifte kayıtları silmek için kullanılır.

```
$ sort a.txt
101
101
academy
business
Linux
Linux
pardus
$ sort a.txt |uniq
101
Academy
business
Linux
pardus
```

Dosya İsmi Örüntüleri

Linux altında dosya isimleri açıkça yazılabildiği gibi belirli kurala uyan dosyaların tamamını ifade etmek üzere örüntüler de kullanılabilir.

? Herhangi bir tek karakteri gösterir.
hd?, hda, hdb, hdc, hdd gibi hd ile başlayıp devamın tek karakter olan bütün dosya isimleriyle eşleşir.

* Hiç veya herhangi sayıdaki karakteri gösterir.

*1.jpg, dosyaadı 1 olan veya 1 ile biten tüm jpg uzantılı dosyalarla eşleşir.

[ ] Aralık gösterir.
a[1-9], a1, a2…a9 isimleriyle eşleşir.
a[6,8], a6 ve a8 isimleriyle eşleşir.

{ } Virgülle ayrılmış seçenekleri gösterir.
{*.doc,*.pdf} .doc ve .pdf dosyaları gösterir.

\ Özel karakterleri korumak için kullanılır.

```
$ ls *.jpg          # Uzantısı jpg olan tüm dosyalar
$ ls ?.jpg          # Dosya adı tek karakter olan jpg dosyalar
$ rm [A-Z]*.pdf     # Büyük harfle başlayan pdf dosyalar
¢ ls report*2013[1-5]
```

file

Dosya türünü belirlemek için kullanılır. /usr/share/file/magic dosyasındaki dosya türlerinin imzası olan sihirli numaraları kullanır.

```
$ file test.tar.gz
test.tar.gz: gzip compressed data, deflated,
 last modified: Sun Sep 16 13:34:51 2001, os: Unix

$ file -z test.tar.gz
test.tar.gz: GNU tar archive (gzip compressed data, deflated,
 last modified: Sun Sep 16 13:34:51 2001, os: Unix)
```

tee

Komut çıktısını hem dosyaya hem de Stdout'a yazar.

-a: Dosyanın sonuna ekleme yapar, mevcut dosyayı sıfırlamaz.

Aşağıdaki komut dizisinde ls'in çıktısı hem dosyaya hem de Stdout'a yazılır.

```
$ ls | tee file.txt
```

Stdout'a yazdığı için Stdin'den okuyan herhangi bir komuta da çıktıyı gönderebilir. Aşağıdaki komut dizisinde çıktı hem dosyaya yazılır hem de sort komutuna girdi olarak geçilir.

```
$ ls | tee file.txt | sort
```

Çıktı birden fazla dosyaya yazılabilir:

```
$ ls | tee file1.txt file2.txt
```

xargs

Bir komutu parametrelerini standart girdiden(stdin) alıp başka komutlara geçebilecek şekilde çalıştırır. Böylece bir defada işleyebileceği parametreden daha fazlası komuta aktarılabilir. Genelde ls veya find komutu tarafından üretilmiş uzun dosya listesini, bu dosyaların herbirinde tek tek işlem yapacak bir başka komuta parametre olarak geçmek için kullanılır.

Örnek: Bir patterni bütün dosyalarda aramakta için:

```
# find /etc | xargs grep pattern
```

Xargs'ın en uygun kullanım yerlerinden birisi de bir dizindeki belli dosyaları silmektir. Bunlar tek satırda rm komutuna parametre olarak verilemediğinden xargs'a ihtiyaç vardır:

```
$ ls "*.tmp" | xargs rm
```

-n: Çalıştırılacak komuta eklenecek parametre sayısı

Örneğin diff komutu iki dosyayı karşılaştırır. Mevcut dizindeki dosyaları ikişer ikişer gruplar halinde karşılaştırmak için:

```
$ ls | xargs –n2 diff
```

```
$ echo 1 2 3 4 | xargs -n 2
1 2
3 4
```

xargs mevcut komutun parametrelerini tamamlamak için de kullanılabilir:

```
$ cat 1.txt
GNU
Linux
Debian
Mint
Ubuntu
$ cat 1.txt | xargs -n2
GNU Linux
Debian Mint
Ubuntu
```

Bir dizindeki tüm dosyaları başka bir dizine tek tek taşımak için:

```
$ ls olddir | xargs -i mv olddir/{ } newdir/{ }
```

-i: { } bulunan her yere Stdin'den okunan değeri (örnekte ls çıktısındaki dosya ismi) yazar.

```
$ find . -maxdepth 1 -type d -print | xargs -I {} echo Directory: {}
```

Normalde Bash kabukta ters tek tırnak ile komut çalıştırılabilir. Ancak bu çok karmaşık olabilmektedir. xargs komutu ters tek tırnak ihtiyacını ortadan kaldırdığı gibi daha fazlasını sunar.

Disk Bölümlendirmesi

Konu:

Sabit diskin bölümlendirilmesi ve kullanıma hazır hale getirilmesi ve bakımı

Hedefler:

- Amaca uygun disk bölümlendirmesini yapabilmek
- Açılış yükleyicisini yapılandırmak ve yüklemek
- Disk sorunlarını çözmek
- Kullanıcı kotalarını aktif etme
- Disk bölümlerini bir dizine bağlama

Anahtar Kelimeler:

MBR, fdisk, GRUB, LILO, mkfs, tune2fs, fsck, mount, /etc/fstab, quotaon, which, locate, whereis

Bölümlendirme

Sabit diskler birden fazla bölümlere ayrılarak her bir bölüm ayrı bir işletim sistemine veya bir işletim sisteminin ayrı bir dizinine tahsis edilebilir. Disk bölümlendirmenin amaçları:

- İşletim sisteminin temel bileşenleri ile kullanıcılara veya kuruma ait bilgilerin ayrılması. Örneğin kullanıcı ev dizinleri, şirketin veri tabanı vs… Bu sayede işletim sisteminde yapılacak güncelleme gibi işlemler kullanıcı verisini etkilemeyecektir.

- Farklı karakteristikteki kullanımları ayırmak için. Örneğin /var dizini sürekli yazılan log dosyalarını içerirken, veritabanları ağırlıklı olarak sorgu için kullanılır. Bunlar için ayrı bölümlendirme yapılarak ayrı biçimlendirilebilir, gereksinime uygun iyileştirmeler yapılabilir.

- Yönetim kolaylığı. Örneğin /boot bölümü ayrı tutarak işletim sisteminin açılış problemlerini yönetmek daha kolay hale getirilir. Örneğin ağ üzerinden paylaştırılacak bir dosya sunucusunun dosyaları içeren bölümü ayrı bir bölüm yapılabilir.

MBR

Linux Intel tabanlı donanımlarda Microsoft Windows İşletim sisteminin kullandığı aynı MBR(Master Book Record) tablosunu kullanır. Bu disk bölümlendirmesi en fazla 4 adet birincil(primary) disk bölümlendirmesine(partititon) izin verir. Bu şemayı genişletmek için genişletilmiş(extended) adı verilen özel bir birincil disk bölümlendirmesi oluşturulmalıdır. Genişletilmiş bölümlendirme ile mantıksal disk bölümleri oluşturulabilmektedir. IDE disklerde Linux en fazla 63 adet kullanılabilir disk bölümlendirmesine izin verirken bu limit SCSI disklerde en fazla 15 adet olmaktadır.

GPT – GUID Disk Bölümlendirme Tablosu

MBR'den açılış yapan BIOS'da bazı kısıtlamalar var. Günüüzde disk boyutları gittikçe büyümektedir. MBR'de kullanılan 32 bit LBA adresleme 2TB üzerini adresleyememektedir. Bu yüzden 64 bit LBA desteği olan GPT kullanımı önem kazanmaktadır. MBR formatındaki disk bölümlerini görmek ve yönetmek için fdisk komutu ile kullanılırken, GPT'de parted komutu kullanılmaktadır.

Linux kurulumu için minimum 3 adet disk bölümlendirmesi tavsiye edilmektedir. Bunlardan birincisi /boot bölümüdür. Bu bölüm linux çekirdeğininin bulunduğu disk bölümüdür. Ortalama olarak 200-300MB disk alanı ayrılması yeterlidir.

İkincisi ise geçici bellek olarak kullanılan swap alanıdır. Bu alan eskiden bellek miktarına göre ayarlanırken günümüzde donanımların çok fazla belleği olduğu bellek ile orantılı bir şekilde boyut vermeye gerek yoktur. Makinede çalıştırılacak servislere göre bu değer değişebilir fakat asla fiziksel bellek miktarı kadar yer ayırmaya gerek yoktur. Çünkü bir sistem eğer swap alanını kullanmaya başlamışsa orada ciddi bir performans sorunu var demektir. Ya sorunu çıkaran uygulama tespit edilerek gerekli düzeltmeler yapılmalı ya da ilave RAM takviyesi yapılmalıdır.

Üçüncü bölüm ise / (root) dosya sistemidir. Bu disk bölümlendirmesi tüm işletim sistemine ve kullanıcılara ait dosyaları tutmaktadır. Bu bölüm oluşturulmadan işletim sistemi kurulum aşamasına geçilemez.

Oluşturulabilecek Disk Bölümleri

Dosyaları saklamak için tek bir / bölümü bazen yetersiz kalmaktadır. Güvenlik, veri yedekleme vb gibi sebeplerden dolayı yukarıdaki bahsedilen üç disk bölümüne ilaveten aşağıdaki gibi disk bölümlendirmesi yapılabilir. En yaygın kullanılan disk bölümlendirmeleri ve amaçları şu şekildedir:

- /home Kullanıcıların ön tanımlı ev dizinidir. Eğer Linux'e giriş yapacak veri saklayacak çok fazla ev kullanıcısı varsa /home dizininin ayrı bir disk bölümü olması tavsiye edilir. Bu sayede kullanıcıların yanlışlıkla veya bilerek / bölümünü doldurmaları engellenmiş olur. Ayrıca ana Pardus sürüm geçişlerinde / bölümünü yeniden formatlamak gerekirse /home bölümü formatlanmadan geçiş yapılabilir. Bu sayede kullanıcıların dosyalarının yedeğinin alınmasına gerek kalmaz.
- /tmp geçici dosyalar dizini. Bu dizine tüm kullanıcılar yazabilmektedir. Ayrıca bu dizin saldırganlar tarafından sunuculardaki güvenlik açığını kullanarak sisteme program yüklemeyip çalıştırmak için kullanılır. Bu disk bölümü ayrı yapılarak disk bölümü bazında bazı güvenlik önlemleri alınabilir ayrıca kullanıcıların / bölümünü doldurmasını engellemek için ayrı bir disk bölümü olarak yapılandırılabilir.
- /var günlük,e-posta ve veri dizini. Bu dizinde çok fazla verinin yazıldığı bir disk bölümüdür. Ön tanımlı olarak günlükler, veritabanı dosyaları,e-posta kuyruğu bu dizinde tutulmaktadır.
- /opt üçüncü parti yazılımların kurulduğu dizindir. Bu bölümü kullanan çok büyük yazılımlar yoksa ayrı bir bölüm yapılmasına gerek yoktur.

Açılış Yükleyicisi

Eski Linux dağıtımlarında ön tanımlı olarak LILO geliyordu. Günümüzde yerini GRUB açılış yükleyicisine bıraktı. GRUB(GRand Unified Bootloader) yüklenicisinin yapılandırması /boot/grub/grub.cfg dosyasında tanımlıdır. Fakat bunun doğrudan düzenlenmesi tavsiye edilmez. /etc/default/grub dosyasında gerekli düzenlemeler yapıldıktan sonra update-grub komutu ile /boot/grub/grub.cfg dosyası güncellenir. Fakat orijinal GRUB dosyası /boot/grub/menu.lst dosyasıdır. Redhat vb bazı dağıtımlarda akılda kalıcı olması için /boot/grub/grub.conf dosyasını /boot/grub/menu.lst dosyasına bağlantı olarak tanımlamıştır.

/etc/default/grub dosyasında menu ayarları hakkında tanımlamalar bulunur. Bu dosyanın içeriği genel olarak aşağıdaki gibidir.

```
# more /etc/default/grub
# If you change this file, run 'update-grub' afterwards to
update
# /boot/grub/grub.cfg.
```

```
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_TIMEOUT=7
GRUB_HIDDEN_TIMEOUT=0
GRUB_DISTRIBUTOR=Pardus
#GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo
Debian`
GRUB_CMDLINE_LINUX_DEFAULT=""
GRUB_CMDLINE_LINUX="modprobe.blacklist=floppy splash quiet --"
```

Bu yapılandırma GRUB ile gelen standart yapılandırma olmayıp Debian tabanlı dağıtımlara özel bir yapıdır.

/etc/grup.d dizininde ise menü oluşturma betikleri bulunur.

```
# ls /etc/grub.d/
00_header  05_debian_theme  10_linux  20_linux_xen  30_os-
prober  40_custom  41_custom  README
```

```
# grub.conf/menu.lst
#
# Global Options:
#
default=0
timeout=15
splashimage=/grub/bootimage.xpm.gz
#
# Kernel Image Options:
#
title Fedora (2.6.25)
root (hd0,0)
kernel /vmlinuz-2.6.25 ro root=/dev/hda5 mem=2048M
initrd /initrd-2.6.25
title Debian (2.6.26-experimental)
root (hd0,0)
kernel (hd0,0)/bzImage-2.6.26-experimental ro root=/dev/hda6
#
```

Grub dosyasındaki temel parametreler şunlardır:

(hd0,0) ifadesi yüklenecek çekirdeğin hangi diskin hangi bölümünde olduğunu ifade eder.

fdisk ile Disk Bölümleme

Fdisk komut satırından interatkif olarak disk yapılandırması yapmaya yarayan bir programdır. Bu komut ile DOS sitilinde bir disk bölümledirmesi yapılmaktadır. Disk

bölümlendirmesi yapılırken her bir disk bölümü için bir disk bölümü kodu tanımlandır. Linux'de yaygın olarak kullanılan disk bölümü kodu şunlardır.

83 Linux dosya sistemi(ext2,ext3,ext4)
82 swap
8e fiziksel hacim
5 genişletilmiş disk alanı

fdisk –l parameresi ile disk üzerindeki disk bölümlemleri listelenebilir.

```
root@pardus:~# fdisk -l /dev/sda
GNU Fdisk 1.2.4
Copyright (C) 1998 - 2006 Free Software Foundation, Inc.
This program is free software, covered by the GNU General Public
License.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.


Disk /dev/sda: 32 GB, 32210196480 bytes
255 heads, 63 sectors/track, 3916 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1               1         192     1542208   83  Linux
/dev/sda2             193        3916    29904997   83  Linux
root@pardus:~#
```

Dosya Sistemi Oluşturma

Dosya sistemi oluşturmak için mkfs komutu kullanılır. Ön tanımlı olarak mkfs komutu ext2 linux dosya sistemi formatında disk bölümünü formatlar. Fakat –t parametresi ile diğer dosya sistemleri de oluşturulabilir. Örneğin /dev/sdb1 disk bölümünde MS-DOS formatında disk bölümü oluşturmak için aşağıdaki komut kullanılır.

```
# mkfs –t msdos /dev/sdb1
```

Aslında mkfs komutu farklı komutlar için bir aracı komut durumundadır. Kendisi herhangi bir dosya sistemi oluşturmaz. Verilen parametreye göre ilgili dosya sistemi oluşturma komutunu çalıştırır. Yukarıdaki örnekteki –t msdos parametresinden dolayı aslında mkdosfs komutunu çalıştırır. –t parametresi verilmediğinde ext2 dosya

sistemini ise mke2fs komutunu çalıştırarak oluşturur. Mkfs yerine doğrudan ilgili dosya sistemine ait komutu da çalıştırabilirsiiz. Bu komutlardan bazıları:

```
mke2fs mkfs.bfs   mkdosfs    mkfs.cramfs   mkfs.ext3
mkfs.ext4dev  mkfs.minix    mkfs.ufs      mkfs.xfs  mkfs.btrfs
mkfs.ext2     mkfs.ext4     mkfs.jfs      mkfs.msdos
mkfs.vfat
```

ext3 dosya sistemi ext2 dosya sisteminin journaling(günlükleme) özelliği eklenmiş sürümüdür. Aşağıdaki üç komut ile de ext3 dosya sistemi oluşturulabilir.

mkfs –t ext3, mkfs.ext3 ve mke2fs -j

Örneğin /dev/sdb2 disk bölümünde ext3 formatında disk bölümü oluşturmak için aşağıdaki komut kullanılır.

```
# mke2fs -j /dev/sdb2
```

Swap Alanı Oluşturma

Bir disk alanı veya dosya üzerinde swap disk bölümü oluşturmak için mkswap komutu kullanılır. Swap için ayrılan disk alanı yetersiz kaldığı durumda dosya sisteminde oluşturulacak bir dosyayı swap alanı olarakta kullanabilirsiniz. Oluşturulan swap dosyasını swapon komutu ile aktif hale getirebilirsiniz.
Örneğin 512 MB swap dosyası oluşturup etkinleştirmek için aşağıdaki komutlar verilir.

```
# dd if=/dev/zero of=/swapdosyasi bs=1024 count=524288
# mkswap /swapdosyasi
# chown root:root /swapdosyasi
# chmod 0600 /swapdosyasi
# swapon /swapdosyasi
```

swapon –s komutu ile dosya tabanlı swap dosyaları listelenebilir.

Dosya Sistemi Bakımı

Dosya sistemini oluşturduktan sonra sistem yöneticisi tarafından bazı bakımların yapılması gerekebilir. Bu bakım ve dosya sistemindeki bazı parametre değişiklikleri tune2fs komutu ile yapılabilmektedir.

Tune2fs komutu ext2,ext3 ve ext4 dosya sistemlerine kullanılabilmektedir. Tunefs ile yapılabilecek bazı değişiklikler:

- ext2 dosya sistemine journaling(günlükleme) desteği ekleyerek ext3 dosya sistemine geçiş yapma
  # tune2fs –j /dev/sda1
- fsck işleminin yapılması için geçmesi gereken zaman aralığı veya mount sayısını belirler (-c ve –i parametreleri)
- Mevcut dosya sistemi hakkında bilgiler verir. (-l parametresi)

```
root@pardus:~# tune2fs  -l /dev/sda2
tune2fs 1.42.5 (29-Jul-2012)
Filesystem volume name:   <none>
Last mounted on:          /
Filesystem UUID:          da02cb85-f80b-4283-ab4e-30079b03d14d
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode
dir_index filetype needs_recovery extent flex_bg sparse_super
large_file huge_file uninit_bg dir_nlink extra_isize
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              1872304
Block count:              7478257
Reserved block count:     373912
Free blocks:              5903589
Free inodes:              1671965
First block:              0
Block size:               4096
Fragment size:            4096
Reserved GDT blocks:      1022
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         8176
Inode blocks per group:   511
Flex block group size:    16
Filesystem created:       Tue Sep 24 11:56:17 2013
Last mount time:          Sat Nov  9 16:54:24 2013
Last write time:          Sat Nov  9 16:54:24 2013
Mount count:              4
Maximum mount count:      -1
Last checked:             Tue Sep 24 11:56:17 2013
Check interval:           0 (<none>)
Lifetime writes:          10 GB
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               256
Required extra isize:     28
Desired extra isize:      28
Journal inode:            8
First orphan inode:       1721463
Default directory hash:   half_md4
Directory Hash Seed:      a1de246f-76ea-47aa-84dd-79a94915cb6d
```

```
Journal backup:           inode blocks
```

tune2fs hem mevcut verileri gösterme hem de düzenleme özelliğine sahipken dumpe2fs komutu ise tune2fs –l çıktısının aynısını vermektedir. Dosya sistemini detaylı incelemek(debug) ve dosya sisteminin durumunu değiştirmek için de debugfs komutu kullanılır.

fsck

Eğer dosya sistemininde elektrik kesintisi vs gibi arızalardan dolayı sorunlar oluşursa fsck komutu ile dosya sistemi kurtarılmaya çalışılabilir. Bu komut parametre olarak disk bölümü adı veya disk bölümünün /etc/fstab dosyasında belirlenmiş bağlanma noktasını alır. Fsck komutu ek bir parametre kullanmadan çalıştırıldığında bulduğu her bir hatanın düzeltmesi için onay isteyecektir. Bu da büyük boyutlu dosya sistemlerinde çok zaman alabilmektedir. Tespit edilen tüm hataların düzeltilmesi için –y parametresi kullanılmalıdır. fsck komutu mutlaka okuma-yazma modunda bağlanmamış dosya sisteminde çalıştırılmalıdır.Eğer okuma ve yazmak için bağlanılmış bir dosya sisteminde çalıştırılısa beklenmedik hatalar oluşabilir.

Örneğin /dev/sda2 disk bölümünün dosya sistemini düzeltmek için aşağıdaki komut verilir.

```
# fsck –y /dev/sda2
```

Bölüm Bağlama (mount) İşlemleri

mount işlemi MS Windows dünyasındaki fiziksel disk bölümlemesinin hangi isme karşılık geldiğini tanımlama (C: D: E: vb) işleminin Linux halidir. Linux'de formatlanan bir işletim sistemine kullanıcı ve uygulamaların erişebilmesi için ilgili dosya sisteminin bir dizin üzerinden erişilebilir olması gerekir. Bu işlemi yapmak için mount komutu kullanılıdır. Bu işlemi sadece root veya root hakkı verilen kullanıcı yapabilir. Açlılış sırasında ön tanımlı olarak hangi disk bölümlerinin hangi parametrelerle bağlanılmas gerektiği /etc/fstab dosyasında tanımlanmaktadır.

Örnek bir fstab dosyası aşağıdaki gibidir.

```
# cat /etc/fstab
# UNCONFIGURED FSTAB FOR BASE SYSTEM
proc    /proc   proc    defaults        0       0
# /dev/sda1
UUID=6e856a2a-be08-4a1d-85ab-8d345d7a5c65    swap    swap    sw      0       0
# /dev/sda2
UUID=da02cb85-f80b-4283-ab4e-30079b03d14d    / ext4  rw,errors=remount-ro   0       1
```

fstab dosyasının eski formatında UUID yerine fiziksel disk bölümünün adı yazardı(/dev/sda2 gibi). Hangi UUID değerinin hangi disk bölümüne ait olduğunu öğrenmek için blkid komutu kullanılabilir.

```
# blkid
/dev/sda1: UUID="6e856a2a-be08-4a1d-85ab-8d345d7a5c65" TYPE="swap"
/dev/sda2: UUID="da02cb85-f80b-4283-ab4e-30079b03d14d" TYPE="ext4"
```

mount komutu parametre kullanılmadan verildiğinde o anda sistemde bağlı olan dosya sistemlerini gösterir.

```
# mount
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
udev on /dev type devtmpfs
(rw,relatime,size=10240k,nr_inodes=125513,mode=755)
devpts on /dev/pts type devpts
(rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs
(rw,nosuid,noexec,relatime,size=102704k,mode=755)
/dev/disk/by-uuid/da02cb85-f80b-4283-ab4e-30079b03d14d on / type ext4
(rw,relatime,errors=remount-ro,user_xattr,barrier=1,data=ordered)
tmpfs on /run/lock type tmpfs
(rw,nosuid,nodev,noexec,relatime,size=5120k)
tmpfs on /run/shm type tmpfs
(rw,nosuid,nodev,noexec,relatime,size=513640k)
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc
(rw,nosuid,nodev,noexec,relatime)
```

```
# mount –t ext3 /dev/sdb2 /mnt
```

Yukarıdaki komut ext3 dosya sistemli /dev/sdb2 disk bölümünü /mnt dizinine bağlar. Bu dosya sistemine ait dosyalara artık /mnt üzerinden erişilebilir.

Eğer bir dosya sistemi /etc/fstab dosyasında tanımlı ise mount işlemi için herhangi bir parametre kullanmadan sadece bağlama noktası veya disk bölümü adı yazmak yeterlidir. Örneğin /etc/fstab dosyasında /dev/sdb2 disk bölümü /mnt bölümüne bağlanacak şekilde yapılandırıldı ise aşağıdaki iki komutta aynı işi yapacaktır.

```
# mount /dev/sdb2 veya
# mount /mnt
```

ext2/ext3 ve reiserfs dosya sistemleride dosya sistemi adı yerine kullanılabilen etiketleme desteği vardır. Bu özellik fstab dosyasında LABEL olarak tanımlanır. Aşağıdaki gibi bir komutla da dosya sisteminin tam yolu verilmeden sadece LABEL verilerek mount işlemi yapılabilir.

```
# mount LABEL=maildizini /mail
```

/etc/fstab dosyasındaki tanımlı tüm dosya sistemlerinin mount edilmesi isteniyorsa mount –a parametresi verilmelidir.

CD-ROM için Linux’de /dev/cdrom aygıtı kullanılır. CD’yi bağlamak için de genelde /media dizini kullanılır.

```
# mount /dev/cdrom /media
```

Dosya sistemi kullanılmadığı durumda umount komutu ile dosyaya erişim kapatılır.

```
# umount /dev/cdrom
```

Eğer dosya sistemi kullanıcı veya bir proses tarafından erişilmekte ise umount komutu dosya sistemi meşgul diye hata verecektir. Hangi dosyanın hangi prosesler tarafından kullanıldığını lsof komutu ile görebilirsiniz. Bu komut ön tanımlı olarak hangi dosyanın ve port numarasının hangi prosesler ve kullanıcılar tarafından kullanıldığını listelemektedir.

```
root@pardus:~# lsof
COMMAND     PID   TID       USER   FD      TYPE             DEVICE SIZE/OFF       NODE NAME
init          1             root  cwd       DIR                8,2     4096          2 /
init          1             root  rtd       DIR                8,2     4096          2 /
init          1             root  txt       REG                8,2    36992     664455 /sbin/init
init          1             root  mem       REG                8,2    14664     654765
/lib/x86_64-linux-gnu/libdl-2.17.so
init          1             root  mem       REG                8,2  1742520     654762
/lib/x86_64-linux-gnu/libc-2.17.so
init          1             root  mem       REG                8,2   126232     658039
/lib/x86_64-linux-gnu/libselinux.so.1
init          1             root  mem       REG                8,2   261184     658040
/lib/x86_64-linux-gnu/libsepol.so.1
init          1             root  mem       REG                8,2   145160     654759
/lib/x86_64-linux-gnu/ld-2.17.so
init          1
```

Fstab Parametreleri

/etc/fstab dosyasındaki 4. Kısım seçenekleri ifade eder.

# <file system>	<dir>	<type>	<options>	<dump>	<pass>
/dev/sda1	/	ext4	defaults,noatime	0	1
/dev/sda2	none	swap	defaults	0	0
/dev/sda3	/home	ext4	defaults,noatime	0	2

Ön tanımlı olarak dosya sistemlerini sadece root kullanıcısı mount veya umount yapabilir. Eğer kullanıcılarında bu işlemleri yapabilmesi isteniyorsa /etc/fstab dosyasındaki seçenekler kısmına user parametresi eklenmelidir. Örnek:

/dev/sda8 /data ext4 rw,user,exec 0 0

Bu dosyadaki bir dosya sisteminin açılışta bağlanıp bağlanmayacağı auto ve noauto parametreleri ile belirlenir.

Kota yapılandırması

Linux dosya sistemleri için kota desteği vardır. Kota ile kullanıcı ve grupların disk kullanımını kısıtlayabilirsiniz. Kota yapılandırması için özetle aşağıdaki süreç uygulanır.

Dosya sisteminin disk kotası desteği ile mount edilmesi
Kota veritabanı oluşturulması
Kotanın aktifleştirilmesi
Kullanıcı ve gruplara kota limitleri tanımlanması

Dosya sisteminin kota desteği ile açılması için /etc/fstab dosyasında ilgili dosya sisteminin seçenekler kısmına usrquota ve/veya grpquota parametresi ekli olmalıdır. Bu parametre eklendikten sonra ilgili dosya sistemi yeniden mount edilmelidir.

Kota genellikle son kullanıcıların yogun kullandığı dosya sistemlerinde kullanılır. Buna en iyi örnek /home bölümüdür. /home dizininde kota uygulaması yapmak için bu dizinin bir dosya sisteminin bağlanma noktası olmalıdır.

/etc/fstab dosyasına kota seçenekleri eklendikten sonra aşağıdaki quotacheck komutu ile kota veritabanı dosyaları(aquota.user ve aquota.group) oluşturulmalıdır.

```
# quotacheck –cugm /home
```

Bu işlemden sonar quotaon komutu ile dosya sisteminde kota aktif edilmiş olur.Eğer bu komut çaktısında /home altında aquota.user.new.tmp dosyası oluşturulamadı gibi bir hata alınırsa sebebi SELinux olabilir. Kota yapılandırması için SELinux özelliği aktif ise kapatılması gerekir.

```
# quotaon /home
```

Geçici olarak kota özelliğini iptal etmek için de quotaoff komutu kullanılır.

```
# quotaoff /home
```

Kullanıcılara kota tanımlamada setquota ve edquota komutları kullanılır. Edquota komutu sayesinde interaktif olarak bir editor ile yapılandırma yapılırken setquota komutu ile parametrelerle kota tanımlaması yapılabilir.

Kota kullanımını raporlamak içinde repquota ve quota komutları kullanılabilir.

Sembolik ve Hard Bağlantılar

Dosyalar ve dizinler arasında link oluşturmak için ln komutu kullanılı. Bu komut, orijinal dosya ile aynı yetkilendirmeye sahip yeni bir dosya kaydı oluşturur.

Link oluşturma genelde bulunan dosya sisteminde disk alanı yetersiz hale geldiğinde dosyaları başka bir dosya sistemine taşıdıktan sonra eski dosya/dizin yolunu kullanan kişi ve uygulamaların hala eski yol üzerinden erişmeleri için kullanılır.

İki çeşit bağlantı vardır. Sıkı (hard) bağlantılarda yapılan değişikler orijinal dosyayı etkiler. Sembolik (soft) bağlantı sadece orijinal dosyayı gösteren bir isimdir. MS Windows sistemlerindeki karşılığı kısa yol oluşturmaktır. Sembolik bağlantının silinmesi orijinal dosyayı silmez. Orijinal dosya silinirse de sembolik link işlevsiz hale gelecektir. Sembolik link üzerinden dosyaya erişilemez. Sembolik link oluşturmak için –s parametresi kullanılır. Hiç parametre kullanılmadan kullanılırsa hard link oluşturulmuş olur.

Kullanımı:

```
# ln seçenek mevcut_dosya/dizin olusturulacak_bağlantı
root@pardus:~# ln -s /data/pardus /var/data/pardus
root@pardus:~# ls -al /var/data/
toplam 8
drwxr-xr-x  2 root root 4096 Kas 29 16:30 .
drwxr-xr-x 13 root root 4096 Kas 29 16:30 ..
lrwxrwxrwx  1 root root   12 Kas 29 16:30 pardus -> /data/pardus
```

Yukarıdaki komut ile /var/data/pardus dizinine yapılan tüm erişimler aslında /data/pardus dizinde uygulanacak.

Hard linkte ise orijinal dosya ile link oluşturulan dosyanın birebir aynısı iki farklı dosya oluşur. Birisinde yapılan işlem diğerine de yansır. Bu durumu disklerde kullanılan RAID-1 teknolojisine benzetebiliriz. Orijinal dosya silinse bile verilere diğer dosya üzerinden erişilebilir. Katı linkteki bu özellik Linux inode yapısındaki dosyaya yapılan link sayısı özelliği ile sağlanmaktadır. Dosyanın biri silindiğinde link sayısı değeri bir azaltılmakta. Diğer dosya(lar) silindiğinde ise link sayısı sıfır olduğunda dosya tamamen silinmektedir. Dolayısıyla hard link dosya sistem ile doğrudan alakalıdır. Farklı dosya sistemlerinin(disk bölümleri) farklı inode yapısı olduğu için hard link farklı dosya sistemleri arasında yapılamaz. Sadece aynı dosya sistemi üzerinde yapılabilir.

Farklı dosya sistemleri arasında link oluşturulacaksa sadece sembolik link kullanılabilir.

Dosya Arama

Linux'de find ile dosya isminde ve içeriğinde arama yapılabilmektedir. Buna ek olarak bir kaç farklı komut ile de arama yapılabilmektedir.

which komutu

$PATH değişkeni içerisinde tanımlı çalıştırılabilir komutların tam yolunu gösterir. Sadece komutlar ve shell scriptler için çalışır.

```
# which cut
/bin/cut
```

locate komutu

find komutu arama yaparken her seferinde belirtilen dizinde tarama yaptığı için çok fazla dosya olan sistemlerde yavaş çalışabilir. Daha hızlı dosya aramak için locate komutu kullanılabilir. Bu komut dosya adı aramalarını direkt dosya sisteminde tarama yapmak yerine özel bir veritabanından getirmektedir. Locate komutu aramayı /var/cache/locate/locatedb veritabanı dosyasında yapmaktadır. Bu dosya locate dışında ayrı olarak çalışan updatedb komutu tarafından güncellenmektedir. Eğer en güncel dosyalarında aramaya dahil olmasını istiyorsanız belirli aralıklarla updatedb komutu çalıştırılmalıdır. Her gece veritabanı dosyasını güncellemek için Pardus tarafından otomatik olarak /etc/cron.daily/locate dosyasındaki cron işi çalıştırılmaktadır. Arama yapmak için dosyanın adı veya adının geçtiği kelimeleri yazmak yeterlidir.

```
root@pardus:/ # locate apache
/etc/apache2
/etc/apache2/apache2.conf
/etc/apache2/conf.d
/etc/apache2/conf.d/charset
/etc/apache2/conf.d/javascript-common.conf
/etc/apache2/conf.d/localized-error-pages
..
```

whereis komutu

Parametre olarak verilen program, kaynak kodu ve yardım dosyalarının yerlerini gösterir.

```
# whereis pwd
pwd: /bin/pwd /usr/include/pwd.h /usr/share/man/man1/pwd.1.gz
```

Linux Yetkilendirme Modeli

Konu:

UNIX yetkilendirme modeli

Hedefler:

- Yetkilendirme modelini anlamak
- Dosyalara erişim gerekli izinlerini tanımlamak
- s ve t bitleri

Anahtar Kelimeler:

chmod, chown, chgrp

Yetkiler

Linux yetkilendirme modeli, UNIX ve BSD'den alınmıştır.

```
# ls -l /bin/cp
-rwxr-xr-x 1 root root 130128 Oca 26  2013 /bin/cp
```

'ls' komutu `-l` parametresi ile kullanıldığında yukarıdaki gibi dosyaların yetkilerini belirten ayrıntılı bir çıktı verir. Satır başında yer alan `-rwxr-xr-x` bu dosyaya ait hakları belirtmektedir. Bu haklar üç kısımdan oluşmaktadır.

```
-rwx  r-x  r-x
```

İlk kısmın hemen başında tek bir simgeyle dosyanın türü ifade edilmektedir. Yukarıdaki gibi eğer boş ise (yani -) normal dosyadır.

```
'd' directory (dizin)
'l' symbolic link (sembolik bağ)
'c' character special device (karakter aygıt)
'b' block special device (blok aygıt)
'p' fifo
's' socket
```

İlk kısım dosya sahibine ait hakları, ikinci kısım aynı gruptaki kullanıcılara ait hakları, son kısım ise diğer kullanıcılara ait hakları belirlemektedir.

```
'r' okuma izni(4)
'w' yazma izni(2)
'x' çalıştırma izni(1)
```

Örneğin: 754 modu Dosya sahibi için 7 (4 + 2 + 1, yani rwx), grubu için 5 (4 + 1, yani rx) ve herkes için 4 (yani r) hakkı vermektedir.

