

# Linux ve Uygulamalar - I

*İşletim sistemi ve uygulamalarından oluşan  
bilimsel çalışma ortamına giriş*

- ◆ Kabuk ve Komut Satırı 
  - ◆ Giriş: Linux kabuktur
  - ◆ Bazı Çevre Değişkenleri
  - ◆ Alias (Komut İsimlerini Değiştirmek)
- ◆ Kabuk İşlemleri
  - ◆ Standart giriş, çıkış ve hata' nın yönlendirilmesi
  - ◆ Süzgeç komutları
- ◆ Kabuk ve Komut Satırı
  - ◆ Özel İşaretler
  - ◆ Kütük ve Dizin Bulmak
- ◆ Kabukta Süreç Yönetimi
  - ◆ Süreçleri İzlemek ve Sonlandırmak
  - ◆ Süreç Öncelikleri ve Ardalanda Çalıştırma
- ◆ Kabuk Programlama
  - ◆ Betik Genel Yapısı ve Çalıştırma
  - ◆ Neden var ?
  - ◆ Bir Örnek
  - ◆ Bilmediğimi Biliyorum
- ◆ Uygulamalar
  - ◆ Ayrımlar
- ◆ Metin Düzenleyiciler
  - ◆ 8 Bitlik Gözlüklerimiz
- ◆ Kütüklerin Anlamlılık Ölçekleri
  - ◆ Ölçek Seçimi Sonuçtur
  - ◆ Ölçek Seçimi Anlamdır
- ◆ Matematik İşlev ve Veri Çiziciler
  - ◆ Verinin Okunabilirliğini Artırmak
- ◆ Bilgisayarlı Cebir
  - ◆ Güvenli Simgesel Hesap
- ◆ Tüm Mühendislik Hesapları
  - ◆ Octave
- ◆ Güvenli Uzak Erişim, Kütük Aktarımı ve Veri Yedeklemek
- ◆ Ararken
- ◆ Akşam Sefası
  - ◆ Ertesi günün sabahına hazırlanması beklenen aksamlık ödevler

# Kabuk ve Komut Satırı - I

## Giriş: Linux kabuktur

- ◆ Kabuk, kullanıcı ile linux işletim sistemi arasındaki en düşük seviyeli yani gerçekte olanla kullanıcı arasında en az yanıtıcı arayüzdür.
- ◆ Linux'un gücü kabuğun etkinliği ve kullanıcıya sunduğu hızdan gelir
- ◆ Linux işletim sisteminde çoğunlukla kullanılan kabuklar: **bash**, csh, ksh, sh, tcsh, zsh v.b.
- ◆ En ileri olan ve hemen her linux sürümünde varsayılan kabul edilen kabuk: bash (bourne again shell)
- ◆ Sistemde kullanılabilen kabuklar “/etc/shells” kütüğünde verilir, o an geçerli kabuğu görmek için SHELL çevresel değişkenine bakılabilir ve kullanılan kabuğu değiştirmek için chsh komutu kullanılır: yapılan değişiklik /etc/passwd kütüğünde ilgili yere yazılır ve bir sonraki oturumda etkin hale gelir.
- ◆ İleriki kısımlarda daha ayrıntılı görüleceği gibi kabuğun görevleri:
  - ➔ *programları çalıştırmak (execution)*,
  - ➔ *değişken ve isim yerine koyma (variable & name substitution)*,
  - ➔ *giriş/çıkış yönlendirme (I/O redirection)*,
  - ➔ *çevre yönetimi/denetimi (environment control) ve*
  - ➔ *betik yorumlamak (Interpreted Programming Language)*
- ◆ Çevresel değişkenler kabukta içinde bulunulan çalışma ortamıyla ilgili bilgi içerir ve sıkça kullanılırlar
- ◆ Bir çevre değişkeninin hangi değere sahip olduğunu görmek için:
  - ➔ echo \$DEĞİSKEN komutu kullanılır
  - ➔ Örnek: echo \$PATH ya da echo \$SHELL
- ◆ Bir oturumda geçerli tüm çevre değişkenlerinin neler olduğunu görmek için “printenv” komutu, çıktısı uzunca olduğundan genellikle “more” komutu ile birlikte belirli harfleri içeren çevresel değişkenleri seçerek kullanılır

```
oc@olmak2:~/HEP_Okulu$ cat /etc/shells
# /etc/shells: valid login shells
/bin/csh
/bin/sh
/usr/bin/es
/usr/bin/ksh
/bin/ksh
/usr/bin/rc
/usr/bin/tcsh
/bin/tcsh
/usr/bin/esh
/bin/dash
/bin/bash
/bin/rbash
/usr/bin/screen
```



```
oc@olmak2:~/HEP_Okulu$ echo $SHELL
/bin/bash
```

```
oc@olmak2:~/HEP_Okulu$ chsh -s /bin/sh
Password:
Changing shell for oc.
Shell changed.
```

```
oc@olmak2:~/HEP_Okulu$ printenv | grep -i path
PATH=/home/oc/bin:/usr/local/sbin:/usr/local/bin
:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/home/
oc/bin:/home/oc/monitorino/bin
WINDOWPATH=7
```

```
oc@olmak2:~/HEP_Okulu$ printenv | grep -i mon
MONITORINO_SITE=/home/oc/monitorino
PATH=/home/oc/bin:/usr/local/sbin:/usr/local/bin
:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/home/
oc/bin:/home/oc/monitorino/bin
```

```
oc@olmak2:~/HEP_Okulu$ echo $PATH
/home/oc/bin:/usr/local/sbin:/usr/local/bin:
/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:
/home/oc/bin:/home/oc/monitorino/bin
oc@olmak2:~/HEP_Okulu$ _
```

# Kabuk ve Komut Satırı - II

## Bazı Çevre Değişkenleri

- ◆ **OSTYPE** : Sistem bağımlı bu değişken işletim sisteminin ne olduğu bilgisini tutar
- ◆ **LANG** : Sistem çıktılarının hangi dilde verileceğini tutan değişkendir
- ◆ **TERM** : Sistemde geçerli terminalin ismini tutar
- ◆ **HOME** : Kullanıcının kök dizinini tutar
- ◆ **USER(LOGNAME)**: Oturum açmış kullanıcının ismini tutar
- ◆ **PATH** : Sistem ya da kullanıcı bir komut verdiğinde kabuğun bu komutun çalıştırılabilir kütüğünü nerlerde arayacağı bilgisini tutar. Dizinler arasında ":" işareti bulunur. "ls" komutunu verdiğimizde "/bin/ls" yazmak zorunda olmayışımız bu çevre değişkeni sayesinde. Bu değişkene yeni bir dizin eklemek için "PATH=\$PATH:/yeni/dizin" komutu kullanılır. Burada dikkat edilmesi gereken PATH değişkeninin eskiden tuttuğu dizinleri kaybetmeme gerekliliğidir ve "\$PATH:" kısmı bunu sağlar
- ◆ **SHELL** : Kullanılan kabuğun sistemdeki yerini (path) tutar
- ◆ **PS1** : Kullanıcıdan komut beklerken gösterilecek metni tutar (command prompt)
- ◆ **HISTFILE** : Verilen komutların geçmişinin tutulacağı kütüğün yerini tutar
- ◆ **HISTSIZE** : En son kaç komutun geçmişinin tutulacağı bilgisini tutar
- ◆ **PWD** : O anda içinde çalışılıyor olan dizin yerini (path) tutar; bu çevre değişkenine bakmaksızın kullanımı daha kolay olan "pwd" komutu da aynı işi görür
- ◆ **\_** (Son Komut) : Kullanıcının çalıştırdığı en son komutu tutar

```
oc@olmak2:~$ echo $OSTYPE
linux-gnu
oc@olmak2:~$ echo $LANG
en_US.UTF-8
oc@olmak2:~$ echo $TERM
xterm
oc@olmak2:~$ echo $HOME
/home/oc
oc@olmak2:~$ echo $USER
oc
oc@olmak2:~$ echo $PATH
/home/oc/bin:/usr/local/sbin:/usr/local/bin:/usr/
sbin:/usr/bin:/sbin:/bin:/usr/games:/home/oc/bin:
/home/oc/monitorino/bin
oc@olmak2:~$ PATH=$PATH:/home/oc/bin
oc@olmak2:~$ echo $PATH
/home/oc/bin:/usr/local/sbin:/usr/local/bin:/usr/
sbin:/usr/bin:/sbin:/bin:/usr/games:/home/oc/bin:
/home/oc/monitorino/bin:/home/oc/bin
oc@olmak2:~$ echo $SHELL
/bin/bash
oc@olmak2:~$ echo $PS1
\[\e]0;\u@\h: \w\a\]${debian_chroot:+
($debian_chroot)}\u@\h:\w\$
oc@olmak2:~$ echo $HISTFILE
/home/oc/.bash_history
oc@olmak2:~$ echo $HISTSIZE
500
oc@olmak2:~$ echo $PWD
/home/oc
oc@olmak2:~$ date
Sun Dec 28 18:26:53 CET 2008
oc@olmak2:~$ echo $_
date
oc@olmak2:~$ pwd
/home/oc
oc@olmak2:~$ _
```

# Kabuk ve Komut Satırı - III

## Alias (Komut İsimlerini Değiştirmek)

- ◆ Bir kabuk komutu olan “alias” ile varolan komutlar için takma isimler ya da özelleştirilmiş komutlar oluşturmak mümkündür. Özelleştirilmiş komut anahtar ve değişkenlerle genişletilebilir ancak asıl komut asla yer değiştirmez. Sıkça verilen uzunca bir komutun kısaltılması amacı ile kullanılabilir:

```
oc@olmak2:~$ ps aux | grep -i monitorino
oc      12638  0.0  0.0  10276  1400 pts/2    S   18:50   0:00 /bin/bash /home/oc/monitorino/bin/Monitorino
oc      12639  0.9  1.2 157748 26244 pts/2    S   18:50   0:00 ./monitorino.executable
oc      12668  0.0  0.0   7452   948 pts/2    R+  18:50   0:00 grep -i monitorino
oc@olmak2:~$ alias bak="ps aux | grep -i monitorino"
oc@olmak2:~$ bak
oc      12638  0.0  0.0  10276  1400 pts/2    S   18:50   0:00 /bin/bash /home/oc/monitorino/bin/Monitorino
oc      12639  0.5  1.2 157748 26244 pts/2    S   18:50   0:00 ./monitorino.executable
oc      12690  0.0  0.0   7452   964 pts/2    S+  18:51   0:00 grep -i monitorino
oc@olmak2:~$ _
```

- ◆ Yukarıdakine benzer yapılan kısaltmalar “unalias” komutu ile kaldırılabilir:

```
oc@olmak2:~$ unalias bak
oc@olmak2:~$ bak
bash: bak: command not found
oc@olmak2:~$ _
```

