

# AWK Programlama Dili

# Nedir?

- AWK, Alfred **A**ho, Peter **W**einberger ve Brain **K**ernighan tarafından 1978 yılında geliştirilmiş ve bir çok UNIX sürümünde (özellikle System V, version 3.1'den sonrakilerde) kendisine yer bulmuş bir programlama dilidir.
- Dilin açık kaynak kodlu GNU versiyonu 1986 yılında Richard Stallman'ın tavsiyesi ile Paul Rubin ve Jay Fenlason tarafından yazılmıştır.
- GAWK'ın yazılması aşamasında ve sonrasında Brain Kernighan projeye fikirleri ve müdahaleleri ile destek olmuştur.
- Zaman içerisinde MS Windows ve Mac üzerinde çalışan versiyonları ve AWK kodunu C, C++ ve Perl kodlarına dönüştüren araçlar programcılar tarafından geliştirilmiştir.

# Nedir? (devam)

- AWK genel olarak metinleri kolayca işlemenize ve raporlar oluşturabilmenize olanak sağlayan data-driven bir programlama dilidir. Awk ile,
  - Küçük kişisel veritabanlarınızı yönetebilir, formatlı raporlar oluşturabilir,
  - Veriler üzerinde aritmetik ve string operasyonlarını gerçekleştirebilir,
  - Genel programlama yapılarını kullanabilir (kıyas operatörleri, döngüler v.s.),
  - Kabukta bir komutun çıktısını on-the-fly işleyebilir ve başka bir komutun girdisi olacak şekilde formatlayabilir,
  - Doğru yerde kullandığınızda C, Pascal gibi dillerle -nispeten- zor olan operasyonları tek satırda gerçekleştirebilirsiniz.

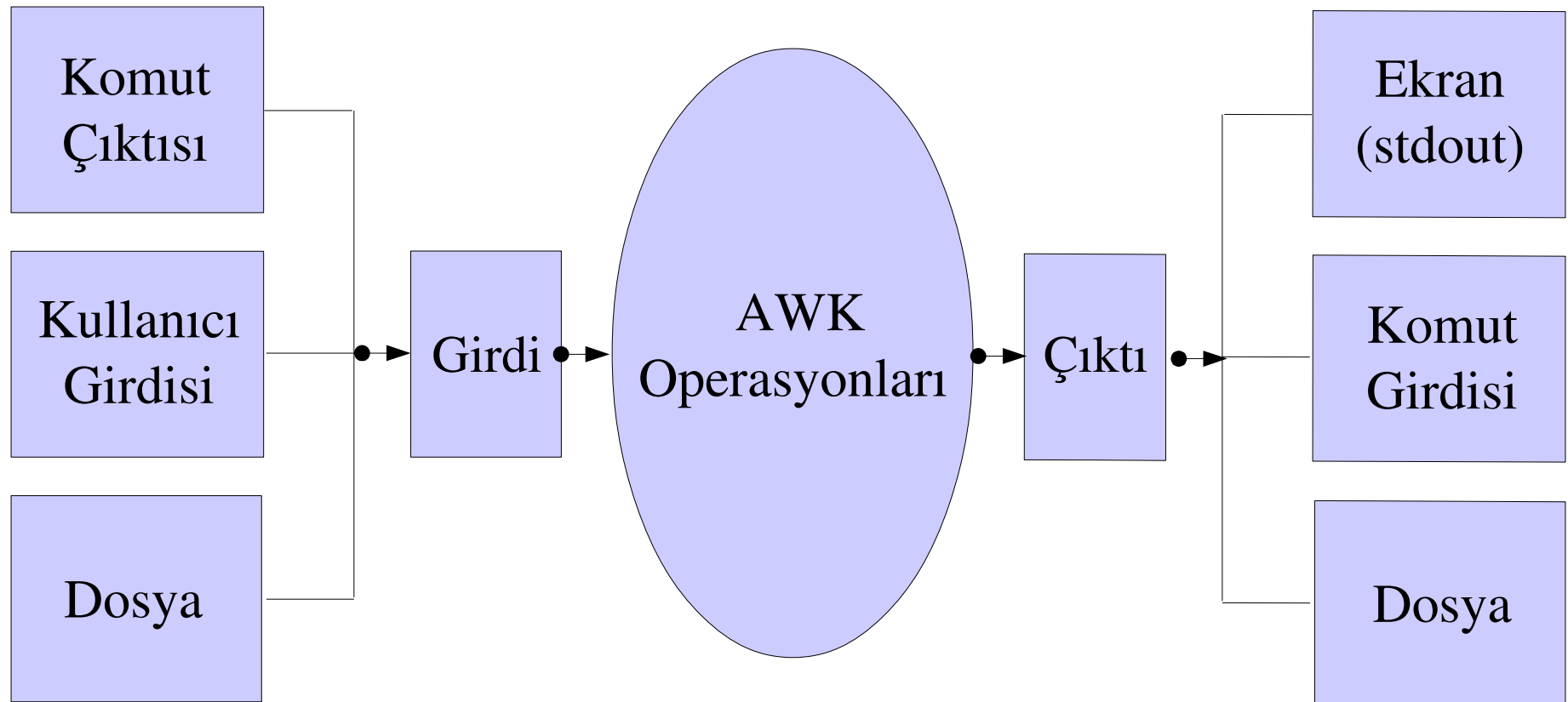
# Nasıl Çalışır?