Yetkileri değiştirmek için 'chmod' komutu kullanılır.

```
# ls -al dosya.txt
-rw-r--r-- 1 root root 0 Kas 29 16:34 file

# chmod 755 dosya.txt
# ls -al dosya.txt
-rwxr-xr-x 1 root root 0 Kas 29 16:35 dosya.txt
```

755 belirtilen üç grup için 7,5,5 haklarını vermektedir. Her bir yetkinin rakamsal değeri vardır. 7,5 ve 5 bu rakamların toplamıdır.

```
r – 4
w – 2
x – 1
```

Bu rakamlara göre 7 = r + w + x anlamında, 5 = r + x anlamındadır. Bunun dışında 'u – user', 'g – group' ve 'o – other' olacak şekilde de yetki verilebilir:

```
# chmod g+w dosya.txt
```

Örnekler:
Hiç hakları olmayan dosya aşağıdaki gibi görünür:

```
# ls -al sirala.sh
---------- 1 root root 0 Kas 29 16:37 sirala.sh
```

Bu dosya için herkese r:okuma hakkı şu şekilde verilir:

```
# chmod +r sirala.sh
# ls -al sirala.sh
-r--r--r-- 1 root root 0 Kas 29 16:37 sirala.sh
```

Eğer yalnızca dosya sahibine (u:user) yazma hakkı verilmek istenirse:

```
# chmod u+w sirala.sh
# ls -al sirala.sh
-rw-r--r-- 1 root root 0 Kas 29 16:37 sirala.sh
```

Eğer gruba (g) ve diğerlerine (o) çalıştırma hakkı verilecekse aşağıdaki komut kullanılır:

```
# chmod g+x,o+x sirala.sh
# ls -al sirala.sh
-rw-r-xr-x 1 root root 0 Kas 29 16:37 sirala.sh
```

Veya tüm bu işlemler rakamlarla yapılabilir.

```
# chmod 754 sirala.sh
# ls -al sirala.sh
-rwxr-xr-- 1 root root 0 Kas 29 16:37 sirala.sh
```

chown

Dosya sahibi ve grubunu değiştirir.
-R
Belirtilen dizinin tüm dosyaları ve alt dizinleriyle birlikte sahip ve grup bilgilerini değiştir.

Aşağıdaki örnekte şimdiki sahibi root:wheel olan beni.oku dosyasının sahibi simsek:admin olarak değiştirilmiştir.

```
# ls -l beni.oku
```

```
-rw-r--r-- 1 root root 0 Kas 29 16:38 beni.oku
```

```
# chown simsek:admin beni.oku
```

```
# ls -al beni.oku
-rw-r--r-- 1 simsek admin 0 Kas 29 16:38 beni.oku
```

chgrp

Sadece dosya ve dizinin grubunu değiştirmede kullanır. Chown komutu gibi –R parametresi kullanılabilir. Aşağıdaki örnekte dosyanın sahibi users grubu yapılmıştır.

```
# chgrp users beni.oku
```

Özel İzinler(t ve s bitleri)

Yukarıdaki örneklerde dosya erişimlerinin 3bit(---) üzerinden tanımlandığı belirtilmişti. Aslında genelde 3 bit üzerinden yetkilendirilir demek daha doğru bir ifadedir. Fakat gerçekte günlük hayatta çok fazla kullanılmayan 4. Bir daha vardır. Dördüncü bit'de t(sticky bit) ve s(suid bit) olmak üzere iki ayrı vardır.

Sticky bit(t) genelde herkesin yazmasına açık olan /tmp gibi dizinlere uygulanır. Bu bitin amacı herkesin erişimi olan dizinlerde kullanıcılar tarafından oluşturulan dosyaların sadece o kullanıcı tarafından silinebilmesini sağlar. Dizinlere sticky bit hakkı chmod 1777 veya chmod +t komutu ile verilir. Dizinin stickey bit hakları olup olmadığı dosya haklarının sonundaki t ifadesi ile anlaşılır.

```
# ls -ald /tmp
```

```
drwxrwxrwt 11 root root 4096 Kas 29 16:15 /tmp
```

Çalıştırma hakkı olmayan dizin veya dosyaya sticky bit hakkı verilirse sondaki t yerine T gösterilir.

Suid bit(s) ise normal kullanıcıların bir komutu çalıştırırken sadece o komut için geçici olarak daha yetkili bir kullanıcı haklarıyla iş yapmasını sağlar. Bu hak chmod 4755 komutu ile verilir.

Linux Dizin Yapısı

Konu:

Standart Linux dizin hiyerarşisi

Hedefler:

- Dosya sisteminin hiyerarşik düzenini anlamak
- Hangi dizinin hangi amaçla kullanıldığını öğrenmek

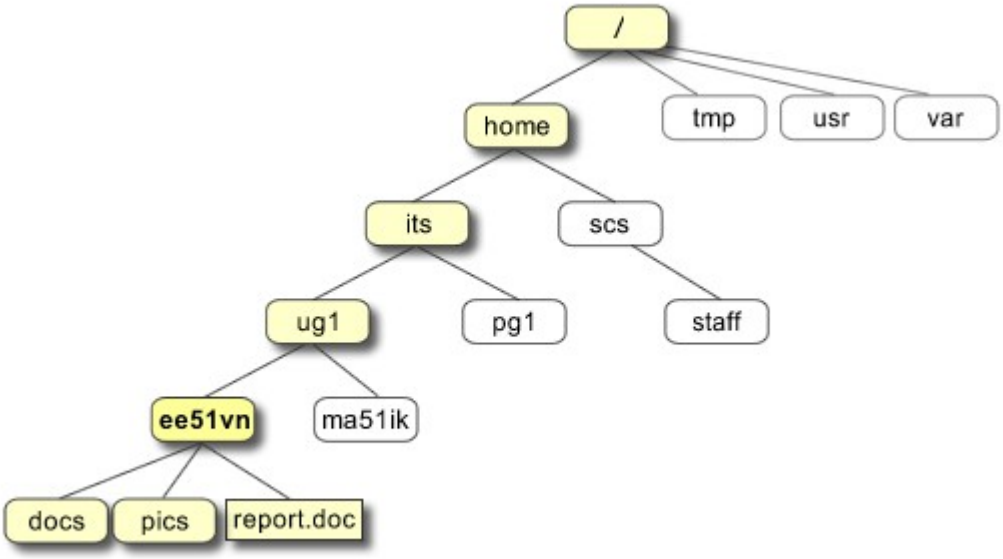
Anahtar Kelimeler:

/boot, /root, /etc, /usr, /bin, /sbin, /mnt, /lib, /dev, /proc, /tmp, /var, /home

Hiyerarşi

Günümüzde yüzlerce linux dağıtımı bulunmaktadır. Her bir dağıtım kendi dizin yapısı hiyerarşisi kullanmaya başladığında her bir linux dağıtımına özel dizin yapısını bilmek gerekiyordu. Bu sorunu aşmak için Dosya sistemi hiyerarşi standartı(Filesystem Hierarchy Standard(FHS) belirlendi. Bu standartın amacı kurulan yazılımların ve kullanıcıların kullanacağı dosya ve dizinleri belirlemektir. Bu standartlara göre man dosyalarının, programların yeri, veri tutan dizinlerin yeri vs hepsi belli bir kurala göre düzenlenmiştir. http://www.pathname.com/fhs/ adresinden en güncel standartı inceleyebilirsiniz.

Linux işletim sisteminde dizin hiyerarşişi genel olarak aşağıdaki şekildeki gibidir.



/bin	Temel komutlar
/boot	Ön yükleyicinin kullandığı dosyalar
/dev	Aygıt dosyaları
/etc	Sistem yapılandırma dosyaları
/home	Kullanıcıların ev dizinleri
/lib	Paylaşılan temel kütüphaneler ve çekirdek modülleri
/media	Çıkarılabilir aygıtlar için bağlama noktası
/mnt	Dosya sistemlerini geçici olarak bağlama noktası
/opt	Sonradan eklenen uygulamalar
/proc	sistemin genel ve anlık bilgilerini içeren sanal dosya sistemi.
/sbin	Temel sistem yöneticisi komutları
/tmp	Geçici dosyalar
/usr	İkincil kök, kullanıcı için yeni dizilim noktası
/var	Değişken veriler, log dosyaları vb.

/bin

/bin dizini sistem yöneticileri ve normal sistem kullanıcıları için kullanışlı temel komutları barındırır. Kullanıcı kabuğu bash ve ls, cp, rm, cat gibi pek çok temel komutu içermesi nedeni ile en basit sistem kurulumunda bile oluşturulur.

/boot

İşletim sisteminin açılması için gerekli bütün dosyalar bu dizindedir. Çekirdekler ve initrd kalıpları bu dizinde yer alır.

```
# ls /boot/
config-3.2.0-4-amd64  grub  initrd.img-3.2.0-4-amd64  System.map-
3.2.0-4-amd64  vmlinuz-3.2.0-4-amd64
```

/dev

Bu dizin özel veya aygıt dosyalarının konumudur. UNIX dünyasındaki “herşey bir dosyadır” kuralının bir göstergesi olarak bilgisayara bağlı aygıtlar, donanımlar /dev altında kendilerine erişimi sağlayan bir dosyaya sahiptirler.

/etc

Sistemin önemli yapılandırma dosyaları bu dizindedir. Yapılandırma dosyaları çalıştırılamayan fakat okunan ve yorumlanan dosyalardır. Bir programın nasıl çalışacağını belirlerler. Buradaki dosyalar olmadan sistem programları düzgün çalışmayacaktır. O nedenle /etc için sistemin kalbidir diyebiliriz. Her zaman yedeklemede önceliğe sahiptir.

/home

Linux çok kullanıcılı bir işletim sistemi olarak kullanıcıların dosyalarını başkalarına karşı koruduğu gibi herkese çalışması için bir ev (home) dizini sunar. Linux'te standart olarak kullanıcı dizinleri /home altında yer alır. Buraya kullanıcı istediği dosyaları yerleştirir, programları kurar veya siler. Çok fazla kullanıcı olan sistemlerde bu dizinin ayrı bir disk bölümü olması tavsiye edilmektedir.
Bir kullanıcı açılırken kullanıcı dizinine bazı dosyalar otomatik olarak kopyalanır. Bu dosyalar /etc/skel altındaki dosyalardır.

/lib

Bu dizin çekirdek modülleri ve sistem kütüphanelerini içerir. Çekirdek modülleri sistemin açılışı için gerekli olup bazı donanımların sistem tarafından tanınmasını

sağlar. Sistem kütüphaneleri ise /bin ve /sbin altındaki programların çalışması için gerekli C kütüphaneleridir.
Kütüphane dosyalarının uzantısı *.so şeklindedir.
Çekirdek modülleri ve aygıt sürücüleri /lib/modules/'kernel-version' dizini altındadır. Linux altında birden fazla çekirdek olabileceği için bu şekilde çekirdek sürümü ile dizinler ayrılmaktadır. Açılış zamanında hangi çekirdek seçilirse onun dizinindeki modüller yüklenir.

/lost+found

Bazı zamanlar bilgisayar doğru kapanmamış olabilir, elektrik kesintisi nedeni ile aniden kapanmış olabilir. Bu durumda bir sonraki açılışta işletim sistemi dosya sistemi kontrolü (fsck programı ile) yapacaktır. Bu kontrol sonucunda bozulmuş veya kaybolmuş bir şey varsa düzeltip veya bulup bu dizin altına koymaktadır.
Her disk bölümü kendi lost+found dizinine sahiptir. lost+found dizinindeki dosyalar eğer düzgün ise olmaları gereken dizine geri taşınırlar.

/media

Disket, cd-rom, dv-rom, cd/dvd-rw gibi sökülüp takılabilir dosya sistemleri bu dizine bağlanmaktadır.

/mnt

Dosya sistemlerini veya aygıtları bağlamak için kullanılan genel bağlama noktasıdır. Bağlama işlemi mount komutuyla yapılır:

```
# mount /dev/hdd /mnt
```

/opt

İşletim sisteminin öntanımlı kurulumu ile gelmeyen tüm yazılım ve paketlerin kurulumu için /opt dizini tahsis edilmiştir. Örneğin, Kylix, StarOffice, WordPErfect vb...

Bu dizin Windows altındaki “Program Files” dizinine benzetilebilir. Her üretici bu dizin altında kendi adıyla veya programın adıyla bir dizin açar ve programlarını bu dizinin içerisine kurar. /opt/surgate, /opt/endersys veya /opt/ecm gibi.

/proc

/proc sanal bir dosya sistemidir. Linux'un sunduğu en güçlü özelliklerden biri de /proc dosya sistemidir. Buradaki dosyalar gerçek içerik taşımaz. Çalışma zamanında çekirdek tarafından içerikler güncellenebilir. Sistem belleği, bağlanmış aygıtlar, donanım bilgisi, süreçlerin durumu gibi çekirdeğe ait bilgileri barındırır. Pek çok sistem programı bilgilerini bu dizinden alırlar:

Yukarıda görüleceği gibi lsmod komutu /proc/modules dosyasındaki bilgileri yorumlar. Bunun gibi 'lspci' komutu da /proc/pci dosyasını okur.

/root

Sistem yöneticisinin ev dizinidir. Sistem kullanıcısı 'root' isimli hesaba sahiptir ve bu hesabın ev dizini /root dizinidir. /home dizini genelde farklı disklerde olduğundan herhangi bir nedenle yetkili kullanıcının sisteme erişememesi durumu olmaması için ev dizini köke alınmıştır.

/sbin

Sistem programları (system binary) bu dizinde bulunur. Sistem programlarının normal programlardan (/bin) farkı sistem yönetimine yönelik olmalarıdır. Büyük bir kısmını çalıştırmaya normal kullanıcılar yetkili değildir. Çalıştırabildikleri ise sistemden bilgi almaya yönelik, sistemde değişiklik yapmayan sistem komutlarıdır.

mkfs disk biçimlendirme komutu, lspci, lsmod, service, sysctl, syslogd, grub, fsck, dump, init, modprobe gibi sistem komutları bu dizindedir.

/usr

Sistemde en çok yer kaplayan dizin /usr dizinidir. Bu dizin kullanıcı için ikinci bir kök dizin olarak düşünülebilir. Nasılki sistemin /bin, /sbin dizinleri var; /usr/bin ve /usr/sbin dizinleri de vardır. Sisteme sonradan kullanıcı tarafından kurulan programlar /usr 'ı kök kabul ederek dosya sistemine yerleşirler. Örneğin kullanıcı programları /usr/bin'e, sistem programları /usr/sbin'e, yapılandırma dosyaları /usr/etc veya /usr/local/etc'ye yerleştirilir.

```
# ls /usr/
bin  games  include  lib  lib32  local  sbin  share  src
```

/var

Genel olarak değişken (variable) veriler/dosyalar bu dizinde tutulur. Örneğin sistem günlükleri, mail kuyruğu, gelen mailler, yazıcı kuyruğu, programların kilit dosyaları vs...

```
# ls /var/
backups  cache  data  lib  local  lock  log  mail  opt  run  spool
tmp  www
```

/var ve /usr'ın ayrı bölümlerde olması sistem yönetimini kolaylaştırır. Çünkü /usr yalnız okunur şekilde bağlanıp korumaya alınabilir. Çünkü /usr altına dosya eklemek veya ordaki dosyaları değiştirmek çok nadiren gerçekleşir. /var ise her saniye değişebilir. Sürekli gelen emailler, sürekli günlük yazan programlar düşünülürse ne kadar yoğun yazmaya tabi tutulduğu daha kolay anlaşılır.

/tmp

Geçici olarak kullanılacak dosyalar bu dizinde oluşturulur. Bütün programlar ve kullanıcılar bu dizine yazabilirler. Buradaki dosyaları ne yaptığınıza emin olmadan silmeniz tehlikeli olabilir. Çünkü silinen dosyaları o an çalışan süreçlerden biri kullanıyor olabilir.

Arşiv ve Yedekleme

- **Konu:**

Linux altında dosya ve dosya sistemi yedekleme

- **Hedefler:**
  - Dosya yedekleme araçlarını tanımak
  - Dosya sıkıştırmak
  - Disk imajı almak

Anahtar Kelimeler:

tar, gzip, gunzip, zcat, bzip2, dd, cpio

tar

Bütün UNIX işletim sistemleri tar (Tape ARchive) ile arşivleme yapmayı destekler. Tar aslında teyp birimlerine arşivleme amacıyla geliştirilmiştir. Teyp üniteleri gelişen yeni teknoloji ile birlikte kullanımdan yavaş yavaş kalkmaktadır. Fakat tar programı halen daha değişik amaçlarla dosya sisteminde kullanılabilmektedir.

Tar kendisi veri sıkıştırması yapmaz. Veri sıkıştırması için “gzip” veya “bzip2” biçimlerini kullanabilmektedir.

```
$ tar -zcvf yedek.tar.gz db.sql mbox
db.sql
mbox

$ ls -lh yedek.tar.gz
-rw-r--r-- 1 ahmet ahmet 176 Ara 13 23:22 yedek.tar.gz
```

Komuttaki z parametresi gzip sıkıştırması da yapılmasını söylemektedir. 'v' ekrana ne yapıldığına dair bilgi dökülecegini gösterir. 'c' parametresi arşiv oluşturma için, 'x' parametresi ise arşiv açmak için kullanılır. Bzip2 biçiminde sıkıştırma yapmak için z yerine j kullanmak gerekir.

'f' parametresi ile arşivleme sonucu oluşacak veya mevcut arşiv dosyanın adını belirtmek içindir. Dısya ismini bu parametreden hemen sonra belirtmek gerekir.

Gzip ile sıkıştırılmış arşiv açılırken yine z parametresi kullanılmalıdır. Bzip için ise aynı şekilde j parametresi kullanılır:

```
$ tar -zxvf yedek.tar.gz
db.sql
mbox
```

Örneğin Gzip ile sıkıştırılmış dosyayı Bzip2 ile açmaya çalışırsanız aşağıdaki gibi hata alırsınız:

```
$ tar -jxvf yedek.tar.gz
bzip2: (stdin) is not a bzip2 file.
tar: Child returned status 2
tar: Error exit delayed from previous errors
```

Dizinler arşivlenirken aşağıdaki gibi bir bilgi mesajı çıkar:

tar: Removing leading `/' from member names

Bu mesaj, arşivde en baştaki kökü ifade eden (/) simgesinin kaldırılacağını söylemektedir. Yani Bu arşiv açıldığında örneği /etc'ye değil bulunulan dizindeki etc/ 'ye açılır. Böylece çalışan sistemin üzerine yazılması engellenmiş olur. Sistem yöneticisi kendisi ihtiyaç gördüğü yere açılan dizini kopyalayabilir.

`'t'` parametresi arşiv içeriğini gösterir. `'v'` ile kullanılırsa daha fazla detay verir:

```
$ tar -ztvf yedek.tar.gz
-rw-r--r-- simsek/wheel 125116416 2008-11-05 16:18:30 db.sql
-rw------- simsek/wheel 2771 2008-11-05 16:18:48 mbox
```

Bzip2 çok daha fazla sıkıştırma imkanı sağlayan bir biçimdir.

cpio

Dosyaları arşive kopyalamak, arşivdeki dosyaları çıkartmak için kullanılır. Arşivlenecek dosyaların listesini STDIN'den alır ve arşivi STDOUT'a yazar. Bu nedenle dosya listesinin oluşturulması ve çıkışın yönlendirilmesi gerekmektedir.

```
$ ls | cpio -ov > directory.cpio
```

-o seçeneği arşiv oluşturmak, -v seçeneği ise yapılan işlemler hakkında bilgi vermek için kullanılır.

Bir dizini alt klasörleriyle tamamen yedeklemek için dosya isimleri find komutuyla oluşturulabilir.

```
$ find . -print -depth | cpio -ov > tree.cpio
```

Arşivi açmak için:

```
$ cpio -idv < tree.cpio
```

-I seçeneği arşivi açmak, -d seçeneği ise arşivin içerdiği klasörleri oluşturmak için kullanılır. Normalde cpio klasörleri oluşturmaz.

Cpio'nun bir kullanım şekli de bir dizini digger bir dizine kopyalamasıdır.

```
$ find . -depth -print0 | cpio --null -pvd new-dir
```

cpio komutu kullanılarak tar arşiv oluşturulabilir. –F seçeneği ile çıktının yazılacağı arşiv dosyasının ismi, -H ile arşiv biçimi belirtilir.

```
$ ls | cpio -ov -H tar -F arsiv.tar
```

Tar arşivi açmak için:

```
$ cpio -idv -F ornek.tar
```

Tar arşivin içindeki dosyaları görüntülemek için:

```
$ cpio -it -F ornek.tar
```

Diğer bazı seçenekler:

-L: Sembolik link yerine, linkin gösterdiği dosya arşivlenir.
-m: Arşiv açılırken dosyaların son değiştirilme zamanını korumak için.

dd

Öntanımlı olarak Stdin'den Stdout'a belirtilen blok boyutlarında dosya kopyalar. Genel kullanımı aşağıdaki gibidir:

```
# dd if=girdi of=cikti bs=blok-boyutu count=blok-sayısı
```

if: Input File, girdi
of: Output File, çıktı
count: bs ile boyutları belirtilen bloklardan kaç adet kopyalanacağı
bs: Bir blogun boyutu (öntanımlı: 512 byte)

dd komutu genel olarak disk imajı almak için yedekleme amaçlarıyla kullanılır. Aşağıdaki komutla sda diski sdb diskine dump edilmektedir.

```
# dd if=/dev/sda of=/dev/sdb
```

Bir diskin imajını alıp dosya olarak saklamak için:

```
# dd if=/dev/hda of=haddisk.img
```

Alınmış imajı herhangi bir diske (örnekte hdb) dump etmek için:

```
# dd if=hdadisk.img of=/dev/hdb
```

Bir disk tamamen dump edilebildiği gibi içindeki bir disk bölümü yalnız başına dump edilebilir.

```
# dd if=/dev/hda1 of=hda1.img
```

Bir CD'den ISO dosyası oluşturulabilir:

```
# dd if=/dev/cdrom of=pardus.iso bs=2048
```

MBR'nin yedeğini almak için:

```
dd if=/dev/hdx of=/path/to/image count=1 bs=512
```

MBR ilk 512 byte olduğundan 512 byte'lık tek bir blok kopyalanmalı.

gzip, gunzip, zcat, bzip2

Verilen dosyayı Lempel-Ziv (LZ77) kodlamasını kullanarak sıkıştırır ve dosya boyutunu küçültür. Sıkıştırılan dosyanın uzantısı otomatik olarak .gz yapılır. Öntanımlı olarak orjinal dosya ismini ve dosya değiştirilme zamanını muhafaza eder.

```
$ gzip test.c
```

Sıkıştırılma sonucunda sıkışmamış dosya dönüşür, yani sıkışmamış hali diskten silinir. –c parametresiyle çıktı STDOUT'a yazılır ve orjinal dosya muhafaza edilir. Arşiv dosyasının saklanması için STDOUT'un bir dosyaya yönlendirilmesi gerekmektedir.

```
$ gzip -c archive.txt > arsiv.txt.gz
```

Gzip ile sıkıştırılmış dosya, gzip –d veya gunzip veya zcat ile açılır.

```
$ gunzip test.c.gz
```

Gunzip komutu, gzip, zip, compress ve pack sıkıştırma biçimlerini otomatik olarak tanıyarak açar. Gunzip –c parametresiyle çıktıyı STDOUT'a yazar. Zcat komutu Gunzip –c ile aynı işlevi görür.

Sıkıştırma işlemini bir dosyaya değil de alt klasörleriyle birlikte bir dizine yapmak için –r seçeneği kullanılır.

```
$ gzip -r directory
```

Bzip2 daha iyi sıkıştırma yapabilen Burrows-Wheeler metin sıkıştırma algoritmasını kullanan bir sıkıştırma komutudur. Kullanımı gzip ile aynıdır. Sadece dosya uzantısı bz2 şeklinde oluşturulur.

```
$ bzip2 -c readme.txt > readme.txt.bz2
```

Çalışan Süreçler

Konu:

Süreç başlatma ve çalışan süreçleri yönetme

Hedefler:

- Süreçleri başlatma ve sonlandırma
- Süreçleri görüntüleme ve izleme
- Süreçlere sinyal gönderme
- Eş zamanlı sanal terminaller açma
- Arkaplanda yürütülen görevler
- Süreçlerde öncelik

Anahtar Kelimeler:

ps, pstree, top, htop, free, uptime, screen, nohup, bg, fg, jobs, &, nice, renice, kill, killall, pkill, pgrep

Süreçler

Süreç (process) genel tanımla çalışan bir programdır. Süreçler, komut satırından çalıştırılan kısa süreli komut olabileceği gibi işletim sisteminin açık olduğu süre boyunca çalışan bir ağ servisi de olabilir.

Kullanıcıların süreçleri yönetebilmesi için her sürece ait bir PID numarası vardır. Süreçle ilgili yapılacak işlemler bu PID numarası üzerinden gerçekleştirilir. Linux altında çalışan ilk süreç init olup PID numarası her zaman 1'dir. Init ilk süreç olduğu için kullanıcı tarafından başlatılan diğer süreçlerin aksine çekirdek tarafından başlatılır.

Bazı süreçler eş zamanlı çalışan alt thread'lere sahip olabilir. Thread'ler aynı PID altındadır, kendilerina ait bir PID numarası olmaz. Örneğin Mozilla Firefox'ta bir thread DNS sorgularını çözerken diğer thread sunucuyla konuşur.

ps , pstree

Çalışan kendine ait süreçleri görüntülemek için temel olarak ps komutu kullanılır.

```
$ ps
  PID TTY          TIME CMD
 6082 pts/0    00:00:00 bash
 6190 pts/0    00:00:00 ps
```

Sistemdeki tüm süreçleri görüntülemek için:

```
$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0  10372   684 ?        Ss   Nov02   0:00 init [3]
root         2  0.0  0.0      0     0 ?        S<   Nov02   0:00 [migration/0]
root         3  0.0  0.0      0     0 ?        SN   Nov02   0:00 [ksoftirqd/0]
root         4  0.0  0.0      0     0 ?        S<   Nov02  16:01 [events/0]
root         5  0.0  0.0      0     0 ?        S<   Nov02   0:00 [khelper]
root        14  0.0  0.0      0     0 ?        S<   Nov02   0:00 [kthread]
```

a: Bütün süreçler
-u: Belirtilen kullanıcıya ait süreçler
-t: Belirtilen terminale ait kullanıcılar

Çıktı parametreler biçimlendirilebilir. Örneğin aşağıdaki komut ağaç yapısında bir çıktı üretecektir.

```
$ ps axjf
```

```
    1  2635  2635  2635 ?           -1 Ss       0   0:00 /usr/sbin/sshd
 2635  6077  6077  6077 ?           -1 Ss       0   0:00  \_ sshd: simsek [priv]
 6077  6081  6077  6077 ?           -1 S      501   0:00      \_ sshd: simsek@pts/0
 6081  6082  6082  6082 pts/0     6207 Ss     501   0:00          \_ -bash
 6082  6207  6207  6082 pts/0     6207 R+     501   0:00              \_ ps axjf
    1  2653  2653  2653 ?           -1 Ss       0   0:00 xinetd -stayalive -pidfile /var/run/xinetd.pid
    1  2669  2669  2669 ?           -1 SLs     38   0:00 ntpd -u ntp:ntp -p /var/run/ntpd.pid -g
```

top

En çok CPU kullanan süreçleri gösterir. Aynı zamanda süreçleri düzenleyebilecek etkileşimli bir arayüz sunar. Görevleri CPU kullanımına, bellek kullanımına ve çalışma süresine göre sıralar ve sistem yöneticisinin kaynak kullanımını analiz etmesini sağlar.

-d: Değerlerin güncellenme aralığını belirlemek için
-p PID: Yalnızca verilen PID numarasına sahip süreci izlemeye alır

-q: Değerler herhangi bir bekleme olmadan güncellenir
-C: Birden fazla işlemci olan sunucularda tek tek CPU değerini göstermek yerine toplam CPU değerlerini gösterir
-c: Sadece süreç adını değil tam komut satırı parametrelerini gösterir
-H: Thread'leri göstermek için

```
top - 19:11:12 up 24 days,  5:29,  1 user,  load average: 0.00, 0.00, 0.00
Tasks: 111 total,   2 running, 109 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.3%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1026140k total,   997220k used,    28920k free,   157784k buffers
Swap:  2064376k total,       12k used,  2064364k free,   452032k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
13222 simsek    15   0 12764 1100  824 R  0.3  0.1   0:00.03 top
    1 root      15   0 10372  684  572 S  0.0  0.1   0:00.77 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.00 ksoftirqd/0
    4 root      10  -5     0    0    0 S  0.0  0.0  17:28.35 events/0
    5 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 khelper
   14 root      10  -5     0    0    0 S  0.0  0.0   0:00.00 kthread
   18 root      10  -5     0    0    0 S  0.0  0.0   0:00.83 kblockd/0
   19 root      20  -5     0    0    0 S  0.0  0.0   0:00.00 kacpid
  180 root      19  -5     0    0    0 S  0.0  0.0   0:00.00 cqueue/0
  183 root      19  -5     0    0    0 S  0.0  0.0   0:00.00 khubd
```

Top çıktısında gösterilen bazı alanların açıklamaları:

İlk satır: Sistemin ne kadar süredir açık olduğunu gösteren uptime (yukarıdaki çıktıda 24 gün) ve son 1,5 ve 15 dakika içerisindeki CPU yükünü gösterir. Uptime komutuyla aynı sonucu verir.

Mem satırı: Toplam bellek, boş bellek ve kullanılan bellek miktarı gibi bellekle ilgili istatistikleri içerir.

PID: Process ID

PPID: Ana sürecin PID numarası

User: Sürecin sahibi kullanıcı

PRI: Sürecin önceliği

Time: Süreç başladığından beri tükettiği CPU zamanı

%CPU: Tüm süreçler içinde CPU zamanının % ne kadarını kullandığı

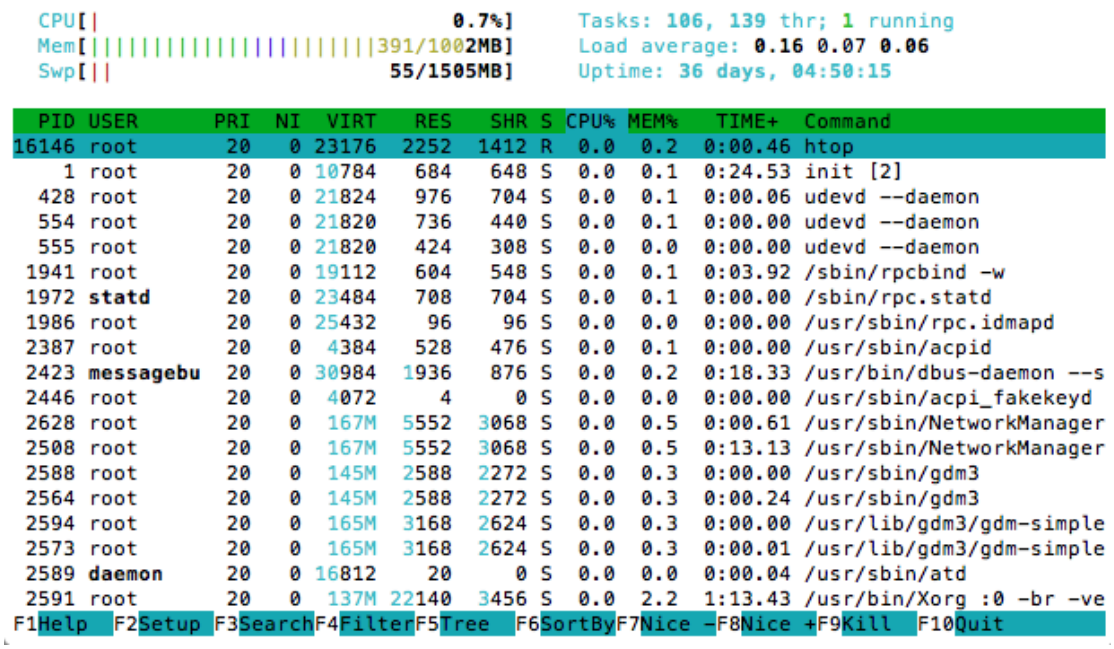
%MEM: Toplam süreçler içinde fiziksel belleği % ne kadar kullandığı

htop

htop, ncurses tabanlı top benzeri çalışan süreçleri gösteren bir uygulamadır.