- ◆ Kullanıcıların tercih ettikleri komut takma isimleri, her seferinde yeniden tanımlama zahmetinden kaçmak için “~/bashrc” kütüğüne yazılır ve her konsol açıldığında kendiliğinden (otomatik olarak) okunan bu kütük içindeki takma isimler etkinleşir:

```
alias ll='ls -l'
alias la='ls -A'
alias l='ls -CF'
alias ..='cd ..'
alias e="nano"
alias x="xterm"
alias xm="xmgrace"
alias n="nedit"
alias g="make"
alias gi="make install"
alias co="./configure"
```

# Kabuk İşlemleri - I

## Standart giriş, çıkış ve hata' nın yönlendirilmesi

- “yedekle” kütüğünün içeriği, “-l” anahtarı ile kullanılmak istenen “wc” komutuna standart giriş verisi olarak *yönlendiriliyor*.

```

l> wc -l < yedekle
l> 58
l> _
    
```

- Girişin yönlendirilmesine benzer biçimde, “ls -l” komutunun çıktısı “lsCiktisi.txt” kütüğüne *yönlendiriliyor*. Yönlendirme yapılırken lsCiktisi.txt kütüğünün varolmaması durumunda kütük kendiliğinden (otomatik) oluşturulur. Kütüğün önceden varolması durumunda ise içeriği silinir ve öyle işlem yapılır.

```

oc@olmak2:~/Documents/HEP_Okulu/workDir$ ls -l
total 120
-rwxr-xr-x 1 oc oc 836 2008-12-26 04:06 analiz.sh
-rw-r--r-- 1 oc oc 466 2008-12-26 02:50 phaseError.dat
-rw-r--r-- 1 oc oc 110539 2008-12-26 04:22 rnd.dat
-rwxr-xr-x 1 oc oc 1237 2008-12-26 03:32 yedekle
oc@olmak2:~/Documents/HEP_Okulu/workDir$ ls -l > lsCiktisi.txt
oc@olmak2:~/Documents/HEP_Okulu/workDir$ ls -l
total 124
-rwxr-xr-x 1 oc oc 836 2008-12-26 04:06 analiz.sh
-rw-r--r-- 1 oc oc 280 2008-12-26 16:32 lsCiktisi.txt
-rw-r--r-- 1 oc oc 466 2008-12-26 02:50 phaseError.dat
-rw-r--r-- 1 oc oc 110539 2008-12-26 04:22 rnd.dat
-rwxr-xr-x 1 oc oc 1237 2008-12-26 03:32 yedekle
oc@olmak2:~/Documents/HEP_Okulu/workDir$ cat lsCiktisi.txt
total 120
-rwxr-xr-x 1 oc oc 836 2008-12-26 04:06 analiz.sh
-rw-r--r-- 1 oc oc 0 2008-12-26 16:32 lsCiktisi.txt
-rw-r--r-- 1 oc oc 466 2008-12-26 02:50 phaseError.dat
-rw-r--r-- 1 oc oc 110539 2008-12-26 04:22 rnd.dat
-rwxr-xr-x 1 oc oc 1237 2008-12-26 03:32 yedekle
oc@olmak2:~/Documents/HEP_Okulu/workDir$ _
    
```

- Giriş yönlendirmesi “>>” işlemcisi ile de gerçekleştirilebilir; bunun “>” işlemcisinden farkı kütüğün içeriğini silmemesi ve verilen komutun çıktısını varolan kütüğün içeriğini koruyarak sonuna eklemesidir. Komutların sadece hata belirten çıktılarının yönlendirmesi “2>” işlemcisi ile yapılır.

# Kabuk İşlemleri - II

## Süzgeç komutları

- Süzgeçler çoğunlukla metin kütükleri üzerinde işlem yapan standart linux komutları ve/veya bu komutlarla kullanıcıların oluşturduğu kullanıcıya özgü komutlardır.
- Süzgeçler çoğunlukla başka bir programın çıktısını incelemek için kullanılır; dolayısı ile boru (“|”) işlemcisi ile beraber kullanılmalarına sıkça ihtiyaç duyulur.
- “**ps aux**” komutu tüm süreçleri terminale basar; bu çok uzun bir çıktıdır ve muhtemelen kullanıcılar bu çıktının bir alt kümesiyle ilgilenerler. Aşağıdaki örnekte kullanıcı içinde “**tracker**” kelimesi geçen süreçleri görmek istemektedir. Grep komutu, kendisine yönlendirilen (boru “|” ile) giriş verisi içinde yine kendisine verilen anahtar kelimeyi arar ve sadece bu kelimenin bulunduğu satırları terminale basar; diğer satırlar grep süzgecine takılır ve çıktıya aktarılmaz. Böylece çıkış kullanıcıya daha anlamlı gelir. Boru, bir komutun çıktısını başka bir komuta girdi olarak aktaran işlemcidir.

```
oc@olmak2:~/Documents/HEP_Okulu/workDir$ ps aux | grep tracker
oc      7179  10.9  2.0 170292 41288 ?        SNl  13:38   20:07 trackerd
oc      7180  0.0   0.3 124400  6972 ?        S    13:38   0:02 tracker-applet
oc@olmak2:~/Documents/HEP_Okulu/workDir$ _
```

- Bu yönlendirme işlemcileri ihtiyaca göre istenildiği sıra ve sayıda kullanılabilir.

```
> cat < chargePump.v | grep C4 | grep C3 >> freq.txt
> _
```

# Kabuk İşlemleri - III

## Süzgeç komutları (devam)

- **wc** [seçenek][kütük] – kütük içindeki sekizlileri (byte), kelimeleri ve satırları sayar:

```
oc@olmak2:~/Documents/HEP_Okulu/workDir$ ls yedekle -l
-rwxr-xr-x 1 oc oc 1236 2008-12-26 17:26 yedekle
oc@olmak2:~/Documents/HEP_Okulu/workDir$ wc yedekle
 58  209 1236 yedekle
oc@olmak2:~/Documents/HEP_Okulu/workDir$ _
```

- **sort** [seçenek][kütük] – Kütük içeriğini verilen sarta göre sıralayan ve sonucu standart çıkışa (öntanımlı olarak terminal) aktarır:

```
oc@olmak2:~/HEP_Okulu/workDir$ more sehirler.txt
Bursa
Adana
Antalya
Canakkale
Izmir
oc@olmak2:~/HEP_Okulu/workDir$ sort sehirler.txt
Adana
Antalya
Bursa
Canakkale
Izmir
oc@olmak2:~/HEP_Okulu/workDir$ _
```

- **nl** [seçenek][kütük] – Kütük içindeki metnin her satırına bir satır sayısı vererek standart çıkışa aktarır.
- **tail** [seçenek][kütük] – kütüğün son kısmını gösterir, çok büyük kütükler için kullanışlıdır:

```
oc@olmak2:~/HEP_Okulu/workDir$ nl rnd.dat | tail -5
1238188 1.394978e+02
1238189 8.199519e+01
1238190 -8.616127e+01
1238191 -1.327350e+02
1238192 -5.484803e+01
oc@olmak2:~/HEP_Okulu/workDir$ _
```

- **head** [seçenek][kütük] – kütüğün baş kısmını gösterir:

```
oc@olmak2:~/HEP_Okulu/workDir$ nl rnd.dat | head -5
 1 1.442444e+02
 2 3.154135e+01
 3 -1.521276e+02
 4 1.903035e+02
 5 -4.211599e+00
oc@olmak2:~/HEP_Okulu/workDir$ _
```

- **grep** [seçenek] kalıp [kütük] – kütük metni içinde kalıbı arar ve kalıbın bulunduğu yeri, “seçenekler” uyarınca standart çıkışa aktarır. Kütük verilmediği takdirde varsayılan, girdinin standart girişten okunacağıdır; dolayısıyla bu komut, boru (“|”) işlemcisi ile kullanıma çok uygundur.

```
oc@olmak2:~/HEP_Okulu/workDir$ grep -i adana *
sehirler.txt:Adana
oc@olmak2:~/Documents/HEP_Okulu/workDir$ _
```

- “Seçenek” kısmında en çok kullanılan anahtarlar aşağıda sıralanmıştır:

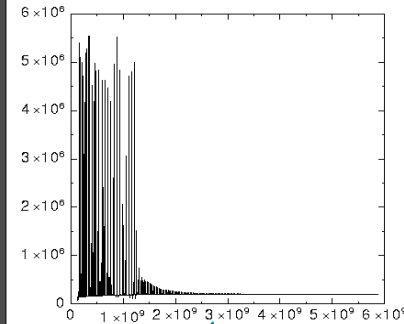
- l : Aranılan kalıbın geçtiği satırlar yerine kütük isimlerinin gösterilmesi için kullanılır; belirli bir dizin içinde aranan kelimeyi içeren kütüklerin isimlerinin bulunması için kullanılır.
- c : Aranılan metin parçasının kütük içinde kaç yerde geçtiğini bulur
- i : büyük/küçük harf ayrımı yapmaksızın arar
- v : arananın bulunmadığı satırları gösterir
- A <sayı> {-B <sayı>}: bulunan satırdan sonra {önce} bulunan “sayı” kadar satırı daha gösterir

# Kabuk İşlemleri - IV

## Süzgeç komutları (devam)

- **awk** – kütük içinde kalıp arama ve işleme programlama dili
  - Asağıdaki örnekte, bir benzetim (simülasyon) sonucunda “phaseError.dat” isimli 4 sütunlu bir veri kütüğü oluşmaktadır.
  - Bu veri kütüğünün çok büyük olduğu ve göreceli olarak çok sistem kaynağı (CPU, MEM, I/O vb.) kullanan gedit ya da kwrite gibi bir uygulama ile açılmadığı da varsayılabilir
  - Kullanıcının ayrıştırmasını (analizini) yapmakla ilgili olduğu bilginin ise 1. ve 3. sütunlarda bulunduğu varsayılmaktadır. Birinci sütun fs cinsinden benzetim (simülasyon) zamanı ve 3. sütun ise bir dinamik sistemin bir referans sisteme göre hesaplanan açı hatasıdır (bu aşamada verinin içeriğinin uygulama açısından önemi yok)
  - Kullanıcı 1. ve 3. sütunları alarak yeni 2 boyutlu bir veri kütüğü oluşturmak ve analize geçmeden önce yaptığı işin doğruluğunu çok hızlı bir biçimde veriyi çizdirerek denetlemek (kontrol etmek) istemektedir.

```
oc@olmak2:~/Documents/HEP_Okulu/workDir$ nl phaseError.dat | tail -5
23427613      5867472000    208000      207938      -12572
23427614      5867680000    208000      208060      -12634
23427615      5867888000    208000      208339      -12574
23427616      5868096000    208000      208334      -12235
23427617      5868304000    208000      208489      -11901
oc@olmak2:~/Documents/HEP_Okulu/workDir$ awk '{print $1" "$3}' phaseError.dat | nl - | tail -5
23427613      5867472000    207938
23427614      5867680000    208060
23427615      5867888000    208339
23427616      5868096000    208334
23427617      5868304000    208489
oc@olmak2:~/Documents/HEP_Okulu/workDir$ awk '{print $1" "$3}' phaseError.dat > yeni.dat
oc@olmak2:~/Documents/HEP_Okulu/workDir$ graph yeni.dat -T x
oc@olmak2:~/Documents/HEP_Okulu/workDir$ _
```