- AWK yorumlanan bir dildir, yani AWK kodları tek başına çalıştırılabilir bir uygulamaya dönüştürülemez (!), AWK scriptlerinin bir sistemde çalışabilmesi için o sistemde AWK yorumlayıcısının bulunması gerekir.
- AWK'nin çalışma şekli genel olarak şöyledir:
  - Bir dosyayı ya da kendisine yönlendirilen bir girdiyi (örneğin bir komut çıktısını) satır satır okur,
  - Her bir girdi satırını alanlara ayırır,
  - İstedığımız durumlarla eşleşen satırlar üzerinde istediğimiz işlemleri gerçekleştirir.

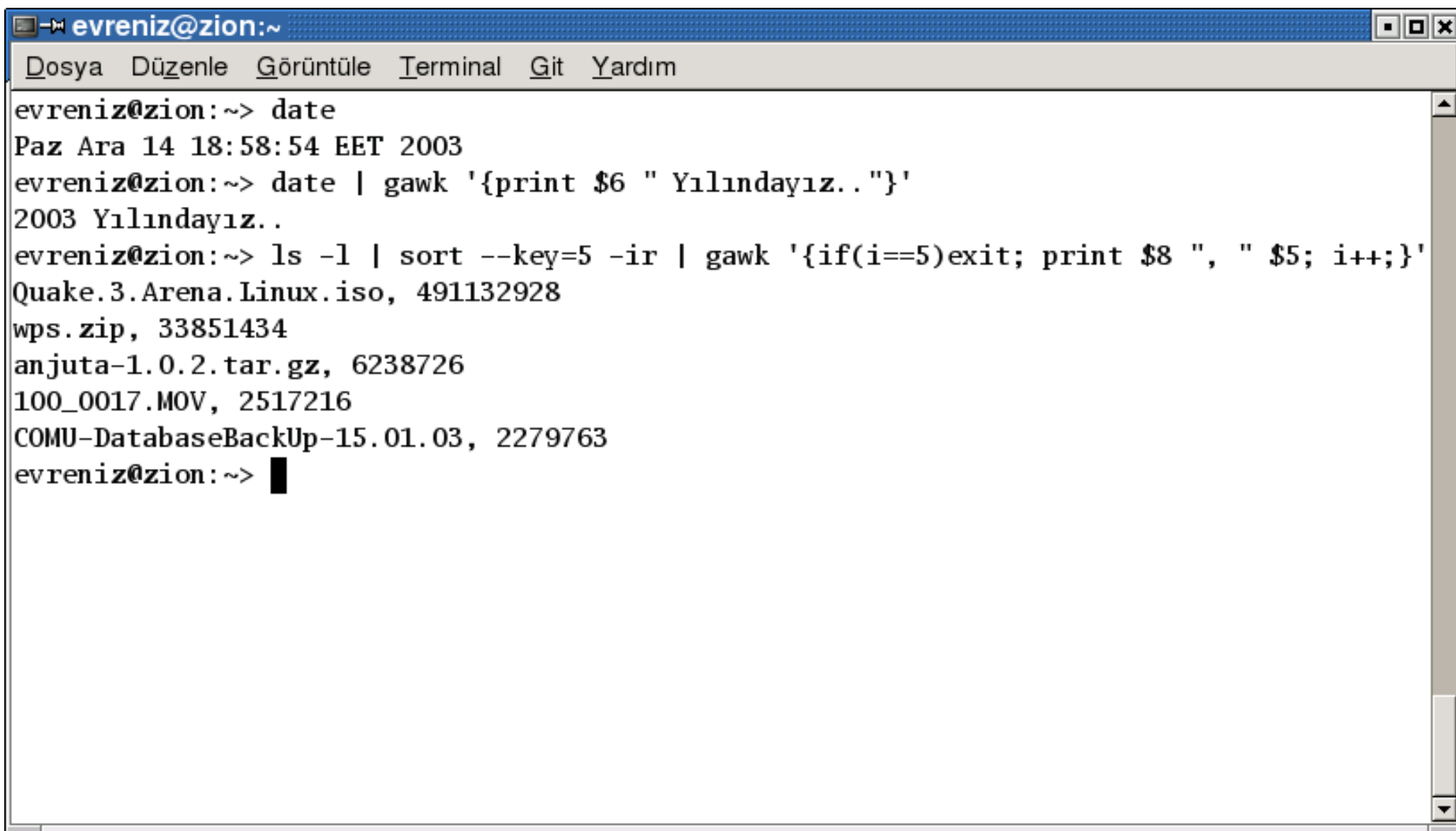
## Nasıl Çalışır? (devam)

- İster komut satırından yazdığımız AWK programı ile:  
    ]`$ awk [options] 'program_kodu' dosyalar`  
    (`awk -F : '/evreniz/ {if($5>1000)print $1 $2}' kullanıcı_dosyaları`)
- İstersek de daha önce bir dosyaya yazmış olduğumuz AWK programı ile çalıştırabiliriz:  
    ]`$ awk [options] -f script.awk dosyalar`  
    (`awk -f rutin_control.awk /etc/passwd`)
- *info awk* ya da *man awk* ile seçenekler ve kullanım ile ilgili ayrıntılı bilgi alınabilir.