Htop programı sistemle birlikte kurulu olmayabilir. Aşağıdaki şekilde kurulur:

```
# apt-get install htop
```



Htop arayüzü üç kısımdan oluşmaktadır:

Başlık kısmı: İşlemci, bellek ve Swap kullanımı gösteren barlar ve çalışan süreç sayısı, işlemci yük durumu ve sistemin ne kadar süredir açık olduğu gibi bilgileri içermektedir. Bu görüntü htop menüsünden değiştirilebilir.

Gövde kısmı: top'a benzer olarak süreçlerin işlemci kullanımına göre listelendiği kısımdır.

Alt kısım: Htop menüsünü ve kısayollarını gösterir.

free

Kullanılabilir boş veya kullanılmış bellek alanını görüntüler. Aynı zamanda swap kullanımı hakkında da bilgi verir.

```
# free
             total       used       free     shared    buffers
cached
Mem:       1006708     935872      70836          0     148244
346656
-/+ buffers/cache:     440972     565736
Swap:       262140     130084     132056
```

Mem olarak gösterilen ilk satır fiziksel bellek kullanımını gösterir.
Swap olarak gösterilen üçüncü satırda ise swap kullanımı gösterilir.

-b: Gösterim birimi byte yapılır
-m: Gösterim birimi megabyte yapılır
-h: Miktarlar insan okuması için kolaylaştırılır
-t: Dördüncğ bir satırda toplamları gösterir
-s: Sürekli güncellenen çıktı üretir. Program Ctrl-C ile sonlandırılana kadar çıktılar ekranda kayar. Parametre ile birlikte güncelleme süresi belirtilir.

```
# free –ms 5
```

Bu komutun verdiği çıktılar proc dosya sisteminde /proc/meminfo olarak bulunabilir.

uptime

Sistemin ne kadar süredir çalıştığını gösterir. Sistem yöneticileri genelde bilgileri dışında sistem kapanması olmuş mu olmamış mı bu komutla kontrol eder.

Bununla birlikte kaç kullanıcının sistemde olduğu, son 1-5-15 dakikalık işlemci yükü gibi ilave bilgiler de verir.

```
$ uptime
21:54:11 up 13 days,  4:29,  1 user,  load average: 0.21, 0.21, 0.12
```

Çıktıdaki 21:54 mevcut zamanı, 13 ise kaç gündür sistemin açık olduğunu gösterir.

screen

Metin terminal ve emülatörler için ekran yöneticidir. Tek bir pencereden bir fazla bağlantıyı yeni terminal penceresi açmaya gerek kalmadan yönetebilir. Screen genel olarak, özellikle uzak sunucu bağlantısı ile yapılan ve uzun zaman alacak işlemlerin, bağlantının kopması durumunda yarıda kesilmesine karşı çözüm olarak kullanılır.

Uzun sürecek bir işlemden önce (örneğin wget ile dosya indirme) screen komutu verilerek oturum çoğaltılır.

```
$ screen
```

Bağlantı koptuğunda screen ile yapılan işlemler yarıda kesilmez. Açılan screen penceresindeki işlem devam ederken Ctrl-a + d tuşlarına basılarak ekran çözülür (detach) ve asıl terminale geri dönülür. Bu şekilde birden fazla terminal açılarak işlemler yapılabilir.

Çözülen terminaller arka planda screen tarafından sanal olarak işletilmeye devam edilir. Kullanıcı sistemden çıkış yaparak başka işlerine devam edebilir.

Sisteme tekrar girildiğinde arkaya gönderilen terminallere tekrar erişilebilir. Birden fazla screen terminal açılmış ise aşağıdaki komut ile listesi alınır:

```
$ screen -ls
There are screens on:
     19019.pts-1.elastix   (Detached)
     19079.pts-1.elastix   (Detached)
2 Sockets in /var/run/screen/S-user1.
```

Bu terminallerden birine dönmek için screen komutuna oturum ID'sini parametre olarak vermek gerekecektir.

```
$ screen –r
$ screen –r 19079
```

Bu şekilde kapatılan terminallere yeniden ulaşılır (re-attach).

Birden fazla screen terminal açmak için iki yol vardır: 1. Yeni bir screen oluşturup çözmek ve ana terminalden yeni bir screen daha açmak. 2. Ctrl-A + c ile iç içe screen oluşturmak. Bu durumda bir sonraki terminal için Ctrl-A + n, bir önceki terminal için Ctrl-A + p tuşları kullanılır.

Screen, yapılan işlemleri loglayabilir. Bunun için Ctrl-A + H (büyük H) tuşları kullanılır. Bu komutu kullandığınızda aşağıdaki gibi bir mesaj göreceksiniz. Bu mesajda loglamanın yapılacağı dosya adı verilmektedir. Bu dosya kullanıcı ev dizinindedir.

```
Creating logfile "screenlog.0"
```

Loglamayı başlatmanın bir diğer yolu da screen komutunu –L parametresiyle çağırmaktır. Loglamayı durdurmak için tekrar Ctrl-A + H yapılır.

```
$ screen -L
```

O anki terminal görüntüsünü almak için Ctrl-A + h yapılır. Çıktı yine ev dizininde hardcopy.0 şeklinde bir dosyada saklanmaktadır.

Screen ile ekran başından ayrılacağınız zaman terminal ekranına hızlıca parola koruması koyabilirsiniz. Bunun için Ctrl-A + X (büyük X) ikilisi kullanılır. Aşağıdaki gibi parola soran bir ekran çıkartır. Tekrar parola yazılarak sisteme girilir.

```
Screen used by  <user1>.
Password:
```

Screen, herhangi bir oturumu tekrar açabilmek için (re-attach) de giriş parolasından bağımsız parola koyabilir. Bu durumda –r ile eski screen terminaline ulaşmadan önce parola sorulur.

Bunun için öncelikle $HOME/.screenrc dosyasında aşağıdaki gibi parolanın şifreli hali (crypted) olmalı:

```
password crypt_password
```

Screen'in gelişmiş bir özelliği de bir terminali başka bir kullanıcı ile paylaşabilmesidir. Böylece bir başkası terminal içerisinde yapılanları izleme şansına sahip olacaktır.

Bunun için öncelikle screen komutunun SUID bitli olması gerekmektedir.

```
# chmod u+s /usr/bin/screen
```

Screen ekranında ":" karakterine basıldığında gelen promptta aşağıdaki yazılır:
:multiuser on
:acladd user2 (user2 terminalin paylaşılacağı kullanıcı)

Diğer kullanıcı (user2) aynı sunucuya bağlanarak aşağıdaki komutla paylaşılmış terminali alır:

```
$ screen –x user1/
```

Screen ekranını sonlandırmak için iki yol bulunmaktadır. Ya Ctrl-A + d (detach) yada Ctrl-A + K (kill) kullanılır.

nohup

Verilecek komutları herhangi bir kesintiye uğramadan çalıştırmayı sağlar. Bütün sinyalleri devre dışı bırakarak çalışan sürecin sinyallerden etkilenmesini engeller. Genelde uzak bağlantılarda uzun sürecek komutların bağlantı sonlandığında bile arka planda devam etmesi için kullanılır. Kullanıcı sistemden çıksa ve terminali kapatsa bile komut çalışmasına devam edecektir. Çünkü nohup komutu terminalin ürettiği ve süreci de sonlandıracak sinyalleri engeller.

```
$ nohup komut &
```

Örneğin çok uzun sürecek bir işlem olan SUID biti set edilmiş dosyaların bulunmasını nohup ile başlatıp terminal kapatılabilir. Normalde ekrana (Stdout) basılacak çıktılar nohup.txt dosyasına yazılır. Arka planda ne olduğunu görmek için bu dosyaya bakılabilir.

```
# nohup find / -xdev -type f -perm +u=s -print &
[1] 19567
nohup: appending output to `nohup.out'
```

Nohup komutu sadece arka planda sinyallere maruz kalmadan çalışmayı engeller. Screen gibi bir terminal yöneticisi değildir.

& bg fg jobs

Bash kabuk altında çalışan bir süreç yarıda kesilebilir, arka plana gönderilebilir, tekrar önplana alınabilir veya arka planda sonlandırılabilir.

Aşağıdaki gibi uzun sürecek bir görev arka plana gönderilebilir. Arka plana göndermek için komut sonuna & simgesi eklenir.

```
# find / -ctime -1 > changed-files.txt &
```

Bir komutu çalıştırıp, bir süre izledikten sonra Ctrl-Z ile kesilip bg komutu verilerek de arka plana gönderilebilir.

```
$ wget http://www.google.com
[1]+  Stopped                 wget http://www.google.com
$ bg
[1]+ wget http://www.google.com &
```

Arkaplana giden bir iş sonlandığında konsola aşağıdaki gibi mesaj verilir:

```
[1]+  Done                    wget http://www.google.com
```

Arka plana gönderilmiş görevleri görmek için jobs komutu kullanılır:

```
# jobs
[1]   Running                 bash script.sh &
[2]-  Running                 sample-soft &
[3]+  Done                    browser .
```

Arka plandaki bir işi tekrar ön plana almak için fg komutu kullanılır. Hiçbir parametre verilmezse en son arka plana gönderilen komutu ön plana alır.

Aşağıdaki komutla 1 numaralı arka plan işi öne alınır:

```
$ fg %1
```

Arka plandaki bir işi sonlandırmak için kill komutu kullanılır. Aşağıdaki komut arka plandaki 2 numaralı işi sonlandırır:

```
$ kill %2
```

nice, renice

Bir komutu işlemci önceliğini değiştirmiş olarak başlatır.

```
# nice –n öncelik komut
```

Öncelik -20 ile 19 arasında bir tamsayıdır. -20 en yüksek öncelik demek oluyorken 19 en düşük öncelik demektir. Komut verilmez ise mevcut öncelik değerini ekrana basar.

Öncelik değeri bir değer belirtilmezse öntanımlı olarak 10 artırılır.

```
$ nice
0
$ nice nice
```

```
10
$ nice -n 10 nice
10
$ nice nice -n 3 nice
13
$ nice -n 1000000 nice
19
```

Bir sürecin öncelik değerini düşürmeyi ancak yetkili kullanıcılar yapabilir:

```
$ nice -n -1 nice
nice: cannot set niceness: Permission denied
0
$ sudo nice -n -1 nice
-1
```

Çalışan bir sürecin önceliğini değiştirmek için renice kullanılır. Parametre olarak çalışan sürecin PID numarası verilir.

```
# renice -n -19 -p 1312
```

-g: Aynı gruptaki tüm süreçler
-u: Aynı kullanıcıya ait tüm süreçler
-p: PID numarasına göre eşleşen süreçler

Aşağıdaki komut grp1 kullanıcı grubuna ait tüm süreçlerin önceliğini 5 yapar.

```
# renice -n 5 -g grp1

# renice -n 5 -u bala
```

Süreç Oluşturma ve Sonlandırma

Süreçler iki şekilde oluşturulur:

1. Çatallanma (fork): Ana süreç kendi kopyasını çıkartarak yeni iş bu kopyaya yaptırtılır. İlk çalışan süreç olan init kendi kopyasını çıkartarark yeni işleri bu kopyaya yaptırtır. Bu şekilde bir soyağacı ile yeni süreçler oluşturulur.
2. Yeni süreç başlatma (exec): exec komutuyla yeni bir süreç başlatılır. Bu süreç ana sürecin yerini alır. Komut satırından exec ile bir komut çalıştırıldığında, komut bitince tekrar komut satırına dönülmez. Çünkü yeni süreç ana sürecin yerini almıştır, artık ana süreç kalmamıştır. Buna en güzel örnek komut satırından exec komutuyla ssh komutunu çalıştırmaktır. Karşı sunuya bağlantı kuran ssh komutu exec ile çalıştırıldığından kendi ana süreci olan kabuğun yerini almıştır. Karşı taraftan exit yapılıp ssh sonlandırıldığında konsol penceresi kapanır. Çünkü artık dönülecek bir kabuk yoktur.

Bash kabuğu komut çalıştırırken fork + exec yapar. Yani kendisinin bir klonunu çıkartır ve bu klon exec yaparak yeni süreci başlatır. Yeni süreç klonun yerini alır. Dolaysıyla komut bittiğinde klon kabuk olmayacak ancak klonu çıkartan esas kabuk hala var olacak.

Ana süreç oluşturduğu alt sürecin sonlanmasını wait(2) veya waitpid(2) sistem çağrısıyla bekler. Eğer bir alt süreç sonlandığı halde ana süreci tarafından beklenmiyor ise o alt süreç zombi durumuna geçer. Sistem kaynağı tüketmez ancak çok sayıda olmaları durumunda PID numaraları karmaşıklaşacaktır. Böyle bir durumda ana sürece SIGCHLD sinyali gönderilerek alt sürecini takip etmesi istenir. Sinyal göndermek için kill komutu kullanılır. Eğer ana süreç halen daha alt sürecinin kendisine dönmesini beklemiyorsa artık ana süreç öldürülür. Bu durumda zombi süreç sahipsiz (orphan) süreç pozisyonuna geçer. Bütün süreçlerin atası olan init süreci belli aralıklarla sahipsiz süreçleri sahiplenerek yok eder.

Programlar, programcıların kodlarındaki exit fonksiyonlarıyla yada dışardan gelecek kesmelerle (interrupt) sonlanır. Programlar sonlanırken 0-255 arası bir çıkış değeri ile sonlanırlar. Bu değer programın neden çıktığını göstermesi açısından önemlidir. Genel olarak 0 ile çıkan programlar başarıyla görevini tamamlayarak, 1-255 arası bir rakamla çıkan programlar ise bir hata ile karşılaşarak sonlanmıştır demektir. Hangi kodun hangi anlam taşıdığı genel olarak programcının belgelendirmesinde belirtilmektedir.

Programlar dışardan müdahalelerle kesme (interrupt) yiyerek sonlanabilirler. Örneğin çalışan harddiskte arıza çıkması bir kesme olarak çekirdek tarafından programa iletilecektir ve program kesmeyi alınca sonlanacaktır.

Sinyaller

Kullanıcı ortamında kill komutuyla süreçlere ellen sinyal gönderilebilir. Sinyallerin her birinin özel bir anlamı olup sürecle haberleşmeyi sağlar. Örneğin bir süreç SIGFPE (Floating Point Error) almışsa sıfıra bölmek gibi anlamsız bir işlem yaptığını haber verir.

Sinyaller eğer program tarafından yakalanmazsa genellikle programın aniden sonlanmasına neden olur. Bu nedenle programcılar kendilerine gelen sinyali yakalayıp programı düzgün bir şekilde kapatırlar. Sinyali işleyen bu kısma programcılıkta sinyal işleyici (signal handler) denir.

Aşağıdaki tabloda sinyallerin bir kısmı verilmektedir:

Sinyal	Değer	Aksiyon	Açıklama
SIGHUP	1	Term	Terminali askıya al yada süreci sonlandır
SIGINT	2	Term	Klavyeden gelen kesme
SIGQUIT	3	Core	Klavyeden gelen çıkış kesmesi
SIGILL	4	Core	Tanımsız CPU işlemi
SIGABRT	6	Core	Abort
SIGFPE	8	Core	Matematik işlemci hatası
SIGKILL	9	Term	Süreci öldür
SIGSEGV	11	Core	Geçersiz bellek adreslemesi
SIGPIPE	13	Term	Okuyucusu olmayan kırık pipe'a yazma hatası

```
SIGALRM       14        Term    Zamanlama sinyali
SIGTERM       15        Term    Sonlandırma sinyali
SIGUSR1    30,10,16     Term    Programcının tanımladığı özel sinyaller
SIGUSR2    31,12,17     Term    Programcının tanımladığı özel sinyaller
SIGCHLD    20,17,18     Ign     Alt süreç durdu veya sonlandı
SIGCONT    19,18,25     Cont    Durmuşsa devam et
SIGSTOP    17,19,23     Stop    Durdur
```

Listede tanımlanmış aksiyonlar:

Term: Süreci sonlandır
Ign: Sinyali gözardı et
Core: Süreci sonlandır ve bellek dump'ı al
Stop: Süreci durdur

Kill, killall

Komut satırından ellen süreçlere sinyal göndermek için kill komutu kullanılır.

Aşağıdaki komut 1120 PID numaralı sürece 9 numaralı (SIGKILL) sinyali gönderir. Dolaysıyla süreci öldürür.

```
$ kill -9 1120
```

Linux altında bir sürecin yapılandırma dosyasını yeniden okuması için normalde sürecin sonlandırılıp yeniden başlatılması gerekmektedir. Süreci sonlandırmadan bunu yapmanın yolu çalışan sürece SIGHUP göndermektir. Bu sinyali alan sürec kendini askıya alır ve aynı PID numarası içerisinde çalışmasına en baştan başlar.

```
# kill -1 1531
# killall –HUP httpd
```

killall komutu PID numarasına bakmadan ismiyle sürece sinyal göndermek için kullanılır.

Pkill, pgrep

Sürecin adı veya sahibi gibi değişik pattern'ler verilerek belirtilen sürece sinyal göndermek için pkill kullanılır.

Aşağıdaki komut user4 kullanıcısının sahip olduğu tüm süreçleri sonlandırır.

```
# pkill –STOP –u user4
```

Aşağıdaki komut ise adında http geçen bütün süreçleri sonlandırır.

```
# pkill http
```

Verilen kritere hangi süreçlerin uyduğunu görmek için pgrep komutu kullanılır. Böylece verilen komuttan etkilenecek süreçlerin istenen süreçler olduğundan emin olunur.

Örnek:

```
# pgrep –u user4
6081
6082
# pkill –u user4
```

veya

```
# pgrep http
3077
3531
3532
3533
3534
# pkill http
```

Dosya Düzenleyiciler

- **Konu:**

Linux altında dosya düzenleme

- **Hedefler:**
  - nano
  - vi
  - Geliştiricilere özel vi kullanımı

Anahtar Kelimeler:

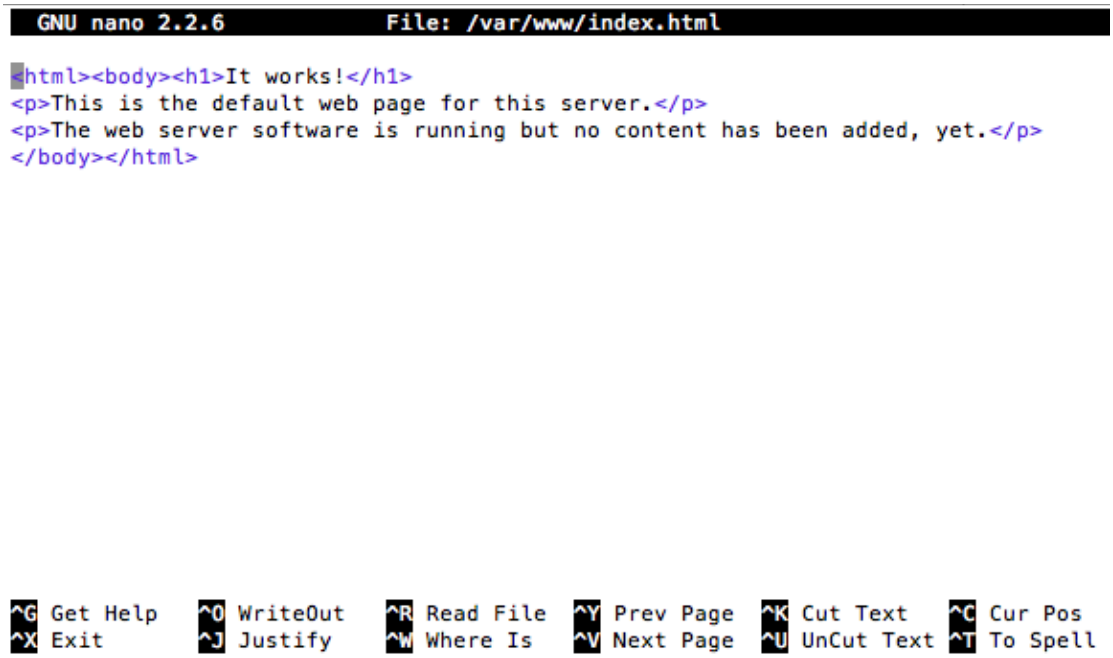
nano, vi

Nano

Nano, Pico'nun kullanıcı dostu kabiliyetlerini sunacak ancak daha basit ve küçük bir dosya düzenleyicisi olması amacıyla geliştirilmiştir.

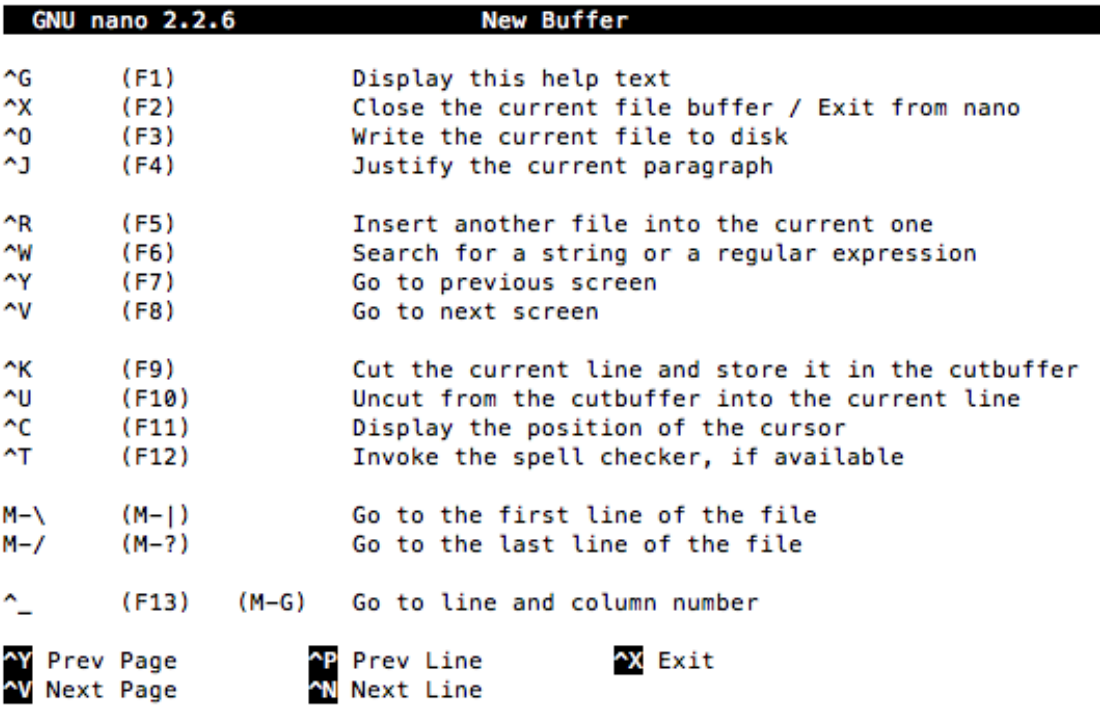
Aşağıdaki gibi çağırılır:

```
$ nano /var/www/index.html
```



nano komutuyla beraber var olan bir dosya adı verilirse o dosyayı açar. Olmayan bir dosya adı verilirse o isimde dosya oluşturulur. Hiçbir isim verilmezse yeni ve boş bir dosya açar ve kaydederken ismini sorar.

Nano içinde işlemlerin çoğu kısayollarla yapılmaktadır. Bu kısayolların önemli olanları nano ekranının altında gözükmektedir. Ancak burada gösterilenden daha fazla kısayol bulunmaktadır. Ctrl-G tuş kombinasyonu ile yardım ekranına ulaşıp kısayolların tamamı hakkında bilgi alınabilir.



Bazı kısayolların açıklamaları:

Ctrl-O: Dosyayı kaydetmek için kullanılır. Kaydedilecek dosya ismini sorar. Grafik düzenleyicilerde kullanılan Save As benzeri bir işlevi var.

Ctrl-R: Başka bir dosyanın içeriğini mevcut dosyanın içine aktarır. İçe alınacak dosyanın yolunu sorar.

Ctrl-C: O an yapılan işlemi kesmek için kullanılır.

Ctrl-X: Dosyadan çıkmak için kullanılır. Eğer dosya içeriği değiştirilmişse içeriğin kaydedilip edilmeyeceğini soracaktır.

Ctrl-A: Satır başına gider (Home tuşu).

Ctrl-E: Satır sonuna gider (End tuşu).

Ctrl-V: Aşağı doğru sayfa sayfa gider.

Ctrl-Y: Yukarı doğru sayfa sayfa gider.

Ctrl-Space: Kelime kelime dosya içinde ilerler.

Vi Dosya Düzenleyici

Genel

Vi editörü tüm UNIX ve Linux dağıtımlarıyla birlikte gelen ve yaygın bir kullanıcı kitlesi olan bir metin editörüdür. Komut satırından çalışmaktadır ve dosya oluşturmak, var olan dosyada değişik yapmak için kullanılır.

Vi içerisinde üç alan vardır:

: ile başlayan Ex komutlarının verildiği satır alanı. (:q :wq :x gibi)

Yazı alanında ESC ile geçilen ve karakterlerin özel fonksiyon icra ettiği komut modu.

Yazı alanı. Karakterlere basınca istenilen metnin yazıldığı alan.

Komut modunda editöre sadece komutlar gönderilir. Verilen komutlar ekranda yazı olarak gözükmez. Bu komutlar satır, kelime silme, dosyada aşağı,yukarı, sağa, sola hareket etme, dosyada kelime arama vs amaçlı verilebilir.

Yazı modunda ise basılan her karakter ekrana veri olarak yazılır. Yazı modundan komut moduna geçmek için ESC tuşu kullanılır. Fakat komut modunda iken bir başka komut çalıştırmak için ESC tuşuna basmaya gerek yoktur.

Genel kullanım biçimi aşağıdaki gibidir.

```
# vi [parametreler] dosyaadı
```

Dosya açmak için

```
# vi dosyaadı
```

Bu komutla istediğiniz isimde bir dosya vi ile açılmış olur. Bu komut aynı zamanda sizi, bu dosyayı düzenlemek üzere vi dosya düzenleyicisinin içine sokar. Verdiğiniz dosya diskte yoksa ve yeni oluşturulacak aşağıdaki gibi bir bilgilendirme ile karşılaşırsınız:

```
# vi deneme
~
~
~
deneme: new file: line 1
```

Buradaki ~ ifadesi boş satırları ifade eder. Satırlara yazı yazıldıkta bu ifade kaybolur. En altta da dosyanın adının deneme olduğu ve yeni oluşturulan bir dosya olduğu belirtiliyor. Yukarıda belirtildiği gibi vi şu anda komut modundandır. Bu durumda iken i tuşuna basılarak yazma moduna geçilir ve yazılmak istenen metin yazılır.

Dosyadan Çıkmak

Öncelikle komut moduna geçmek için ESC tuşuna basılır. Komut modunda : (iki nokta üst üste) karakterine basarak nasıl çıkış yapacağınızı belirtebilirsiniz:

Kaydederek çıkmak için:
:wq!

Kaydetmeden çıkmak için:
:q!

Çalışma Modları

Vi düzenleyicisi iki çalışma moduna sahiptir. Birincisi, basılan her tuşun özel bir işleve sahip olduğu komut modudur. Diğeri ise basılan her tuşun dosyaya aynı şekilde yazıldığı yazma modudur.

Komut modunun kullanım biçimi:

[adet] komut [nereye]
Köşeli parantez içindekiler seçimliktir. Çoğu komut tek karekterden ibarettir.

Adet seçeneği, o komutun kaç kere yerine getirileceğini belirtir. Örneğin x komutu karakter silmek içindir. 15x komutu verilirse 15 karakter silinir.

Bazı komutlar, komutların kaç satıra ve nereye kadar uygulanacağını belirtmek üzere ilave bir seçenek daha alırlar.

Basit Vi Komutları

a

Ekleme moduna geçer. İmleçin bulunduğu pozisyondan itibaren basılan karakterler dosyaya yazılır.

h

İmleci bir karakter sola taşır.

j

İmleci bir satır aşağı alır.

k

İmleci bir karakter yukarı alır.

l

İmleci bir karakter sağa alır.

r

İmlecin altındaki karakteri basılan karakter ile değiştirir.

u

En son yapılan değişikliği geri alır (undo). Tekrar basılırsa yapılan değişikliği yeniden yapar. (redo)

x

İmlecin altındaki karakteri siler.

Kesme ve Yapıştırma Komutları

Kesmek için kullanılan komut genelde d komutudur.

d^

İmlecin konumundan satır başına kadar siler.

d$

İmlecin konumundan satır sonuna kadar siler.

dw

İmlecin konumundan kelime sonuna kadar siler.

3dd

İmlecin konumundan itibaren 3 satır siler.

Kesilen verileri yapıştırmak için p veya P komutu kullanılır. P komutu imlecten geriye yapıştırırken, p komutu imlecin önüne yapıştırır.

Kopyalama içinse aşağıdaki komutlar kullanılır.

y Yakalamak (yank) manasına gelir. Tek bir karakteri kopyalar.

yy İmlecin olduğu satır kopyalanır. Yine p komutu ile yapıştırılır.

Örneğin

Aşağıdaki dosyanın ilk satırını dosyanın sonuna kopyalamak için birinci satırda iken yy komutu verilir dosyanın sonuna :$ ile geçilir ve p komutu verilir.

```
Bu bir deneme metnidir.

Bu bolumde vi editoru anlatilacaktir.
~
```

Sonuç aşağıdaki gibi olacaktır:

```
Bu bir deneme metnidir.

Bu bolumde vi editoru anlatilacaktir.
Bu bir deneme metnidir.
~
```

Dosyanın en son satırında iken p yerine P komutu verilseydi sonuç aşağıdaki gibi olacaktır:

```
Bu bir deneme metnidir.
```

```
Bu bir deneme metnidir.
Bu bolumde vi editoru anlatilacaktir.
~
```

Hata yapıldığında yapılan değişiklik u komutu ile geri alınır. Bu komut ile sadece son komutta yapılan değişiklikler geri alınır. Tekrar u komutu verilirse yapılan değişiklik tekrar gelir (redo).

Arama

Vi düzenleyici iki çeşit arama imkanı sunar: Kelime arama ve karakter arama.

Kelime arama için / ve ? Komutları kullanılır. Bu iki komuttan herhangi birine bastığınızda aramak istediğiniz kelimeyi girmeniz istenecektir. Her ikisi de arama yaparak bulduğu ilk kelimenin üzerine konumlanır. Aralarındaki tek fark, / komutu ileri doğru arama yaparken, ? Komutu geri doğru arama yapar. N ve n komutları, bir önceki komutu ters yönde veya aynı yönde tekraren arama yapmayı sağlar.

Not: Vi için özel anlamı olan karakterleri aramak için \ işareti ile korumaya almak gerekir.

Özel karakterler:

^
: Satır başı

.
: Tek bir karakterle eşleşir.

*
: Bir önceki karakterden sıfır veya daha fazla adet eşleştirir.

$
: Satır sonu

[

İfade yazmak için kullanılır.

< >

Kelimenin başını veya sonunu bulmak için kullanılırlar.

Harf aramak için f ve F komutları kullanılır. Harf arama yalnızca üzerinde bulunulan satırda yapılır. F komutu geri doğru ararken, f komutu ileri doğru arar.

Metinde geçen bir kelimeyi değiştirmek için :%s komutu kullanılır. Komutun kullanım şekli aşağıdaki gibidir.

:s/eski/yeni Sadece imlecin bulunduğu satırdaki eski değeri yeni olarak değiştirir.

:%s/eski/yeni/g Tüm metin içerisinde değişiklik yapar. g değerinin yanına c değeri de eklenerek :%s/eski/yeni/gc) değişiklik yapılmadan önce onay alınabilir.

Aslında buradaki eski ve yeni değerleri birer düzenli ifadedir (regular expression). Düzenli ifadeler kullanılarak ileri seviye değişiklikler yapılabilir.

Örneğin aşağıdaki metindeki Pazar ifadesini Pazartesi yapmak için:

```
Bu Pazar Pardus toplantisi yapilacaktir.
Pazar gunu gorusmek uzere....
~
```

`:%s/Pazar/Pazartesi/g`

```
Bu Pazartesi Pardus toplantisi yapilacaktir.
Pazartesi gunu gorusmek uzere....
~
```

Geliştiricilere Özel

Vi düzenleyicisi programcılara yardımcı olacak pek çok özelliğe sahiptir. Bunlardan birisi de kod bloklarının içeri atılması veya dışarı çekilmesidir. Bu şekilde kodlarınızı daha okunaklı hale getirebilirsiniz.

<<

Satırı belirlenen kaydırma genişliği kadar sola kaydırır.

>>

Satırı belirlenen kaydırma genişliği kadar sağa kaydırır.

Kaydırma genişliğini tanımlamak için:

```
:set sw=4
```

Satırlar aşağıdaki şekilde numaralandırılır:

```
:set nu
```

komutu verildiğinde ekran aşağıdaki gibi gözükecektir:

```
1 Bu bir deneme metnidir.
2
3 Bu bolumde TUBITAK yayinlarindan cikan Pardus kitabi
hakkinda bilgi verilecektir.
~
~
```

Satırları tekrar numarasız hale getirmek için de

`:set nonu` veya `:set nonumber`

komutu verilir.

Vi düzenleyicisinin geliştiriciler için sunduğu özelliklerden biri de unutulan parantezleri kontrole imkan tanımasıdır. % komutu, üzerinde durduğu sağa doğru olan açma parantezi için uygun bir sola doğru kapatma parantezi bulup üzerinde konumlanır.