# Kabuk ve Komut Satırı - I

## Özel İşaretler

- ◆ Komut çıktılarının aynı satırda başka komutlara değişken olarak girilmesi “ ` ” ters tırnak işareti ile sağlanır; örnekte kullanılan “which” komutu, kendisine verilen anahtar kelimenin sistemdeki çalıştırılabilir kütüklerden biri olması durumunda bu kütüğün yerini döndürür. “du \$(which ls)” komutu da aynı görevi görür.
- ◆ Kabuklarda \*, ?, [, ], {, } gibi karakterlerin sahip oldukları özel anlamlar, komut satırını grafik kullanıcı arayüzleri karşısında güçlü kılar. Bu işaretler özel anlamları dışında kullanılmak istendiklerinde kaçış simgesi (escape character) de denen “\” işareti ile kullanılırlar.
  - ◆ \* : sıfır ve daha fazla işaret anlamına gelir
    - ➔ \* : Tüm kütük ve dizinler anlamına gelir
    - ➔ \*.\* : adında nokta olan tüm kütük ve dizinler
    - ➔ \*.dat : sonu “.dat” ile biten tüm kütük ve dizinler
    - ➔ a\*b : a harfi ile başlayıp b harfi ile biten tüm kütük ve dizinler
  - ◆ ? : tek bir karakter anlamına gelir
    - ➔ a?b : a ile başlayıp b ile biten 3 harfli tüm kütük ve dizinler
    - ➔ a??b : a ile başlayıp b ile biten 4 harfli tüm kütük ve dizinler
  - ◆ [ ] : veya anlamına gelir, ^ ve !, değil anlamı katar
    - ➔ [eE]\* : “e” veya “E” ile başlayan tüm kütük ve dizinler
    - ➔ [^eE]\* ya da [!eE]\* : “e” veya “E” ile başlamayan tüm kütük ve dizinler
    - ➔ [k-z][0-9] : “k” ile “z” harfleri arasındaki harflerle başlayan ve 0 ile 9 arasındaki rakamlarla son bulan tüm kütük ve dizinler
    - ➔ [k-z]\*[0-9]\* : “k” ile “z” harfleri arasındaki harflerle başlayan ve içinde herhangi bir yerinde 0 ile 9 arasındaki rakamları barındıran tüm kütük ve dizinler
  - ◆ { } : Köseli parantez ile benzerdir ancak harf grupları üzerinde de çalışır
    - ➔ {uyg,Ek}\* : “uyg” veya “Ek” ile başlayan tüm kütük ve dizinler
- ◆ Yandaki basit örneklerde yapılan hassas seçimin, içinde yüzlerce kütük ve dizinin bulunduğu bir dizinde, grafik bir arayüz ile nasıl yapılabileceğini düşünün.

```
oc@olmak2:~$ du `which ls`
108 /bin/ls
oc@olmak2:~$ du /bin/ls
108 /bin/ls
oc@olmak2:~$ du $(which ls)
108 /bin/ls
oc@olmak2:~$ _

oc@olmak2:/dev$ ls *2_ep00
usbdev2.2_ep00 usbdev3.2_ep00
oc@olmak2:/dev$ _

oc@olmak2:/dev$ ls s*t
snapsho  sndstat  stdout
oc@olmak2:/dev$ _

oc@olmak2:/dev$ ls tty?4
tty14  tty34  tty54  ttys4  ...
oc@olmak2:/dev$ _

oc@olmak2:/dev$ ls [t]*[w]*[5-7]
ttyw5  ttyw6  ttyw7
oc@olmak2:/dev$ _

oc@olmak2:/dev$ ls [t]*[w]*[!5-7]
ttyw0  ttyw2  ttyw4  ttyw9  ...
oc@olmak2:/dev$ _

oc@olmak2:/dev$ ls -l | wc -l
713
oc@olmak2:/dev$ _
```

# Kabuk ve Komut Satırı - II

## Kütük ve Dizin Bulmak

- ◆ Şu ana kadar içinde bulunduğumuz dizin içinde kütük ve dizinleri düzenlemek ve seçmek gibi etkilikleri nasıl gerçekleştireceğimizi çalıştık. Pekiyi içinde çalışmakla ilgilendiğimiz bu kütük ve dizinlerin yerlerini bilmiyorsak ? O zaman linux' un hızlı dizin ve kütük arama yöntemlerini kullanacağız.
- ◆ find : Disk üzerinde komutun çalıştırıldığı anda bulunulan yeri (path) kök (root) kabul ederek, kütük sisteminde kendisine “-name” anahtarı ile verilen kelimeyi içeren kütük ve dizinleri döndürür. Aramayı gerçekten disk üzerinde yapar ve arananı bulması, arama yapılan ortamın büyüklüğü ve hızına bağlı olmakla birlikte, göreceli olarak uzun zaman alır:

```
oc@olmak2:~$ find -name dürüstlükPolitikasi.pdf
find: `./Documents/gbt13/CaPPeLLo/iVerilog/oc_ser_v4/inca_libs': Permission denied
./Documents/HEP_Okulu/f77/dürüstlükPolitikasi.pdf
oc@olmak2:~$ _
```

- ◆ locate: Linux açılırken (boot ederken) kendisine bağlanan (mount edilen) tüm disklerin dizin yapısını (directory structure) bir veri tabanına yazar. Bu veri tabanını oluşturan ve işletim sistemi tarafından kendiliğinden (otomatik) çalıştırılan sürecin ismi “updatedb”dir ve sadece açılışta değil, kullanıcı istediği zaman da bu veri tabanını yeniler (update). “locate” komutu, disk içinde fiziksel olarak arama yapmak yerine, zaten oluşturulmuş bu veri tabanı üzerinde arama yapar ki bu göreceli olarak karşılaştırılmayacak kadar hızlıdır. “locate” komutu bu veri tabanı üzerinde “grep” gibi davranır ve kendisine verilen anahtar kelimeyi içeren veri tabanı girdilerini sıralar.

```
oc@olmak2:~$ locate baslik.pdf
/home/oc/Documents/HEP_Okulu/f77/baslik.pdf
/sdb3/eskiler/g77/fortranUygulama/baslik.pdf
/sdb3/eskiler/priv-html/lowlevelstuff/g77/baslik.pdf
oc@olmak2:~$ _
```

- ◆ Bu iki arama yöntemi arasındaki farkın anlaşılması durumunda, gerçekleştirilen bir arama sonucunda kullanıcı diskte aslında varolan bir kütüğün var olmadığı sonucuna varılabilir. Dikkat edilecek nokta şudur ki eğer bir kütük ya da dizin, sistemin düzenli olarak ya da kullanıcının keyfi bir zamanda “updatedb” komutunu çalıştırmamasından **sonra** oluşturulur ise, bu kütük ya da dizin sadece find komutu ile bulunabilir. Bunun nedeni yeni oluşturulanın henüz ilgili veri tabanına girmemiş olmasıdır. Bu durumda ya önce updatedb komutu ile veritabanı yenilenmeli ve ardından locate kullanılmalı ya da doğrudan find komutu kullanılmalıdır.

# Kabukta Süreç Yönetimi - I

## Süreçleri İzlemek ve Sonlandırmak

- Linuxta başlatılan her komut ve uygulama bir süreçtir. İşletim sistemi ilk açıldığında (boot) *init* adlı kök süreç (PID=1) başlar ve diğer tüm süreçler bu sürecin yarattığı (fork) oğul süreçler veya yaratılmış süreçlerin yarattığı diğer süreçler olarak yaşar ve ölürler. Her bir süreç PID denen ve sürecin doğumundan ölümüne kadar değişmeden kalan essiz bir kimlik numarasına sahiptir. Süreçler üzerindeki denetim işte bu PID kullanılarak gerçekleştirilir.

- Süreç denetiminde sıkça kullanılan komutlar:

- ps**: o anda çalışan süreçlerin neler olduğunu gösterir en çok **-aux** anahtarı ile kullanılır

```
ps aux ya da daha seçici olmak için
```

```
ps aux | grep -i monitorino
```

```
oc      11565  0.0  0.0  10276  1400 pts/0    S   23:35   0:00 /bin/bash /bin/Monitorino
oc      11566  0.8  1.2  157744 26232 pts/0    S   23:35   0:00 ./monitorino.executable
oc      11606  0.0  0.0   7452   952 pts/0    R+  23:36   0:00 grep -i monitorino
```

- top**: ps gibidir ancak çıktısını belirli aralıklarla yeniler ve süreçlerin canlı takibine yarar. Bu komutun öntanımlı davranışı süreçleri CPU kullanımına göre sıralamaktır. Komut çalıştıktan sonra "M" komutu ile bu davranış, hafızaya göre sıralamaya ve tekrar "P" ile öntanımlı davranışa geri değiştirilebilir.

- pidof**: çalışıyor olan bir sürecin ve yarattığı (fork) ikincil süreçlerin PID'lerini döndürür.

```
pidof gnuplot
12412 12405 12402 12399 12394
```

- uptime**: sistemin ne kadar süredir çalışıyor durumda bulunduğunu gösterir

```
uptime
23:31:40 up 11 days,  2:32,  3 users,  load average: 0.09, 0.09, 0.08
```

- kill**: süreçlere, belirli seyleri yapmaları emrini vermek için, işaret göndermekte kullanılır. Pek çok işaret olmasına karşın (kill -l ile tümü görülebilir) en sık kullanılanlar:

- kill PID: PID sayılı süreci sıradan (normal) yolla sonlandırmak için kullanılır; sürecin açık kütüklerini ve ağ bağlantılarını kapatması ve varolmaları durumunda gerekli diğer kapanış işlemlerini yapması için kendisine zaman tanınır. Bazı durumlarda başarısız olabilir.
- kill -9 PID: PID sayılı süreci olağanüstü yolla sonlandırmak için kullanılır; sürecin hiç bir iş yapması beklenmez. Çok nadir başarısız olur.

- lsdf**: kendisine verilen bir dizinin hangi süreçler tarafından kullanıldığını sıralar.