# Nasıl Çalışır? (devam)



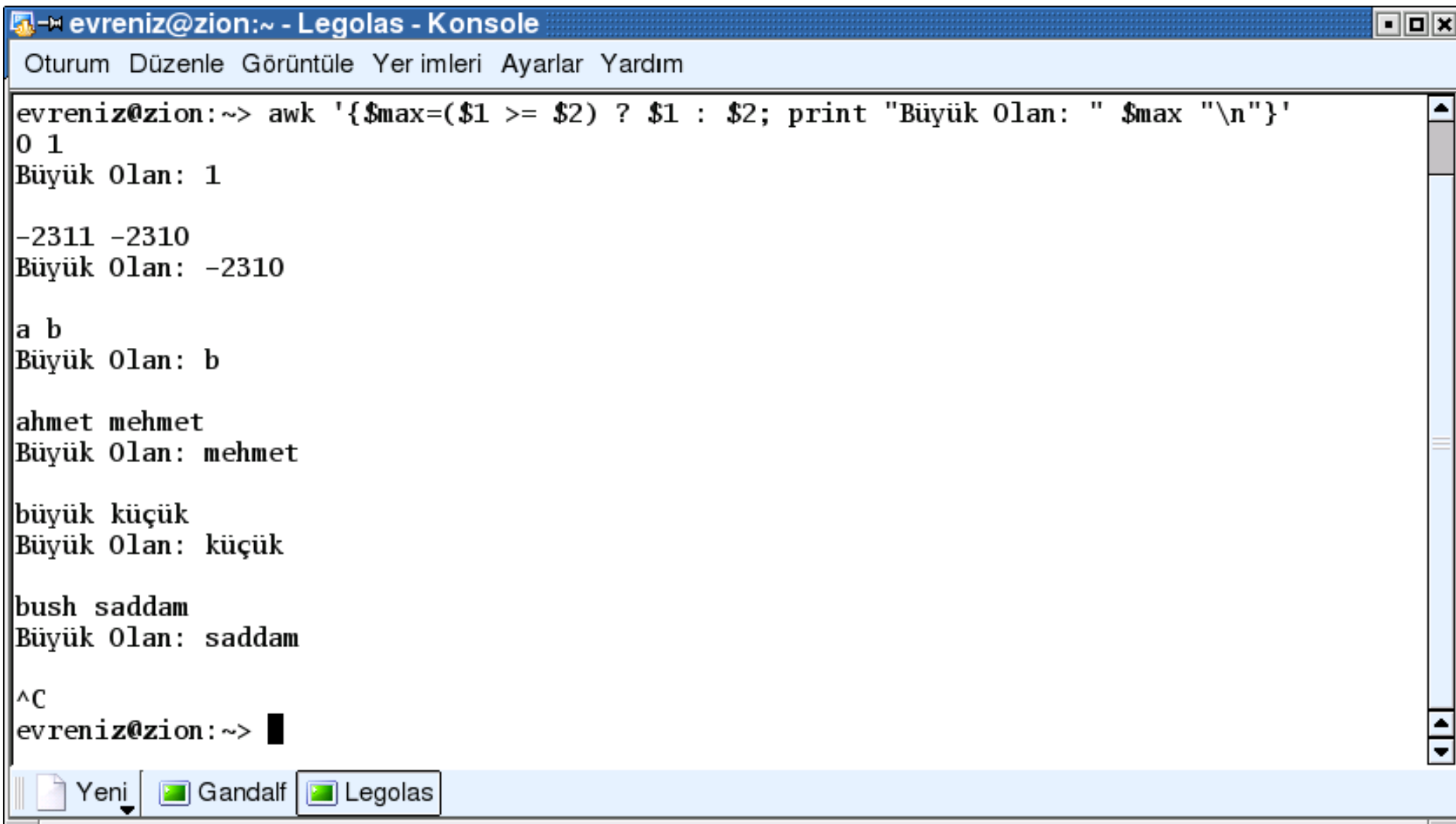
# Komut Çıktısı (örnek)



The image shows a terminal window titled "evreniz@zion:~". The window has a menu bar with options: Dosya, Düzenle, Görüntüle, Terminal, Git, and Yardım. The terminal displays the following commands and their outputs:

```
evreniz@zion:~> date
Paz Ara 14 18:58:54 EET 2003
evreniz@zion:~> date | gawk '{print $6 " Yılındayız.."}'
2003 Yılındayız..
evreniz@zion:~> ls -l | sort --key=5 -ir | gawk '{if(i==5)exit; print $8 " ", " $5; i++;}'
Quake.3.Arena.Linux.iso, 491132928
wps.zip, 33851434
anjuta-1.0.2.tar.gz, 6238726
100_0017.MOV, 2517216
COMU-DatabaseBackup-15.01.03, 2279763
evreniz@zion:~> █
```