BASH Betik Yazma

- **Konu:**

Bash kabuk programları (betik) yazmak

- **Hedefler:**
  - Bash açılış yapılandırma dosyaları
  - Kabuk programların genel yapısı
  - Kabuk programları çalıştırma

- Global ve yerel çevre değişkenleri
- Fonksiyonlar
- Şartlı ifadeler
- Dosya ve değişken test etme
- Döngüler

Anahtar Kelimeler:

source, set, unset, /etc/profile, .bashrc, echo, read, alias, if, case, while, for, until

Kabuk Ortamı

Bash ilk başladığında eğer varsa /etc/profile dosyasını okur. Daha sonra kullanıcı ev dizinindeki ***~/.bash_profile***, ***~/.bash_login*** ve ***~/.profile*** dosyaları okunur. Bu dosyalarla kullanıcılar kendi ortamlarını özelleştirebilir.

Profile dosyaları çevre değişkenlerinin ayarlanması için, rc dosyaları ise alias ve fonksiyon tanımları için kullanılır.

Bash'den çıkış yapılırken eğer varsa ***~/.bash_logout*** çalıştırılır.

/etc/profile dosyasında yapılan ayarlar:

- öntanımlı komut çalıştırma yolu (PATH)
- Prompt ayarları (PS1, PS2)
- Öntanımlı kullanıcı sınırlandırmaları (ulimit)

Başka yazılımların da profile tanımı yapabilmeleri için /etc/profile.d dizini tanımlanmıştır. Böylece yüklenen programlar kendi profile tanımlarını sistemin genel profile dosyasını etkilemeden bu dizine koyabilirler.

```
# ls -l /etc/profile.d/
toplam 4
-rw-r--r-- 1 root root 660 Haz 17  2012 bash_completion.sh
```

Kabuk altında bir komut verildiğinde kabuk önce kendi klonunu oluşturur ve komutu bu klon çalıştırır. Böylece klon kabuk isteği yerine getirirken kabuğun kendisi yeni istekleri almaya devam edebilir. Bu çalışma şeklinde alt kabukların çalışma ortamında yaptığı değişiklikler ana kabuğu etkilemeyecektir. Alt kabuk sonlanır sonlanmaz ortamları da silinecektir. Eğer bir betiğin mevcut kabuk ortamında çalıştırılması isteniyorsa nokta (.) veya source komutu kullanılır.

```
$ . script.sh [parametre]
$ source script.sh [parametre]
```

Bu şekilde başlatılan bir betik mevcut kabuk içinde çalıştırır ve betiğin oluşturduğu değişkenler, betik sonlandıktan sonra da kullanılabilir.

Örneğin bir kullanıcı kendi ***~/.profile*** dosyasında değişkenler tanımlayıp bunları yürürlüğe koyması için mevcut kabuk altında çalıştırması gerekir. Yeni bir kabuk altında çalıştırırsa değişkenler geçerli olamayacaktır.

```
$ source ~/.profile
```

Kabuğun başlama şekline göre farklı dosyalar kullanılır: Interaktif ve giriş kabukları, interaktif ve giriş olmayan kabuklar, interaktif olmayan kabuklar.

Interaktif ve giriş kabuğu: Başarılı bir giriş işleminden sonra başlatılan kabuk. Terminalden veya uzaktan sisteme giriş, su – komutuyla giriş. Bu durumda /etc/profile ve ***~/.bash_profile*** okunur.

Interaktif ve giriş olmayan kabuk: Komut satırından kabuk başlatılarak (kabuk içinde bash komutunu vererek) veya su komutunu – olmadan çalıştırarak başlayan kabuktur. Grafik arayüzden başlatılan xterm konsole programları da bu sınıfa girer. Buna alt kabuk da denilebilir. Böyle bir kabuk ana kabuğun ortamını kopyalar ve ***~/.bashrc*** dosyasını çalıştırır.

Interaktif olmayan bir kabuk, betik çalıştıran kabuktur. Bu durumda kullanıcıdan giriş beklenmez.

Bir kabuğun interaktif olup olmadığını anlamanın iki yolu vardır:

$- özel değeri kontrol edilir, i parametresini içeriyorsa interaktif kabuktur.
$PS1 prompt değişkeni atanmış ise interaktif kabuktur.

Bash yapılandırma dosyaları en çok çevre değişkeni tanımlama ve alias tanımlama için kullanılır.

Kabuk Değişkenleri

Kabuk değişkenleri anahtar-değer ikilisi şeklindeki değişkenlerdir. Tanımlı bütün değişkenlerin listesi için:

```
# set
```

```
BASH=/bin/bash
BASH_VERSION='3.2.25(1)-release'
COLORS=/etc/DIR_COLORS
COLUMNS=125
[...]
```

Değişkenler üç yolla atanır:

1. Ana kabuktan miras gelen değişkenler
2. /etc/profile, .bash_profile, .bashrc gibi başlangıç dosyalarıyla atanan değişkenler
3. Kullanıcı tarafından elle komut satırından tanımlanan değişkenler

Değişkenleri göstermek için:

```
$ echo $VAR
$ echo $PS1
[\u@\h \W]\$
```

Aşağıda bir değişken tanımlanmakta ve sonra başka bir kabuğa geçilip bu değişken değeri gösterilmekte ancak alt kabukta değişken değeri görülmemiştir.

```
$ FILE=/tmp/x.out
$ echo $FILE
/tmp/x.out
$ bash
$ echo $FILE

$
```

Tanımlanan değişkenler kendi kabukları için geçerlidir. Başka bir kabuğa geçiş yapıldığında global olmayan değişkenler geçerli olmazlar. Global olacak çevre değişkenleri export ile tanımlanır.

```
$ export FILE=/tmp/x.out
$ bash
$ echo $FILE
/tmp/x.out
```

Tanımlı değişkeni silmek için unset komutu kullanılır:

```
$ unset FILE
```

Çevre değişkenlerini görüntülemek için env komutu kullanılabilir:

```
# env
HOSTNAME=beta.pardus.org.tr
SHELL=/bin/bash
HISTSIZE=1000
```

```
USER=root
MAIL=/var/spool/mail/root
```

Bir komut çalıştırılırken çevre değişkeni tanımlanabilir. Örneğin crontab düzenlenmek istendiğinde EDITOR değişkeninde tanımlı düzenleyici ile dosyayı açar. Normalde EDITOR olarak vi dosya düzenleyicisi tanımlıdır. Crontab'ı nano düzenleyicisi ile açmak için EDITOR değişkeni komut satırından atanabilir:

```
# EDITOR=nano crontab -e
```

Normalde çevre değişkenler alt süreçlere de geçer. Belli bir değişkenin alt sürece geçmesini engellemek için:

```
$ env -u SSH_AGENT_PID ssh user1@pardus
```

Bütün çevre değişkenlerini temizleyerek alt süreci başlatmak için:

```
$ env -i command
```

Alias Tanımlama

Alias, bir komutun yerini alacak kısayoldur. Genel olarak .bashrc dosyasında tanımlanırlar. Aşağıdaki gibi tanımlanır:

```
$ alias ll="ls -l"
$ ll
toplam 0
-rw-r--r-- 1 simsek simsek 0 Ara 15 18:32 config.sh
-rw-r--r-- 1 simsek simsek 0 Ara 15 18:32 dosyalar.txt
```

Uzun uzun ls –l yerine artık sadece ll komutu kullanılabilir.

Bir alias'ı silmek için unalias komutu kullanılır:

```
$ unalias ll
$ ll
-bash: ll: command not found
```

Fonksiyon Tanımlama

Fonksiyonlar belirli görevleri yapan betik program parçalarıdır.

```
myecho () {
    echo "> $1"
}
```

Bu fonksiyon tanımlandıktan sonra daha sonra betik içinden şu şekilde çağrılır:

```
myecho "hello world"
```

Fonksiyon çağrılırken verilen parametreler $1 (1.parametre), $2... şeklinde fonksiyon içinde kullanılabilir.

Bash Kabuk Programlama

Sistem yöneticileri ve işletmecileri sürekli olarak yaptıkları işleri kabuk program haline getirerek hem zamandan tasarruf ederler, hemde işlerini kolaylaştırırlar. Kabuk programlar, tekrar tekrar yapılan işlerin bir dosyaya sıralı olarak yazılmasıyla oluşturulurlar.

Betikler C programlardaki gibi derlemeli değil yorumlamalıdır. Betik dosyası satır satır kabuk programı tarafından yorumlanır.

Aşağıda çok basit bir betik görülmektedir. Bu betiği bir dosya düzenleyici ile yazıp merhaba.sh diye kaydedin.

```
#!/bin/bash

echo "Merhaba dünya."
echo "Bu benim ilk betiğim."
exit 0
```

```
# chmod +x merhaba.sh
# ./merhaba.sh
Merhaba dünya.
Bu benim ilk betiğim.
```

Betik dosyasının .sh şeklinde bir uzantısı olma zorunluluğu yoktur. Programın doğru olması ve dosyanın çalışabilir olması yeterlidir. Ancak diğer dosyalardan ayırt etmek için .sh uzantısı vermek yararlı olacaktır.

İlk satırda yer alan `#! /bin/bash` bu betiği yorumlayacak programın tam yoludur. Örnek bash betiği olduğu için 'bash' in yolu yazıldı. Eğer bu bir perl betik olsaydı oraya `#!/usr/bin/perl` yazmak gerekecekti.

Zorunluluk olmamakla birlikte, bütün betiklerin bir çıkış değeri ile 'exit' yapması tavsiye edilir. Programların hangi durumda çıktığı bu çıkış değerlerine bakarak anlaşılır.

En son komutun çıkış değerini elde etmek için:

```
$ echo $?
1
```

Betik dosyalarında açıklama yapmak için # işareti kullanılır. Satırdaki # işaretinden sonrası kabuk tarafından yorumlanmaz. Aşağıdaki betikte ilk satır ve ikinci satırdaki # işareti ve sonrası sistem yöneticisinin kendi için düştüğü bir açıklama olup kabuk tarafından yorumlanmadan geçilir.

```
# Print filename
echo $filename # filename in /etc/
```

Parametreler

Komut satırından çalıştırılacak betiğin ardına parametreler eklenerek betiğin davranışı değiştirilebilir.

```
$ komut param1 param2 param3
```

Betik içerisinde betiğe gönderilen parametrelerin alınması ve yorumlanması gerekir. Bu şekilde parametrelere göre betik iş yapar. Komut satırından gönderilen parametreleri almak için $0, $1... $N şeklinde özel değişkenler kullanılır.

```
$ cat param.sh
#!/bin/bash

echo "parametre 0: $0"
echo "parametre 1: $1"
echo "parametre 2: $2"

echo "Toplam parametre sayisi: $#"
echo "Parametre listesi: $@"

$ ./param.sh bir iki
parametre 0: ./param.sh
parametre 1: bir
parametre 2: iki
Toplam parametre sayisi: 2
Parametre listesi: bir iki
```

İlk parametre $0 her zaman programın kendi ismini verir.
Diğer parametreler $1 $2 $3 ... $10... şeklindedir.

Programa gönderilen parametre sayısı özel olarak $# değişkeninde tutulur.
Parametre listesi topluca $@ özel değişkeninde tutulur.

Okuma Yazma

Programlarda olduğu gibi betiklerde de asıl amaç kullanıcı girdilerini alıp elde edilen sonuçları ekrana basmaktır.

Betik çalışıyorken kullanıcıdan girdi istemek için read kullanılır.

```
echo –n “Adinizi girin: “
read name
```

Kabuk read satırına geldiğinde kullanıcıdan bir girdi bekler. Kullanıcı girdi yapıp ENTER’a bastığında bu girdi name değişkenine aktarılır.

Read komutunun diğer parametreleri:

-t 10: Girdi için en fazla 10 sn beklenecek
-n 8: 8 karakter girdi alınacak

Ekrana metin veya değişken değerlerini basmak için echo kullanılır.

```
echo $name
```

Echo normalde satır satır ekrana basar. Yani her echo ‘dan sonra bir sonraki satıra devam eder. Aynı satırda devam etmesi için –n parametresi kullanılır.

```
$ cat inout.sh
#!/bin/bash

echo –n "Adinizi girin: "
read name
echo "Adiniz: $name"
$ ./inout.sh
Adinizi girin: Ali
Adiniz: Ali
```

Echo –e seçeneği ile özel karakterleri basabilir:

\0NNN: NNN referans numarali ASCII karakter (man 7 ascii)
\\ Backslash
\a Alert (Bip sesi)
\b Backspace
\n Newline
\r Carriage return
\t Tab

Şartlı İfadeler

Bir işlemi yapıp yapmayacağına karar vermek için şartlı ifadeler kullanılır. Şartlı ifadeler şu şekilde bina edilir:

```
if [ şartlı ifade ]
then
      İfade doğru olduğunda çalışacak kısım
      command 1
      command 2
      ...
else
   command 3
   command 4
   ...
fi
```

Veya basitçe şu şekilde de kullanılabilir:

```
if [ şartlı ifade ]; then
      İfade doğru olduğunda çalışacak kısım
      command 1
      command 2
fi
```

Örnek 1:

```
echo "Testing \"0\""
if [ 0 ]      # zero
then
  echo "0 is true."
else          # Or else ...
  echo "0 is false."
fi            # 0 is true.
```

Örnek 2:

```
echo "Testing \"1\""
if [ 1 ]      # one
then
  echo "1 is true."
else
  echo "1 is false."
fi            # 1 is true.
```

Örnek 3:

```
# cat sartli.sh
#!/bin/bash

SAYAC=6
LIMIT=5
```

```
if [ $SAYAC -lt $LIMIT ]; then
        echo "Sayac limitten küçük"
else
        echo "Sayac limitten küçük değil"
fi

exit 0

# ./sartli.sh
Sayac limitten küçük değil
```

Sayac=6 ve Limit=5 olduğundan sayaç limitten küçük değildir. Buradaki şartların anlamları şöyledir:

-lt
: Less than: Daha küçük ise.

-gt
: Great than. Daha büyük ise.

=
: Eşit ise.

Stringler ve değişkenler aşağıdaki gibi kıyaslanır:

```
if [ "$distri" = "Pardus" ] then
       ...
else
       ...
fi
```

Birden fazla şartlı durumlar için aşağıdaki gibi if-elif yapısı kullanılır:

```
if [ condition1 ]
then
   command1
   command2
   command3
elif [ condition2 ]
then
   command4
   command5
else
   default-command
fi
```

Kıyaslamak için test şartları kullanılabilir. Pek çok test şartı vardır. Bazıları:

[-b file]
: Dosya var ve blok dosya ise.

[-c file]

Dosya var ve karakter dosya ise.

[-d path]

Belirtilen path var ve dizin ise.

[-e file]

Eğer dosya var ise.

[-f file]

Eğer dosya var ve normal dosya ise.

[-z string]

String boyu 0 ise.

[-x file]

Dosya var ve çalıştırılabilir ise.

[string]

String boş değil ise.

[s1 = s2]

s1 ve s2 stringleri birbirinin aynı ise.

[s1 != s2]

s1 ve s2 stringleri birbirinin aynı değil ise.

[s1 < s2]

Karakter sıralamasına göre s1, s2'den önce geliyor ise.

[s1 > s2]

Karakter sıralamasına göre s1, s2'den sonra geliyorsa.

[n1 -eq n2]

n1 ve n2 tamsayıları eşit ise.

[n1 -ne n2]

n1 ve n2 tamsayıları farklı ise.

[n1 -gt n2]

n1, n2'den büyük ise.

[n1 -ge n2]

n1, n2'den büyük veya eşit ise.

[n1 -lt n2]

n1, n2'den küçük ise.

[n1 -le n2]

N1, n2'den küçük veya eşit ise.

Örnek:

```
file=/etc/passwd

if [[ -e $file ]]
then
  echo "Password file exists."
fi
```

! simgesi ifadenin değili alınabilir.

-a ile iki ifade AND işlemine tabi tutulabilir.

-o ile iki ifade OR işlemine tabi tutulabilir.

Case-Esac Yapısı

Şartlı ifadeler için daha anlaşılır ve okunaklı bir yapı sunar. Belirtilen değişkenin değerine göre bloklardan biri çalıştırılır. Aşağıda Linux altındaki pekçok start/stop betiğinin kullandığı case yapısı örneği görülmektedir.

```
case "$1" in
      start)
            start
            ;;

      stop)
            stop
            ;;

      restart)
            stop
            start
            ;;
      *)
            echo $"Usage: $0 {start|stop|restart}"
            exit 1
esac
```

En sondaki * şartının olduğu blok yukarıdaki hiç bir şart sağlanmazsa çalıştırılacak bloktur.

Döngü Kurmak

Gelişmiş programlama dillerinde kullanılan döngüleri Bash içerisinde de kullanma imkanı vardır. Bu sayede bir algoritma çercevesinde yüzlerce adımdan oluşan bir hesabı tek bir döngü ile yapabiliriz.

Bash içerisinde üç tane döngü ifadesi vardır: for, while ve until.

for ... done

Programlama dillerindeki for döngüsünden farklı olarak seri halindeki kelimeler üzerinde hareket eder.

```
# cat for.sh
#!/bin/bash

for i in $( ls ); do
    echo "File: $i"
done

exit 0
```

```
# ./for.sh
File: secret.txt
File: sartli.sh
File: zZZ.txt
```

For ifadesi, ls komutunun ürettiği kelime katarı üzerinde kelime kelime hareket ederek döngü yapmıştır. Bu nedenle dosya sayısı kadar döngü çalıştırılmıştır.

Döngü for ile başlayıp done ile biter. Kabuk, done özel kelimesini gördüğünde kelime katarından bir sonraki kelimeyi alarak döngüye yeniden girer. Bu işlem katardaki kelimeler bitene kadar devam eder.

while … done

Şartlı döngü sağlar. While sözcüğü ile belirtilen şart sağlandıkça döngü devam eder.

```
# cat while.sh
#!/bin/bash

COUNTER=0
LIMIT=5

while [ $COUNTER -lt $LIMIT ]; do
    echo "Sayac: $COUNTER"
    let COUNTER=COUNTER+1
done

exit 0

# ./while.sh
Sayac: 0
Sayac: 1
Sayac: 2
Sayac: 3
Sayac: 4
```

Buradaki şart [$COUNTER -lt $LIMIT] ‘dır; yani birer birer artan sayaç LIMIT’den (burada limit 5) küçük oldukça dön.

Döngü while ile başlayıp done ile biter. Kabuk done sözcüğünü gördüğünde belirtilen şartın hala geçerli olup olmadığına bakar. Eğer geçerli ise yeniden döngüye girer. Aksi halde döngüden çıkıp devam eder.

Not: let komutu COUNTER değerini her defasında 1 artırmak için kullanılmıştır.

until … done

Olumsuz şartlı döngü kurmak için kullanılır. Yani belirtilen şart geçerli olmadıkça döngüye girer:

```
# cat until.sh
#!/bin/bash

COUNTER=10
LIMIT=5

until [ $COUNTER -lt $LIMIT ]; do
    echo "Sayac: $COUNTER"
    let COUNTER=$COUNTER-1
done

exit 0

# ./until.sh
Sayac: 10
Sayac: 9
Sayac: 8
Sayac: 7
Sayac: 6
Sayac: 5
```

Belirtilen şart sayacın limitten küçük olması şartıdır. Until döngüsü olduğu için kabuk bunu “sayaç limitten küçük olmadığı sürece dön” şeklinde yorumlayacaktır. While olsa idi “sayaç limitten küçük olduğu sürece dön” yorumunu yapacaktı.

Sayaç 10’dan başlayıp let ile birer birer azaltılmaktadır. Limit ise 5’dir. Dolaysıyla 5’den küçük olmadıkça dönecektir. 5’den küçük olduğu an döngü kesilecektir. Komutun çıktısından da anlaşılacağı üzere dörde kadar döngüye girecektir. Sayaç 4 olduğunda döngüden çıkacaktır.

SQL ile Veri Yönetimi

- **Konu:**

SQL dili

- **Hedefler:**
  - Veritabanı ve tablo şeması
  - Temel SQL ifadeleri

Anahtar Kelimeler:

Insert, Update, Select, Inner Join, Delete, Order By, Group by

SQL Veritabanları

SQL (Structured Query Language), ilişkisel veritabanı sunucularında verilerin yönetilmesi için geliştirilmiş amaca özel programlama dilidir. En temel amacı, veri sorgulamak ve veri yazmaktır.

Linux dünyasında kullanılan pek çok açık kaynak kodlu ve ticari veritabanı sunucusu bulunmaktadır. Bunlardan en çok kullanılanları:

- SQLite
- PostgreSQL
- MySQL

Bütün veritabanı sunucuları standart bir SQL dili kullanarak veri üzerinde çalışırlar.

SQL dili belli amaçları olan SQL ifadelerinden oluşmaktadır.

Veritabanı üzerinde çalışmak için veritabanının oluşturulmuş ve veritabanı şemasının tanımlanmış olması gerekmektedir. Veritabanı birden fazla tablodan oluşur. Örneğin bir okul veritabanı için öğrenciler tablosu, öğretmenler tablosu, demirbaş tablosu gibi farklı alanlar içeren (örneğin öğrenciler için öğrenci nosu gerekirken öğretmenler için TC No'su veya kıdemi gerekli) tablolar tanımlanabilir.

Veritabanı şeması, hangi verilerin ne türde ve ne boyutta saklanacağını belirler. Aşağıda örnek bir şema bulunmaktadır:

```
mysql> DESCRIBE pet;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | YES  |     | NULL    |       |
| owner   | varchar(20) | YES  |     | NULL    |       |
| species | varchar(20) | YES  |     | NULL    |       |
| sex     | char(1)     | YES  |     | NULL    |       |
| birth   | date        | YES  |     | NULL    |       |
| death   | date        | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
```

Field, saklanacak verinin alan adı
Type, saklanacak verinin türü ve boyutu
Null, verinin boş olup olamayacağı
Key, bu alanın anahtar olup olmadığı

Insert

Var olan bir veritabanı tablosuna veri eklemek için kullanılır. Hangi alana hangi değerin yazılacağı belirtilir.

```
INSERT INTO tbl_name () VALUES();
INSERT INTO tbl_name (col1,col2) VALUES(15,col1*2);
```

Burada tbl_name tablonun adı, tablodaki col1 alanı yada sütunu için 15 değeri, col2 alanı için ise col1*2 değeri yazılır.

Aşağıdaki örnekte ise aynı tabloya 3 satır eklenmektedir:

```
INSERT INTO tbl_name (a,b,c) VALUES(1,2,3),(4,5,6),(7,8,9);
```

Aşağıdaki INSERT yapısı hata verecektir. Çünkü sütun sayısı ile girilecek veri sayısı uyuşmuyor. Tabloda yer alan sütundan fazla değer girilmek istenmektedir.

```
INSERT INTO tbl_name (a,b,c) VALUES(1,2,3,4,5,6,7,8,9);
```

Update

Veritabanındaki belirli kriterlere uyan satır veya satırlardaki verileri güncellemek için kullanılır.

Aşağıdaki örnekte col1 sütunundaki değerler mevcut col1 değerlerinin bir fazlası olarak güncellenir.

```
UPDATE t1 SET col1 = col1 + 1;
```

Aynı anda iki alan güncellenebilir. Aşağıda col1 ve col2 birlikte güncelleniyor.

```
UPDATE t1 SET col1 = col1 + 1, col2 = col1;
```

Belli kriterleri sağlayan satırları güncellemek için aşağıdaki genel yapı kullanılır:

```
UPDATE table_name SET column1=value, column2=value2,...
WHERE some_column=some_value
```

Örneğin aşağıdaki sorguda CustomerName yalnızca belirtilen şirket olan satırlardaki ContactName ve City alanları güncellenir.

```
UPDATE Customers
SET ContactName='Ali Al', City='Istanbul'
WHERE CustomerName='Computer Co';
```

Select

Veritabanı tablosundan veri satırlarını çekmek için kullanılır.

Bütün alanlarıyla bütün satırları çekmek:

```
SELECT * FROM employee;
```

Belirli alanlarıyla bütün satırları çekmek:

```
SELECT name, salary FROM employee;
```

Şartlı veri çekmek:

Aşağıdaki örnekte sadece maaşları 6000'den fazla olan satırlar çekilir.

```
SELECT * FROM employee WHERE salary > 6000;
```

Aşağıdaki örnekte bölümü Finance olan çalışanlar çekilir.

```
SELECT * FROM employee WHERE dept = 'Finance';
```

Bir kısmı benzeyen şartla veri çekmek:

Aşağıdaki örnekte çalışan ismi JOHN ile başlayan satırlar çekilecektir.

```
SELECT * FROM employee WHERE name LIKE 'JOHN%';
```

Aşağıdaki örnek ise adı SMITH ile biten satırları çeker:

```
SELECT * FROM employee WHERE name LIKE '%SMITH';
```

Mantıksal işlem sonucuna göre veri çekmek:

Aşağıdaki örnekte bölümü Finance olup maaşı 6000 veya üstü olanlar çekilir.

```
SELECT * FROM employee WHERE dept = 'Finance' AND salary >=
6000;
```

Inner Join

İyi tasarlanmış bir veritabanı gereksiz ve çift kayıt tutulmasına neden olmamalıdır. O nedenle bir veri başka bir tabloda varsa ihtiyaç duyulsa bile diğer bir tabloda da tutulmaz.

Örneğin bir tabloda (ismi articles olsun) sadece yazı ve yazıya ait kategori ID'si olsun. Diğer tabloda (ismi cats olsun) ise kategori ID'lerine karşılık gelen kategori ismi olsun.

Bu senaryoda articles tablosundaki yazıları ekrana basarken kategori ID'sini değil de kategori adını basmak istersek cats tablosundan faydalanmamız gerekecek. İşte bu işlem INNER JOIN işlemidir.

```
SELECT articles.article, cats.name FROM articles INNER JOIN
cats ON articles.catid=cats.catid;
```

Bu örnekte articles tablosundan yazıyı, cats tablosundan ise kategori ismini alıyoruz. Kategori ismi olarak articles'da belirtilen kategori ID ile cats'de belirtilen kategori ID'si aynı olan değer seçilir.

Delete

Veritabanı tablosundan veri satırı silmek için kullanılır. Diğer yapılarda olduğu gibi WHERE şartıyla kullanılabilir.

```
DELETE FROM table_name [WHERE Clause]

DELETE FROM somelog WHERE user = 'jill'
DELETE FROM tutorials_tbl WHERE tutorial_id=3
```

Delete yapısı WHERE olmadan genel olarak kullanılmaz. Çünkü WHERE olmadan yazılacak bir delete ifadesi tablodaki bütün satırların silinmesine neden olur.

Order By

Çekilen verileri sıralamak için Order By kullanılır ve bir sıralama kriteri verilir.

Aşağıdaki örnekte doğum tarihine göre küçükten büyüğe sıralama yapılmıştır.

```
mysql> SELECT name, birth FROM pet ORDER BY birth;
+----------+------------+
| name     | birth      |
+----------+------------+
| Buffy    | 1989-05-13 |
| Bowser   | 1989-08-31 |
| Fang     | 1990-08-27 |
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
```

```
| Slim     | 1996-04-29 |
| Puffball | 1999-03-30 |
+----------+------------+
```

Büyükten küçüğe sıralamak için DESC kullanılır:

```
mysql> SELECT name, birth FROM pet ORDER BY birth DESC;
+----------+------------+
| name     | birth      |
+----------+------------+
| Puffball | 1999-03-30 |
| Slim     | 1996-04-29 |
| Claws    | 1994-03-17 |
| Fluffy   | 1993-02-04 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Buffy    | 1989-05-13 |
+----------+------------+
```

İki sütuna göre sıralama yapılabilir:

```
mysql> SELECT name, species, birth FROM pet
    -> ORDER BY species, birth DESC;
+----------+---------+------------+
| name     | species | birth      |
+----------+---------+------------+
| Chirpy   | bird    | 1998-09-11 |
| Fluffy   | cat     | 1993-02-04 |
| Fang     | dog     | 1990-08-27 |
| Bowser   | dog     | 1989-08-31 |
| Buffy    | dog     | 1989-05-13 |
| Puffball | hamster | 1999-03-30 |
| Slim     | snake   | 1996-04-29 |
+----------+---------+------------+
```

Group By

Verileri belli bir alana göre gruplamak için kullanılır. Sonuçlar grup adına gösterilir.

```
SELECT status FROM orders GROUP BY status
```

Orders tablosunda yüzlerce satır vardır ve her satırın bir statusu vardır. Ancak bu SELECT cümlesi status'e göre gruplandığı için her bir status çeşidi için tek satır gösterilir. Örneğin çıktı şu şekilde olur:

```
Cancelled
In Progress
On Hold
Resolved
Shipped
```

Aslında tüm bu statuslerden onlarca satırda var. Grup yapıldığı için her tür status için tek bir satır yazılmıştır.

X Grafik Sunucu

- **Konu:**

X grafik sunucu mimarisi

- **Hedefler:**
  - X istemci/sunucu mimarisi
  - X grafik kartı ve monitor yapılandırma
  - xorg.conf
  - X sunucuyu başlatma ve grafik açılış
  - Giriş yöneticileri

Anahtar Kelimeler:

X, xinit, startx, /etc/X11/xorg.conf, xhost, DISPLAY, xwininfo, xdpyinfo, xdm, gdm, kdm

X Window Sunucu

MS Windows işletim sisteminden farklı olarak Linux altında grafik ortam işletim sisteminin ana parçası olmayıp herhanbi bir uygulama gibi ayrıca bir bileşendir. İşletim sistemi grafik ortama ihtiyaç duymadan tüm fonksiyonlarıyla iş görebilmektedir. Bu mimari sayesinde bilgisayar kaynaklarını en çok tüketen kısım olan grafik arayüz bir ihtiyaç olmaktan çıkmış, ayrıca grafik ortamda çıkacak bir sorun işletim sisteminin çalışmasını etkilemeyecektir.

Linux altında yaygın olarak kullanılan grafik ortam X Window'dur. X Windows, ekran kartını kontrol etmenin yanı sıra içerisinde uzak bilgisayarlara grafik ucu vermek için geliştirilmiş ağ protokolleri de içeren gelişmiş bir sistemdir. Sunucu-istemci mimari ile farklı bir işletim sistemindeki aynı protokolü kullanan X istemcilerine de hizmet verebilir.

X Window sistemi 1984'te MIT üniversitesi tarafından geliştirilmiş

olup hakları 1993’te X Consontium’a, 1997’de Open Group’a geçmiş, bu el değiştirmeler nedeniyle proje yavaşlamıştır. 1992’de itibaren faaliyet gösteren Xfree86 projesi (http://www.xfree86.org) X kodlarını iyileştirmiş, pek çok grafik kart desteği eklemiş, OpenGL grafik hızlandırma gibi gelişmiş özellikleri eklemiştir. Uzunca süre Linux ‘lerin ön tanımlı X sunucusu olarak kullanılmıştır. XFree86 projesi de bir süre sonra yavaşladı ve 2004’te lisans değişiklikleri nedeniyle bir grup geliştirici tarafından X.Org Foundation kurularak kaynak kod çatallanmıştır. Mevcut Linux dağıtımları XFree86’dan X.org’a geçiş yapmıştır.

X Window, oldukça esnek modüler bir yapıda tasarlanmıştır. Örneğin fontlar bilgisayarın kendi diskinden yüklenebileceği gibi uzak sistemden de yüklenebilir. Bu sayede bir font sunucusu kurarak bütün organizasyona bu fontlar sunulabilir.

X Windows içinde pencere yöneticisi (window manager) denen bileşen mutlaka çalışması gerekmektedir. Pencere yöneticisi, uygulamaların grafik ekrana yerleşimi, sanal masaüstleri, pencere küçültme büyütme, yeni pencerelerin açılması ve yerleştirilmesi gibi masaüstünün organizasyonunu yapar. Modüler yapı sayesinde geliştiriciler X Window üzerinde çalışacak kendi pencere yöneticilerini yazmıştır. Pek çok pencere yöneticisi geliştirilmiş olup kullanıcılar kendi beğenilerine göre istediği pencere yöneticisini kullanabilir. Örnek: Window Maker, Enlightenment, Sawfish, Blackbox…

X Window modüllerinden birisi de giriş yöneticisidir (display manager). Linux’te grafik arayüze iki şekilde ulaşılabilir.

1. Eğer açılış seviyesi grafik arayüzden açılacak şekilde ayarlanmışsa (init 5) işletim sistemi açıldıktan sonra init programı grafik açılışla ilgili betikleri çalıştırarak doğrudan grafik arayüzü açar.

2. Kullanıcının önce text konsola girip oradan X Window’u başlatması.

Eğer init doğrudan grafik arayüze geçerse text konsolda yapılan kullanıcı adı ve parola sorma işleminin grafik arayüzde yapılması gerekecektir. Giriş yöneticileri bu işlemi yapmak içindir. Xdm, Kdm ve Gdm gibi farklı farklı giriş yöneticileri bulunmaktadır.

X Windows modüllerinden bir diğeri grafik arayüzdeki widget denilen butonlar, kaydırma çubukları, mesaj kutucukları gibi grafik öğelerdir. QT ve GTK gibi farklı widget kütüphaneleri bulunmaktadır.

Xorg.conf

X yapılandırma dosyası /etc/X11/xorg.conf dosyasıdır. Bu dosya Section anahtar kelimesiyle başlayıp EndSection anahtar kelimesiyle biten bölümlerden oluşmaktadır.

```
Section "SectionName"
SectionEntry
...
EndSection
```

Sıfırdan X yapılandırması zaman alabilir. Bu nedenle dağıtımlar kendi yapılandırma araçlarını geliştirmişlerdir. Red Hat altında system-config-display, SUSE altında sax2, Ubuntu altında displayconfig-gtk kullanılabilir. Genelde kurulum sırasında bu araçlar çalıştırılarak grafik ortam hazırlanır. Bunun yanı sıra XOrg kendisi –configure parametresi ile system donanımını otomatik tespit eder ve şablon xorg.conf dosyasını root ev dizinine yazar. Daha sonar bu dosya isteğe göre düzenlenebilir.