# Kabukta Süreç Yönetimi - I

Süreçleri İzlemek ve Sonlandırmak



**Biz senin yerine  
süreçleri izledik ve  
sonlandırdık !..**

# Kabukta Süreç Yönetimi - II

## Süreç Öncelikleri ve Ardalanda Çalıştırma

- ◆ Yetkili bir kullanıcı süreçlerin çalışma önceliklerini (*nice* sayısı) “*renice*” komutu ile değiştirebilir. Böylece çok yüklü çalışan süreçlerin daha fazla CPU zamanı kullanmaları sağlanmış olur.
- ◆ Süreçlerin -20 (en yüksek öncelik) ile 20 (en düşük öncelik) arasında olabilen *nice* değeri, öntanımlı olarak 0 dır. Değiştirilmiş olması durumunda eğer *fork* ile yaratılan bir oğul süreç varsa *oğul*, *ana* sürecin önceliğini alır.
- ◆ Aşağıdaki örnek 20256 PID sayılı sürece en yüksek önceliği vermek için kullanılmıştır:

```
root@olmak2:~# renice -20 20256
20256: old priority 0, new priority -20
root@olmak2:~# _
```

- ◆ Komut satırında bir süreç başlatıldığında (*fork*), o süreç sonlanana kadar komut satırı başka bir komut kabul edemez. Bu uygulamada çoğunlukla kullanışlı değildir; komut satırının başka komutları da kabul edebilmesini isteriz. Bunu sağlamanın yollarından biri komutları ya da uygulamaları “&” özel işaretini kullanarak “*ardaland*” (*background*) başlatmaktır. Diğer bir yol ise komut ya da uygulamaları “Ctrl-Z” ile *duraklatıp* (*interrupt*) “*bg*” komutu ile ardalana almaktır.
- ◆ Aşağıdaki örnekte uygulama doğrudan ardalanda başlatılmaktadır:

```
oc@olmak2:~/Documents/HEP_Okulu/workDir$ xmgrace veri.dat &
```

- ◆ Aşağıdaki örnekte ise uygulama başlatılmakta ve konsol başka komut kabul edemediği için ardalana alma ihtiyacı doğmakta, uygulama duraklatılmakta ve ardalana alınmaktadır:

```
oc@olmak2:~/Documents/HEP_Okulu/workDir$ xmgrace veri.dat
^Z
[1]+  Stopped                  xmgrace veri.dat
oc@olmak2:~/Documents/HEP_Okulu/workDir$ bg
[1]+ xmgrace veri.dat &
oc@olmak2:~/Documents/HEP_Okulu/workDir$ jobs
[1]+  Running                  xmgrace veri.dat &
```

- ◆ Ardalanda çalışıyor olan uygulamaları veya komutları görmek için “*jobs*” komutu kullanılır.

# Kabuk Programlama - I

## Betik Genel Yapısı ve Çalıştırma

```
File Edit Search Preferences Shell Macro Windows Help
1 #!/bin/bash
2
3 # TISEAN uygulamasini kullanarak rastgele sayi uret ve rnd.dat kutugunu yarat
4 makenoise -g -I 0 -r 100 -0 -l 8192 -o rnd.dat
5
6 # Bir OCTAVE betigi kullanarak rnd.dat icin degisen ortalamayi hesapla
7 octave runningAverage.m
8
9 # NEDIT ile ilgili kutugu ac ve gerekli degisiklikleri yap VE
10 # NEDIT saglikli bir bicimde islemi tamamladiginda TISEAN
11 # uygulamasini kullanarak degisen ortalamainin spectrumunu hesapla
12 nedit runningAverage.dat && spectrum runningAverage.dat -o runningAverageSpectrum.dat
13
14 # TISEAN uygulamasini kullanarak en basta uretilen rastgele
15 # sayilarin spectrumunu hesapla
16 spectrum rnd.dat -o rndSpectrum.dat
17
18 # GRAPH uygulamasini kullanarak degisen ortalamainin ve en basta
19 # uretilen rastlantisal sayilarin spectrumlarının grafiklerini cizdir
20 graph runningAverageSpectrum.dat -T x
21 graph rndSpectrum.dat -T x
22
```

Kütük: analiz.sh

- “#!” ile kütük içeriğinin *hangi* kabuk tarafından *yorumlanacağı* belirtilir
- “#” ile betiğe yorumlanmayacak olan *açıklamalar* eklenir
- Makenoise, octave, nedit, spectrum ve graph, yolları \$PATH çevresel değişkeninde bulunan çalıştırılabilir sıradan kabuk komutlarıdır
- “analiz.sh” isimli yukarıdaki kütük:
  - “chmod +x analiz.sh” ile çalıştırılabilir hale getirildikten sonra
  - “./analiz.sh” komutuyla çalıştırılır

# Kabuk Programlama - II

## Neden var ?

- Linux komutlarının kullanıcı tarafından belirlenen bir görevi yerine getirmek için bir araya gelmesi çok sık rastlanan bir gereksinimdir
- Gerçek işletim sistemleri, her iş için ayrı bir uygulama sunmaya çalışmak yerine, her işi yerine getirebilecek farklı uygulamaların kolayca üretilebileceği bir komutlar birlikteliği sunmayı amaçlar.
- Sunulan bu komutlar/uygulamalar birbirlerinden bağımsız geliştiriciler tarafından üretilmiş ve/veya belgelendirilmiştir. Örnek olarak bir önceki sayfadaki betik için:
  - TISEAN zaman serisi analizi (ayrıştırması) için tasarlanmıştır
  - OCTAVE MatLab<sup>(R)</sup> uygulamasının açık-kaynak karşılığıdır
  - NEDIT bir metin düzenleyicidir ve
  - GRAPH bir veri çizicidir
- Kullanıcıya büyük esneklik sağlar:
  - Betikler içinde kullanılan komutlar/uygulamalar kendi anahtarları ile kullanılabilir; örneğin:
    - Bir önceki sayfadaki betik için: “graph rndSpectrum.dat -T x” komutunda olduğu gibi “-T” anahtarı ve ilgili “x” değeri, “graph” komutuna giriliyor
  - Programlama dillerinde bulunan değişken (integer, float vb), döngü (for, while vb.) ve denetim (if, else vb) yapıları kullanılabilir
  - Sadece birkaç satırda kullanıcı ile etkileşimli (grafik ortamda ve/veya komut satırında) uygulamalar gerçekleştirilebilir

# Kabuk Programlama - III

## Bir Örnek

İçinde bulunulan dizindeki tüm kütükleri, kullanıcının verdiği yeni bir dizine çoğaltan (kopyalayan), bu dizini tar'layıp ardından gzip'leyerek bir \*.tgz kütüğü oluşturan ve yeni dizini silen bir uygulamaya sıkça ihtiyaç duyulması durumunda tüm bu işlemler güvenli bir biçimde bir betiğe derlenebilir ve tek bir linux komutu haline getirilebilir.

### Değer Döndürme

- 0 standart giriş
- 1 standart çıkış (24. satır)
- 2 standart hata (12. satır)

### Değişkenler

- Tanımlama: değişken=değer
- Çağırma: \$değişken, "\$değişken", \${değişken}
- Kullanıcıdan okuma: read değişken
- Okunarı basma: echo \$değişken (5. satır)

### Anahtarlar

- \$0 çalıştırılan kütüğün kendisinin adı (5. satır)
- \$1, \$2 .. birinci, ikinci, .. anahtarlar (17. satır)
- \$@ girilen tüm anahtarlar (31. satır)
- \$# girilen anahtar sayısı (8. satır)

### Koşullular

- if [ koşul/sınama ]; then (else) fi (37-43. satırlar)
- koşul: -eq, -ne, -lt, -gt, -le, -ge (eşit, eşit değil, küçük, büyük, küçük eşit, büyük eşit) (8. satır)
- sınama: -e, -f, -d, -h, -s, -(r/w/x), -nt, -ot (kütük var mı, kütük mü, dizin mi, bağlantı mı, kütük boyu sıfırdan farklı mı, okuma/yazma/çalıştırma izinleri var mı, birinci ikinciden yeni, birinci ikinciden eski) (37. satır)
- !: değil (koşul yanlışsa) (37. satır)
- &&: tüm koşullar doğruysa
- ||: koşullardan biri doğruysa (8. satır)

### Döngüler

- for değişken do komut done
- for değişken in liste do komut done (47. satır)
- while [ koşul ] do komut done

```
File Edit Search Preferences Shell Macro Windows Help
1 #!/bin/bash
2
3 # Calistirdigimiz komutun ismine ulasiyoruz
4 echo " "
5 echo "Komutunuza verdiginiz isim : " $0
6
7 # Girilen anahtar sayisini denetliyoruz
8 if [ $# -eq 0 ] || [ $# -gt 1 ] ; then
9     echo " "
10    echo "HATA : Bir anahtar girilmeli."
11    echo " "
12    exit 2
13 fi
14
15 # Komutumuzun hangi anahtarlari kabul
16 # ettigi soruldugunda cevabimizi olusturuyoruz
17 if [ $1 == "-h" ] ; then
18     echo " "
19     echo "Icinde bulunulan dizindeki tum kutukleri"
20     echo "ve dizinleri, yeni <dizin ismi> dizinine"
21     echo "coğaltir ve *.tgz ile sikistirir. Kullanim sekli:"
22     echo $0 "<dizin ismi>"
23     echo " "
24     exit 1
25 fi
26
27 # Hersey yolunda, girilen anahtari
28 # kullaniciya gosteriyoruz
29 if [ $# -gt 0 ] ; then
30     echo " "
31     echo "Girdiginiz dizin ismi : " @$
32 fi
33
34 # Dizin var mi yok mu denetliyoruz; yoksa olusturuyoruz
35 # varsa karisikliklari onlemek icin baska bir isim istiyoruz
36
37 if [ ! -e $1.tgz ] ; then
38     echo $1.tgz "yok, olusturuluyor..."
39     mkdir $1
40 else
41     echo "HATA : " $1.tgz "zaten var."
42     exit 2
43 fi
44
45 # Gorevi yerine getiriyoruz
46
47 for x in *
48 do
49     if [ $x != $1 ] ; then
50         cp -avuf $x $1
51     fi
52 done
53
54 tar cfz $1.tgz $1
55 echo $1.tgz "bohcase olusturuldu."
56 rm -rf $1
57 echo "Gecici dizin" $1 "silindi."
58 echo "Komut tamamlandi."
```

Kütük: yedekle



# Kabuk Programlama - IV

## Bilmediğimi Biliyorum

- İstisnasız tüm linux komutları ve uygulamaları geniş biçimde belgelendirilmişlerdir. Çoğu zaman bir komut ya da uygulama birden fazla veri tabanında farklı biçimlerde belgelerle açıklanır. En çok kullanılan standartlaşmış linux belgeleme düzeni (sistemi) komutların kendilerinden “-h” ya da “--help” gibi anahtarlarla istenen açıklamalar ile işletim sisteminin kullanıcıya sunduğu “man” ve “info” sayfalarından oluşur.
- Bir komutun ya da uygulamanın kullanım kılavuzu ya da belgesine ya da man (info) sayfasına ulaşmak için kabukta “man <komut ismi>” ya da “info <komut ismi>” komutlarını vermek yeterlidir.
- Bu man ve info sayfalarından komutla ilgili tüm ayrıntılara ulaşmak ve komutun kabul ettiği tüm anahtarları öğrenmek mümkündür. “/” ile metin araması yapmak da mümkündür.
- Bir uygulama ya da komut geliştirildiğinde, komut satırından “-h” anahtarını kabul edecek şekilde uygulamaya kısa da olsa bir belgelendirme gömülmesi diğer geliştiricilerce olumlu karşılanır.