# Kullanıcı Girdisi (örnek)



```
evreniz@zion:~ - Legolas - Konsole
Oturum Düzenle Görüntüle Yer imleri Ayarlar Yardım

evreniz@zion:~> awk '{ $max=($1 >= $2) ? $1 : $2; print "Büyük Olan: " $max "\\n"}'
0 1
Büyük Olan: 1

-2311 -2310
Büyük Olan: -2310

a b
Büyük Olan: b

ahmet mehmet
Büyük Olan: mehmet

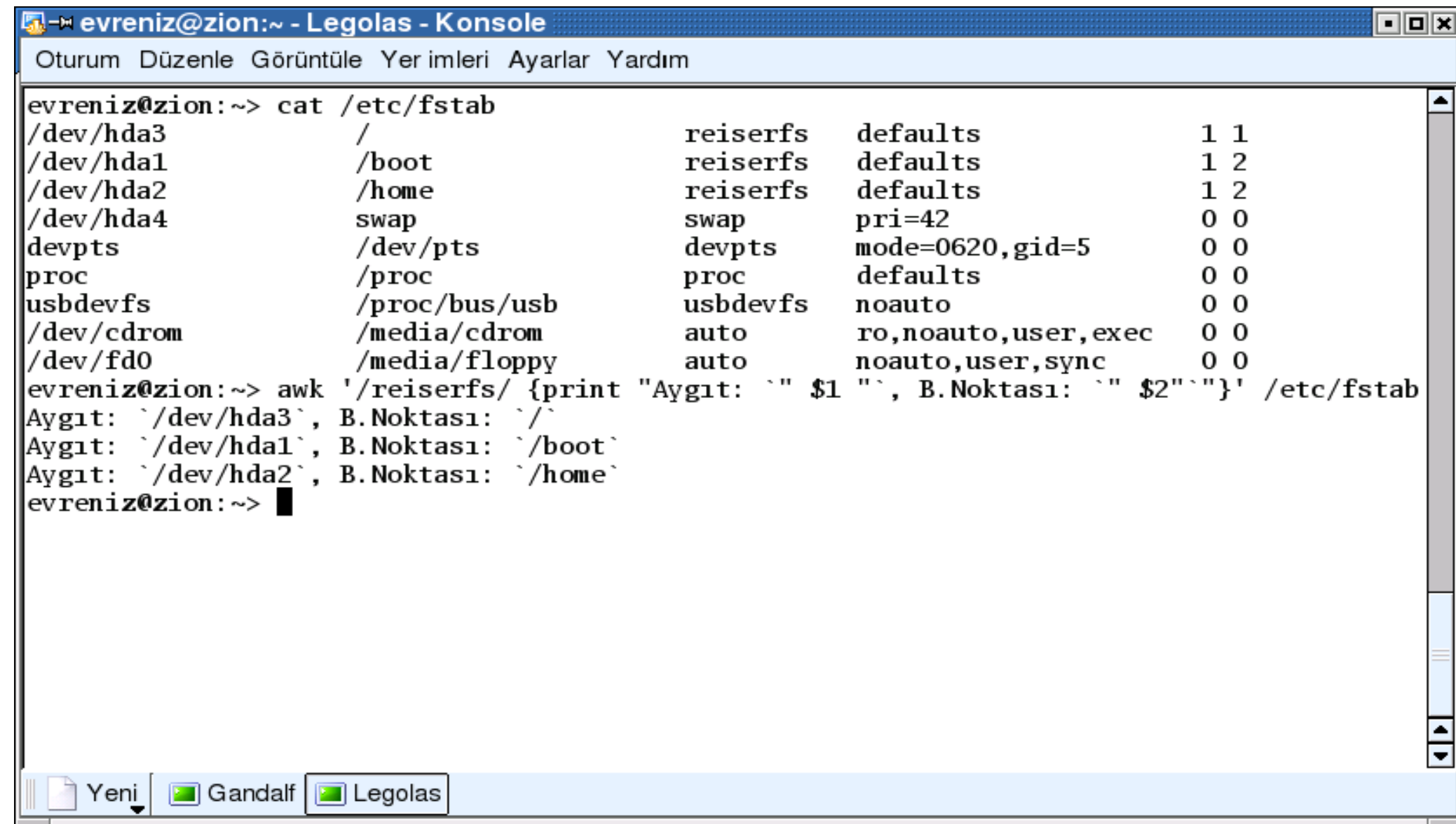
büyük küçük
Büyük Olan: küçük

bush saddam
Büyük Olan: saddam

^C
evreniz@zion:~> █
```



# Dosyadan (örnek)



```
evreniz@zion:~> cat /etc/fstab
/dev/hda3          /                  reiserfs  defaults          1 1
/dev/hda1          /boot             reiserfs  defaults          1 2
/dev/hda2          /home             reiserfs  defaults          1 2
/dev/hda4          swap              swap      pri=42            0 0
devpts             /dev/pts          devpts    mode=0620,gid=5   0 0
proc              /proc            proc      defaults          0 0
usbdevfs           /proc/bus/usb     usbdevfs  noauto            0 0
/dev/cdrom         /media/cdrom      auto      ro,noauto,user,exec 0 0
/dev/fd0           /media/floppy     auto      noauto,user,sync  0 0
evreniz@zion:~> awk '/reiserfs/ {print "Aygıt: `" $1 "` , B.Noktası: `" $2"\"}`' /etc/fstab
Aygıt: `/dev/hda3`, B.Noktası: `/`
Aygıt: `/dev/hda1`, B.Noktası: `/boot`
Aygıt: `/dev/hda2`, B.Noktası: `/home`
evreniz@zion:~>
```

# Pattern ve Action

- AWK betikleri, kalıp (pattern) ve eylem (action)'lerden meydana gelirler.

```
]$ awk 'pattern {action} pattern {action} pattern {action} ... '
```

- Eğer eylemin başında bir kalıp yoksa eylem tüm kayıtlar için gerçekleştirilir.
- Eğer kalıbın ardından bir eylem yoksa tüm uygun kayıtlar hiç bir işleme tabi tutulmadan ekrana yazılır.
- Kalıplar düzenli ifadelerden, mantıksal kıyas ifadelerinden ya da virgül ile birbirini takip eden kalıp dizilerinden oluşabilir. Ayrıca BEGIN ve END adında iki özel kalıp vardır.