Xorg.conf dosyasında yer alan bazı önemli bölümler:

Files

Sunucu tarafından gereken dosyaların yolunu belirtmek için kullanılır.

```
FontPath "/usr/X11R6/lib/X11/fonts/misc/"
FontPath "/usr/X11R6/lib/X11/fonts/75dpi/"
FontPath "/usr/X11R6/lib/X11/fonts/100dpi/"
```

ServerFlags

Sunucunun davranışlarını etkileyen bayrakları belirler.

```
Option "DontZoom" "boolean"
Option "StandbyTime" "time"
```

Module

Hangi dinamik modüllerin yükleneceğini (Load) veya yüklenmeyeceğini

(Disable) belirler.

```
Load "modulename"
Disable "modulename"
```

InputDevice

Klavye fare gibi girdi aygıtlarını tanımlamak için kullanılır.

```
Section "InputDevice"
Identifier "name"
Driver "inputdriver"
options
...
EndSection
```

Device

Grafik kartı tanımlamak için kullanılır. En az bir tane tanımlanması gerekmektedir.

```
Section "Device"
Identifier "name"
Driver "driver"
entries
...
EndSection
```

ServerLayout

Genel yapılandırmanın olduğu kısımdır. Identifier ve Screen mecburidir.

```
Section "ServerLayout"
         Identifier     "1600x1200"
         Screen         "Screen0" 0 0
         InputDevice    "Mouse0" "CorePointer"
         InputDevice    "Keyboard0" "CoreKeyboard"
 EndSection
```

Monitor

Monitorlerin tanımı bu bölümde yapılır.

```
Section "Monitor"
Identifier "name"
entries
...
EndSection
```

Screen

En az bir tane olması gereken ekran yapılandırma bölümüdür. Birden fazla olabilir olabilir.

```
Section "Screen"
Identifier "name"
Device "devid"
Monitor "monid"
entries
...
SubSection "Display"
entries
...
EndSubSection
...
EndSection
```

```
Section "Screen"
         Identifier   "Screen0"
         Device       "S3 Savage/MX"
         Monitor      "Monitor0"
         DefaultDepth 16

         Subsection "Display"
                 Depth  16
                 Modes  "1600x1200" "1280x1024" "1024x768"
         EndSubsection
 EndSection
```

xwininfo

X pencereleri hakkında bilgi verir. Hiçbir parameter verilmezse –stats varsayılır.

```
$ xwininfo
Absolute upper-left X: 2
Absolute upper-left Y: 85
Relative upper-left X: 0
Relative upper-left Y: 25
Width: 579
Height: 316
Depth: 8
Visual: 0x1e
Visual Class: PseudoColor
Border width: 0
Class: InputOutput
Colormap: 0x27 (installed)
Bit Gravity State: NorthWestGravity
Window Gravity State: NorthWestGravity
Backing Store State: NotUseful
Save Under State: no
Map State: IsViewable
Override Redirect State: no
Corners: +2+85 -699+85 -699-623 +2-623
-geometry 80x24+0+58
```

xdpyinfo

X sunucu hakkında bilgi gösterir. Sunucunun yeteneklerini anlamak, öntanımlı değerleri görmek ve ekran türlerini listelemek gibi amaçlarla kullanılır. Örneğin X sunucunun çözünürlüğünü control etmek için:

```
$ xdpyinfo | grep resolution
resolution: 111x111 dots per inch
```

Bu çözünürlük monitörünüz için uygun değilse uygun olan şekilde X başlatılır:

```
$ startx -- -dpi 100 -depth 16
```

X Window'u Başlatma

X pencere sunucusu komut satırından çalıştırılabildiği gibi bilgisayar açılışı grafik olarak yapılabilir.

Grafik açılış için /etc/inittab içerisinde açılış çalışma seviyesinin 5 yapılması gerekmektedir.

```
id:5:initdefault:
```

Komut satırından basitçe aşağıdaki komutla çalıştırılır:

```
$ startx
```

startx komutu aslında xinit komutunu çalıştıran bir betiktir. Xinit, pencere yöneticisi olmadan boş boş bir grafik ekranı açar. Kullanıcı ev dizininde .xinitrc dosyası varsa bunun içeriğini çalıştırarak masaüstünü donatır. Eğer bu dosya yoksa şu komutu çalıştırarak ilk açılışta bir terminal yerleştirir:

```
xterm -geometry +1+1 -n login -display :0
```

Aşağıdaki örnek .xinitrc dosyası bazı X istemci uygulamalarını ve twm pencere yöneticisini çalıştırır:

```
xrdb -load $HOME/.Xresources
xsetroot -solid gray &
xbiff -geometry -430+5 &
oclock -geometry 75x75-0-0 &
xload -geometry -80-0 &
xterm -geometry +0+60 -ls &
xterm -geometry +0-100 &
xconsole -geometry -0+0 -fn 5x7 &
exec twm
```

X Güvenlik

X sunucu ağ üzerinden hizmet verebilen gelişmiş bir grafik sunucudur. Ağ üzerinden çalışmanın getirdiği iki güvenlik sorunu vardır:

1. Yetkisiz erişimlerle veri çalınması
2. Şifresiz X trafiğinin görülmesi

X haberleşmesi şifresiz olmaktadır. Bu nedenle X trafiğin VPN üzerinden taşınması veya SSH üzerinden tünellenmesi gerekmektedir.

X sunucuya uzaktan bağlantı yapılmayacaksa internet soketini dinlemeyecek şekilde yapılandırılması ve UNIX domain soketlerin kullanılması gerekmektedir. UNIX domain soketlere yalnızca sunucu üzerinden ulaşılabilir.

Xhost ve DISPLAY

X sunucuya erişim yapacak veya yapması engellenecek hostları veya kullanıcıları tanımlamak için kullanılır. Herhangi bir parametresiz verildiğinde mevcut erişim control listesini gösterir:

```
$ xhost
access control enabled, only authorized clients can connect
SI:localuser:user
SI:localuser:gdm
SI:localuser:root
```

Belirli bir hosta X üzerinden erişim izni vermek için:

```
$ xhost +foo.example.com
```

Belirli bir hostun erişim yetkisini kaldırmak için:

```
$ xhost -foo.example.com
```

Erişimi herkese açmak için host belirtilmez:

```
$ xhost +
access control disabled, clients can connect from any host
$ xhost
access control disabled, clients can connect from any host
INET:platon
```

xhost – komutu bütün hostlara erişimi kapatır ancak daha önce verilmiş yetkileri iptal etmez. Daha önce verilmiş yetkiler, tek tek host adları yazılarak iptal edilebilir.

X Sunucu çok gelişmiş ağ protokolleri içerir. Uzak bir bilgisayarda çalıştırılan bir X istemci uygulama başka bilgisayarın fiziksel ekranında gösterilebilir. Bir X istemci çalıştırıldığında hangi ekranın kullanılacağına DISPLAY çevre değişkenine bakarak karar verilir. DISPLAY yerel veya uzak bir monitor olabilir.

```
$ export DISPLAY=remote.example.com
```

Bu tanımdan sonra örneğin xterm veya xclock gibi bir X istemci programı çalıştırılırsa görüntü kendi bilgisayarınızın monitöründe değil remote.example.com isimli bilgisayarın monitöründe gözükür. Bunun gerçekleşebilmesi için uzak bilgisayarda X sunucu çalışması ve bağlanacak bilgisayara erişim izni verilmesi gerkemektedir.

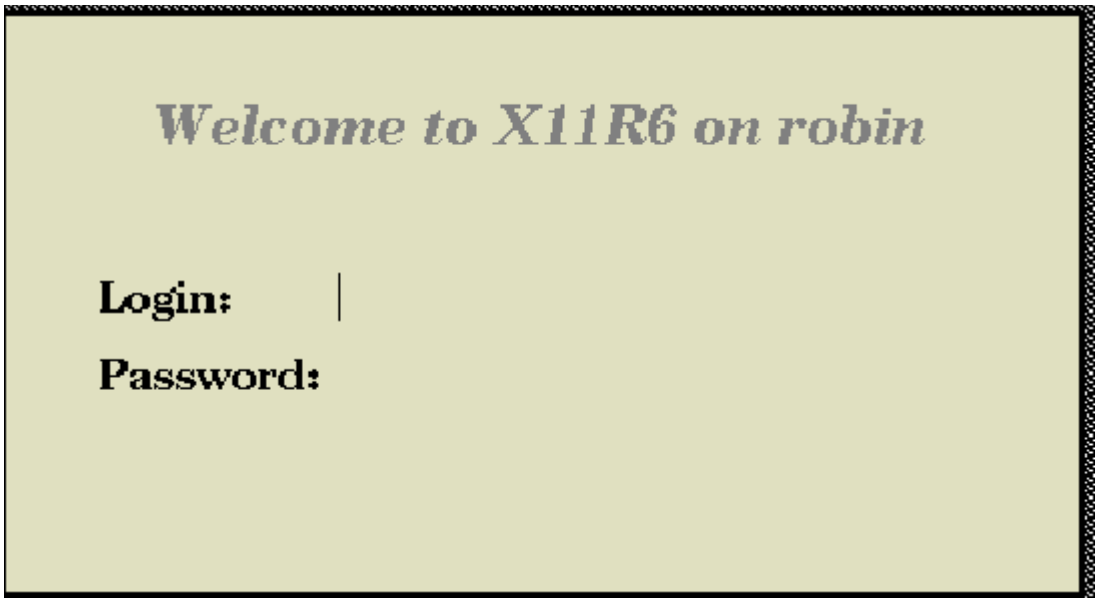
```
$ xhost +me.example.com
$ xclock
```

Grafik Giriş Yöneticisi

X Display Manager, kullanıcı adı ve parola sorarak grafik bilgisayar üzerinden veya uzaktan oturum açmayı sağlayan yazılımdır. Text konsoldaki `login` programının grafik benzeridir. X sunucu kendisi istemci pozisyonunda giriş yöneticisine bağlanır. Aralarında 177. Portu kullanan XDMCP protokolünü kullanırlar.

X'in modüler yapısı sayesinde grafik giriş yöneticileri de ayrı bir modüldür ve başkalarının da benzer modül geliştirmesini sağlamaktadır. X ile birlikte öntanımlı olarak Xdm giriş yöneticisi gelmektedir. Bununla birlikte Linux dünyasında yaygın olarak Gdm (Gnome Display Manager) ve KDM (Kde Display Manager) kullanılmaktadır.

XDM, X11R3 sürümüyle geliştirilen X sunucuyla gelen grafik giriş yöneticisidir. Yapılandırma dosyaları /etc/X11/xdm altındadır.



GDM, Gnome masaüstü projesi için tamamen sıfırdan yazılmış bir grafik giriş yöneticisidir. Yapılandırma dosyaları /etc/gdm altındadır.

KDM, KDE masaüstü projesi için tamamen sıfırdan yazılmış bir grafik giriş yöneticisidir. Yapılandırma dosyası /usr/share/kde.X altında kdmrc dosyasıdır.

Öntanımlı giriş yöneticisi Debian’da /etc/X11/default-display-manager dosyasında tanımlıdır.

Sistem Yönetimi

- **Konu:**

Linux sistem yönetimi

- **Hedefler:**
  - Kullanıcı yönetimi
  - Görev zamanlama
  - Türkçe yerelleştirme ayarları
  - Zaman ayarları

Anahtar Kelimeler:

useradd, userdel, /etc/skel, cron, crontab, at, locale, tzselect

Kullanıcılar

Linux’e giriş yapabilmeniz için bir kullanıcı adınız olması gerekir. Ön tanımlı olarak sadece root kullanıcısı gelmektedir. Tüm yetkilendirmeler kullanıcı adı veya kullanıcı grubu üzerinden yapılmaktadır.

Eklenen kullanıcıların kaydı /etc/passwd ve /etc/shadow dosyalarında tutulmaktadır. /etc/passwd dosyası kullanıcının şifrelenmiş parolası ve parola ilgili tanımlamalar dışında kalan tüm tanımlamaları tutmaktadır. Parola ve parola kullanım süresi, değiştirme süresi gibi tanımlar ise /etc/shadow dosyasında tutulmaktadır. /etc/passwd dosyasını kullanıcılar okuyabiliyorken /etc/shadow dosyasına sadece root kullanıcısı erişebilmektedir.

Örnek bir shadow dosyasının içeriği aşağıdaki gibidir.

```
root:
$6$cFNz8/e8$z9cQK3sUC0OPVWw6nu/QGV1cnKpBJj20ESD2RnWRNWylvLQW5q
.kmeAHrrOfAPLB33JpoqpEQTkFiovqDbFe50:15972:0:99999:7:::
daemon:*:15788:0:99999:7:::
bin:*:15788:0:99999:7:::
```

/etc/shadow dosyasında “:” karakteri ayraç karakteri olarak kullanılır. Dosyadaki değerler sırayla şu anlama gelir.

- Kullanıcı adı
- Şifrelenmiş parola
- Parolanın en son değişim tarihi
- Asgari parola değiştirme periyodu
- En uzun parola geçerlilik süresi
- Parolanın son kullanma tarihi
- Parolanın etkisizleştirilmesinden sonraki toplam gün sayısı

Örnek bir /etc/passwd dosyası aşağıdaki gibidir.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
```

Dosyadaki alanların açıklaması

- Kullanıcı adı
- x karakteri. Eskiden parolanın kendisi tanımlanırdı. Artık parola /etc/shadow dosyasında tanımlanıyor.
- Kullanıcı numarası(ID)
- Grup numarası(GID)
- Kullanıcı açıklaması
- Ev dizini
- Kabuk yolu

Kullanıcı grupları ise /etc/group dosyasında tutulmaktadır.

Örnek bir grup dosyası içeriği:

```
root:x:0:
daemon:x:1:
```

```
bin:x:2:
sys:x:3:
adm:x:4:
```

Dosyadaki alanların açıklaması

Sisteme kullanıcı eklemek için grafik arayüzü araçları kullanılabileceği gibi kabukta useradd komutu kullanılabilir. Bu komut root kullanıcısı tarafından çalıştırılır. Parametre olarak sadece kullanıcı adı verilirse aynı kullanıcı adına ait bir grup da oluşturulup kullanıcı grubu olarak bu grup tanımlanacaktır. Komutun genel parametreleri şunlardır.

- -m kullanıcı ev dizinini oluştur
- -d kullanıcı ev dizini yolu
- -s kullanıcı kabuğu
- -g kullanıcı grubu
- -c Açıklama

Örnekler:

```
# useradd ismail
# useradd –s /bin/csh –m  -d /home/ismail –g admin ismail
```

Belirtilmeyen parametrelerle ilgili ön tanımlı değerler /etc/login.defs dosyasından alınır.

Kullanıcı tanımlandıktan sonra parola ataması yapılmalıdır. Parola tanımlamak için ise passwd komutu kullanılır.

```
# passwd ismail
```

Kullanıcı oluşturulduktan sonra aşağıdaki komutlarla kullanıcı ayarlarında değişiklik yapılabilir.

usermod

Bu komut ile kullanıcı ilave gruplara eklenebilir , mevcut kullanıcı adı başka bir kullanıcı ile değiştirilebilir, kullanıcı hesabı kilitlenebilir veya hesap kilidi kaldırılabilir.

ahmet kullanıcısının hesabını kilitlemek için –L parametresi kullanılır.

```
# usermod –L ahmet
```

Pardus kullanıcısını admin ve apache grubuna eklemek için aşağıdaki komut kullanılır.

```
# usermod –G admin,apache pardus
```

Grup yönetimi

Grup eklemek için groupadd, silmek için de groupdel komutu kullanılır. Groupadd komutu otomatik olarak sıradan bir Grup numarası(GID) atar. Farklı bir GID vermek için –g parametresi kullanılır.

```
# groupadd sistemci
```

Kullanıcının kullanıcı ve grup numarasını öğrenmek için id komutu kullanılır.

```
# id pardus
uid=1004(pardus) gid=1006(pardus) gruplar=1006(pardus),1003(admin),1007(apache)
```

Parola ömür süresi yapılandırması

/etc/shadow dosyasında şifrelenmiş parolaya ek olarak parolanın ne kadar süre sonra değiştirileceğini bilgisi tutulmaktadır. Bu süreyi değiştirmek için chage komutu kullanılır. Bu komut parolanın son değiştirme tarihinden değiştirilmesi gerekecek güne kadar olan geçen sürenin gün cinsinden miktarını değiştirir. Bu bilgi sistem tarafından kullanıcının ne zaman parolasını değiştirmek gerektiğini bulmakta kullanılır.

-M parametresi ile azami gün sayısı belirtilir. –m ile de parola değişikliği için geçecek asgari süre belirtilir.

Bu değerleri sistem genelinde tanımlamak için /etc/login.defs dosyasındaki PASS_MAX_DAYS ve PASS_MIN_DAYS değerleri kullanılır.

Kullanıcı Silme

Kullanıcıyı silmek için userdel komutu kullanılır. Bu komut sadece kullanıcı adı verilerek kullanıldığında kullanıcının /etc/passwd /etc/shadow dosyalarından kaydını siler.

```
# userdel ahmet
```

Yukarıdaki komut çalıştırıldığında ev dizini silinmez. Kullanıcıyı ev dizinindeki dosyalarla birlikte tamamen silmek için –r parametresi kullanılır.

```
# userdel –r ahmet
```

Kullanıcı Girişlerini Kısıtlama

Root dışındaki kullanıcıların geçici olarak sisteme girişini engellemek için /etc/nologin dosyası oluşturmak yeterlidir. Kullanıcı giriş yapmaya çalıştığında bu dosyadaki yazan metin ekrana yazılacak kullanıcının girişine izin verilmeyecektir. Eğer tek bir kullanıcının giriş yapılmaması isteniyorsa usermode –L veya passwd –l komutu ile kullanıcı hesabı kilitlenebilir. Veya kabuğu /sbin/nologin yapılabilir. FTP yapabilsin fakat ssh yapamasın isteniyorsa da kullanıcı kabuğunun /bin/false olarak değiştirilmesi gerekir.

Kullanıcı taslak dosyaları

Her yeni kullanıcı açıldığında kullanıcı ev dizinine otomatik olarak bazı dosyaları kopyalamak için /etc/skel dizini kullanılır.

```
# ls -al /etc/skel/
toplam 40
drwxr-xr-x   5 root root  4096 Kas  9 16:48 .
drwxr-xr-x 153 root root 12288 Ara  4 14:00 ..
-rw-r--r--   1 root root   220 Ara 30  2012 .bash_logout
-rw-r--r--   1 root root  3206 Eki 10  2012 .bashrc
drwxr-xr-x   3 root root  4096 Mar 24  2013 .config
drwxr-xr-x   4 root root  4096 Mar 24  2013 .mozilla
-rw-r--r--   1 root root   675 Ara 30  2012 .profile
drwxr-xr-x   4 root root  4096 Mar 24  2013 .thunderbird
root@pardus:~#
```

Bu dizine açılacak her bir yeni dosya yeni kullanıcıların ev dizinine kullanıcı hakları ile kopyalanacaktır.

su ve sudo komutu

su komutu mevcut kabukta iken çıkmadan başka bir kullanıcı hakları ile kabuk açmak için kullanılır. Geçiş yapılmak istenen kullanıcı ise genelde root kullanıcısıdır. Bu komutun en yaygın parametresi – dir. Bu parametre kabuk geçişinde yeni kullanıcının çevre değişkenleri ile geçiş yapılmasını sağlar.

```
$ su -
Parola:
```

Parola kısmına hedef (burada root) kullanıcısının parolası girilmelidir. Bu durumda root haklarına sahip olmak isteyen herkesin root parolasını bilmesi gerekecekir. Root parolasının birden fazla kişi

tarafından bilinmesi güvenlik zaafiyeti oluşturacağı için su komutuna alternatif olarak sudo komutu geliştirilmiştir.

Sudo komutunun su komutundan en önemli farkı kullanıcıya verilen yetkileri yine kullanıcı kendi parolasını girerek kullanabilmektedir. Bu sayede kullanıcıların root parolasını bilmesine gerek kalmadan root gibi işlemler yapabilmektedir. /etc/sudoers dosyasında izin verilen komutları çalıştırmak için ilgili komutun başına sudo komutunu eklemek yeterli olacaktır.

Cron Görevleri

Cron sistemi UNIX dünyasındaki zamanlayıcının adıdır. Zamanlanan iş olup olmadığı crond servisi tarafından dakikada bir kontrol edilir. Vakti geldiğinde belirtilen komutlar işletilir. Sistem genelinde olduğu gibi her bir kullanıcı da –izin verilmişse- kendi zamanlayıcısını tanımlayabilir. Sistem geneline ait işler /etc/crontab dosyasında tanımlanır.

```
# cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *  * * * root    cd / && run-parts --report /etc/cron.hourly
25 6  * * * root  test -x /usr/sbin/anacron || ( cd / && run-parts
--report /etc/cron.daily )
47 6  * * 7 root  test -x /usr/sbin/anacron || ( cd / && run-parts
--report /etc/cron.weekly )
52 6  1 * * root  test -x /usr/sbin/anacron || ( cd / && run-parts
--report /etc/cron.monthly )
#
```

ile başlayan satırlar açıklama satırıdır cron tarafından yorumlanmaz.

SHELL, PATH ve HOME çevre değişkenleridir, tanımlı olmalıdırlar. Eğer SHELL tanımlı olmazsa geçerli olan ‘sh’ kabuğu kullanılacaktır. PATH değişkeni tanımlanmazsa geçerli bir path değeri kullanılmaz, bütün dosya yolları mutlak olarak verilir. HOME değişkeni tanımlı değilse cron, kendisini çalıştıran kullanıcının ev dizinini kullanır.

Dosyada da görüleceği gibi crontab satırının hangi formatta olacağı şu şekilde tanımlanıyor:

minute Dakika. 0-59 arası bir değer alır.
hour Saat. 0-23 arasında bir değer alır.
mday Ayın günü. 0-31 arası bir değer alır.
month Ay. 1-12 arası bir değer alır.
wday Haftanın günü. 0 ile 7 arası değer alır. 0 ile 7 Pazar günüdür.

Yukarıdaki değerlerin yerine '*' sembolü konulursa geçerli bütün değerler için çalıştır anlamına gelir. Her bir sütun için 2-6 gibi aralık belirtilebildiği gibi, 1,5,7 gibi birden fazla değer de belirtilebilir. Periyodik yapılacak işler için 0-59/15, */5 gibi kullanımlar da geçerlidir. Kullanıcı kısmı, komutun hangi sistem kullanıcısı yetkileriyle çalıştırılacağını belirler. Bunları daha iyi anlayabilmek için aşağıdaki örneklere bakınız.

```
# Her gün gece 02:18 de yedekle.sh'ı çalıştır
18 2 * * * root /root/scripts/yedekle.sh


# Haftanın 1. günü saat 5'te her 10 dakikada bir çalıştırır
0-59/10 5 * * 1 root /usr/local/bin/gonder

# Her 5 dakikada bir çalıştırır
*/5 * * * * root /usr/libexec/atrun
```

/etc/crontab dosyasında Pardus'un kendisiyle ilgili günlük, haftalık ve aylık işler aşağıdaki satırlarla tanımlanmıştır.

```
17 *   * * *  root    cd / && run-parts --report /etc/cron.hourly
25 6   * * *  root   test  -x  /usr/sbin/anacron  ||  (  cd  /  &&  run-parts
--report /etc/cron.daily )
47 6   * * 7  root   test  -x  /usr/sbin/anacron  ||  (  cd  /  &&  run-parts
--report /etc/cron.weekly )
52 6   1 * *  root   test  -x  /usr/sbin/anacron  ||  (  cd  /  &&  run-parts
--report /etc/cron.monthly )
```

Günlük yapılan işler /etc/cron.daily/ dizininde, haftalıklar /etc/periodic/weekly, aylıklar /etc/periodic/monthly dizininde tanımlanmıştır. Linux vakti geldiğinde bu dizinlerdeki tüm dosyaları isim sırasına göre çalıştırır. Dizinlerin içeriği aşağıdaki gibidir. Başlarındaki sayı değerleri ise öncelik sırasını ayarlamak için tanımlanmıştır.

```
# ls /etc/cron.daily/
apache2  apt  aptitude  apt-show-versions  bsdmainutils  dpkg  locate
logrotate  man-db  ntp  passwd  quota  samba  sysstat


# ls /etc/cron.weekly/
apt-xapian-index  man-db

# ls /etc/cron.monthly/
readahead-monthly
```

Crontab'ın Yüklenmesi

Bir önceki bölümde anlatılan işler /etc/crontab dosyasında tanımlanarak yapılmıştı. İstenirse her bir kullanıcı kendi işlerini crontab komutunu kullanarak tanımlayabilir.

Crontab komutuna –l parametresi verildiğinde var olan işler listelenir.

```
# crontab –l
# Her gün gece 02:18 de yedekle.sh'ı çalıştır
18 2 * * * /root/scripts/yedekle.sh > /dev/null 2>&1


# Her 5 dakikada bir çalıştırır
*/5 * * * * /usr/libexec/atrun > /dev/null 2>&1
```

'-e' parametresi ile cron işleri düzenlenebilir. Bu parametre ile çalıştırıldığında geçerli metin düzenleyici –ön tanımlı olarak vi editörü- ile birlikte bu kullanıcıya ait bütün işler açılır. İstenilen değişiklikler veya eklemeler yaptıktan sonra kaydedip çıkıldığında otomatik olarak yeni işler yüklenmiş olacak. Öntanımlı olarak cron işlemi yapıldıktan sonra komutların çıktısı ilgili kullanıcıya e-posta olarak gönderilecektir. Her crontab çalıştığında e-posta gelmesini engellemek için komutların sonuna "`> /dev/null 2>&1`" ibaresi eklenir.

Crontab komutu çalıştırılarak oluşturulan işlerde /etc/crontab dosyasındakinden farklı olarak işin çalıştırılacağı kullanıcıyı tanımlamaya gerek yoktur. Bu değer tanımlanmadığı için crond servisi bu işleri ön tanımlı olarak crontab'ın sahibi kullanıcı haklarıyla çalıştırılacaktır. root kullanıcısı isterse başka kullanıcıların crontab'larına müdahale edebilir. Bunun için –e parametresinden sonra –u parametresi ile aşağıdaki gibi crontab'ında değişiklik yapılacak kullanıcının adı verilir.

```
# crontab –e –u pardus
```

Kullanıcılara ait cron tanımları /var/spool/cron/crontabs dizini altında saklanır.

```
# ls /var/spool/cron/crontabs/
pardus  root
```

Kullanıcılar için Crontab Sınırlaması

Ön tanımlı olarak Pardus'ta tüm kullanıcılar kendi crontab dosyalarını oluşturabilirler. Bunu engellemek için /etc/cron.allow veya /etc/cron.deny dosyaları kullanılabilir. /etc/cron.allow dosyası her satırında bir kullanıcı olacak şekilde oluşturulursa sadece bu

kullanıcıların crontab oluşturulmasına izin verilir. Bu kullanıcılar dışında kimse crontab oluşturamaz. /etc/cron.deny oluşturulursa bu dosyada tanımlı kullanıcılar dışındaki tüm kullanıcıların crontab oluşturmasına izin verilir. Her iki dosyada mevcut ise öncelik /etc/cron.allow dosyasına aittir.

at komutu

Crontab servisine tanımlanan komut belirlenen zaman aralıklarında tekrar tekrar çalıştırılır. Eğer bir iş ileri bir zaman diliminde sadece bir kere çalıştırılmak isteniyorsa cron yerine at komutu kullanılmalıdır. At komutu ile tanımlamış işleri listelemek için atq komutu kullanılırken, tanımlı bir işi silmek için atrm komutu kullanılır. Crontab'da olduğu gibi kullanıcıların bu servisi kullanmalarını engellemek için /etc/at.deny, izin vermek için /etc/at.allow dosyaları kullanılır.

Yerelleştirme ve Saat Dilmi Ayarları

Kabukta kullanılan dili belirlemek için LANG değişkeni kullanılır. Mevcut tanımılari ögrenmek için locale komutu kullanılır.

```
# locale
LANG=tr_TR.UTF8
LANGUAGE=
LC_CTYPE="tr_TR.UTF8"
LC_NUMERIC="tr_TR.UTF8"
LC_TIME="tr_TR.UTF8"
LC_COLLATE="tr_TR.UTF8"
LC_MONETARY="tr_TR.UTF8"
LC_MESSAGES="tr_TR.UTF8"
LC_PAPER="tr_TR.UTF8"
LC_NAME="tr_TR.UTF8"
LC_ADDRESS="tr_TR.UTF8"
LC_TELEPHONE="tr_TR.UTF8"
LC_MEASUREMENT="tr_TR.UTF8"
LC_IDENTIFICATION="tr_TR.UTF8"
LC_ALL=
```

Bu değerlere göre sistem saati, hata mesajları vb bilgilerin hangi dil de gösterileceği belirlenir. Sistemde tanımlanabilecek tüm locale değerlerini listelemek için –a parametresi kullanılır.

```
# locale -a
```

```
aa_DJ
aa_DJ.utf8
aa_ER
aa_ER@saaho
aa_ET
...
```

Örnek olarak date komutunun farklı dil ayarlarındaki çıktıları aşağıdaki gibidir.

```
root@pardus:~ # date
Prş Ara  5 11:56:32 EET 2013
root@pardus:~ # LANG=en_US.UTf8 date
Thu Dec  5 11:56:45 EET 2013
```

iconv komutu ile bir karakter kodlamasında oluşturulmuş dosyayı başka bir karakter kodlamasına çevirebilirsiniz. –f ile kaynak karakter kodlması –t ile de hedef karakter kodlaması tanımlanır.

```
# iconv –f iso-8859-9 –t UTF-8 turkce.txt > turkce-utf8.txt
```

Linux saat dilimi bilgisini öğrenmek için /etc/localtime dosyasına bakar.

```
# ls -al /etc/localtime
-rw-r--r-- 1 root root 2721 Kas  9 16:47 /etc/localtime
```

Bu dosya bir saat dilimi dosyası olabilir veya /usr/share/zoneinfo dizini altında tanımlı bir saat dilimi dosyasına link olabilir.

```
# ls -al /etc/localtime
lrwxrwxrwx 1 root root 35 Ara  5 12:11 /etc/localtime ->
/usr/share/zoneinfo/Europe/Istanbul
```

Bu dosya dışında Pardus ve bazı Linux dağıtımları /etc/timezone dosyasında metin olarak zaman dilimi dosyasını tutarlar.

```
# cat /etc/timezone
Europe/Istanbul
```

Tanımlanabilecek saat dilimi dosyalarının hepsi /usr/share/zoneinfo altında listelenmiştir. Örneğin Avrupa bölgesi için tanımlanabilecek dosya adları aşağıdaki gibidir.

```
# ls /usr/share/zoneinfo/Europe/
Amsterdam  Berlin      Busingen    Guernsey     Kaliningrad  Luxembourg
Monaco   Podgorica  San_Marino  Stockholm  Vaduz      Warsaw
Andorra    Bratislava  Chisinau    Helsinki     Kiev         Madrid
Moscow   Prague     Sarajevo    Tallinn    Vatican    Zagreb
```

```
Athens     Brussels    Copenhagen  Isle_of_Man  Lisbon       Malta
Nicosia  Riga       Simferopol  Tirane     Vienna     Zaporozhye
Belfast    Bucharest   Dublin      Istanbul     Ljubljana    Mariehamn
Oslo     Rome       Skopje      Tiraspol   Vilnius    Zurich
Belgrade   Budapest    Gibraltar   Jersey       London       Minsk
Paris    Samara     Sofia       Uzhgorod   Volgograd
root@pardus:~#
```

Saat dilimini Helsinki yapmak istenirse ilgili dosya /usr/share/zoneinfo/Europe/Europe/Helsinki olacak, saat dilimi tanımı ise Europe/Helsinki olacaktır.

Elle yapılandırmaya ek olarak tzselect, tzsetup ve tzconfig(yeni adıyla dpkg-reconfigure tzdata) gibi araçlarla da saat dilimi öğrenilip ayarlanabilir.

```
root@pardus:~# tzselect
Please identify a location so that time zone rules can be set
correctly.
Please select a continent or ocean.
 1) Africa
 2) Americas
 3) Antarctica
 4) Arctic Ocean
 5) Asia
 6) Atlantic Ocean
 7) Australia
 8) Europe
 9) Indian Ocean
10) Pacific Ocean
11) none - I want to specify the time zone using the Posix TZ format.
#? 8
Please select a country.
 1) Aaland Islands        18) Greece                35) Norway
 2) Albania          19) Guernsey              36) Poland
 3) Andorra          20) Hungary               37) Portugal
 4) Austria          21) Ireland               38) Romania
 5) Belarus          22) Isle of Man     39) Russia
 6) Belgium          23) Italy           40) San Marino
 7) Bosnia & Herzegovina  24) Jersey                41) Serbia
 8) Britain (UK)     25) Latvia                42) Slovakia
 9) Bulgaria               26) Liechtenstein         43) Slovenia
10) Croatia          27) Lithuania             44) Spain
11) Czech Republic        28) Luxembourg      45) Sweden
12) Denmark          29) Macedonia             46) Switzerland
13) Estonia          30) Malta                 47) Turkey
14) Finland          31) Moldova               48) Ukraine
15) France           32) Monaco                49) Vatican City
16) Germany          33) Montenegro
17) Gibraltar             34) Netherlands
#? 47

The following information has been given:

      Turkey

Therefore TZ='Europe/Istanbul' will be used.
```

```
Local time is now:      Thu Dec  5 12:14:02 EET 2013.
Universal Time is now:  Thu Dec  5 10:14:02 UTC 2013.
Is the above information OK?
1) Yes
2) No
Please enter 1 for Yes, or 2 for No.
#? 1

You can make this change permanent for yourself by appending the line
      TZ='Europe/Istanbul'; export TZ
to the file '.profile' in your home directory; then log out and log
in again.

Here is that TZ value again, this time on standard output so that you
can use the /usr/bin/tzselect command in shell scripts:
Europe/Istanbul
```

Kabukta TZ değişkeni ile de aşağıdaki gibi saat dilimi(time zone) bilgisi tanımlanabilir.