Komuta verilen “-h” anahtarı

```
oc@olmak2:~/Documents/HEP_Okulu$ xmgrace -h | more
Usage of xmgrace command line arguments:
-autoscale [x|y|xy|none]      Set autoscale type
-barebones                    Turn off all toolbars
-batch [batch_file]          Execute batch_file on start up
-block [block_data]          Assume data file is block data
-bxy [x;y;etc.]              Form a set from the current block data set
                              using the current set type from columns
                              given in the argument
-datehint [isoleuropeanus
           |days|seconds|nohint] Set the hint for dates analysis
                              (it is only a hint for the parser)
-dpipe [descriptor]          Read data from descriptor on startup
-fixed [width] [height]      Set canvas size fixed to width*height
-free                          Use free page layout
-graph [graph_number]        Set the current graph number
-graphtype [graph_type]      Set the type of the current graph
-hardcopy                      No interactive session, just print and
                              quit
-hdevice [hardcopy_device_name] Set default hardcopy device
-install                      Install private colormap
-legend [load]                Turn the graph legend on
-log [x|y|xy]                 Set the axis scaling of the current graph
                              to logarithmic
--More--
```

```
MAN(1) Manual pager utils MAN(1)
NAME
  man - an interface to the on-line reference manuals

SYNOPSIS
  man [-c|-w|-tZ] [-H[browser]] [-T[device]] [-X[dpi]] [-adhu7W] [-il-I]
  [-m system[,...]] [-L locale] [-p string] [-C file] [-H path] [-P
  pager] [-r prompt] [-S list] [-e extension] [-warnings [warnings]]
  [[section] page ...] ...
  man -l [-7] [-tZ] [-H[browser]] [-T[device]] [-X[dpi]] [-p string] [-P
  pager] [-r prompt] [-warnings [warnings]] file ...
  man -k [apropos options] regexp ...
  man -f [whatis options] page ...

DESCRIPTION
  man is the system's manual pager. Each page argument given to man is
  normally the name of a program, utility or function. The manual page
  associated with each of these arguments is then found and displayed. A
  section, if provided, will direct man to look only in that section of
  the manual. The default action is to search in all of the available
  sections, following a pre-defined order and to show only the first page
  found, even if page exists in several sections.

Manual page man(1) line 1
```

```
File: plotutils.info, Node: Top, Next: Plotutils Intro, Prev: (dir), Up: (d\
ir)

This is the documentation for version 2.5 of the GNU plotting
utilities package, including GNU libplot 4.2. The package consists of
programs and functions for vector graphics and data plotting.

* Menu:
* Plotutils Intro::      GNU plotting utilities
* graph::                graph, a program for plotting datasets
* plot::                 plot, a plot format conversion program
* pic2plot::             pic2plot, a translator for files in the pic language
* tek2plot::             tek2plot, a translator for legacy Tektronix files
* plotfont::            plotfont, a program for plotting font character maps
* spline::               spline, an interpolation program
* ode::                  ode, a differential equation integrator
* libplot::              A library for device-independent 2-D vector graphics
* Appendices::          Additional Information

--zz-Info: (plotutils.info.gz)Top, 20 lines --All-----
Welcome to Info version 4.11. Type ? for help, m for menu item.
```

info plotutils

# Linux ve Uygulamalar - I

*İşletim sistemi ve uygulamalarından oluşan  
bilimsel çalışma ortamına giriş*

- ◆ Kabuk ve Komut Satırı
  - ◆ Giriş: Linux kabuktur
  - ◆ Bazı Çevre Değişkenleri
  - ◆ Alias (Komut İsimlerini Değiştirmek)
- ◆ Kabuk İşlemleri
  - ◆ Standart giriş, çıkış ve hata' nın yönlendirilmesi
  - ◆ Süzgeç komutları
- ◆ Kabuk ve Komut Satırı
  - ◆ Özel İşaretler
  - ◆ Kütük ve Dizin Bulmak
- ◆ Kabukta Süreç Yönetimi
  - ◆ Süreçleri İzlemek ve Sonlandırmak
  - ◆ Süreç Öncelikleri ve Ardalanda Çalıştırma
- ◆ Kabuk Programlama
  - ◆ Betik Genel Yapısı ve Çalıştırma
  - ◆ Neden var ?
  - ◆ Bir Örnek
  - ◆ Bilmediğimi Biliyorum
- ◆ Uygulamalar 
  - ◆ Ayrımlar
- ◆ Metin Düzenleyiciler
  - ◆ 8 Bitlik Gözlüklerimiz
- ◆ Kütüklerin Anlamlılık Ölçekleri
  - ◆ Ölçek Seçimi Sonuçtur
  - ◆ Ölçek Seçimi Anlamdır
- ◆ Matematik İşlev ve Veri Çiziciler
  - ◆ Verinin Okunabilirliğini Artırmak
- ◆ Bilgisayarlı Cebir
  - ◆ Güvenli Simgesel Hesap
- ◆ Tüm Mühendislik Hesapları
  - ◆ Octave
- ◆ Güvenli Uzak Erişim, Kütük Aktarımı ve Veri Yedeklemek
- ◆ Ararken
- ◆ Akşam Sefası
  - ◆ Ertesi günün sabahına hazırlanması beklenen aksamlık ödevler

# Uygulamalar

## Ayrımlar

- **Marka - Teknoloji Ayrımı:** Ticari marka isimleri ile bilişim teknolojilerine karşılık gelen isimlerin birbirine karıştırılması, algıda bulanıklığın ve dolayısı ile hangi sorun karşısında hangi araçların kullanılabileceği ile ilgili kararsızlığın kaynaklarından biridir. Örnek:
  - **Ayrim:** “Excel” bir markadır, teknoloji olan “spreadsheet” kavramıdır; spreadsheet fikrini uygulayan pek çok ofis uygulaması vardır: MS Excell, OpenOffice.org Calculator, koffice, spreadsheet, xsheet v.b. Bu uygulamalar, spreadsheet kavramını kendi ürünlerine programlamışlardır.
  - **Belirsizlik:** “Bu bir excel kütüğüdür.” önermesinde bulunan kişi o kütüğün uzantısının \*.xls olduğunu mu yoksa yapılan hesabın bir spreadsheet uygulaması ile yapıldığını mi anlatmak ister ? Ya da “Bir milyon noktalık bir veri excel ile çizilemez.” önermesinde bulunan kişi (*ben*), MS Excel' in bu işi yapamayacağını mi yoksa bir teknoloji olarak spreadsheet kavramının yapılacak iş için uygun olmadığını mi belirtmektedir ?
- **Teknoloji - Topoloji Ayrımı:** YEF ve nükleer fizikte dedektörlerden gelen veri olay olay kaydedilmektedir ve her olay içinde fiziksel bilgi, belirli bir biçimde özel ayrılmış veri alanlarını dolduran belirli alt kısımları içermektedir. Topolojik açıdan bakıldığında işlenmemiş YEF verisi ile bir SQL veritabanı aynı farzedilebilir. Fakat deneysel fizikçiler fizik olaylarını işlenmemiş YEF verisinden, SQL sorgulama dilini kullanarak çekme yolunu tercih etmemektedirler; ya da tersten gidelim: veritabanlarıyla çalışanlar deneysel fizikçilerin kullandıkları yöntemlerle veri tabanlarına sorgu yöneltmemektedirler. Örneğin veri tabanlarıyla çalışanlar SQL sorgu cümlelerine “software trigger” dememektedirler. Bunun nedeni, tarihsel olmaktan çok, topolojik olarak aynı ya da benzer olmalarına rağmen farklı düşünce yöntemlerine ihtiyaç duymalarıdır.
  - Benzer şekilde tamamen aynı olan iki hesap için düşünülen ölçek de farklı teknolojileri gerekli kılabilir. Tavla oynayan iki bilim adamının zarlarını kaybettiklerini düşünelim: yaklaşık iki saniyede bir, 1 ile 6 arasında iki tamsayı rastlantısal sayıya ihtiyaçları vardır. Bu ihtiyaç basit bir hesap makinasının rnd işlevi ile çözülebilirken bir nükleer bozunma benzetimi (simülasyonu) için çok daha yüksek hızda, hassaslıkta ve doğrulukta rastlantısal sayılara ihtiyaç vardır. Bu durumda standart c/c++ kütüphanesindeki rand() işlevinden fazlasına ihtiyaç duyulacaktır.

# Uygulamalar

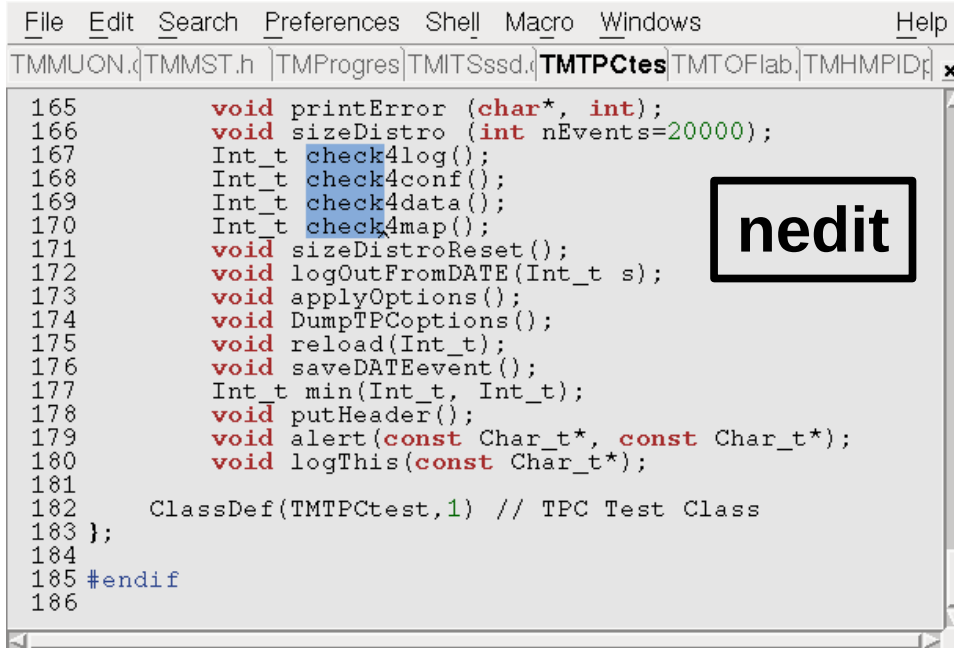
## Ayrımlar (devam)