# Pattern ve Action

BEGIN {eylem; eylem; ...}

```
{  
...  
bütün kayıtlar için  
ana hesaplama  
ve operasyonlar  
...  
}
```

END {eylem; eylem; ...}

BEGIN {eylem; eylem; ...}

/kelime/ {eylem; eylem; ...}

.....  
\$1 !~ /regexp/ {eylem; ...}

.....  
\$1 == 25 {eylem; eylem; ...}

.....  
\$1 > \$5 {eylem; eylem; ...}

END {eylem; eylem; ...}

# BEGIN ve END

- BEGIN ve END özel tanımlanmış iki kalıptır. BEGIN kalıbından sonra gelen eylem ilk girdi satırının işlenmesinden hemen önce, END kalıbından sonra tanımlanmış eylem ise son girdi satırının işlenmesinden hemen sonra çalışır.

```
BEGIN {  
    FS=":";  
    counter=0;  
}
```

```
/root/ {  
    counter++;  
}
```

```
END {  
    print "root kelimesi geçen " counter " satıra rastlandı"  
}
```

```
root:x:0:0:root:/root:/bin/bash  
bin:x:1:1:bin:/bin:/bin/bash  
daemon:x:2:2:Daemon:/sbin:/bin/bash  
...  
]$ awk -f script.awk /etc/passwd  
root kelimesi geçen 1 satıra rastlandı  
]$
```

# Built-In Değişkenler

- AWK çalışma zamanında kimi bilgilere kolaylıkla erişim için bir takım özel değişkenler ile beraber çalışır.

FILENAME	Yürürlükteki dosyanın adı
FS	Girdi alan ayracı (öntanımlı olarakSPACE)
RS	Girdi kayıt ayracı (öntanımlı olarak \n)
NF	Yürürlükteki kayıdın alan sayısı
NR	Yürürlükteki kayıdın satır numarası
OFS	Çıktı alan ayracı (öntanımlı olarak SPACE)
ORS	Çıktı kayıt ayracı (öntanımlı olarak \n)
\$0	Girdi kaydının tümü
\$n	Girdi kaydının n. alanı.

# Operatörler

- Mantıksal Kıyas Operatörleri
  - && (Mantıksal VE)
  - || (Mantıksal VEYA)
  - ! (Mantıksal DEĞİL)
- Aritmetik Operatörler
  - \*, /, +, - (Çarpma, Bölme, Toplama, Çıkarma)
  - % (Modül)
  - ^ (Üs)

# Operatörler (devam)

- İlişkisel Operatörler

- < (Küçük)
- > (Büyük)
- == (Eşit)
- != (Eşit Değil)
- <=, >= (Küçük Eşit, Büyük Eşit)
- ~ (regexp ile Uygun [ x ~ /regexp/ ])
- !~ (regexp ile Uygun Değil [ x !~ /regexp/ ])

# Kontrol İfadeleri

- if - else

```
if ( durum ) { yapılacaklar } else { yapılacaklar }
```

```
if ( x % 2 == 0 )  
    print “x çift sayıdır”  
else  
    print “x tek sayıdır”
```

```
]$ awk 'if( $1 % 2 == 0) print $1 “ çift sayıdır”; else print $1 “ tek  
sayıdır”’
```



# Kontrol İfadeleri (devam)

- while

```
while ( durum ) { yapılacaklar }
```

```
i=1  
while ( i <= 3 ){  
    print i  
    i++  
}
```

```
]$ awk 'BEGIN {i=3; while ( i <= 3 ) {print i; i++}}'
```

# Kontrol İfadeleri (devam)

- for

```
for ( başlangıç_durumu ; koşul ; artış ) { yapılacaklar }
```

```
for(i = 0; i <= 100; i += 5){  
    print i  
}
```

```
]$ awk 'BEGIN {for(i = 0; i <= 100; i += 5) print i}'
```

# Kontrol İfadeleri (devam)

Ayrıca AWK,

- continue
- break
- next
- exit

akış kontrol ifadelerini de destekler.

# Built-In Fonksiyonlar

- AWK'nin beraberinde gelen bir çok önceden tanımlanmış fonksiyon mevcuttur. Bu hazır fonksiyonlar da programcıya kolaylık sağlar..
  - Numerik Fonksiyonlar: `int(x)`, `sqrt(x)`, `exp(x)`, `log(x)`, `sin(x)`, `cos(x)`, `atan2(x, y)`, `rand()`, `srand(x)`, ...
  - Karakter Dizileri ile İş Yapan Fonksiyonlar: `index(in, find)`, `length(string)`, `match(string, regexp)`, `split(string, array, fieldsep)`, `sub(regexp, replacement, target)`, `substr(string, start, length)`, `tolower(string)`, `toupper(string)`, ...
  - Girdi/Çıktı Fonksiyonları: `system(command)`, ...