```
# export TZ='Europe/Istanbul'
```

Temel Sistem Hizmetleri

- **Konu:**

Linux sistemin sağlıklı çalışması için gerekli temel hizmetler

- **Hedefler:**
  - Sistem tarihi ve saatinin ayarlanması
  - Donanım saatinin ayarlanması
  - NTP sunucusu kullanımı
  - Saat dilimi(timezone) ayarı
  - Sistem ve çekirdek loglama hizmeti
  - Mail gönderim hizmetleri
  - Printer sunucu yapılandırması ve yönetimi

Anahtar Kelimeler:

date, ntpd, ntpdate, hwclock, /etc/localtime, /etc/timezone, /etc/ntp.conf, /usr/share/zoneinfo, pool.ntp.org, syslog,conf, syslogd, klogd, logger, .forward, mail, mailq, sendmail, qmail, exim, postfix, CUPS, lpr, lprm, lpq

Sistem Saati

X86 tabanlı donanımlarda çalışan Linux işletim sistemi için donanım saati ve yazılım saati olmak iki farklı saat türü vardır. Yazılım saati date komutu ile donanım saati de hwclock komutu ile görülebilir.

```
root@pardus:~# date
Cts Kas  9 12:52:45 EET 2013
root@pardus:~# hwclock
Cts 09 Kas 2013 12:52:40 EET  -0.645602 saniye
root@pardus:~#
```

Donanım saatinin UTC(Coordinated Universal Time Eşgüdümlü Evrensel Zaman) olarak tanımlanması tavsiye edilmektedir. Donanım saati ve yazılım saati çok güvenilir olmadığı için NTP(Network Time Protocol) kullanılarak saat sorunları aşılmaktadır.

Date komutu

Bu komut ile sistem saatini görebilir ve değiştirebilirsiniz. Ayrıca sadece günün saati, dakikası, ayın günü gibi değerleri de sorgulayabilirsiniz. Bu tür sorgulamalar genellikle betik(script) içerisinde sıkça kullanılmaktadır.

Hiç bir parametre verilmediğinde o anki tarih ve saati vermektedir.
date
Cts Kas 9 13:01:35 EET 2013

Sadece o anki tarihe ait bir değeri çekmek için ise date +%değişken parametresi kullanılır. Yaygın olarak kullanılan değişkenler şunlardır:

%M	Dakika (00..59)
%m	Ay (01..12)
%k	saat (0..23)
%l	Saat (1..12)
%H	Saat (00..23)
%I	Saat (01..12)
%d	Ayın günü(01..31)

Örnekler:

```
root@pardus:~# date
Cts Kas  9 14:20:24 EET 2013
```

```
root@pardus:~# date +%M
20
root@pardus:~# date +%m
11
root@pardus:~# date +%H
14
root@pardus:~# date +%I
02
root@pardus:~# date +%d
09
```

Bu komutumuz ile sistem tarihini değiştirmek de mümkün. Bunun için aşağıdaki format kullanılır.

 # date AyGünSaatDakikaYıl

Örneğin tarihi 10 kasım 2014 saat 15:35 yapmak için

```
# date 111015352014

#  date 1110153520114
Pzt Kas 10 15:35:00 EET 2014
```

Hwlock komutu

Bu komut ile sistem saatini donanım tarihini gösterdiği tarih olarak tanımlayabilir veya tam tersi tanımlamayı yapabilirsiniz.

hwclok –s veya --hctosys (hardware clock to system) komutu ile Sistem saatini donanım saatinden alması sağlanır.
hwlock –w veya --systohc komutu donanım saatini sistemin şu anki saati olarak tanımlar.

Örnek olarak önce sistem tarihini ileri bir tarih alalım sonrasında sistem tarihini donanım tarihinden alacak komutu verelim.

```
root@pardus:~#  date 1110153514
Pzt Kas 10 15:35:00 EET 2014
root@pardus:~# date
Pzt Kas 10 15:37:04 EET 2014
root@pardus:~# hwclock --hctosys
root@pardus:~# date
Cts Kas  9 14:41:43 EET 2013
root@pardus:~#
```

Aşağıdaki iki komut ise BIOS saatini sistem saatini baz alarak yerel saat veya UTC’ye göre ayarlar.

```
root@pardus:~# hwclock  --systohc --localtime
root@pardus:~# hwclock  --systohc --utc
```

Zaman Dilimi Yapılandırması

Tanımlanabilecek tüm saat dilimleri /usr/share/zoneinfo dizinde bulunmaktadır.

Örneğin Europe/Istanbul saat dilimi bilgileri /usr/share/zoneinfo/Europe/Istanbul dosyasında tutulmaktadır.

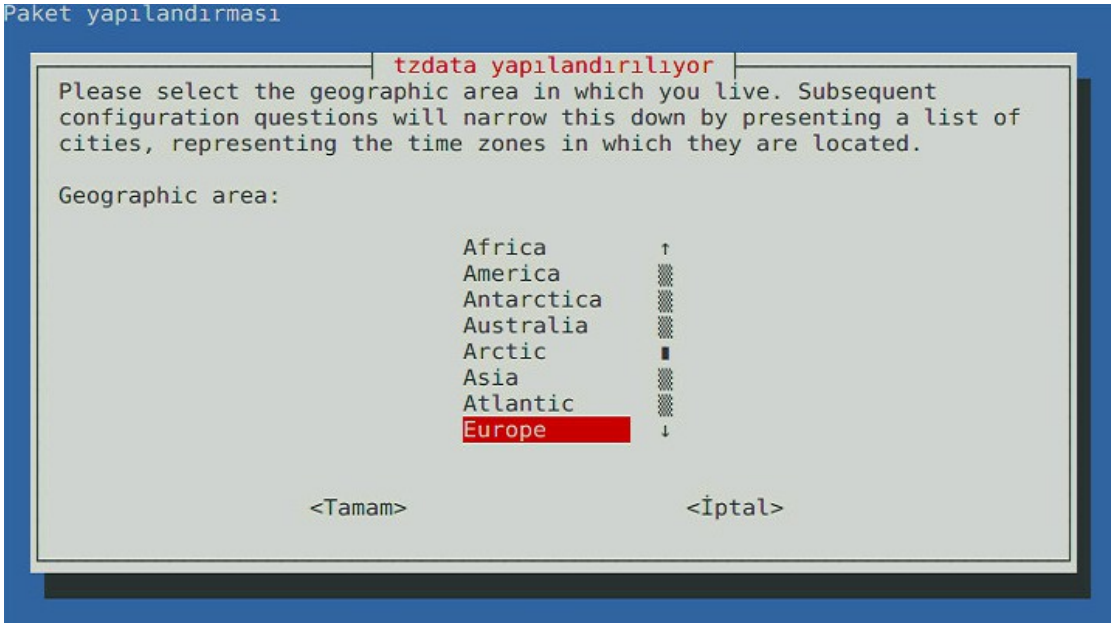
İşletim sistemi saat dilimi değerini ikili /etc/localtime dosyasında tutar. Saat dilimi değiştirilecekse kullanılacak saat dilimi dosyası /etc/localtime dosyasını olarak kaydedilmelidir.

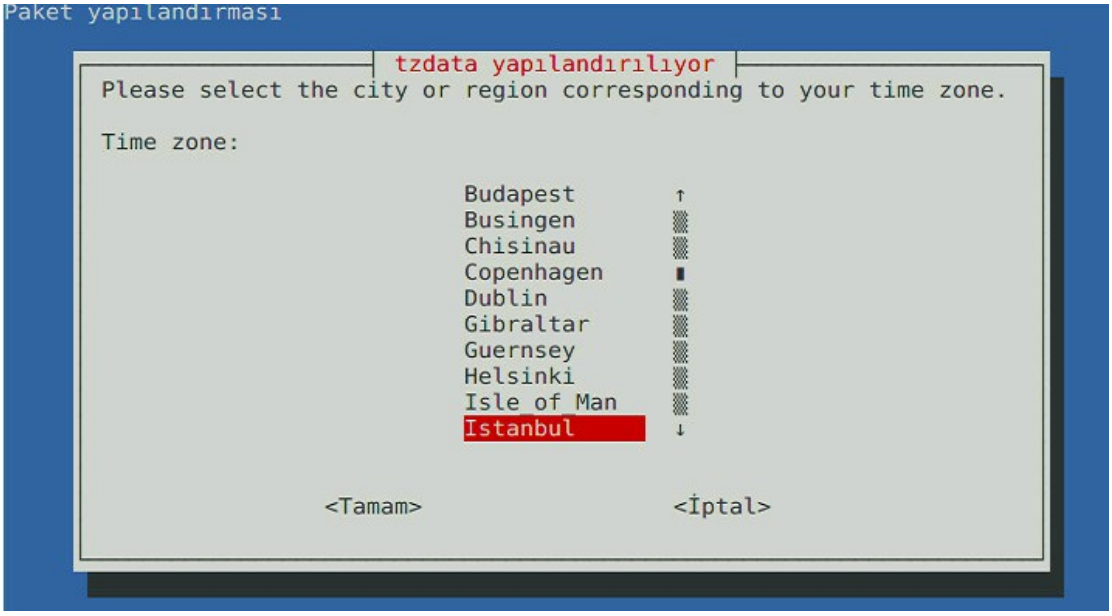
Buna ek olarak /etc/timezone dosyasında okunabilir formatta saat diliminin adı yazar.

```
# cat /etc/timezone
Europe/Istanbul
```

Saat dilimi Pardus'ta zaman dilimi tanımlamak için bir kaç yöntem vardır. En kolayı
dpkg-reconfigure tzdata komutu ile grafiksel olarak yapılandırmak.

```
# dpkg-reconfigure tzdata
```





Current default time zone: 'Europe/Istanbul'
Local time is now: Sat Nov 9 14:59:51 EET 2013.
Universal Time is now: Sat Nov 9 12:59:51 UTC 2013.

Elle yapılandırmak için ise /etc/localtime ve /etc/timezone dosyaları güncellenmelidir.

Saat dilimini Europe/Berlin olarak tanımlamak isterseniz aşağıdaki komutlar verilir.

```
root@pardus:~# date
Cts Kas  9 15:21:25 EET 2013
root@pardus:~# ln -sf /usr/share/zoneinfo/Europe/Berlin
/etc/localtime
```

```
root@pardus:~# echo "Europe/Berlin" > /etc/timezone
root@pardus:~# date
Cts Kas  9 14:21:49 CET 2013
root@pardus:~#
```

Yukarıdaki date komutlarından da görüleceği gibi sistem saati Berlin'e göre ayarlanmıştır.

NTP Yapılandırması

Ağdaki tüm bilgisayarların aynı saat bilgisine sahip olması için NTP(Network Time Protocol) kullanılması gerekmektedir. NTP servisini kullanmak için Linux sistemlerinde ntpd isimli bir servis bulunmaktadır. NTP servisinin durumu öğrenmek, servisi durdurup başlatmak için /etc/init.d/ntp servis dosyası kullanılır. Veya service ntp komutu kullanılır.

```
root@pardus:~# service ntp status
[ ok ] NTP server is running.
root@pardus:~# service ntp status
[ ok ] NTP server is running.
root@pardus:~# /etc/init.d/ntp status
[ ok ] NTP server is running.
root@pardus:~# /etc/init.d/ntp stop
[ ok ] Stopping NTP server: ntpd.
root@pardus:~# service ntp start
[ ok ] Starting NTP server: ntpd.
```

NTP servisi yapılandırma dosyası /etc/ntp.conf'dur. Bu dosyada yapılan bir değişikliğin etkin olması için ntp servisinin yeniden başlatılması gerekir.

Hangi ntp sunucusunun kullanılacağı /etc/ntp.conf dosyasındaki "server" parametresi ile belirtilir. Ön tanımlı olarak Pardus'ta aşağıdaki sunucular tanımlanmıştır.

```
# pool.ntp.org maps to about 1000 low-stratum NTP servers.  Your server will
# pick a different set every time it starts up.  Please consider joining the
# pool: <http://www.pool.ntp.org/join.html>
server 0.debian.pool.ntp.org iburst
server 1.debian.pool.ntp.org iburst
server 2.debian.pool.ntp.org iburst
server 3.debian.pool.ntp.org iburst
```

Elle ntp sunucusundan saati güncellemek için ise ntpdate komutu kullanılır. Bu komuta parametre olarak kullanılmak istenilen ntp sunucu adı verilir. Fakat bu komutun çalışması için ntp servisinin kapalı olması gerekir.

```
# service ntp stop
[ ok ] Stopping NTP server: ntpd.
root@pardus:~# ntpdate 0.debian.pool.ntp.org

 9 Nov 14:37:46 ntpdate[12215]: adjust time server
62.12.173.12 offset -0.012969 sec
# service ntp start
[ ok ] Starting NTP server: ntpd.
```

NTP servisi kapalı değilse aşağıdaki gibi hata alınacaktır.

```
# ntpdate 0.debian.pool.ntp.org
 9 Nov 14:36:18 ntpdate[11936]: the NTP socket is in use, exiting
```

Uzak NTP sunuculardan saatin güncellenebilmesi için güvenlik duvarı üzerinde ntp port numarasına(UDP 123) erişim verilmelidir.

Sistem Günlükleri

Syslog servisi

Linux günlük yönetimi için syslogd servisini kullanmaktadır. Günümüzde rsyslogd servisin ek olarak yeni nesil syslog servisi olan syslog-ng servisi de yaygın olarak kullanılmaktadır. Debian tabanlı sistemlerde ön tanımlı syslog sunucusu olarak rsyslogd gelmektedir.

Rsyslogd servisinin ön tanımlı yapılandırma dosyası /etc/rsyslog.conf'dur. Daha eski sürümlerde /etc/syslog.conf dosyası bulunmaktadır.
Bu dosyada bulunan aşağıdaki tanımla /etc/rsyslog.d dizininde .conf ile biten tüm dosyalarda yapılandırma dosyası olarak kullanılmaktadır.

```
# Include all config files in /etc/rsyslog.d/
#
$IncludeConfig /etc/rsyslog.d/*.conf
```

Örnek günlük tanımları aşağıdaki gibidir.

```
# First some standard log files.  Log by facility.
#
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
```

```
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
user.*                          -/var/log/user.log


#
# Logging for the mail system.  Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info                       -/var/log/mail.info
mail.warn                       -/var/log/mail.warn
mail.err                        /var/log/mail.err
```

Yukarıda görüldüğü gibi dosya iki ana bölümden oluşmaktadır. Soldaki bölümde servis ve şartlar(facility) tanımlanmakta, sağ tarafta ise günlüklerin nasıl işleneceği tanımlanır. Sağ tarafa eylem (action) bölümü de denilir. Sol taraf ile sağ taraf birbirinden TAB komutlarıyla ayrılmıştır.

Sol taraftaki servisler de servis adı.öncelik şeklinde iki kısımdan oluşur. Bu öncelik değerleri `emerg, alert, crit, err, warning, notice, info ve debug` olabilir. Servis adı `ise auth, authpriv, console, cron, daemon, ftp, kern, lpr, mail, mark, news, ntp, security, syslog, user, uucp, ve local0 den local7` ye kadar olan herhangi bir değer olabilir. Tüm servisleri veya öncelikleri ifade etmek için * karakteri kullanılır. Sağ taraftaki eylem kısmında dosya belirtilirse günlükler dosyaya yazılır, uzak sunucu , konsol da tanımlanabilir.

Örneğin ilk satırdaki

```
mail.*                          -/var/log/mail.log
```

ifadesi şu şekilde yorumlanır. Mail servisine ait tüm öncelik değerleri /var/log/mail.log dosyasına yazılacaktır. Dosya başlarındaki “-“ ifadesi kaydın asenkron bir şekilde yazılacağını ifade etmektedir. Rsyslog ile ilgili herhangi bir yapılandırma değişikliği olduğunda rsyslogd servisi yeniden başlatılmalıdır. Aşağıdaki komutlarlar servis durumu öğrenilebilir, servis durdurup başlatılabilir veya yeniden başlatılabilir.

Eğer bir günlük türü ayrıca uzak bir syslog sunucusuna iletilmek istenirse dosya adı yerine sunucu adı başına @ işareti konularak belirtilir.

```
mail.*                          @172.16.1.10
```

Ön tanımlı olarak syslog sunucuları UDP 514 portu kullanılır. Eğer sunucu farklı bir port numarası kullanıyorsa :port ile port numarası belirlenir. Örneğin 5410 nolu port numarasına göndermek için:

```
mail.*                          @172.16.1.10:5140
```

```
root@pardus:/etc# service rsyslog status
[ ok ] rsyslogd is running.
root@pardus:/etc# service rsyslog stop
[ ok ] Stopping enhanced syslogd: rsyslogd.
root@pardus:/etc# service rsyslog start
[ ok ] Starting enhanced syslogd: rsyslogd.
root@pardus:/etc# service rsyslog restart
[ ok ] Stopping enhanced syslogd: rsyslogd.
[ ok ] Starting enhanced syslogd: rsyslogd.
root@pardus:/etc#
```

/etc/rsyslog.d dizininde ise genelde sonradan kurulan servislere ait yapılandırma dosyaları olur. Bu sayede ana yapılandırma ile ek servislere ilişkin yapılandırılma ayrıştırılmış olur.

```
# ls /etc/rsyslog.d
postfix.conf
```

Logger Komutu

Bu komut elle syslog kaydı oluşturmak için kullanılır. Genelde yapılandırmanın doğru bir şekilde olup olmadığını kontrol amacıyla kullanılır.

```
root@pardus:~# logger sunucu tekrar baslatilacak
root@pardus:~# tail /var/log/messages
...
Nov  9 22:33:43 pardus root: sunucu tekrar baslatilacak
```

Belli bir kayıt türü belirtmek için –p parametresi kullanılır.

```
root@pardus:~# logger -p mail.info "eposta geldi"
root@pardus:~# tail /var/log/mail.info
Nov  9 22:35:49 pardus root: eposta geldi
```

Günlük Dosyalarının Döndürülmesi

Günlük dosyalarının döndürülmesi için logrotate komutu kullanılır. Bu komut ayrıca döndürülen dosyalar ile ilgili aşağıdaki özelliklere sahiptir.

- Sıkıştırma
- belirli bir sayıda dönderilen dosyayı saklama
- günlük,haftalık, aylık aralıklarla döndürme yapma
- döndürme işleminden sonra gerekiyorsa servisi yeniden başlatma

Ön tanımlı yapılandırma dosyası /etc/logrotate.conf dosyasıdır. Bu dosyadaki aşağıdaki ifade ile /etc/logrotate.d dizinindeki dosyalarda yapılandırma dosyaları olarak işlem görür.
packages drop log rotation information into this directory
include /etc/logrotate.d

Aşağıda ön tanımlı logrotate.conf dosyasındaki tanımlamaları görebilirsiniz.

```
# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0660 root utmp
    rotate 1
}
```

Bazı logrote parametreleri:

- rotate ifadesi ile tutulacak eski günlük dosya sayısı belirtilir.
- create ile dosyanın oluşturma modu ve sahibi belirlenir.
- missingok parametresi ile de bu dosya yoksa bir hata dönmemesi sağlanır.

- montly parametresi aylık bazla dosyanın dönderileceğini ifade eder.
- Size ile dosyanın dönderilmesi için gerekli boyut belirtilir.
- Compress ile dosyanın sıkıştırılması gerektiği belirtilir.

Örnek bir /etc/logrotate.d dizini içeriği aşağıdaki gibidir.

```
root@pardus:~# ls /etc/logrotate.d
apache2  apt  aptitude  consolekit  cups  dpkg  pm-utils  ppp
preload  rsyslog  samba  speech-dispatcher  unattended-
upgrades  winbind
```

Aşağıda görüldüğü gibi sıkıştırılan dosyalar

```
# ls /var/log/
alternatives.log       bootstrap.log  daemon.log.1     dpkg.log.1
lpr.log        messages.3.gz       syslog                user.log.1
alternatives.log.1     btmp           daemon.log.2.gz  dpkg.log.2.gz
lpr.log.1      messages.4.gz       syslog.1              user.log.2.gz
alternatives.log.2.gz  btmp.1         daemon.log.3.gz  error
mail.err       news                syslog.2.gz           wtmp
apache2                ConsoleKit     daemon.log.4.gz  faillog
mail.info      ntpstats            syslog.3.gz           wtmp.1
apt                    cron.log       debug            fontconfig.log
mail.log       pm-powersave.log    syslog.4.gz           Xorg.0.log
auth.log               cron.log.1     debug.1          fsck
mail.log.1     pm-powersave.log.1  syslog.5.gz
Xorg.0.log.old
auth.log.1             cron.log.2.gz  dmesg            gdm3
mail.log.2.gz  preload.log         syslog.6.gz
auth.log.2.gz          cron.log.3.gz  dmesg.0          hp
mail.warn      preload.log.1.gz    syslog.7.gz
auth.log.3.gz          cron.log.4.gz  dmesg.1.gz       kern.log
messages       pycentral.log       sysstat
auth.log.4.gz          cups           dmesg.2.gz       kern.log.1
messages.1     samba               unattended-upgrades
boot.log               daemon.log     dpkg.log         lastlog
messages.2.gz  speech-dispatcher   user.log
```

E-posta Yönetimi

Bir linux sistem hem e-posta sunucusu hem de e-posta istemcisi olarak hizmet verebilir. Linux üzerinde kullanılabilecek bir çok e-posta sunucusu ve yazılımı vardır. Burada kısaca bu yazılımlar hakkında bilgiler verilecektir.

E-posta Sunucuları

Sendmail: Linux üzerinde çalışan en eski e-posta sunucusudur. Sendmail hala yaygın olarak kullanılsa da performans,güvenlik

sorunları ve yapılandırmasının diğerlerine göre biraz kompleks olmasından dolayı eskisi kadar tercih edilmemektedir. Sendmail hakkında detaylı bilgiyi www.sendmail.org sitesinden edinebilirsiniz. Tüm işi tek bir proses yapmaktadır.

Postfix: Sendmail'e alternatif,modüler bir dizayn ile geliştirildi. Postfix birden fazla program kullanarak farklı işleri farklı prosesler ile yapmaktadır. Sendmail'e göre daha kolay yapılandırması vardır. Günümüzde bir çok Linux dağıtımının ön tanımlı e-posta sunucusudur. Web sayfası: www.postfix.org

Exim: Sendmail gibi tek bir proses ile tüm süreci yönetmektedir. Sendmail'e göre daha kolay bir yapılandırması vardır. Bir kaç Linux dağıtımı tarafından ön tanımlı e-posta sunucusu olarak kullanılmaktadır. Web sayfası:www.exim.org

Qmail: Güvenlik ve performans amaçlı geliştirilmiş bir e-posta sunucusudur. Yapılandırması diğerlerine göre farklılık gösterir. İlk başlardaki lisanslama konusundan dolayı herhangi bir Linux dağıtımı tarafından ön tanımlı olarak kullanılmamaktadır. Fakat performansı ve güvenlik açığı olmamasından dolayı bir çok sistem yöneticisi tarafından tercih edilen bir e-posta sunucusudur. Gayrı resmi web sayfası: www.qmail.org, qmail yazarına ait resmi web sayfası http://cr.yp.to/qmail.html

E-postaları çekmek için bir IMAP/POP3 sunucusuna ihtiyaç vardır. Günümüzde yaygın olarak Dovecot(www.dovecot.org) sonrasında ise Cyrus IMAP(http://www.cyrusimap.org/) sunucuları kullanılmaktadır. Bu sunuculardan farklı olarak uzaktaki e-postaları belirli aralıklarla bilgisayara indirip size ait bir SMTP sunucusu kuyruğuna veren fetchmail yazılımı vardı. Fetchmail bir sunucu yazılımı olmayıp, uzak sunucu ile yerel sunucu arasındaki bağlantıyı sağlıyor.Fakat günümüzde her geçen gün geçerliliğini yitirmektedir.

Temel E-posta Sunucu Ayarları

E-posta Yönlendirme

Bir kullanıcıya gelen e-postayı başka bir adrese yönlendirmek için hem kullanıcı seviyesinde hem de sistem seviyesinde tanımlama yapılabilir.

Kullanıcı seviyesine kullanıcının ana dizininde .forward dosyası oluşturup buraya e-postanın iletileceği yeni adres yazılır. Bu dosyaya adres yerine bir komut tanımlayarak her e-posta geldiğinde bu komutun çalışması da sağlanabilir.

Sistem genelinde ise /etc/aliases dosyası kullanılır. Bu dosyanın ön tanımlı hali aşağıdaki gibidir:

```
# cat /etc/aliases
# See man 5 aliases for format
postmaster:    root
```

Bu dosyada yapılacak herhangi bir değişikliğin etkinleşmesi için newaliases komutu kullanılır. Bu dosya hem postfix hem de sendmail tarafından kullanılmaktadır.

E-posta Kuyruk Raporlaması

mailq komutu ile kuyrukta bekleyen e-postalar hakkında kısa bilgi alınabilir.
Komut hiç parameter verilmede çalıştırılabilir.

```
# mailq
Mail queue is empty
```

E-posta İstemcileri

Linux'de hem komut satırından hem de grafik arayüzden kullanabileceğiniz bir çok e-posta istemcisi vardır. En yaygın grafik tabanlı e-posta istemcileri:

Thunderbird (www.mozilla.org/tr/thunderbird)
Evolution (https://projects.gnome.org/evolution/)
Kmail (http://userbase.kde.org/KMail).

Komut satırından e-posta okumak için ise mutt, alpine ve mail komutu bulunmaktadır.

mail komutu hem e-posta okuyabilir hem de e-posta gönderebilir.

Mail komutu parametre vermeden çalıştırılırsa kullanıcıya ait okunmamış e-posta var mı yok mu diye kontrol eder. Aşağıdaki gibi parametlerlerle de bir e-posta adresine bir dosyanın içeriğini gönderebilir.

```
# mail –s "konu"  admin@pardus.org.tr <  /var/tmp/mesaj.txt
```

-s ile e-posta konusu belirlenir. < ile de belirtilen dosyadaki içerik e-postanın gövdesi olarak karşı tarafa iletilir.

Yazıcı Sistemi

Linux sistemlerde yazıcı işlemleri CUPS(Comon Unix Printing System) servisi tarafından yönetilir. Yapılandırmalar /etc/cups dizininde bulunur.

```
# ls /etc/cups
cupsd.conf          cups-files.conf  ppd        raw.types  ssl
subscriptions.conf.O  cupsd.conf.default  cups-pdf.conf
raw.convs  snmp.conf  subscriptions.conf
```

bu servise “service cups stop|start|restart” komutları ile yönetilir. CUPS öncesinde LPD ve LPRng gibi yazıcı alt sistemleri kullanılıyordu. LPD kullanan birisi lpr, lprm,lpq gibi komutlarına aşinadır. CUPS eski servisleri kullanan system yöneticileri için bu komutların aynısını geliştirmiştir.

Ayrıca CUPS servisinin web yönetim arayüzü de bulunmaktadır. Ön tanımlı olarak sadece http://localhost:631 adresinden erişilebilmektedir. Ek izinler için ve digger yapılandırmalar için /etc/cups/cupsd.conf dosyası düzenlenmelidir.

CUPS servisi /var/spool/cups dizinini kuyruk dizini olarak kullanmaktadır.

Yazıcı komutları:

lpr Komutu

Bu komut yazıcıya iş göndermek için kullanılır. Bu komutu kullanmadan once yazıcı ayarlarının yapılmış olması gerekir. Grafik arayüz üzerinden çıktı alınacaksa bu detayları bilmeye gerek yoktur. Arayüz lpr komutlarını daha açıklayıcı bir şekilde tanımlamaktadır.

En yaygın parametreleri:

- -r Bu parametre ile yazma işlemi bittikten sonra dosyanın silinmesini sağlanır.
- -h ile banner yazılması iptal edilir
- -# sayı: Yazmanın kaç kopya olacağını belirtir.
- -P hangi yazıcı kuyruğunun kullanılacağını belirtir.

Örnek:

HPLazer kuyruğunda dosya.txt dosyasından 3 kopya çıktı almak için:

```
# lpr -P   HPLazer  -#3  dosya.txt
```

lpq Komutu

lpq komutu ile yazıcı kuyruğu istatistikleri görüntülenir. –P parametresi ile kuyruk adı belirtidir.

lprm/cancel

lprm ve cancel komutları aynı işi yapmaktadır.Bu komutlar ile de kuyruktaki bekleyen işler silinebilir. Parametre olarak iş numarası(job id) alır.

Silinecek işe ait iş numarası ise yukarıda anlatılan lpq veya lpstat komutu ile görüntülenebilir.

“-“ parametresi ile kuyruktaki bekleyen tüm işler iptal edilir.

cupsenable/cupsdisable

cupsenable ve cupsdisable komutu ile de yazıcı kapatılıp açılabilir. Ayrıca cpudisable komutuna –c parametresi verilecek yazıcı kuyruğundaki tüm işler iptal edilebilir. İptal mesajını tüm kullanıcılara bildirmek için ise –r parametresi ile mesaj tanımlanabilir.

Ağ Yönetimi

- **Konu:**

Linux ağ yönetimi

- **Hedefler:**
  - IP adresleme
  - Temel protokoller, TCP ve UDP farkları
  - Temel portlar ve üzerlerinde çalışan hizmetler
  - Ağ arayüzlerinin yapılandırılması
  - Yönlendirme
  - IPv6

- Host DNS yapılandırması ve sorgulama
- Ağ sorunlarını çözme araçları

Anahtar Kelimeler:

IPv4, IPv6, /etc/services, ftp, telnet, ifconfig, ifup, ifdown, route, /etc/network/interfaces, ethtool, ip, /etc/hosts, /etc/resolv.conf, /etc/nsswitch.conf, host, dig, ping, traceroute, netstat, netcat

Adresleme

TCP/IP ağlarının oluşturulması sırasında yapılması gereken ilk görev ağ üzerinde bulunan tüm noktalara ağ adreslerinin atanmasıdır. IPV4 (32 bit) ve IPV6 (128 bit) olmak üzere iki çeşit IP adresi vardır.

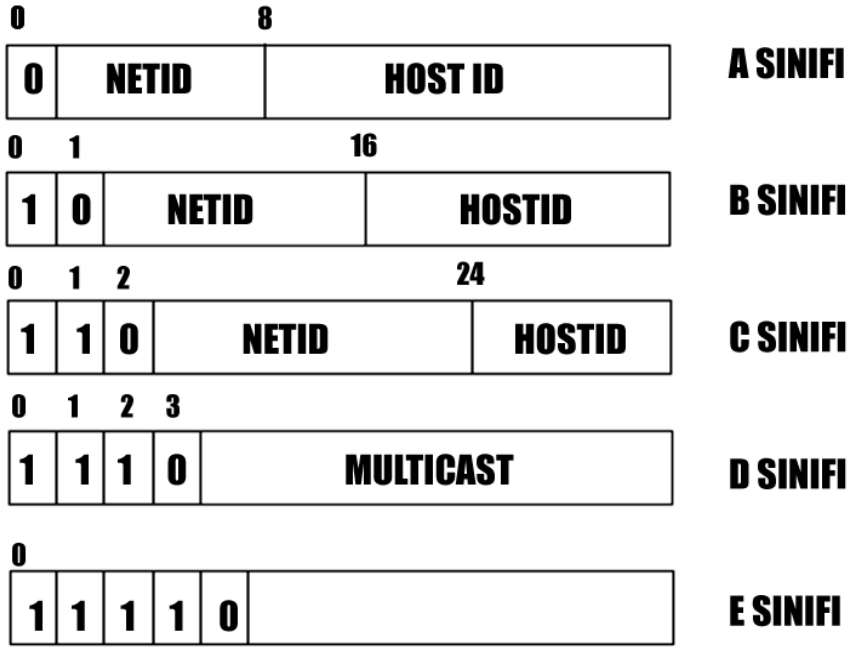
IP adresleri 4 oktetlik birimler halinde gösterilir. Onluk tabanda IP adresi, dört parçaya ayrılmış noktalı (194.27.200.20) gösterimle tanımlanır. Noktalar ile ayrılmış olan her bir parça 0-255 arasında değer alabilir. Bu yapıya “onluk tabanda noktalı gösterim” adı verilir.

IP adresleri temel olarak A, B, C olmak üzere 3 sınıfa ayrılır.

A Sınıfı – 127 adet ağı ve her ağ içerisinde yer alabilecek olan 16,777,214 milyon bilgisayarı tanımlayabilir.

B Sınıfı – 16,383 adet ağı ve her ağ içerisinde yer alabilecek olan 65,534 bilgisayarı tanımlayabilir.

C Sınıfı – 2,097,151 milyon ağı ve her ağ içerisinde yer alabilecek olan 254 bilgisayarı tanımlayabilir.



D sınıfı IP'ler multicast için, E sınıfı IP'ler ise test amaçlı kullanılan özel IP'lerdir.

Ağlar kimi zaman yönetimi kolaylaştırmak ve güvenliği artırmak için küçük parçalara ayrılabilir. Bu durumda bağlantıyı sağlayan yönlendiricilerin alt ağ maskesini göz önünde bulundurması gerekir.

Alt ağların tanımlanması durumunda "Hedef IP Adres" alanını alt ağ maskesi ile "&" (and) ([1 &1 = 1], [1 & 0 = 0]) mantıksal işlemine sokar. "&" işleminden sonra elde edilen değer ağ adresi ile aynı değilse yönlendirme işlemi yapılır.

IP sınıfına göre ağ maskeleri şu şekildedir:

Sınıf	İlk 8'li	Öntanımlı Maske
A	1-126	255.0.0.0
B	128-191	255.255.255.0
C	192-223	255.255.255.0
D	224-239	255.255.255.255
E	240-254	-

IPv6 adresleri bünyesinde IPv4 adreslerini de barındırabilir. Bu nedenle IPV6 adreslerinin gösterimi bu ayrımı yapmak içinde kullanılabilir. "X:X:X:X:X:X:d.d.d.d" olarak verilen gösterimde "X"

değerleri onaltılık sayı sisteminde sayıları “d” değerleri onluk tabanda sayıları ifade eder.

Özel IP’ler

Özel ağlarda adreslemeyi basitleştirmek, internet ağlarıyla çakışmayı önlemek için bazı IP’ler rezerve edilmiştir. Özel IP’ler internet ağ geçitleri tarafından yönlendirilmez, yalnızca iç ağlarda kullanılırlar. RFC 1918’de tanımlanan özel ağlar:
10.0.0.8/8
172.16.0.0/12
192.168.0.0/16

Protokoller

Protokol iki ağ blgisayarının birbiriyle konuşma dilidir. TCP/IP katmanlı bir mimari olup her protokol, gerçek pakete bir takım bilgiler ekleyerek sarmalar ve bir sonraki katmanın protokolüne iletir. Bu şekilde sarılıp sarılıp en son fiziksel katmandan (bu genelde bir ethernet veya fiber kart oluyor) karşı tarafa iletilir. Karşı tarafta en dıştan başlanarak paket açılır. Bir katmanda yönlendirme bilgisi varken, diğer katmanda hangi hizmetin (portun) hedeflendiği bilgisi bulunur. Tüm bunların yanında mesajların doğru gelip gelmediğini kontrol eden hata kontrol mekanizmaları ve bu mekanizmalara ait ilave alanlar vardır.

Kaynak bilgisayarda bir katmandaki bir protokol tarafından pakete eklenmiş veriyi ancak hedef bilgisayardaki aynı katmandaki aynı protokol çözümleyebilir, anlayabilir.

TCP Protokolü

TCP protokolü, bağlantı bazlı güvenli veri akışını sağlayarak iletim katmanına (transmission layer) çok önemli hizmetler sunar. Çoğu uygulama kendi veri iletişim kontrol mekanizmasını oluşturmaktansa

TCP protokolünün sağlamış olduğu hizmetleri kullanır. TCP sunduğu hata denetimi, veri akış kontrolü gibi hizmetler sayesinde kendisini kullanan uygulamalara tatmin edici düzeyde güvenlik, hata denetimi ve akış kontrolü sağlar.

TCP protokolünün en önemli özellikleri şunlardır;

- Bağlantı noktaları arasında veri iletişimini sağlaması
- Güvenli veri iletimine olanak tanıması
- Bağlantıda olan iki bilgisayar arasında akış kontrolü sağlaması
- Çoklama (Multiplexing) yöntemi ile birden fazla bağlantıya izin vermesi
- Sadece bağlantı kurulduktan sonra veri iletimi sağlaması
- Gönderilen mesaj parçaları için öncelik ve güvenlik tanımlaması yapılabilmesi

FTP, SMTP, HTTP, IMAP gibi uygulama katmanı protokolleri TCP kullanırlar.

UDP Protokolü

UDP güvenli olmayan, bağlantısız veri akışı sağlar. Bağlantının açılması, kapatılması gibi operasyonlara gerek yoktur. Datagramların hedeflerine varmaları farklı olarak yönlendirilmeleri sebebi değişik zamanlarda olabilir. Bu nedenle UDP protokolü kullanılarak gönderilen verilerin aynı sıra ile hedeflerine ulaşması mümkün olmayabilir. UDP protokolü, TCP protokolünden daha hızlı ve daha kolaydır. Buna karşın sağlamlık, güvenilirlik gibi kriterler göz önüne alındığı zaman TCP protokolüne nazaran çok fazla dezavantajı vardır.

UDP protokolünün temel özellikleri şunlardır.

UDP protokolü datagram tabanlıdır. Bu nedenle bağlantının açılması, kapatılması gibi operasyonlara gerek yoktur.
UDP toplu yayın (broadcast) – grup yayın (multicast) mesajları için son derece kullanışlıdır.
Datagramların hedeflerine varmaları farklı olarak yönlendirilmeleri sebebi değişik zamanlarda olabilir. Bu nedenle UDP protokolü kullanılarak gönderilen verilerin aynı sıra ile hedeflerine ulaşması mümkün olmayabilir.
UDP protokolü içerisinde sadece isteğe bağlı olarak hata kontrol mekanizması yürütülür.
UDP protokolü, TCP protokolünden daha hızlı ve daha kolaydır. Buna karşın sağlamlık, güvenilirlik gibi kriterler göz önüne alındığı zaman TCP protokolüne nazaran çok fazla dezavantajı vardır.

DNS ve SNMP uygulama protokolleri UDP üzerinde taşınırlar.

ICMP Protokolü

Internet Protokolü, ICMP Protokolünü kullanarak işlem gören datagramların iletimi sırasında meydana gelen hataları, uyarı ve kontrol bilgilerini iletir. Bu mesajlar ağ yöneticileri tarafından değerlendirilerek ağ içerisinde meydana gelen aksaklıkların belirlenmesinde kullanılabilir.

ICMP mesajları genel olarak şu durumlarda üretilir.

IP datagramların hedeflerine ulaşamaması durumunda
Ağ geçitlerinin datagramları hedeflerine yönlendiremeyecek kadar yoğun olması durumunda
Datagramların hedeflerine yönlendirilebileceği daha kısa bir rota olması durumunda

Port Numaraları

Aynı hedefte birden fazla servis çalıştırmak için port numarası kullanılır. IP adresi bir bilgisayarı adreslerken, port numarası o bilgisayardaki çalışan servisi adresler. IP adresi bir binanın kapı numarası ise, port numarası kaçıncı kat olduğunu veya daireyi gösterir. Böylece aynı hedefte birden fazla servis varsa bunlar birbirinden ayırt edilir.

Örneğin aynı bilgisayarda hem mail gönderim (SMTP) hem de web sunma (HTTP) hizmetleri aynı anda bulunabilir. Mail gönderim için örneğin Thunderbird uygulamasını, sunulan web sayfasını görüntülemek için örneğin Chrome uygulaması kullanılır. Her iki uygulamada aynı sunucuya bağlanacak. Ancak iki uygulamanın konuşma dili (protokol) birbirinden farklıdır. Thunderbird, 25 numaralı porttaki SMTP sunucuya bağlanırken Chrome 80 numaralı porttaki web sunucuya bağlanır.

1024'den küçük port numaraları standarttır ve belli protokollere tahsis edilmiştir. Linux altında standart hizmetlerin listesi /etc/services dosyasında bulunur.