- **Komut Satırı - Grafik Arayüz Ayrımı:** Kullandığımız uygulamalar sadece kabuk kullanıyor ve hiç bir biçimde grafik bir arayüze (GUI) ihtiyaç duymuyor ise (*batch mode*) uzaktan eriştiğimiz bilgisayarlar üzerinde güvenle ve hızlı bir biçimde kullanılabilir. Aksi durumda uygulamalar ya aradaki ağ bağlantısı yeterince hızlı olan uzak bilgisayarlarda ya da yerel bilgisayarlarda hiç bir ağ bağlantısına gerek duymaksızın ancak kullanılabilir.
- Her iki dünyanın da kendine göre kolaylıkları vardır. Örnek olarak uzaktaki bir bilgisayara bağlandığınızı ve bir ayar (konfigürasyon) kütüğü içindeki belirli bir satırı değiştirmeniz gerektiğini varsayalım. Buna ek olarak ağ bağlantısının göreceli olarak yavaş olduğunu ve uzak bilgisayarın işlem gücünün de düşük olduğunu varsayalım ki bu varsayımlar ender rastlanır cinsten değildir. “ssh” ile uzak makineye bağlandığınızda kullanabileceğiniz metin düzenleyiciler sadece kabukta çalışan standart linux düzenleyicileri olacaktır. Üstelik seçim de size ait olmayacak zira uzak makine her ne yüklü ise onu kullanmak zorunda olacaksınız. Şu durumda kabuk uygulamaları daha çekici görünüyor. Bir başka örnek olarak kendi bilgisayarınızda benzer bir iş yaptığınızı farzedelim ve metin düzenleyici olarak da “pico” kullanıyor olalım. Bu durumda gelişmiş pek çok metin düzenleyicide varolan işlevlerden yararlanamayacağız: makrolar, metin renklendirme, komutları hatırlama zorunluluğunun olmaması, aynı anda birden fazla kütüğü düzenleme, normal metin seçimine ek olarak blok metin seçme, akıllı içerik düzenlemesi v.b.
- Bu iki dünyadan farklı uygulamaları bilmek (uzman olmaya gerek olmaksızın) hangi durumda neyin ise yarayacağına daha kolay karar verebilmek anlamına gelecektir.

# Metin Düzenleyiciler

## 8 Bitlik Gözlüklerimiz

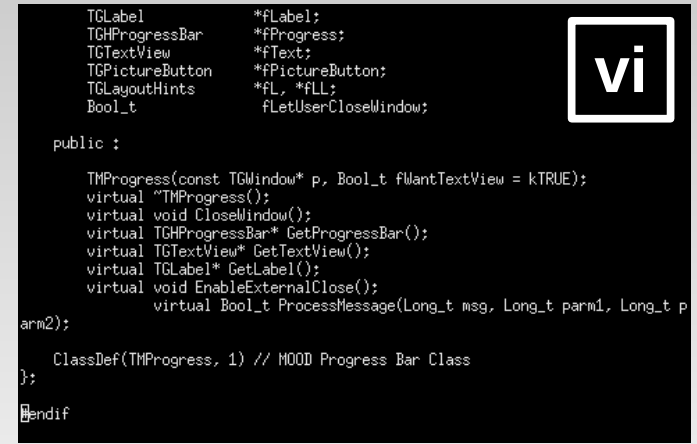
- ◆ Kelime işlemcilerle karıştırılmamalıdır: Örneğin MS Word ya da OpenOffice Writer metin düzenleyicileri değildirler (\*.doc ya da \*.sxi uzantılı bir kütüğün içeriğini “cat \*.doc” ya da “cat \*.sxi” komutu ile konsolunuza bastırmaya çalışın)
- ◆ Temel iki dünya: kabukta çalışanlar (batch mode) ve grafik kullanıcı arayüzüne sahip olanlar
- ◆ Küçük, hızlı ve/veya uzak bilgisayarlar üzerindeki işler için kabukta çalışan düzenleyiciler daha uygun olabilir
- ◆ Kabukta çalışanlar bazı yüksek seviyeli işlemlere sahip olmayabilir: makrolar, tab'lar, eş-zamanlı kütük düzenleme, blok metin seçimi, akıllı içerik düzenleme v.b.
- ◆ Grafik arayüze sahip düzenleyiciler daha fazla sistem kaynağına ihtiyaç duysalar da sezgiseldirler (kullanımları kolaydır) ve işlevleri hatta ihtiyaç duyulandan fazladır



```
File Edit Search Preferences Shell Macro Windows Help
TMMUON.d TMMST.h TMTProgres TMITSSsd.d TMTPCtes TMTOfLab TMTMHPID
165 void printError (char*, int);
166 void sizeDistro (int nEvents=20000);
167 Int_t check4log();
168 Int_t check4conf();
169 Int_t check4data();
170 Int_t check4map();
171 void sizeDistroReset();
172 void logOutFromDATE (Int_t s);
173 void applyOptions();
174 void DumpTPCOptions();
175 void reload(Int_t);
176 void saveDATEevent();
177 Int_t min(Int_t, Int_t);
178 void putHeader();
179 void alert(const Char_t*, const Char_t*);
180 void logThis(const Char_t*);
181
182 ClassDef(TMTPCtest,1) // TPC Test Class
183 };
184
185 #endif
186
```

nedit

Renkli grafik dünya



```
TGLabel *fLabel;
TGHProgressBar *fProgress;
TGTextView *fText;
TGPictureButton *fPictureButton;
TGLayoutHints *fL, *fLL;
Bool_t fLetUserCloseWindow;

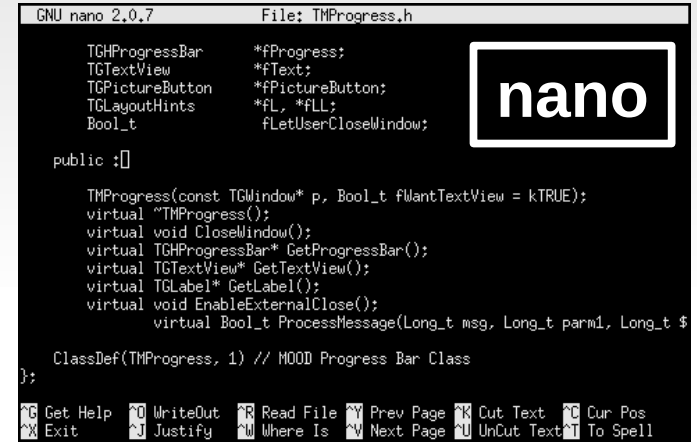
public :

TMTProgress(const TGWindow* p, Bool_t fWantTextView = kTRUE);
virtual ~TMTProgress();
virtual void CloseWindow();
virtual TGHProgressBar* GetProgressBar();
virtual TGTextView* GetTextView();
virtual TGLabel* GetLabel();
virtual void EnableExternalClose();
virtual Bool_t ProcessMessage(Long_t msg, Long_t parm1, Long_t p
arm2);

ClassDef(TMTProgress, 1) // MOOD Progress Bar Class
};

#endif
```

vi



```
GNU nano 2.0.7 File: TMTProgress.h

TGHProgressBar *fProgress;
TGTextView *fText;
TGPictureButton *fPictureButton;
TGLayoutHints *fL, *fLL;
Bool_t fLetUserCloseWindow;

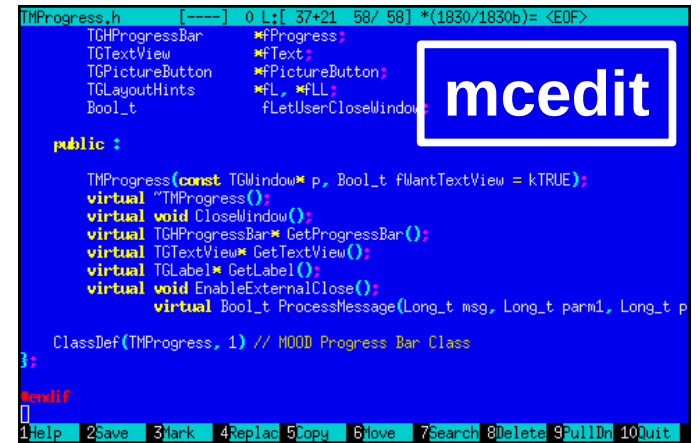
public :[]

TMTProgress(const TGWindow* p, Bool_t fWantTextView = kTRUE);
virtual ~TMTProgress();
virtual void CloseWindow();
virtual TGHProgressBar* GetProgressBar();
virtual TGTextView* GetTextView();
virtual TGLabel* GetLabel();
virtual void EnableExternalClose();
virtual Bool_t ProcessMessage(Long_t msg, Long_t parm1, Long_t $
);

ClassDef(TMTProgress, 1) // MOOD Progress Bar Class
};

Get Help WriteOut Read File Prev Page Cut Text Cur Pos
Exit Justify Where Is Next Page UnCut Text To Spell
```

nano



```
TMTProgress.h [----] 0 L: [ 37+21 58/ 58] *(1830/1830b)= <EOF>
TGHProgressBar *fProgress;
TGTextView *fText;
TGPictureButton *fPictureButton;
TGLayoutHints *fL, *fLL;
Bool_t fLetUserCloseWindow;

public :

TMTProgress(const TGWindow* p, Bool_t fWantTextView = kTRUE);
virtual ~TMTProgress();
virtual void CloseWindow();
virtual TGHProgressBar* GetProgressBar();
virtual TGTextView* GetTextView();
virtual TGLabel* GetLabel();
virtual void EnableExternalClose();
virtual Bool_t ProcessMessage(Long_t msg, Long_t parm1, Long_t p
);

ClassDef(TMTProgress, 1) // MOOD Progress Bar Class
};

#endif

1|help 2|save 3|mark 4|replac 5|copy 6|move 7|search 8|delete 9|pullIn 10|uit
```

mcedit

Yüksek verimli kabuk dünya

# Kütüklerin Anlamlılık Ölçekleri

## Ölçek Tahmini



★ Yukarıdaki resimlerin ölçeklerini tahmin edebilir misiniz ?  
→ Makro ? Mikro ? Nano ?