# Fonksiyon Tanımlama

- AWK'de -elbette- fonksiyon tanımlamak ve bunu çağırmak mümkündür, fonksiyonlar şu şekilde tanımlanır ve çağırılır:

```
function fonksiyon_adi ( parametre1, parametre2, ... ) {  
  
    fonksiyon gövdesi  
  
    ...  
    return değer  
}
```

```
fonksiyon_adi ( parametre1, “para metre 2”, parametre3, ... )
```

# Örnekler

# Örnek 1

```

evreniz@zion:~ - Gandalf - Konsole
Oturum Düzenle Görüntüle Yer imleri Ayarlar Yardım

root      1749   2.8   5.2  64636 13384 ?      S    18:41   2:08 /usr/X11R6/bin/X vt7 -auth /var/lib/xdm/authdir/authfiles/A:0-hF
root      1750   0.0   0.5   3200 1384 ?      S    18:41   0:00 -:0
evreniz   1780   0.0   0.4   4568 1252 ?      S    18:41   0:00 /bin/sh /usr/X11R6/bin/kde
evreniz   1796   0.0   0.9   7760 2376 ?      S    18:41   0:00 /opt/gnome/bin/medusa-idled
evreniz   1819   0.0   2.0  21784 5344 ?      S    18:41   0:00 kdeinit: Running...
evreniz   1822   0.0   3.1  23656 8164 ?      S    18:41   0:00 kdeinit: dcopserver --nosid
evreniz   1825   0.0   3.7  27112 9564 ?      S    18:41   0:00 kdeinit: klauncher
evreniz   1828   0.0   5.0  30504 13044 ?    S    18:41   0:00 kdeinit: kded
evreniz   1834   0.0   2.2   9752 5660 ?      S    18:42   0:01 /opt/kde3/bin/artsd -F 10 -S 4096 -s 5 -m artsmessage -l 3 -f
evreniz   1845   0.0   5.3  30984 13640 ?    S    18:42   0:00 kdeinit: knotify
evreniz   1846   0.0   0.1   1344  316 ?      S    18:42   0:00 kwrapper ksmserver
evreniz   1848   0.0   4.1  25104 10608 ?    S    18:42   0:00 kdeinit: ksmserver
evreniz   1849   0.0   5.2  27924 13324 ?    S    18:42   0:03 kdeinit: kwin -session 11c1ff6111000106389896500000101660000_107
evreniz   1851   0.0   4.6  26060 11796 ?    S    18:42   0:00 kdeinit: kwrited
evreniz   1853   0.0   6.4  31128 16432 ?    S    18:42   0:03 kdeinit: kdesktop
evreniz   1856   0.0   5.8  31424 15088 ?    S    18:42   0:02 kdeinit: kicker
evreniz   1863   0.0   5.0  27156 12848 ?    S    18:42   0:00 susewatcher -caption SuSE Update Checker -icon kinternet.png -mi
evreniz   1864   0.0   4.8  26300 12300 ?    S    18:42   0:00 kpgp
wwwrun    1870   0.0   6.3  99408 16260 ?    S    18:42   0:00 /usr/sbin/httpd -f /etc/httpd/httpd.conf
evreniz   1875   0.0   5.3  27908 13576 ?    S    18:42   0:00 kdeinit: kmix -session 11c1ff6111000106389897300000101660008_107
evreniz   1877   0.0   5.5  27216 14136 ?    S    18:42   0:00 kget -session 1053fc353c000106680735800000017810034_1070957561_2
evreniz   1883   0.0   6.6  31776 16984 ?    S    18:42   0:02 sim -session 105f2b1091000106853402200000017800011_1070957561_12
evreniz   1884   0.0   7.8  34104 20100 ?    S    18:42   0:02 kdeinit: konqueror --preload
evreniz   1886   0.0   3.3  24908 8456 ?      S    18:42   0:00 kalamd --login
evreniz   1891   0.0   5.1  27760 13092 ?    S    18:42   0:03 extensionproxy --configfile taskbar_panelextensionrc --callbacki
evreniz   1902   2.0  19.3 120236 49536 ?    S    18:44   1:28 /opt/OpenOffice.org/program/soffice.bin /home/evreniz/Desktop/AW
evreniz   1935   0.0  19.3 120236 49536 ?    S    18:44   0:00 /opt/OpenOffice.org/program/soffice.bin /home/evreniz/Desktop/AW
evreniz   1936   0.0  19.3 120236 49536 ?    S    18:44   0:00 /opt/OpenOffice.org/program/soffice.bin /home/evreniz/Desktop/AW
evreniz   1937   0.0  19.3 120236 49536 ?    S    18:44   0:00 /opt/OpenOffice.org/program/soffice.bin /home/evreniz/Desktop/AW
evreniz   1938   0.0  19.3 120236 49536 ?    S    18:44   0:00 /opt/OpenOffice.org/program/soffice.bin /home/evreniz/Desktop/AW
evreniz   1939   0.0  19.3 120236 49536 ?    S    18:44   0:00 /opt/OpenOffice.org/program/soffice.bin /home/evreniz/Desktop/AW
evreniz   1960   0.1   4.9  30860 12748 ?    S    18:47   0:05 kdeinit: konsole
evreniz   1961   0.0   0.6   4768 1664 pts/1    S    18:47   0:00 /bin/bash
evreniz   2127   0.9  10.5  44464 27052 ?    S    19:19   0:21 kdeinit: konqueror --silent
evreniz   2128   0.0   2.1  24656 5416 ?      S    19:19   0:00 kdeinit: kio_file file /tmp/ksocket-evreniz/klauncherux6Jwb.slav
evreniz   2139   0.0   4.4  27256 11508 ?    S    19:19   0:00 kdeinit: kio_uiserver
evreniz   2227   0.0   2.6  22564 6656 ?      S    19:52   0:00 kdeinit: kio_http http /tmp/ksocket-evreniz/klauncherux6Jwb.slav
evreniz   2229   0.0   2.5  22532 6400 ?      S    19:52   0:00 kdeinit: kio_http http /tmp/ksocket-evreniz/klauncherux6Jwb.slav
evreniz   2230   0.0   2.5  22336 6456 ?      S    19:53   0:00 kdeinit: kio_http http /tmp/ksocket-evreniz/klauncherux6Jwb.slav
evreniz   2244   0.0   0.2   2512  656 pts/1    R    19:55   0:00 ps aux
evreniz@zion:~>

```

Örnek bir `ps -aux`  
çıktısı..