```
$ cat /etc/services
[...]
ftp-data      20/tcp
ftp           21/tcp
fsp           21/udp
ssh           22/tcp
ssh           22/udp
telnet        23/tcp
smtp          25/tcp
time          37/tcp
time          37/udp
[...]
```

Bazı port ve hizmetler ağ yöneticileri tarafından sıklıkla kullanılır:

Protokol Port	Hizmet
TCP 20	FTP Veri
TCP 21	FTP Kontrol
TCP 22	SSH

TCP 23	Telnet
TCP 25, 465	SMTP, SSMTP
UDP 53	DNS
TCP 80, 443	HTTP, HTTPS
TCP 110, 945	POP3, POP3S
UDP 139	NetBIOS
TCP 143, 993	IMAP, IMAPS
UDP 161	SNMP

Ağ Arayüzleri

Linux altında ağ arayüzleri /dev altında sanal bir aygıt olarak yönetilir. Fiziksel erişim türüne göre arayüzler farklı isimlerle çağrılır:

lo: loopback
eth: Ethernet kartları
slip: Seri port modemler
ppp: Noktadan noktaya
isdn: ISDN hat
fddi: Fiber hat

Arayüzler sırasıyla isimlendirilir. Örneğin bir sunucuda iki adet Ethernet kart varsa bunlar ana kart üzerindeki sırasına göre eth0 ve eth1 olarak isimlendirilirler. Numaralandırma her zaman 0'dan başlar.

Loopback (lo) arayüzü makinanın kendisine dönen özel bir arayüzdür. Ağ testlerini yapmak için kullanılır. Bu aygıta yapılacak erişimlere yine kendisi cevap verir. Böylece TCP/IP protokolü gerçek bir IP olmadan çalıştırılmış olur. Bazı uygulamalar gerçek anlamda bu arayüzü kullanıyor olabilir. Örneğin sunucu üstündeki bir kullanıcıya mail göndermek isteyen mail transfer ajanı.

Linux altında 127.0.0.1/8 özel ağındaki bütün aygıtlar loopback aygıtına atanmıştır. Genelde 127.0.0.1 IP adresi kullanılır. Bazı Linux dışı işletim sistemleri sadece 127.0.0.1'i kullanıyor.

ethtool, mii-tool

Ethtool, kart hızını, portu, auto-negotiation, PCI lokasyonu gibi değerleri sorgulamak ve değiştirmek için kullanılan ağ kartı yapılandırma aracıdır.

Bir ağ arayüzüne ait istatistikleri almak için:

```
# ethtool –S eth0
```

Bir ağ arayüzü hakkında bilgi almak için:

```
# ethtool –i em1
driver: e1000e
version: 2.0.0-k
firmware-version: 0.13-3
bus-info: 0000:00:19.0
supports-statistics: yes
supports-test: yes
supports-eeprom-access: yes
supports-register-dump: yes
```

```
# ethtool eth0
Settings for eth0:
    Supported ports: [ TP ]
    Supported link modes:   10baseT/Half 10baseT/Full
                            100baseT/Half 100baseT/Full
                            1000baseT/Full
    Supported pause frame use: No
    Supports auto-negotiation: Yes
    Advertised link modes:  10baseT/Half 10baseT/Full
                            100baseT/Half 100baseT/Full
                            1000baseT/Full
    Advertised pause frame use: No
    Advertised auto-negotiation: Yes
    Speed: 1000Mb/s
    Duplex: Full
```

```
Port: Twisted Pair
PHYAD: 0
Transceiver: internal
Auto-negotiation: on
MDI-X: Unknown
Supports Wake-on: d
Wake-on: d
Current message level: 0x00000007 (7)
                drv probe link
```

Arayüzler üzerinde değişiklik yapmak için –s parametresi kullanılır.

```
--speed number devname
--duplex [half | full] devname
--port value devname
--autoneg [yes | no] devname
--phyad HEX-VALUE devname
--xcvr [internal | external] devname
--wol [pumbgsd] devname
```

```
# ethtool –s eth0 speed 100 duplex full autoneg off
```

Fiziksel arayüz hakkında bilgi görüntüleyen diğer bir araç da mii-tool'dur.

```
# mii-tool
eth0: negotiated 1000baseT-FD flow-control, link ok
```

Ağ Yapılandırması

Ağ kartlarına IP vermek ve devreye almak için ifconfig komutu kullanılır. Hiçbir parametre vermeden kullanılırsa mevcut yapılandırmayı gösterir.

```
# ifconfig
eth0  Link encap:Ethernet  HWaddr 00:50:56:b2:6d:8a
      inet addr:172.16.45.219 Bcast:172.16.47.255 Mask:255.255.248.0
      inet6 addr: fe80::250:56ff:feb2:6d8a/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
      RX packets:7811035 errors:0 dropped:1879 overruns:0 frame:0
      TX packets:136191 errors:0 dropped:0 overruns:0 carrier:0
```

```
      collisions:0 txqueuelen:1000
      RX bytes:549190868 (523.7 MiB)  TX bytes:27303204 (26.0 MiB)

lo    Link encap:Local Loopback
      inet addr:127.0.0.1  Mask:255.0.0.0
      inet6 addr: ::1/128 Scope:Host
      UP LOOPBACK RUNNING  MTU:16436  Metric:1
      RX packets:996 errors:0 dropped:0 overruns:0 frame:0
      TX packets:996 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:0
      RX bytes:149463 (145.9 KiB)  TX bytes:149463 (145.9 KiB)
```

Ağ arayüzünü aktif etmek ve IP/netmask ikilisini tanımlamak için şu şekilde kullanılır:

```
# ifconfig eth0 172.16.45.219 netmask 255.255.248.0 up
```

Aktif edilmiş arayüzü kapatmak için:

```
# ifconfig eth0 down
```

Bilgisayarın IP adresi olması ağ üzerinde paket alışverişi yapmasını temin eder. Ancak başka ağlara ulaşamaz. Bir ağdaki bilgisayarın kendi ağında olmayan bir hedefe ulaşabilmesi için yönlendirme girilmesi gerekmektedir.

Nereye yönlendirileceği bilinmeyen her paketin yönlendirildiği ağ geçidine varsayılan ağ geçidi denir ve şu şekilde tanımlanır:

```
# route add default gw 10.0.0.1
```

Bunun dışında belirli ağlar veya IP’ler için aynı komut kullanılarak yönlendirme yapılabilir.

Aşağıdaki yönlendirme 10.0.2.X ağına giden istekleri eth1 arayüzüne yönlendirir:

```
# route add -net 10.0.2.0 netmask 255.255.255.0 eth1
```

Aşağıdaki örnekte 10.0.2.5 IP'li hosta erişimleri eth1 üzerinden yapmak üzere bir yönlendirme yapılıyor:

```
# route add -host 10.0.2.5 eth1
```

Yazılan bütün yönlendirmeler yönlendirme tablosunda tutulur. Bir paket hedefe doğru yola çıktığında bu tablo okunarak paketin nasıl yönlendirileceğine karar verilir. Herhangi bir yönlendirme bilgisi bulunmayan hedefler için varsayılan ağ geçidine yönlendirme yapılır.

Yönlendirme tablosunu görmek için:

```
# route -n
Kernel IP routing table
Destination   Gateway   Genmask  Flags Metric Ref    Use Iface
10.0.0.0      0.0.0.0   255.255.255.0   U   0   0     0  eth0
169.254.0.0   0.0.0.0   255.255.0.0     U   0   0     0 eth0
0.0.0.0       10.0.0.1  0.0.0.0         UG  0   0     0 eth0
```

0.0.0.0 ile belirtilen hedef bütün ağ anlamına gelip varsayılan ağ geçidine yönlendirilir. Dolaysıyla yukarıdaki yönlendirme tablosunda varsayılan ağ geçidi son satırdaki tanım olup 10.0.0.1'dir. netstat –rn komutu da aynı çıktıyı verir.

Komut satırından yapılacak yapılandırmalar bilgisayar yeniden başlatıldığında geçersiz olacaktır. Bilgisayar her açıldığında yapılandırmanın yeniden yapılabilmesi için yapılandırma dosyaları kullanılır. Linux dağıtımları ağ arayüzünü aktif eden açılış betiklerine sahiptir. Bu betikler yapılandırma dosyalarını okuyarak komut satırından verilen yapılandırma komutlarını her açılışta çalıştırır.

Ağ arayüzlerini kontrol eden ana yapılandırma dosyası /etc/network/interfaces dosyasıdır.

```
# cat /etc/network/interfaces
auto lo
 iface lo inet loopback
```

```
auto eth0
 iface eth0 inet static
      address 172.16.45.219
      netmask 21
      gateway 172.16.41.1
```

Bu dosyada her bir ağ arayüzü için bir blok oluşturulur ve bu blokta IP adresi, netmask ve varsayılan ağ geçidi tanımlanır. Ağ maskesi CIDR veya 4 noktalı biçimde olabilir.

Kalıcı özel yönlendirmeler eklemek için ilgili arayüz bloğuna aşağıdaki gibi post-up (arayüz up olduktan sonra) ve pre-down (arayüz down olmadan önce) satırları eklenir:

```
post-up ip route del -net 10.1.2.0 netmask 255.255.255.0
post-up ip route add -host 10.1.2.51 eth1
pre-down ip route add -net 10.1.2.0 netmask 255.255.255.0
pre-down ip route add -host 10.1.2.51 eth1
```

Ağ yapılandırması yapılmış bir sistemde ağ arayüzlerini başlatmak için ifup komutu kullanılır. Bu komut `/etc/network/interfaces` gibi yapılandırma dosyalarını okuyarak arayüzleri yapılandırır.

```
# ifup
# ifdown
```

Ağ hizmetlerini başlatmak için:

```
# /etc/init.d/networking start
```

Ağ hizmetlerini yeniden başlatmak için:

```
# /etc/init.d/networking restart
```

Gnome grafik ortamda ağ yapılandırmasını yapmak için network-admin yapılandırma aracı kullanılır.

ip komutu

Linux 2.2 çekirdek sürümüyle birlikte gelişmiş fonksiyonlar sunan iproute2 paketi getirildi. Bu paketin içindeki ana araç 'ip' komutu olup, ağ bağlantılarını ve arayüzlerini, yönlendirme tablolarını, yönlendirme politikalarını, ARP tablolarını ve ağ tünellerini yapılandırmak için esneklik sunmaktadır.

ifconfig, route gibi komutların yerini artık daha güçlü ve esnek olan ip komutu almakta, açılış betikleri bu komutu kullanmaktadır.

ip ile ağ arayüzüne IP adresi tanımlamak için:

```
# ip addr add 192.168.50.5 dev eth1
```

Arayüzler hakkında bilgi almak için:

```
# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state
UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
pfifo_fast state UP qlen 1000
    link/ether 00:50:56:b2:6d:8a brd ff:ff:ff:ff:ff:ff
    inet 172.16.45.219/21 brd 172.16.47.255 scope global eth0
    inet6 fe80::250:56ff:feb2:6d8a/64 scope link
       valid_lft forever preferred_lft forever
```

Arayüze ait IP adresini silmek için:

```
# ip addr del 192.168.50.5/24 dev eth1
```

Ağ arayüzünü aktif etmek için:

```
# ip link set eth1 up
```

Ağ arayüzünü kapatmak için:

```
# ip link set eth1 down
```

Yönlendirme tablosunu görüntülemek için:

```
# ip route show
10.0.0.0/24 dev eth0  proto kernel  scope link  src 10.0.0.100
169.254.0.0/16 dev eth0  scope link
default via 10.0.0.1 dev eth0
```

Sabit yönlendirme eklemek için:

```
# ip route add 10.0.2.0/24 via 192.168.5.10 dev eth0
```

Sabit yönlendirmeyi silmek için:

```
# ip route del 10.0.2.0/24
```

Sarsayılan ağ geçidini tanımlamak için:

```
# ip route add default via 192.168.50.100
```

Aşağıdaki tabloda net-tools paketinde yer alan komutlara karşılık gelen ip komutları verilmiştir:

net-tools	iproute2
ifconfig	ip addr
route	ip route
arp	ip neigh
ipmaddr	ip maddr
iptunnel	ip tunnel
nameif	ifrename
mii-tool	ethtool

IPv6

IPv4 protokolünün tasarımından kaynaklanan sorunların aşılması ve verimliliğin sağlanması IPv6 protokolünün tasarımında görülmüştür. IPv6 datagramları içerisinde taşınabilecek verinin IPv6 başlığına oranı IPv4 protokolünde hesaplanandan çok daha fazladır.

IPv6 128 bitlik adres büyüklüğüne sahiptir. Bu nedenle 32 bitlik IPv4 yapısına oranla daha büyük miktarlarda adres tanımlanmasına olanak sağlar.

IPv4 başlık alanı içerisinde yer alan bazı bölümler bu protokol içerisinde “isteğe bağlı” olarak tanımlanarak performans artırılmıştır.

Tercih alanlarında meydana gelen değişim IP datagramların daha kolay yönlendirilmesini sağlar.

IPv6 protokolüne eklenen bazı özellikler sayesinde IPv4 protokolünün tasarımsal yapısından kaynaklanan problemlerin önüne geçilmeye çalışılmıştır. IPv6 protokolü, veri akışı sırasında belirli bağlantılara öncelik tanımlanmasını sağlayan belirteçler kullanır ve bağlantılara daha iyi hizmet sunulması sağlayabilir.

Bağlantılar arasında doğruluğun kesinleştirilmesi (authentication), verilerin gerçekten bağlantı veri dizisi içerisinde yer aldığının tespiti, verilerin gizliliğinin sağlanması IPv6 protokolü mimarisi içerisinde tanımlanan tedbirlerdir.

Dahili IPSec ile uçtan uca güvenlik geldi.

Kapsamlı multicast desteği getirildi.

IPv6 yapılandırması ***/etc/network/interfaces*** dosyasında yapılır:

```
#IPV6 static configuration
iface eth0 inet6 static
pre-up modprobe ipv6
address 2607:f0d0:2001:000a:0000:0000:0000:0002
netmask 64
gateway 2607:f0d0:2001:000a:0000:0000:0000:0001
# END IPV6 configuration
```

Ipv6 yönlendirmeyi kontrol etmek için:

```
# ip -6 route show
```

Ipv6 ping atmak için:

```
# ping6 ipv6.google.com
```

Host DNS Yapılandırması

İster sunucu olsun ister çalışma istasyonu veya masaüstü, Linux'un internete veya komşu bilgisayarlara isimleriyle ulaşabilmesi için DNS istemci ayarlarının yapılması gerekir.

Bir bilgisayara ismiyle ulaşmanın en kolay yolu /etc/hosts dosyasında isim adres eşleşmesi yapmaktır.

```
# cat /etc/hosts
127.0.0.1  localhost
10.34.1.21 wless21.example.com
172.16.45.2     pardus
```

Dünyadaki bütün bilgisayarların isim ve ip'lerini bu dosyaya yazmak mümkün olmadığından hiyerarşik bir yapı olan DNS sistemi geliştirilmiştir. Bu sisteminin bilşenleri şu şekildedir:

İş istasyonu/istemci <---> DNS cache <---> DNS Sunucu

DNS cache, istemcilerin çözümleme isteklerine cevap veren, DNS sunuculara sorgu yapan, yaptığı sorguları tekrar kullanmak üzere saklayan sunuculardır.

Linux'u iş istasyonu yada masaüstü olarak kullandığımızda DNS çözümlemelerini yapması için /etc/resolv.conf içinde DNS cache tanımı yapmak gerekir.

```
# cat /etc/resolv.conf
nameserver 8.8.8.8
nameserver 4.4.4.4
```

Birden fazla isim çözümleme mekanizması olduğundan (dosyalar, DNS, NIS...) bunların hangi sırayla bakılacağını belirlemek lazım.

/etc/nsswitch.conf yapılandırma dosyası bazı hizmetler için sıralamayı belirler. Bilgisayar ismi bulma (hosts mu DNS mi NIS mi), kullanıcıları bulma (passwd mi LDAP mı?) gibi işlemler için sıralamayı belirler.

```
# cat /etc/nsswitch.conf
passwd:         compat
```

```
group:          compat
shadow:         compat

hosts:          files mdns4_minimal dns mdns4
networks:       files

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       nis
```

host Komutu

DNS sorguları yapan basit bir araçtır. İsimleri IP adresine çevirir.

host [-t type] [-4] [-6] {name} [server]

name: IP'ye çevrilecek alan adı.
server: Sorgu yapılacak DNS cache. Belirtilmezse /etc/resolv.conf'ta belirtilen kullanılır.
type: DNS sorgu türü.

```
$ host www.pardus.org.tr
www.pardus.org.tr is an alias for portal.pardus.org.tr.
portal.pardus.org.tr has address 193.140.100.202
```

Bir alan adına ait mail sunucuyu bulmak için

```
$ host -t mx pardus.org.tr
pardus.org.tr mail is handled by 5 mail.pardus.org.tr.
```

Bir etki alanına ait yetkili DNS sunucuyu bulmak için:

```
$ host -t ns pardus.org.tr
pardus.org.tr name server ns2.ulakbim.gov.tr.
pardus.org.tr name server ns1.ulakbim.gov.tr.
```

dig Komutu

DNS sorgularını yapmak için kullanılan nslookup'ın yerini almış daha kapsamlı bir komuttur. Genel biçimi:

```
$ dig @server name type
```

server: DNS cache sunucu
name: Sorgulanacak alan adı
type: Sorgu türü, ANY, A, MX, SIG gibi.

```
$ dig www.google.com
```

Dig ile birlikte kullanılan bazı seçenekler:

+[no]tcp

TCP kullan veya kullanma, normalde DNS sorguları UDP ile yapılır

+[no]all

Bütün görüntülüme seçeneklerini aç veya kapat

+time=T

Sorgu için T sn'lik bir zaman aşımı

+[no]answer

DNS cevabını görüntüle veya görüntüleme

```
$ dig pardus.org.tr NS +noall +answer

; <<>> DiG 9.8.4-rpz2+rl005.12-P1 <<>> pardus.org.tr NS +noall
+answer
;; global options: +cmd
pardus.org.tr.        20909 IN    NS    ns1.ulakbim.gov.tr.
pardus.org.tr.        20909 IN    NS    ns2.ulakbim.gov.tr.
```

Ağ Sorunlarını Çözme

ping Komutu

Hedef bilgisayara ICMP ECHO_REQUEST göndererek göndererek ICMP_ECHO_RESPONSE bekler. Basit bir ICMP haberleşmesidir. Ping komutunun amacı hedef bilgisayarın ayakta olup olmadığını, paket kaybı oranını ve cevap süresini tespit etmektir.

Aşağıda çok geç cevap veren (156 ms) bir bilgisayar görülmektedir:

```
$ ping www.example.com
PING www.example.com (93.184.216.119) 56(84) bytes of data.
64 bytes from 93.184.216.119: icmp_req=1 ttl=47 time=156 ms
64 bytes from 93.184.216.119: icmp_req=2 ttl=47 time=155 ms
64 bytes from 93.184.216.119: icmp_req=3 ttl=47 time=156 ms

--- www.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time
10498ms
rtt min/avg/max/mdev = 155.899/156.049/156.244/0.478 ms
```

5 sn aralıklarla ping atmak için:

```
$ ping -i 5 www.google.com
```

Belli sayıda ICMP isteği göndererek sonlandırmak için (örnekte 2 istek gönderilmekte):

```
$ ping -c 2 173.194.39.169
PING 173.194.39.169 (173.194.39.169) 56(84) bytes of data.
64 bytes from 173.194.39.169: icmp_req=1 ttl=44 time=84.0 ms
64 bytes from 173.194.39.169: icmp_req=2 ttl=44 time=84.9 ms

--- 173.194.39.169 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 84.026/84.464/84.903/0.526 ms
```

Sadece ping istatistiklerini görmek için:

```
$ ping -c 5 -q 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.

--- 127.0.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 3996ms
rtt min/avg/max/mdev = 0.021/0.029/0.033/0.005 ms
```

traceroute Komutu

Hedef bilgisayara giderken paketlerin yönlendirme neticesinde hangi geçitlerden geçerek ilerlediğini görmek için kullanılır. Bu sayede hedef bilgisayara ulaşmada bir sıkıntı varsa sıkıntının aradaki hangi noktada olduğu tespit edilebilir.

Aşağıda komutun verildiği bilgisayardan belirtilen hedefe (www.pardus.org.tr) 9 farklı yönlendiriciden geçilerek gidilmektedir. Aradaki bir yönlendiricide sorun olursa komut orda takılıp kalacaktır. Böylece sorunlu nokta bulunmuş olacak.

```
# traceroute www.pardus.org.tr -n
traceroute to www.pardus.org.tr (193.140.100.202), 30 hops
max, 60 byte packets
 1  172.16.41.1  0.397 ms  0.471 ms  0.519 ms
 2  195.214.145.105  3.385 ms  3.390 ms  3.387 ms
 3  172.16.0.1  0.987 ms  1.030 ms  1.075 ms
 4  85.29.23.97  3.551 ms  3.586 ms  3.583 ms
 5  82.222.224.9  4.660 ms  4.661 ms 10.36.1.109  4.761 ms
 6  10.36.2.94  9.379 ms  9.249 ms  9.195 ms
 7  10.36.1.54  8.407 ms  7.972 ms  7.935 ms
 8  85.29.25.10  9.596 ms  8.992 ms  7.992 ms
 9  193.140.100.202  7.265 ms  7.169 ms  7.139 ms
```

Komuttaki –n parametresi verilmezse IP adresleri için DNS çözümlemesi yapacaktır. Bu durumda çıktının gelmesi yavaşlayacaktır.

Traceroute komutu yönlendirme paketine göre yolunu bulur. Varsayılan ağ geçidi yerine özel bir arayüzden yol bulunmak isteniyorsa –i parametresi ile ağ arayüzü seçmek gerekir.

netstat

Ağ bağlantılarını, yönlendirme tablolarını ve ağ arayüzlerinin istatistiklerini görüntüler. Parametreler:

-rn: Yönlendirme tablosunu gösterir

```
# netstat -rn
Kernel IP routing table
Destination  Gateway   Genmask         Flags   MSS Window  irtt
Iface
10.0.0.0     0.0.0.0   255.255.255.0   U         0 0
0 eth0
169.254.0.0  0.0.0.0   255.255.0.0     U         0 0
0 eth0
0.0.0.0      10.0.0.1  0.0.0.0         UG        0 0
0 eth0
```

-s: Protokol istatistiklerini gösterir

```
# netstat -s
Ip:
    2374203 total packets received
    9728 with invalid addresses
    0 forwarded
    0 incoming packets discarded
    2364474 incoming packets delivered
    2196461 requests sent out
Icmp:
    5375 ICMP messages received
    1487 input ICMP message failed.
    ICMP input histogram:
        destination unreachable: 2706
        timeout in transit: 324
        redirects: 48
           [...]
```

-ta: Bütün TCP bağlantılar ve dinlenen portları gösterir

```
# netstat -ta
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address  Foreign Address  State
tcp     0      0 *:imap        *                   LISTEN
tcp     0      0 *:pop3s       *:*                 LISTEN
tcp     0      0 *:mysql       *:*                 LISTEN
```

-ua: Bütün UDP bağlantılar ve dinlenen portları gösterir

-ape: TCP, UDP ve UNIX soket bağlantıları ile bu bağlantıların durumlarını gösterir

netcat Komutu

netcat, istemci/sunucu mimarisiyle çalışan ağ trafiğini analiz etmeye yarayan bir araçtır. Ağ sorunlarını çözmek için karşıdaki bir porta istemci olarak bağlanarak sunucunun cevaplarını analiz edebilir veya bir sunucu gibi bir portu dinleyip istemcilerden gelen istekleri analiz edebilirsiniz. Böylece haberleşmeden kaynaklı sorunları çözebilirsiniz.

En temel kullanımı telnet gibi bir sunucu portuna bağlanmaktır.

```
$ nc -v google.com 80
Connection to google.com 80 port [tcp/http] succeeded!
GET /
HTTP/1.0 302 Found
Location: http://www.google.com.tr/?
gws_rd=cr&ei=u12sUu72AcXBswbt-IGYBw
Cache-Control: private
Content-Type: text/html; charset=UTF-8
[...]
```

Belli bir portta soket açıp dinlemek için:

```
$ nc -l -v 1234
Listening on [0.0.0.0] (family 0, port 1234)
```

Yetkisiz kullanıcılar 1024’ten büyük port numaralarını kullanmalı.

Nc’nin dinlediği porta dışardan veya üzerinden telnet ile bağlanıp veri gönderelim:

```
$ telnet localhost 1234
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Mrb server
```

Nc tarafında bu mesajlar görülür:

```
$ nc -l -v 1234
Listening on [0.0.0.0] (family 0, port 1234)
Connection from [127.0.0.1] port 1234 [tcp/*] accepted (family
2, sport 36652)
Mrb server
```

Bir portun açık olup olmadığını test etmek için:

```
$ nc -vz 127.0.0.1 22
Connection to 127.0.0.1 22 port [tcp/ssh] succeeded!
$ nc -vz 127.0.0.1 23
nc: connect to 127.0.0.1 port 23 (tcp) failed: Connection
refused
```

Bir port aralığındaki port erişilebilirliğini test etmek için:

```
$ nc -v -w 1 10.0.0.10 -z 20-25
nc: connect to 10.0.0.10 port 20 (tcp) failed: Connection
refused
Connection to 10.0.0.10 21 port [tcp/ftp] succeeded!
Connection to 10.0.0.10 22 port [tcp/ssh] succeeded!
nc: connect to 10.0.0.10 port 23 (tcp) failed: Connection
refused
nc: connect to 10.0.0.10 port 24 (tcp) failed: Connection
refused
nc: connect to 10.0.0.10 port 25 (tcp) failed: Connection
refused
```

-w parametresi zaman aşımı değeri belirtmek içindir.

Bir servisle karşılıklı konuşmak için:

```
$ nc -vt 10.0.0.10 21
Connection to 10.0.0.10 21 port [tcp/ftp] succeeded!
220 (vsFTPd 2.0.5)
QUIT
221 Goodbye.
```

Güvenlik

- **Konu:**

Linux ağ yönetimi

- **Hedefler:**
  - Dosya güvenliği
  - SUID bitli dosyaların bulunması
  - Parola güvenliği ve gölge parola sistemi
  - Kullanıcı erişimini kapatma
  - Kaynak kullanımını sınırlandırma
  - Açık dosya ve portları görüntüleme
  - Uzak sunucudaki açık hizmetleri tespit etme
  - Tcpwrapper ile hizmetlere erişimleri sınırlandırmak
  - Gölge parola sistemi
  - Dosya imzalama ve şifreleme
  - SSH hizmetinin yapılandırılması

Anahtar Kelimeler:

find, passwd, pwconv, lsof, nmap, chage, /etc/nologin, sudo, ulimit, usermod, lsof, nmap, xinetd, /etc/hosts.allow, /etc/hosts.deny, gpg, ssh, sshd, ssh-keygen, scp

Dosya Sistemi Güvenliği

SUID (Set User ID) Linux altındaki dosyaların özel bir yetki bitidir. Normalde bir program çalıştırıldığında çalışan kullanıcının yetkilerini miras alarak çalışır. Eğer SUID bit verilmiş ise bu durumda komut, çalıştıranın yetkileriyle değil komut dosyasının sahibinin yetkisiyle çalıştırılır. Bu yetki komutun çalışma süresince geçici olarak kullanıcıya verilir. Bunun yanında SGID (Set Group ID) biti aynı davranışı kullanıcı grupları için yapar.

Böyle bir şeye neden ihtiyaç duyulduğunu anlamak için passwd örneğini inceleyelim:

Parolalar /etc/passwd dosyasında kayıtlıdır. Normal bir kullanıcının /etc/passwd dosyasına yazma yetkisi yoktur. Bu durumda normal bir kullanıcı kendi yetkileriyle passwd komutunu vererek parolasını değiştirmesi mümkün olmayacaktı. Ancak passwd komut dosyasının etkileri aşağıdaki gibidir:

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 27936 Aug 11  2010 /usr/bin/passwd
```

Görüldüğü gibi komut dosyasının sahibi root ve yetkiler içerisinde SUID bit de (s) vardır. SUID bit olduğundan dolayı passwd komutunu çalıştıran herhangi bir komut, kendi yetkileriyle değil komut dosyasının sahibi olan root yetkileriyle işlem yapar. Bu durumda normal bir kullanıcı /etc/passwd'de değişiklik yapacak parola değiştirme işlemini gerçekleştirebilir.