\* Anonim

\*\* Fotoğrafçı Alp Alper' in çalışmalarından alınmıştır

# Kütüklerin Anlamlılık Ölçekleri - I

## Ölçek Seçimi Sonuçtur

- Bir kütük, sabit diskte tanım gereği tek bir biçimde saklanır: ardışık 1'ler ve 0'lardan oluşan dizilimler olarak; bu kütüklerin anlamlı oldukları “ölçekler” aşağıda sıralanan farklı kütük tiplerini meydana getirir:
  - ➔ İkilik (binary) bir kütük : 1'li 1'li (bit) okunduğunda anlam taşıyan kütük
  - ➔ Metin (text, ascii) kütüğü : 8'li 8'li (byte) okunduğunda anlamlı olan kütük
  - ➔ Yukarıdakiler dışında başka ölçekler ile oluşturulmuş kütükler de mevcuttur
- Aşağıdaki örnekte 36 tane 8'liden (byte) oluşan “sehirler.txt” isimli kütüğün 8'lik (cat komutu ile) ve ikilik okunduğunda neye benzediği görülmektedir. İkilik okunduğunda çıktı çok uzun olacağından çoğunlukla uygulanan yöntem, çıktıyı 16'lik tabanda (hexdump komutu ile) basmaktır:

```
oc@olmak2:~/Documents/HEP_Okulu/workDir$ ls -l sehirler.txt
-rw-r--r-- 1 oc oc 36 2008-12-26 18:41 sehirler.txt
oc@olmak2:~/Documents/HEP_Okulu/workDir$ cat sehirler.txt
Bursa
Adana
Antalya
Canakkale
Izmir
oc@olmak2:~/Documents/HEP_Okulu/workDir$ hexdump -C sehirler.txt
00000000  42 75 72 73 61 Ca 41 64 61 6e 61 Ca 41 6e 74 61 |Bursa.Adana.Anta|
00000010  6c 79 61 Ca 43 61 6e 61 6b 6b 61 6c 65 Ca 49 7a |lya.Canakkale.Iz|
00000020  6d 69 72 Ca                                     |mir.|
00000024
oc@olmak2:~/Documents/HEP_Okulu/workDir$ _
```

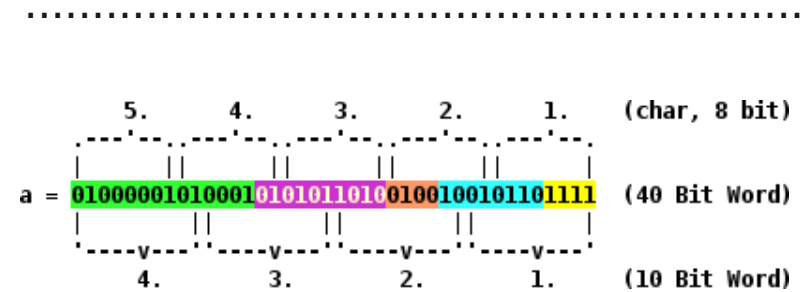
Yandaki örnek khexedit ile de uygulanabilirdi: khexedit, grafik bir arayüze ve en gelişmiş özelliklere sahip, çok kullanışlı bir KDE uygulamasıdır.

# Kütüklerin Anlamlılık Ölçekleri - II

## Ölçek Seçimi Anlamdır

- Yukarıdaki örneklerde ilgilenilen kütüğün sakladığı bilginin anlamlı olduğu ölçek hep sabit kalmaktadır: 1'li ya da 8'li olarak. Bu bir zorunluluk değildir (özellikle deneysel fizik donanımlarında); kütük içinde anlamlılık ölçeği değişebilir.
- Yandaki örnek ALICE deneyinin TPC (Time Projection Chamber) algılayıcısının (dedektör) canlı veri izleyici (On-Line Data Quality Monitor) yazılımından alınmıştır ve yaklaşık 60 Mbyte büyüklüğündeki bir TPC “olay”ının başlangıcını göstermektedir.
- Sarı, yeşil ve pembe alanlar farklı donanım seviyelerinde üretilen başlıklar (event/equipment/hardware headers) ve devamındaki uzun veri ise asıl algılanan veridir.
- Asağıdaki resimler asıl veriyi oluşturan 40 tane 1'li uzunluğundaki tek bir TPC “kelimesi”ni ve ilgili alanların anlamlarını göstermektedir.
- Verinin biçiminden görüldüğü gibi:
  - 10 tane 1'liden oluşan 4 kelimelik cümle,
  - 8 tane 1'liden oluşan 5 kelimelik cümle,
  - 40 tane 1'liden oluşan 1 kelimelik cümle ve
  - sağdan sola 4, 8, tekrar 4, 10 ve 14 olmak üzere boyları birbirinden farklı toplam 5 anlamlı alt “kelimecik”ten oluşan cümle (asıl ilgilendiğimiz)
- aslında tek bir gerçekliğe karşılık gelmektedir: TPC verisi
- Dolayısı ile bu veri bir metin düzenleyici ile okunursa 5 anlamsız isaret, 16'lık tabanda (hexdump ile örneğin) bastırılırsa ya yine anlamsız bir sayılar dizilimi ya da veri 16'nın tam katı olmadığı için kullanılan uygulamaya da bağlı olarak bir hata çıktısı, yüksek çözünürlüklü bir matematik uygulamasıyla açıldığında 40 bit'lik çok büyük anlamsız bir sayı ... v.b. Gözlenecektir.
- Bu nedendir ki üzerinde çalışılan bilgilerin saklandığı kütüklerin biçimleri hakkında bilgi sahibi olmak, birlikte çalışılacak komutların ve uygulamaların seçiminde kullanışlı olacaktır.

0000:0000	8c870000	fe5aleda	44000000	04000300	07000000	78130000	00000000	f842bf06
0000:0020	01000000	00000000	00000000	f0000000	00000000	20000000	03000000	ffffff
0000:0040	d30e0b43	48870000	11000000	69000000	00000000	00000000	00000000	04000000
0000:0060	01000000	02000000	03000000	04000000	05000000	06000000	07000000	08000000
0000:0080	3ef4d043	0f3df4e0	430f3df8	d0430f3d	f8e0430f	3df4d043	0f3ef4d0	430f3ef4
0000:00a0	d0430f3d	f4d0830f	3ef4d043	0f3ef8d0	430f3df4	d0430f3d	f8d0830f	3df8e003
0000:00c0	0f3ef8d0	430f3df4	d0430f3d	f4e0830f	3ef4d083	0f3df0d0	830f3df4	d0830f3d
0000:00e0	f4e0430f	3ef4d043	0f3ef8d0	430f3df4	e0830f3d	f0d0430f	3df4e043	0f7298a1
0000:0100	aaaa00a8	66a8aa3d	f8d0830f	3ef4d043	0f3ef4d0	430f3df4	e0830f3e	f4d0430f
0000:0120	3df0d043	0f3ef8d0	430f3df8	e0430f3d	f4e0430f	3df4e043	0f3df8e0	430f3df4
0000:0140	d0830f3e	f4d0830f	3ef4e083	0f3ef8d0	430f3ef4	e0430f3d	f4d0430f	3df4d043
0000:0160	0f3df4e0	430f3df4	d0830f3d	f4d0430f	3df4e043	0f3ef8c0	430f3df4	e0430f3c
0000:0180	f8e0830f	7298a1aa	aa01a866	a8aa2bac	b0c20a2a	b0c0c20a	2bacb0c2	0a2bb0b0
0000:01a0	c20a2aac	b0c20a2b	acb0c20a	2ab0b082	0a2aacb0	c20a2bb0	b0820a2b	acb0c20a
0000:01c0	2bacb0c2	0a2bb0c0	c20a2bac	b0820a2b	b0b0c20a	2bb0b0c2	0a2bacc0	020b2bac
0000:01e0	b0c20a2b	acb0c20a	2aacb0c2	0a2bacb0	c20a2bac	b0020b2b	acb0c20a	2bacb0c2
0000:0200	0a2bacc0	c20a2cac	b0c20a72	98a1aaaa	02a866a8	aa2bacb0	020b2cac	c0c20a2c
0000:0220	b0c0020b	2cb0c0c2	0a2cb0c0	c20a2cb0	c0c20a2b	acc0020b	2cb0c002	0b2bb0c0
0000:0240	020b2cac	c0c20a2c	b0c0020b	2cb0c002	0b2cb0b0	c20a2bb0	c0020b2c	acc0c20a
0000:0260	2bacc002	0b2cb0c0	c20a2cb0	c0c20a2c	acb0c20a	2bb0c002	0b2bacc0	c20a2cb0
0000:0280	c0020b2c	b0c0020b	2bacc0c2	0a2bb0c0	c20a7298	a1aaaa03	a866a8aa	28a070c2
0000:02a0	09279c60	c209279c	80020a27	9c70c209	289c70c2	09279c70	c20928a0	80c20928
0000:02c0	a070c209	27a070c2	0927a080	c209279c	70020a27	9c80c209	279c70c2	09289c70



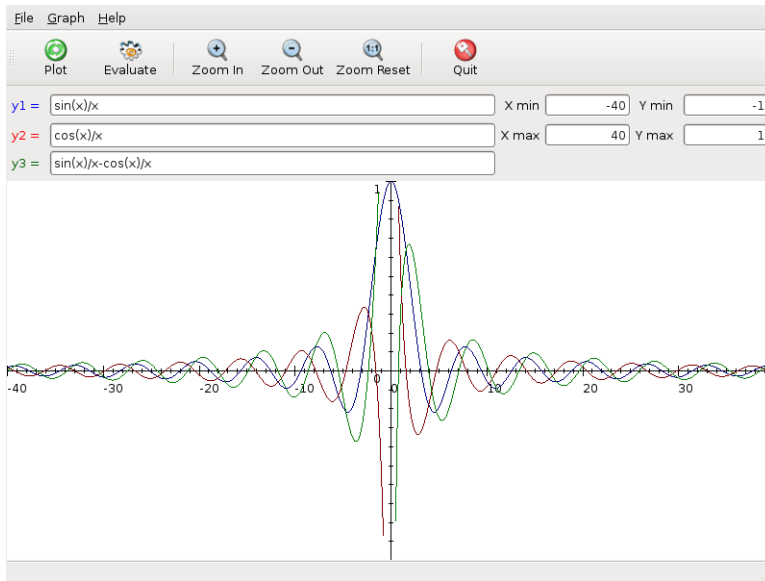
Seq.	bin	dec	(10 Bit Words, trailer)
Channel.	1111	15	(yellow)
Hardware.	10010110	150	(light blue)
Pattern A	0100	4	(orange)
10BitWord	0101011010	346	(magenta)
2AAA Pat.	01000001010001	4177	(green)