## Örnek 1 (devam)

`ps -aux` çıktısı, sistemde çalışan süreçlerin ve o süreçlere ait kimin tarafından ne zaman çalıştırıldığı, ne kadar işlemci ve bellek kullandığı gibi kimi bilgileri verir. Aşağıdaki tek satırlık komut ile, root kullanıcısı dışındaki kullanıcıların çalıştırdığı süreçleri öldürülebilir:

```
]# ps aux | awk '$1 !~ /root/ {if(NR > 1) print $2}' | xargs kill -9
```

Aynı çıktıdan, en çok işlemci kullanan süreç de kolaylıkla elde edilebilir:

```
]$ ps aux | awk '
BEGIN {max=0}
      {if($3>max) {user=$1; max=$3; surec=$2; cmd=$11;}}
END   {print "\n"cmd " komutu ile " user " tarafından çalıştırılmış olan " surec " id`li süreç,
        işlemcinin yüzde " max " kadarlık kısmını kullanmaktadır. \n"}'
```

Bu komutun bilgisayarımdaki çıktısı şu oldu:

```
/usr/X11R6/bin/X komutu ile root tarafından çalıştırılmış olan 1749 id`li süreç, işlemcinin
yüzde 2.6 kadarlık kısmını kullanmaktadır.
```



## Örnek 2

Aşağıdaki AWK kodu bir metin içerisinde geçen tüm kelimelerin geçiş sıklıklarını gösterir:

```
#!/usr/bin/awk -f
```

```
{
  for (i=1; i <= NF; i++){
    kelime=tolower($i)
    if(index(kelime, ",") || index(kelime, ".")){
      kelime=substr(kelime, 0, (length(kelime)-1))
    }
    freq[kelime]++
  }
}
```

```
END {
  for (kelime in freq)
    if(freq[kelime]>4)
      printf "\"%s\"  %d\n", kelime, freq[kelime]
}
```

Bu betik freq.awk adı ile kaydedilip herhangi bir dosya için,

]\$ ./freq.awk makale

biçiminde kullanabilir.

## Örnek 2 (devam)

```
evreniz@zion:~> ./freq.awk makale.1 | sort -key=2 -nr
```

```
"ve" 62  
"bir" 49  
"ile" 26  
"bu" 19  
"daha" 14  
"da" 14  
"Internet" 12  
"de" 11  
"toplumsal" 10  
"en" 10  
"şekilde" 9  
"bilgisayar" 9  
"için" 8  
"azalma" 8  
"çok" 7  
"haline" 7  
"olarak" 6  
"olan" 6  
"kadar" 6  
"arama" 6  
"ya" 5  
"son" 5
```

```
evreniz@zion:~> ./freq.awk makale.2 | sort --key=2 -nr
```

```
"bir" 82  
"ve" 57  
"yazılım" 24  
"ile" 24  
"kaynak" 23  
"özgür" 21  
"bu" 21  
"çok" 17  
"linux" 17  
"için" 17  
"da" 17  
"olarak" 13  
"gnu" 10  
"açık" 10  
"gibi" 9  
"tarafından" 8  
"sahip" 8  
"olan" 7  
"de" 7  
"bilgisayar" 7  
"üzerinde" 6  
"yeni" 6  
"sistem" 6  
"lisans" 6
```

## Örnek 3

evreniz@zion:~> cat ogrenciler.BM113

Öğrenci Adı:Numarası:Ödev 1:Ödev 2:Ödev 3:Vize Notu

Sanem Çelik:030401016:+++:74

Gökçen Eraslan:030401008:+:-+:92

Atacan Tufan:030401003:-:-+:72

Kerem Ilıca:030401010:++:-:66

İşbaran Akçayır:030401004:+++:55

Aysun Özdemir:030401017:-:-+:57

Selda Yurtdaş:030401005:+++:54

Eray Kaplan:020401038:-:+++:49

Fatih Şen:010401023:-:-:-:80

Aykut Aksoy:030401021:+++:86

Rafet Cambaz:030401010:+++:80

Şule Gök:030401009:-:+++:55

Mevlüt Uşan:0304010022:+:-:-:69

Elimizde alanları birbirinden `:` ile, kayıtları da birbirinden `\n` ile ayrılmış küçük bir veritabanı olduğunu farzedelim (benim var :)). Bu veritabanındaki bilgileri zaman zaman html'ye dönüştürüp kişilere duyurmak istiyor olalım.