Bir dosyanın SUID bitini set etmek için:

```
# chmod u+s command
```

Küçük 's' olarak görülen SUID bit komutun çalıştırılabilir olduğunu (x yetkisine sahip), büyük 'S' olarak görülen SUID bit ise komutun SUID'li ancak çalışma yetkisine sahip olmadığını gösterir. Çalışma yetkisi vermek için:

```
# chmod u+x command
```

SUID bit yetkisi fazla olduğundan yalnızca bilinen programlarda olmalı. SUID bitli bir programdaki açık sadırganın root yetkileriyle istediği işi yapmasına neden olabilir.

Find komutu kullanılarak sistemdeki SUID/SGID bitli dosyalar tespit edilmeli ve gereksiz olanlar kaldırılmalıdır.

```
# find / -type f \( -perm -04000 -o -perm -02000 \)
```

Belli dosya sistemleri için SUID biti veya çalıştırma yetkisini devre dışı bırakmak için /etc/fastab kullanılabilir. Ilgili dosya sistemine ait satıra özellik olarak ‘nosuid’ veya ‘noexec’ eklenebilir. Böylece sistem açılırken bu dosya sistemi için SUID bit veya çalıştırma yetkisi verme iptal edilebilir.

Güvenlik için diğer tehlikeli dosyalar ise herkesin yazmasına açık (chmod 777) dosya ve dizinlerdir. Bu dosyalar saldırganların hedefi olabilir ve istedikleri kodları yükleyebilecekleri bir ortamı onlara sunabilir. Herkesin yazmasına açık dosyaları tespit etmek için:

```
# find / -perm -2 ! -type l -ls
```

Verilen komutta aygıt ve sembolik linkler hariç tutulmuştur.

Tespit edilmesi gereken güvenlik açığı ortaya çıkartabilecek diğer bir dosya grubu ise sahibi olmayan dosyalardır.

```
# find / \( -nouser -o -nogroup \) –print
```

Uzaktan sunuculara giriş sağlayan .rhosts dosyaları da tespit edilmelidir:

```
# find /home -name .rhosts -print
```

Bu dosyaların tespiti silinmesini gerektirmemektedir. Çoğu durumda tehlikeli bu dosyaların olması gerekmektedir. Buradaki amaç bunları tespit ederek bildiğimiz bir amaç dışında bu tehlikenin var olup olmadığının tespitidir.

Gölge Parola Sistemi

Linux’un ilk zamanları parolalar doğrudan /etc/passwd dosyasında şifreli olarak tutulmakta idi. Bu dosya herkes tarafından okunabildiğinden güvenlik tehlikesi oluşturmaktaydı. Bu nedenle Shadow suite geliştirildi. Shadow suite aşağıdaki orjinal programları değiştirir:

su, login, passwd, newgrp, chfn, chsh ve id

Suitle birlikte yeni programlar da gelmektedir:

chage, newusers, dpasswd, gpasswd, useradd, userdel, usermod, groupadd, groupdel, groupmod, groups, pwck, grpck, lastlog, pwconv, and pwunconv

Yapılandırma dosyası olarak /etc/login.defs 'i kullanır.

Shadow parola sisteminde kullanıcılara ait şifreli parola dışındaki bilgiler /etc/passwd'te tutulmaya devam ederken şifreli parolalar root dışında kimsenin okuyamayacağı /etc/shadow dosyasında tutulmaktadır. Yetkisiz kullanıcılar sisteme girişlerinde veya parola değiştirdiklerinde bu dosyaya SUID bit sayesinde erişir. SUID bit için lütfen 10.1 bölümüne bakınız.

Normal passwd dosyasını Shadow dosyasına çevirmek için pwconv komutu kullanılır.

Parola Güvenliği

Linux altında kullanıcılar root tarafından useradd komutuyla eklenir. Useradd komutunun nasıl davranacağını aşağıdakiler belirler:

- Komut satırından verilen parametreler
- /etc/login.defs yapılandırma dosyası
- /etc/default/useradd yapılandırma dosyası

```
# useradd -m -d /home/user1 -s /bin/bash -c "User 1" -U user1
# passwd user1
Changing password for user user1.
New UNIX password:
BAD PASSWORD: it is based on a dictionary word
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

Kullanıcı hesabının bir süre sonra kapanması için –e (expire) parametresi kullanılır. Biçimi:

-e {yyyy-mm-dd}

Kullanıcı açılırken otomatik olarak ev dizinine kopyalanması gereken yapılandırma dosyaları /etc/skel içerisindedir. Örneğin .bashrc dosyası gibi.

Usermod

Linux altında var olan kullanıcı bilgilerini ve özelliklerini değiştirmek için usermod komutu kullanılır. Useradd komutuyle neredeyse aynıdır. Tek farkı useradd sıfırdan kullanıcı oluştururken usermod mevcut kullanıcı bilgilerini düzenler:

```
# usermod [-c comment] [-d home_dir [-m]] [-e expire_date] [-f inactive_time] [-g initial_group] [-G group [,...]] [-l login_name] [-p passwd] [-s shell] [-u uid [-o]] [-L|-U] login
```

Kullanıcı geçerlilik süresi tanımlamak için:

```
# usermod -e 2010-10-10 user15
```

Kullanıcıyı geçici olarak devre dışı bırakmak için:

```
# usermod –L user18
```

Bu durumda kullanıcının kodlanmış /etc/shadow dosyasında şifresinin başına ! işareti konularak şifre devre dışı bırakılır.

Kullanıcı Erişimini Kapatma

Bazı durumlarda kullanıcıların kabuğa erişimlerini engellemek gerekebilir. Örneğin IMAP hizmetinden yararlanan bir kullanıcının kabuğa erişimi olmasına gerek yoktur. Bu tür kullanıcıların kabuğu /bin/false veya /sbin/nologin yapılır.

Kabuğu /bin/false olan bir kullanıcı sisteme giriş yapar yapmaz kabuk sonlanır ve 1 ile çıkış yapar. /sbin/nologins ise bir hata mesajıyla çıkış yapar. Her iki durumda da kullanıcı Bash'e ulaşamaz.

```
# usermod -s /sbin/nologin user13

$ ssh user13@10.0.0.10
user13@10.0.0.100's password:

Welcome to Pardus

This account is currently not available.
Connection to 10.0.0.10 closed.
```

Nologins için hata mesajı ***/etc/nologin.txt*** dosyasından kişiselleştirilebilir.

Eğer sistemde /etc/nologin dosyası root hariç bütün kullanıcıların erişimleri engellenecek ve kullanıcılara bu dosyanın içeriği gösterilecektir. Pam_nologin.so modülünün aktif olmasını gerektirir.

Chage

Kullanıcı, parolasını belirli zaman aralıklarında değiştirtmek zorunda bırakılabilir. Chage komutu ile en son parola değişikliğinden kaç gün sonra parolanın değiştirilmesi gerektiği ayarlanabilir.

Parolanın yaş durumunu göstermek için:

```
$ chage -l user1
Last password change                                    : Sep 26, 2013
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires: 7
```

Normal kullanıcılar yalnızca –l parametresini kullanmaya yetkilidir. Diğer parametreler için yetkili kullanıcı olmak gerekir:

```
$ chage -d 2013-12-01 user1
chage: Permission denied.
```

-d parametresi ile parolanın en son değiştirildiği tarih (YYYY-AA-GG biçiminde) set edilebilir.

```
# chage -d 2012-12-01 user1
# chage -l user1
Last password change                                    : Dec 01, 2012
```

-E parametresi ile parolanın hangi tarihte (YYYY-AA-GG biçiminde) geçersiz olacağı belirtilir.

```
# chage -E 2013-12-01 user1
```

Zaman aşımına uğramış kullanıcı şu uyarı mesajını alır:

```
user1@10.0.0.100's password:
Your account has expired; please contact your system
administrator
```

-W parametresi kullanılarak parola geçersiz olmadan belirtilen gün kadar önce uyarı verilmesi sağlanabilir. Öntanımlı olarak 7 gün önce haber verilir.

/etc/login.defs

Linux shadow parola sisteminin yapılandırma /etc/login.defs dosyasıdır. Bu dosyanın olmaması sistemin işlememesine neden olur. Global oturum açma kuralları buradan tanımlanır.

Bütün parolaların periyodik değiştirilmesini zorlamak için aşağıdaki anahtar yapılandırma kelimeleri kullanılır:

PASS_MAX_DAYS: Bir parolanın geçerli olabileceği en fazla gün sayısı. Parola yaşı bu değeri geçerse değiştirilmesi için kullanıcı zorlanır. Herhangi bir değer belirtilmemesi veya -1 olması bir sınırlama olmaması anlamına gelir.

PASS_MIN_DAYS: Bir parolayı değiştirebilmek için geçmesi gereken en az gün sayısı. Bundan daha erken parola değiştirme istekleri red edilecektir.

PASS_WARN_AGE: Bir parolanın süresi dolmadan kaç gün önce uyarı vermeye başlayacağı.

Bu üç değer yeni oluşturulacak hesaplara etki eder, mevcutlar üzerinde herhangi bir etkisi yoktur.

Passwd

Parola değiştirmek için passwd komutu kullanılır:

```
$ passwd
Changing password for user user1.
Changing password for user1
(current) UNIX password:
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

Normal kullanıcılar kendi parolalarını değiştirebilirken yetkili kullanıcı başka kullanıcıların da parolalarını değiştirebilir.

```
# passwd user1
```

Ulimit ve limit.conf

Kabuk ve kabuk tarafından başlatılmış süreçlerin kullanacağı sistem kaynaklarını sınırlandırmak için kullanılır. Bash'in gömülü komutlarındandır yani bu komutun gereğini Bash kabuk yapar, ayrıca ulimit isimli bir komut dosyası yoktur.

```
# ulimit [-SHacdflmnpstuv [limit]]
```

İki çeşit sınır vardır:

-H: Hard limit, yalnız yetkili kullanıcı tarafından artırılabilir
-S: Soft limit , normal kullanıcılar tarafından artırılabilir

Eğer sınır verilmezse mevcut sınır değeri gösterilir. Bütün geçerli sınır değerlerini göstermek için:

```
$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 8192
max locked memory       (kbytes, -l) 32
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 10240
cpu time               (seconds, -t) unlimited
max user processes              (-u) 8192
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

Bu sınırlandırma değerlerinin hepsi değiştirilebilir. Örneğin sürekli fork yaparak yeni süreçler başlatılırsa bir süre sonra sistem kaynakları tükenecek ve sunucu cevap veremez hale gelecektir. Bunu engellemek için:

```
# ulimit –u 64
```

-c: Core dosyaların boyut sınırı
-f: Oluşturulabilecek en büyük dosya büyüklüğü
-n: En fazla açılabilecek dosya sayısı
-t: Kullanılabilecek en fazla işlemci zamanı (saniye)
-u: Bir kullanıcının oluşturabileceği en fazla süreç sayısı

/etc/security/limits.conf dosyası kullanılarak sınırlandırmalar kalıcı ve bütün kullanıcılar için yapılabilir. Pam_limits.so modülü tarafından kullanılır. Sınırlandırmalar her bir oturum için ayrı ayrıdır.

Limits.conf dosyasının biçimi şu şekildedir:

Kim Tür Kaynak Değer

Kim: Sınırlamanın uygulanacağı kullanıcı veya gruplar. Gruplar @groupname şeklinde gösterilir. Öntanımlı sınırlama için * kullanılır.
Tür: Sınırlandırmanın türü, hard veya soft
Kaynak: Hangi kaynağın sınırlandırılacağı
Değer: Sınırın ne olacağı

Sınırlandırılabilecek bazı kaynaklar:

Kaynak	Açıklama
core	Kb olarak core dosya boyutu
fsize	Dosya boyutu
nofile	Açık dosya sayısı
cpu	Dk olarak işlemci zamanı
nproc	Açık süreç sayısı
maxlogins	Bir kullanıcı için eş zamanlı oturum sayısı
maxsyslogins	Sistemde kullanıcılar tarafından açılabilecek oturum sayısı

Örnekler:

```
*               soft    core            0
*               hard    nofile          512
@student        hard    nproc           20
@faculty        soft    nproc           20
@faculty        hard    nproc           50
ftp             hard    nproc           0
@student        -       maxlogins       4
:123            hard    cpu             5000
@500:           soft    cpu             10000
600:700         hard    locks           10
```

PAM modülleri giriş ve yetkilendirmeleri düzenleyen güçlü bir çerçeve sunmaktadır.

su

Başka bir kullanıcıya geçiş yapmak için kullanılır.

Su komutu verildiğinde sistem root parolasını sorar. Parola doğrulandıktan sonra kullanıcı artık yetkili kullanıcı (root) olarak işlem yapar. Parametre verilirse herhangi bir normal kullanıcıya da geçiş yapılabilir.

BSD gibi UNIX türevi işletim sistemlerinde su komutunu herkes kullanamasın diye belirli gruba (wheel) su komutu çalıştırma yetkisi tanınmıştır. Linux'ta da bu özelliği kullanabilmek için /etc/pam.d/su dosyasında aşağıdaki tanım yapılmalıdır:

```
auth       required     pam_wheel.so use_uid
```

Artık bu komutu kullanmasına izin verilecek kullanıcıları wheel grubuna eklemek gerekir:

```
# usermod -G wheel <username>
```

Normal kullanıcılara, geçiş yapılacak kullanıcının parolası sorulurken yetkili kullanıcıya parola sorulmaz.

```
$ su user13
Password:
```

Bu şekilde kullanımında sadece kullanıcı değişir, kullanıcı ortamı değişmez. Kullanıcı ortamını da değiştirmek için:

```
$ su - user13
```

Bu durumda user13 kullanıcısına sıfırdan giriş yapılmış gibi başlangıç betikleri çalıştırılır ve çevre değişkenleri ayarlanır.

sudo ve sudoers

Sudo ile güvenilir bir kullanıcı kendi şifresini girerek istenen bir yetki ile işlem yapabilir.

```
$ cat /etc/shadow
cat: /etc/shadow: Permission denied
$ sudo cat /etc/shadow
root:$1$RweCoHMa$r9/977cwKrigKplI/Wdnv1:15974:0:99999:7:::
bin:*:15701:0:99999:7:::
daemon:*:15701:0:99999:7:::
```

Sudo daha esnek bir yapı sunar. Sadece `/etc/sudoers` dosyasında yer alan kullanıcılar yetki kazanır ve bu kullanıcılar yetkili işlemleri kullanıcı kabuğunda yapar, root kabuğunda değil. Root kullanımı tamamen kaldırılır.

Sudo girişleri /`var/log/messages` ve detayları `/var/log/secure` dosyasında loglanır.

Sudo'nun diğer bir avantajı da kullanıcı bazlı yetki sınırlandırması yapılabiliyor olmasıdır. Farklı kullanıcılar farklı komutlar için yetkilendirilebilir.

Sudo yapılandırma dosyası `/etc/sudoers` içinde hangi kullanıcı hangi komutu hangi IP'den ve parola gerekip gerekmediği tanımları yapılır:

User1 kullanıcısı parola girerek bütün komutları her yerden çalıştırabilir:

```
user1 ALL=(ALL) ALL
```

Users grubu üyeleri shutdown komutunu localhost (yani sunucunun kendisi) üzerinden parola girerek verebilir.

```
%users localhost=/sbin/shutdown -h now
```

Wheel grubu üyeleri parola girmeden heryerden her komutu çalıştırabilir.

```
%wheel ALL=(ALL) NOPASSWD: ALL
```

Asterisk kullanıcısı aşağıdaki komutları parola girmeden her yerden çalıştırabilir:

```
asterisk ALL = NOPASSWD: /sbin/shutdown
asterisk ALL = NOPASSWD: /usr/bin/yum
asterisk ALL = NOPASSWD: /bin/touch
asterisk ALL = NOPASSWD: /bin/chmod
```

Sudoers dosyasını daha okunabilir yapmak için alias'lar tanımlanabilir. Bu sayede belli komut seti, kullanıcı listesi ve bağlantı noktası tanımlanabilir.

```
User_Alias      OPERATORS = user1, user3, user5
Runas_Alias     OP = root, operator
Host_Alias      OFNET = 10.1.2.0/255.255.255.0
Cmnd_Alias      PRINTING = /usr/sbin/lpc, /usr/bin/lprm

Host_Alias CUNETS = 128.138.0.0/255.255.0.0
Host_Alias CSNETS = 128.138.243.0, 128.138.204.0/24
Host_Alias SERVERS = master, mail, www, ns
```

Sudo'nun genel iki kullanımı vardır:

İstenen kullanıcı hesabının kabuğuna su komutuyla giriş yapılır ve ondan sonra bu kabukta o kullanıcının yetkileriyle istenen komutlar çalıştırılır

```
$ sudo su – (Yetkili kullanıcı kabuğuna erişim)
# cat /etc/shadow
$ sudo su – user1 (user1'in kabuğuna erişim)
user1$ cat myfile.txt
```

Yada kabuğa giriş yapmadan sudo ile komutu çalıştırıp kendi kabuğuna dönmek:

```
$ sudo cat /etc/shadow
```

Tabi bu ikincisi için sudo yapacak kullanıcıya sudoers içinde bu komut için yetki verilmiş olması gerekir.

lsof

Açık dosyaları görüntülemek için kullanılır.

```
# lsof
COMMAND   PID    USER   FD   TYPE   DEVICE    SIZE     NODE
NAME
init        1    root  cwd    DI     253,0    4096        2 /
init        1    root  rtd    DIR    253,0    4096        2 /
init        1    root  txt    REG    253,0   43496   451683
/sbin/init
init        1    root  mem    REG    253,0   23360   999947
/lib64/libdl-2.5.so
```

Her bir açık dosya için bir satır gösterir.

TYPE sütunu açık dosyanın cinsini belirtir.

REG: Normal dosya
DIR: Dizin
FIFO: FIFO hat
CHR: Karakter aygıt

Bildiğimiz bir dosyayı hangi süreç kullanıyor bulmak için:

```
# lsof /var/log/messages
COMMAND  PID USER   FD   TYPE DEVICE  SIZE  NODE NAME
syslogd 2302 root    1w   REG  253,0 37826 70887
/var/log/messages
```

Yukarıdaki komutun anlamı şu: /var/log/messages dosyasını syslogd hizmeti açmıştır.

Belirtilen bir dizin altındaki açık dosyaları görüntülemek için:

```
# lsof +D /var/log/
```

Belli bir patternle başlayan bir komutun açtığı dosyaları bulmak için:

```
# lsof -c httpd
```

Belli bir kullanıcı tarafından açılmış dosyaları listelemek için:

```
# lsof -u simsek
```

Belirli bir PID numarasına ait açık dosyaları görüntülemek için:

```
# lsof -p 15653
```

Linux altında herşeyin bir dosya olduğunu ve ağ bağlantılarının da buna dahil olduğunu unutmayın. Ağ bağlantılarını tespit etmek için:

```
# lsof -i
```

```
COMMAND     PID     USER   FD   TYPE DEVICE SIZE NODE NAME
portmap    2336      rpc    3u  IPv4   7001       UDP *:sunrpc
rpc.statd  2376  rpcuser    3u  IPv4   7173       UDP *:859
rpc.statd  2376  rpcuser    6u  IPv4   7159       UDP *:856
sshd       2616     root    3u  IPv4   8968       TCP *:ssh
(LISTEN)
xinetd     2634     root    5u  IPv4   9036       UDP *:tftp
ntpd       2650      ntp   16u  IPv4   9052       UDP *:ntp
mysqld     2743    mysql   10u  IPv4   9212       TCP *:mysql
(LISTEN)
```

Belli bir port numarasını kullanan süreçleri tespit etmek için:

```
# lsof -i :25
```

nmap

lsof komutu yerel makina üzerinde hangi hizmetlerin ve portların açık olduğunu gösterirken nmap, uzak sunucuları tarayarak açık port ve hizmetleri tespit eder.

-O: Sunucu OS bilgisini iste
-p: Port numarası veya aralığı belirtmek için
-V: Porttaki hizmetin sürüm bilgisini al
-F: Yalnızca Nmap'in tanıdığı portları tara
-v: Detay bas

Aşağıdaki yapılandırılmamış basit bir nmap komutu vardır.

```
$ nmap www.microsoft.com

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2013-
12-07 13:10 EET
Interesting ports on 65.55.57.27:
Not shown: 1678 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap finished: 1 IP address (1 host up) scanned in 156.078
seconds
```

OS ve Sürüm bilgisi (-A) verecek daha hızlı (-T4) çalışacak bir komut:

```
$ nmap -A -T4 -F www.microsoft.com

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2013-
12-07 13:09 EET
Interesting ports on 65.55.57.27:
Not shown: 1237 filtered ports
```

```
PORT    STATE SERVICE  VERSION
80/tcp  open  http     Microsoft IIS webserver 8.0
443/tcp open  ssl/http Microsoft IIS webserver 8.0
Service Info: OS: Windows

Nmap finished: 1 IP address (1 host up) scanned in 68.656
seconds
```

Çıktıda görüldüğü gibi sunucu üzerinde 80 ve 443 nolu portlar açık ve bu portlarda Microsoft IIS web sunucu yazılımının 8.0 sürümü çalışmaktadır.

Port tarama teknikleri:

-sS: SYN taraması, yarım bağlantılı tarama da denir. Sadece SYN gönderilerek cevap alınır, TCP bağlantı kurulmaz
-sT: TCP bağlantı taraması
-sU: UDP bağlantı taraması, DNS gibi UDP hizmetler için
-sA: TCP ACK taraması
-sO: IP protokol taraması

NMAP çıktısının biçimlendirilmesi:
-oN filename: Verilen dosyaya normal biçimde yazar
-oX filename: Verilen dosyaya XML olarak yazar.

Bazı taramaların yapılması için root yetkisi gerekir.

```
$ nmap -sU 10.0.0.10
You requested a scan type which requires root privileges.
Sorry dude.

QUITTING!
$ sudo su -
# nmap -sU 10.0.0.10

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2013-
12-07 13:29 EET
Interesting ports on hq.srv.endersys.com (10.0.0.10):
Not shown: 1475 closed ports
PORT     STATE         SERVICE
53/udp   open|filtered domain
67/udp   open|filtered dhcps
69/udp   open|filtered tftp
111/udp  open|filtered rpcbind
137/udp  open|filtered netbios-ns
138/udp  open|filtered netbios-dgm
514/udp  open|filtered syslog
765/udp  open|filtered webster
797/udp  open|filtered unknown
970/udp  open|filtered unknown
973/udp  open|filtered unknown
2049/udp open|filtered nfs
MAC Address: 00:1A:64:71:92:7C (Unknown)
```

```
Nmap finished: 1 IP address (1 host up) scanned in 1499.014
seconds
```

Tcp Wrapper

Linux, ağa dahil olup bir istemci olarak ağ hizmetlerinden yararlanabildiği gibi uygun yapılandırma ile kendisi de pek çok ağ hizmeti sunabilmektedir. Örneğin telnet, ftp, ssh, talk vs...

Linux altında bütün ağ isteklerini kabul eden ve ilgili uygulamaya aktaran super-daemon veya TCP wrapper olarak isimlendirilen yada “servislerin servisi” diyebileceğimiz bir servis bulunmaktadır.

TCP Wrapper, bütün hizmetleri tek bir merkezden yönetmeyi sağlar. Bütün hizmetler TCP Wrapper yapılandırma dosyası ile aktif edilirken /etc/hosts.{allow,deny} dosyaları ile de erişim denetlenir.

Bazı ağ hizmetleri (http gibi) inetd dışında kendi başlarına çalışabilirler. Bu durumda bu hizmet çalıştığı portu kendisi dinlemektedir. Bu tür sunuculara standalone sunucu denmektedir. Bir uygulama istenildiğinde stand-alone veya TCP wrapper ile çalışacak şekilde yapılandırılabilir.

Hangi hizmetin hangi portta çalışacağı /etc/services dosyasında belirtilmiştir. Örneğin http için TCP port 80, smtp mail için TCP port 25 kullanılmaktadır.

```
# service-name port/protocol [aliases ...] [# comment]
ftp 21/tcp
ssh 22/tcp # SSH Remote Login Protocol
ssh 22/udp # SSH Remote Login Protocol
telnet 23/tcp
```

Inetd eskiden beri kullanılan bir TCP wrapper uygulamasıdır.

Inetd yapılandırma dosyası /etc/inetd.conf dosyasıdır. Aşağıda ftp ve telnet servisinin yapılandırması bulunmaktadır. En son sütun servisi verilecek ilgili komutu ve parametrelerini içermektedir.

```
ftp    stream      tcp    nowait      root  /usr/sbin/tcpd     in.ftpd -l -a
telnet      stream      tcp    nowait      root  /usr/sbin/tcpd     in.telnetd
```

Inetd eski, yavaş ve güvenlik zaafiyetine neden olan bir tasarıma sahiptir. Bunun yerine alternatif olarak xinetd geliştirilmiştir. Xinetd'nin sundukları:

- Erişim kontrolü (kendi içinde tcpwrapper vardır)
- Gelişmiş loglama (bağlantı zamanı, başarısız istekler...)
- Başka bir bilgisayardan hizmet yönlendirme
- Ipv6 desteği
- Tek bir yapılandırma dosyası yerine her bir hizmet için gelişmiş bir yapılandırma dosyası

/etc/xinetd.conf yapılandırma dosyası xinetd ‘nin kontrolündeki tüm hizmetleri etkiler.

```
defaults
{
       instances          = 60
       log_type           = SYSLOG authpriv
       log_on_success     = HOST PID
       log_on_failure     = HOST
       cps                = 25 30
}
includedir /etc/xinetd.d
```

cps (connection per second): Sn’de 25 bağlantı sonrasında 30 sn bekleme.

En sondaki /etc/xinetd.d dizininde hizmetlere ait yapılandırmalar yer alır.

```
service telnet
{
       flags           = REUSE
       socket_type     = stream
       wait            = no
       user            = root
       server          = /usr/kerberos/sbin/telnetd
       log_on_failure  += USERID
       disable         = yes
}
```

disable satırı hizmetin aktif olup olmayacağını belirler.

/etc/hosts.{allow,deny}

TCP Wrapper bir erişime izin verip vermeyeceğine karar vermek için hosts.allow ve hosts.deny dosyalarına bakar.

Öncelikle hosts.allow dosyasına bakar. Satır satır bu dosyayı okuyarak mevcut bağlantı için giriş izni arar. Bulursa bağlantıyı kabul eder. Bulamazsa hosts.deny dosyasını okur ve bir yasaklama kuralı olup olmadığına bakar. Varsa bağlantıyı reddeder yoksa bağlantıyı kabul eder.

Dolaysıyla bir bağlantının kabul edilmesi için ya allow kuralı olmalı yada hakkında bir deny kuralı olmamalı.

İlk önce hosts.allow dosyası okunduğundan bir bağlantıyı kabul eden bir allow kuralı ve reddeden bir deny kuralı aynı anda varsa bağlantı kabul edilir.

Bir bağlantı için herhangi bir kural yoksa o bağlantı kabul edilir.

TCP Wrapper önbellek tutmadığından hosts.allow ve hosts.deny dosyalarına yazılacak birşey hizmeti yeniden başlatmaya gerek olmaksızın anında geçerli olacaktır.

Her iki dosyanın kural biçimi aynı olup şu şekildedir:

<daemon list>: <client list> [: <option>: <option>: ...]

Örneğin aşağıdaki kural vsftpd servisine example.com domaininden yapılan (host1.example.com gibi alt hostları da kapsar) istekleri ifade eder. Bu kuralı hosts.allow'a yazmanız erişim izni anlamına gelirken hosts.deny dosyasına yazmanız erişimi red anlamına gelir. Yani kural biçimi aynı, kuralın anlamı ise içinde bulunduğu dosyaya göredir.

vsftpd : .example.com

Aşağıdaki satırın hosts.deny dosyasında olması hosts.allow da olmayan bütün erişim isteklerini red edecektir.

ALL: ALL

Güvenli bir kullanım için hosts.deny'da herşeyi red edip sadece gerekenleri hosts.allow ile açmak tavsiye edilir.

Bazı terimler:

ALL: Herşey
LOCAL: İçinde nokta (.) olmayan bütün domainler (localhost, mail, test gibi)

Aşağıdaki yapılandırma ile 192.168.X.X IP'lerinin tamamı kast edilmektedir.
ALL : 192.168.

Hosts.allow'a yazılacak aşağıdaki kural ile telnet hizmetine, yalnızca belirtilen dosyada yazılan hostlar erişebilecektir:

in.telnetd : /etc/telnet.hosts

GPG ile Şifreleme ve Sayısal İmza

Gpg, şifreleme ve sayısal imza atmak için kullanılan bir araçtır.

Bir dosyayı şifrelemek için:

$ gpg -c secret.txt
Enter passphrase:
Repeat passphrase:

Bu işlemin sonucunda secret.txt.gpg şifreli dosyası oluşturulmuştur.

Şifreli dosyayı açmak için:

$ gpg -o output.txt secret.txt.gpg
gpg: CAST5 encrypted data
gpg: encrypted with 1 passphrase
gpg: WARNING: message was not integrity protected

Geri açılan dosyanın orjinal dosya ile aynı olduğunu kontrol edin:

$ ls -l secret.txt output.txt
-rw-r--r-- 1 simsek wheel 461672 Dec 7 16:05 output.txt
-rwxr-xr-x 1 simsek wheel 461672 Dec 7 16:04 secret.txt
$ diff secret.txt output.txt
$

SSH Hizmeti

Linux makineye uzaktan bağlanmak için gerekli bir servistir. Ayrıca ssh isimli program sayesinde uzaktaki sshd servisi açık makinelere bağlanılabilir.

Ssh istemcisinin yapılandırma dosyası /etc/ssh/ssh_config iken sshd servisinin yapılandırması /etc/ssh/sshd_config dosyasında yapılır.

Pardus üzerinden uzak makineye ssh ile bağlanmak için aşağıdaki gibi bir komut çalıştırılır.

ssh kullanıcadı@IPadresi veya ssh –l kullanıcıadı IPadresi

```
# ssh ahmet@172.16.1.10
# ssh –l ahmet 172.16.1.10
```

SSH Sunucu Yapılandırması

/etc/ssh/sshd_config dosyasındaki bazı kritik parametreler şunlardır:

Protocol 2

Ön tanımlı olarak SSHD protokolünün 2. Sürümü kullanılır. 1. Sürüm güvenlik açısından riskli olduğu için kullanılmamalıdır.

PermitRootLogin yes

Ön tanımlı olarak root kullanıcısının ssh ile sisteme bağlanmasına izin verilir. Bu bazı güvenlik risklerini getirdiği için bu parametrenin değerini no yapılması tavsiye edilir. Root ile işlem yapmak isteyen kullanıcıların standart bir kullanıcı ile ssh bağlantısı yaptıktan sonra su – veya sudo ile root işlemleri yapması daha uygundur.

Yapılan değişiklerin etkin olması için sshd servisi yeniden başlatılmalıdır.

```
# service sshd restart
```

Parolasız SSH erişimi

Karşı sunucuya parolasız olarak ssh üzerinden erişmek veya login parolasından farklı bir parola ile erişmek için RSA anahtarı oluşturma yöntemi kullanılır. Kullanıcı anahtar oluşturmak için ssh-keygen komutunu kullanabilir.

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/data/ahmet/.ssh/id_rsa):
Created directory '/data/ahmet/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /data/ahmet/.ssh/id_rsa.
Your public key has been saved in /data/ahmet/.ssh/id_rsa.pub.
The key fingerprint is:
a7:50:1a:c1:c2:5e:b3:94:cd:0e:72:38:35:7f:f6:6c ahmet@pardus
The key's randomart image is:
+--[ RSA 2048]----+
|    . +++        |
|     * O+o       |
|    . B.=o o     |
|     . .+.o o    |
|       o S . E   |
|        . o .    |
|         .       |
|                 |
|                 |
+-----------------+
```

Yukarıdaki Enter passphrase kısmında eğer bir parola girerseniz karşı sunucuya sadece bu makineden artık bu parola ile giriş

yapabilirsiniz. Bu kısmı boş geçerseniz parolasız girebileceksiniz.

Bu komut ev dizininizde .ssh/id_rsa.pub dosyasını oluşturur.

```
$ ls -al .ssh/
toplam 16
drwx------ 2 ahmet ahmet 4096 Ara  9 23:46 .
drwxr-xr-x 6 ahmet ahmet 4096 Ara  9 23:46 ..
-rw------- 1 ahmet ahmet 1675 Ara  9 23:46 id_rsa
-rw-r--r-- 1 ahmet ahmet  394 Ara  9 23:46 id_rsa.pub
```

id_rsa.pub dosyasını karşı sunucudaki bağlanacağınız kullanıcının ev dizinine .ssh/authorized_keys olarak kopyalayın. Eğer authorized_keys dosyası varsa eski dosyanın üzerine yazmayıp 46 id_rsa.pub içeriğini authorized_keys dosyasına eklemeniz gerekir.

Artık ssh veya scp ile karşı sunucudaki bu kullanıcıya parolasız bağlanabilirsiniz.

SCP ile Dosya Transferi

Scp komutu ile iki sshd servisi üzerinden dosya transferi gerçekleştirilebilir.

```
$ scp dosya kulllanıcıadı@uzakIP:/dosyayolu
```

Örnek:

```
# scp pardus.txt ahmet@172.16.45.219:/var/tmp
ahmet@172.16.45.219's password:
pardus.txt
100%    0     0.0KB/s   00:00
```