## Örnek 3 (devam)

```
evreniz@zion:~> cat convert2html.awk
```

```
#!/usr/bin/gawk -f
```

```
BEGIN {
    FS=":";
    print "<TABLE cellpadding=\"8pt\" BORDER=\"2pt\" CELLSPACING=\"0pt\"\"
    print "bgcolor=\"\#ffffff\" bordercolor=\"\#000000\">\n";
}

(NR==1){
    print "<TR bgcolor=\"\#dfdfdf\">\n"
    for( i=1; i<=NF; i++ )print "<TD><center>\"$i\"</center></TD>\n";
    print "</TR>\n"
}

(NF>0 && NR>1){
    printf "<TR>\n"
    for(i=1; i<=NF; i++){
        if (i==1) print "<TD align=left>\"$i\"</TD>\n";
        else print "<TD align=center>\"$i\"</TD>\n";
    }
    print "</TR>\n";
}

END {
    print "</TABLE>\n";
}
```

Bu minik scripti komut satırından  
ogrenciler.BM113 dosyasını formatlaması  
için,

```
]$ ./convert2html.awk ogrenciler.BM113
```

komutu ile çağırabiliriz. Ekrana göndereceği  
HTML kodlarını `>` ile BM113.html  
dosyasına yönlendirebiliriz...

# Örnek 3

(sonuç)

file:/home/evreniz/BM113.html - Konqueror

Konum Düzenle Görüntüle Git Yer İmleri Araçlar Ayarlar Pencere Yardım

Konum: file:/home/evreniz/BM113.html

e-posta UYDULAR SuSE Networking LT NEC CiteSeer

Öğrenci Adı	Numarası	Ödev 1	Ödev 2	Ödev 3	Vize Notu
Sanem Çelik	030401016	+	+	+	74
Gökçen Eraslan	030401008	+	-	+	92
Atacan Tufan	030401003	-	-	+	72
Kerem Ilıca	030401010	+	+	-	66
İşbaran Akçayır	030401004	+	+	+	55
Aysun Özdemir	030401017	-	-	+	57
Selda Yurtdaş	030401005	+	+	+	54
Eray Kaplan	020401038	-	+	+	49
Fatih Şen	010401023	-	-	-	80
Aykut Aksoy	030401021	+	+	+	86
Rafet Cambaz	030401010	+	+	+	80
Şule Gök	030401009	-	+	+	55
Mevlüt Uşan	0304010022	+	-	-	69

Page loaded.

## Örnek 3 (güzel)

Elbette biraz daha uğraşarak daha güzel ve kullanışlı sonuçlar da almak mümkündür :)

Vize Sonuçları (puana göre sıralı) - Konqueror

Konum Düzenle Görüntüle Git Yer İmleri Araçlar Ayarlar Pencere Yardım

Konum: http://zion.comu.edu.tr/~evreniz/ogrenciler/BM111/puan.html

e-posta UYDULAR SuSE Networking LT NEC CiteSeer UF foldoc Webopedia

---

**BM111 Algoritma ve Programlama I dersi Vize Sonuçları**

---

\* [Numaraya göre sıralı](#) - [İsme göre sıralı](#) - Puana göre sıralı \*

No	İsim	Quiz (15/10/20003)	Vize (03/11/2003)
020401037	KAPLAN, Eray	89	96,5
030401008	ERASLAN, Gökçen	91	85
030401006	YARIMPETE, Oğuz	100	82
020401024	YALMAN, Recep	-	78,5
020401032	İLKBAHAR, İbrahim	-	78
030401001	ATTAR, Eser	65	77,5
030401003	TUFAN, Atacan	60	77,5
000401014	GİYİK, Buluş	-	76,5
030401011	TEKİN, Eren	74	74,5
030401036	YEREBASMAZ, Halit	72	67
030401016	ÖZEY, İbrahim	68	64
030401019	KOÇ, Taşkın	63	62,5
030401014	YILMAZ, Selçuk	45	61,5
030401002	ŞENTÜRK, Pınar	43	60
020401028	TEKİN, Ferhan	-	59,5
030401012	YANARDAĞ, Pınar	10	58
030401031	UÇAN, Sanem	55	55,5
030401010	CANBAZ, Rafet	67	55
030401021	AKSOY, Aykut	52	55
000401012	ÇALIŞKAN, Mehmet	-	50
020401004	KAÇAN, Kirami	-	50

# Neden AWK?

- AWK Perl, C gibi dillere nazaran daha kolaydır.
- Sözdizimi daha anlaşılırdır, bu da yeni başlayanlar için büyük bir avantaj teşkil eder.
- Ufak tefek işlerinizi halledebilecek kadar AWK, zaten biliyor olabilirsiniz, belki tek ihtiyacınız olan bir kaç deneme yapmak.
- Ayrıntılarla boğuşmadan algoritmanızı implemente edebilmenin mutluluğunu yaşarsınız.
- AWK ile ufacık programlar yaratılabilir, bu sayede çok hızlı üretilmesi ve sonuç alınması gereken küçük işler için bire birdir.
- Temiz bir Perl ya da C kodu temiz olmayan bir AWK kodundan çok daha iyidir fakat, temiz olmayan bir Perl kodunun (ya da C kodunun) yanına hiç bir şey yaklaşamaz :)

# Daha Fazlası?

- AWK için faydalı bir FAQ:  
<http://www.faqs.org/faqs/computer-lang/awk/faq/>
- GAWK Kullanım Klavuzu  
<http://www.gnu.org/manual/gawk-3.0.3/gawk.html>
- Effective GAWK Programming  
<http://zion.comu.edu.tr/~evreniz/gawkbook/egp2002.pdf>
- AWK Haber Grubu (25.800 mesaj)  
<http://groups.google.com/groups?group=comp.lang.awk>



# Teşekkürler

A. Murat Eren

[meren@comu.edu.tr](mailto:meren@comu.edu.tr)

<http://zion.comu.edu.tr/~evreniz>