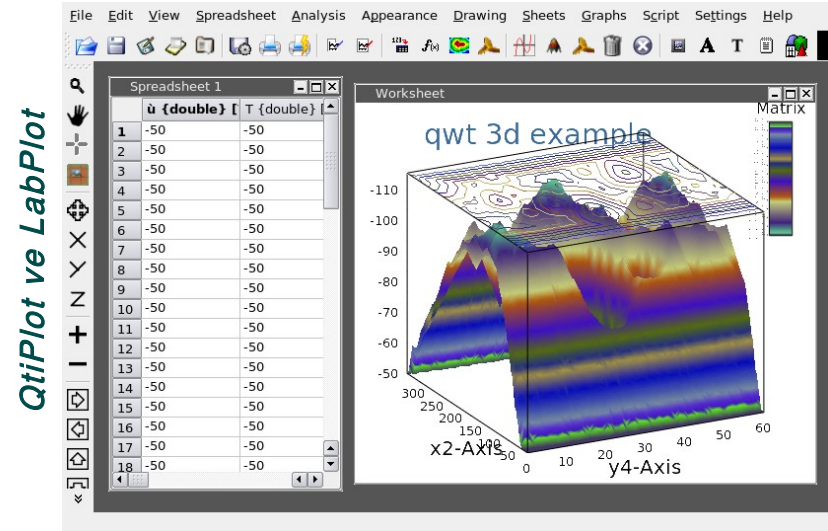
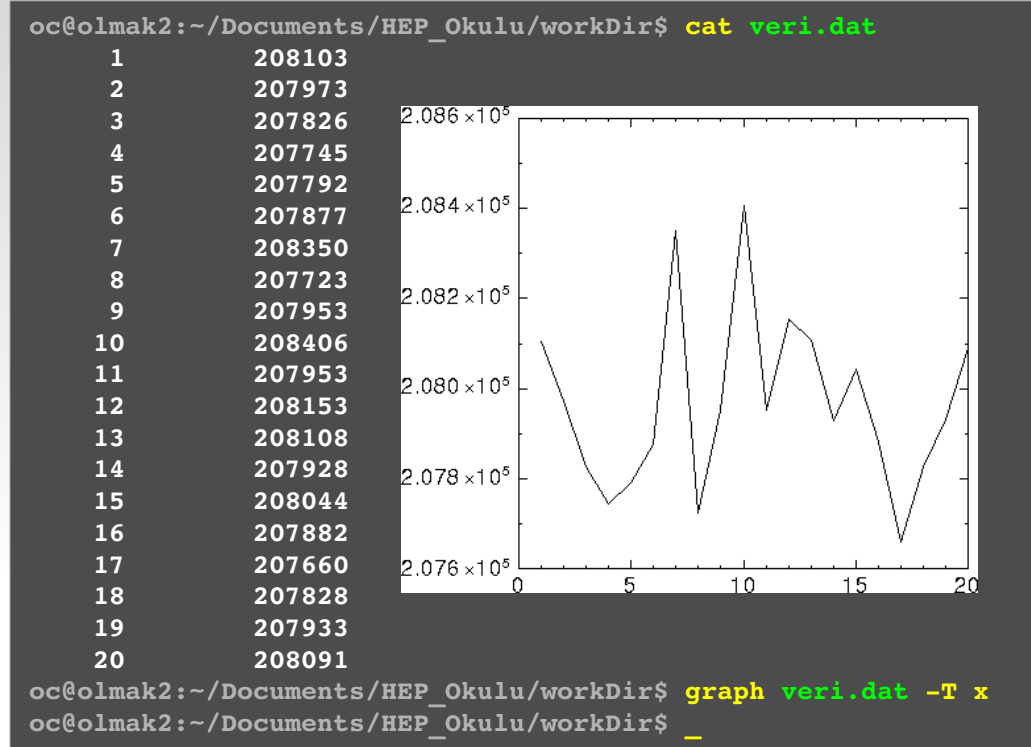
# Matematik İşlev ve Veri Çiziciler - I

## Verinin Okunabilirliğini Artırmak

- Hızlı bir biçimde iki boyutlu bir veriyi çizdirmek için *graph*
- Daha derin veri ayrıştırmaları (analizleri), 2 ve 3B renkli grafikler ve grafik canlandırmaları (animasyonları) için *xmgrace* ve *QtiPlot* veya benzeri *LabPlot*
- Veri değil de matematik işlev (fonksiyon) çizdirmek için *lybniz* ve *xmgrace*
- Benzer işler *spreadsheet* uygulamalarıyla da yapılabilir ancak veri büyüklüğü bir kaç 100 noktayı geçtiğinde ve veri ayrıştırma (analiz) gereksinimleri sadece basit uyum (fit) eğrileri çizmeyi aştığında -ki deneysel fizikte durum çoğunlukla budur- spreadsheet uygulamaları ihtiyaca cevap veremezler; özel uygulamalara ihtiyaç vardır.



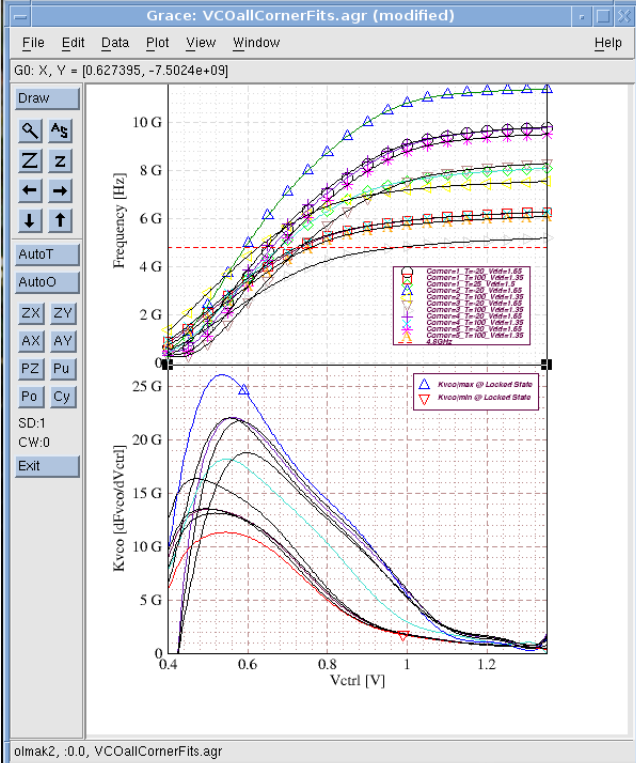
PlotUtils'den "graph"



# Matematik İşlev ve Veri Çiziciler - II

## Verinin Okunabilirliğini Artırmak

xmgrace



Grace: Non-linear curve fitting

Source: G0 (23 sets), G1 (24 sets)

Destination: G0 (23 sets), G1 (24 sets)

Set: G0.S0[2][20], G0.S1[2][20], G0.S2[2][20], G0.S3[2][20]

Formula:  $y = A0 \cdot \exp(A1 \cdot x(A2))$

Parameters: 3, Tolerance: 0.01, Iterations: 5

A0: 1, A1: 1, A2: 1

Buttons: Apply, Accept, Close

Regression

Apply to set: All sets

Type of fit: Linear

Load: Fitted values

Restrictions: None

Buttons: Accept, Close

Transform

Symbol properties: Type: Plus, Size: 100, Color: violet, Symbol char: 65

Line properties: Type: None, Style: Solid, Width: 1, Color: violet

Legend: String: Corner=4\_T=-20\_Vdd=1.65

Buttons: Apply, Accept, Close

Messages

```

SSerr = 2.00337e+15
Variable order: 0 1 2 3 4 5 6
Vector Beta:
0: 1.776e+10
1: -1.556e+11
2: 5.006e+11
3: -7.054e+11
4: 5.042e+11
5: -1.781e+11
6: 2.442e+10
Vector d:
0: 20
1: 1.663
2: 0.1097
3: 0.006895
4: 0.0094203
5: 2.487e-05
6: 1.425e-06
rbar matrix:
0.84875 0.888125 0.980433 1.10401 1.32424
1.75 2.446 3.20163 4.09903 5.21402
2.625 4.80571 7.62656 11.2705 14.8312
6.1425e-06 7.92917 14.8312 11.6165 5.25
Vector thetab:
0: 5.892e+09
1: 5.953e+09
2: -1.102e+10
3: 9.213e+09
4: 1.347e+10
5: -4.993e+09
6: 2.442e+10
y = 1.7764e+10 - 1.5637e+11 * x + 5.0064e+11 * x^2 - 7.0538e+11 * x^3 + 5.0426e+11 * x^4 - 1.7814e+11 * x^5 + 2.4422e+10 * x^6
Regression of set 4 results to set 23
    
```

- ◆ Windoz' daki Origin adlı veri çizici (plotter) ve ayrıştırıcı (analizer) yazılıma karşılık
- ◆ Hem veri hem de matematik islev çizici
- ◆ Kabuk programıymış gibi komut satırından anahtar kelime kabul edebilir ve dolayısı ile betiklerde komut olarak kullanılabilir
- ◆ X sunucusuna gereksinim duymaksızın grafik çıktı üretebilir (grafik gösterilmeksizin diske yazılır)
- ◆ Öntanımlı uyum eğrilerine ek olarak kullanıcının tanımladığı standart olmayan uyum (fit) işlevlerini (fonksiyonlarını) hesaplayabilir.
- ◆ Histogram, fourier transform, running average, türev ve integral, convolution v.b. ayrıştırma (analiz) işlevlerini barındırır.
- ◆ Kendi programlama dili aracılığı ile istenirse, tüm işlevleri (fonksiyonları) betiklerde (script) kullanılabilir.

# Bilgisayarlı Cebir - I

## Güvenli Simgesel Hesap

- **CAS** - Computer Algebra System (*Bilgisayarlı Cebir*) yazılımları son yüzyıla kadar elle yapılagelinen simgesel hesaplamalarda kolaylık sağlar ve hata payını düşürür.
- **MuPad/SciLab, Maxima, GiNaC, Octave, Mathematica** gibi büyük CAS yazılımları ve kütüphaneleri yanında, **Mathomatic** gibi etkin ve göreceli olarak küçük olanlar da mevcuttur.
- Mathomatic bir kabuk programıdır ve sadece C dili kullanılarak yazılmıştır, dolayısı ile **tek gerekliliği bir C derleyicisi ile standart C kütüphanesidir** (büyük yazılımlar pek çok başka şeye dayanıyor olabilir: yazılımlar, kütüphaneler, varolan kütüphanelerin uygun sürümleri v.b.)
- Sayısal hesaplamalarla birlikte simgesel hesapları da gerçekleştirebilir (eşitlikleri farklı şekillerde yeniden yazma, düzenleme, eşitliklerin çözülmesi, karmaşık (complex) sayılar, türev ve integral alma, laplace dönüşümü v.b.)
- Mathomatic ayrıca, eşitliklerden C, Java ve Python kodu üretebilmektedir (analiz yazılımı geliştirirken eşitliklerin doğrulanması bakımından kullanışlı olabilir)
- Yandaki örnekte basit bir f matematik işlevi (fonksiyonu) tanımlanmış (yeşil satır) ve bunun b için çözülmesi istenmiştir (yeşil b harfinden sonra enter).
- "f" işlevinin girilmesinden sonra araç ne anladığını terminale basmış ve bunu yaparken her bir parantez derinliği için ayrı renk kullanarak okunabilirliği artırmıştır.
- Benzer biçimde eşitliğin b için çözülmesi istendiğinde (b ve ardından enter) sonuç yine her parantez derinliği için ayrı renk kullanılarak konsola aktarılmıştır.
- Aşağıdaki pencerede mathomatic uygulamasının eşitlikler üzerinde gerçekleştirebileceği işlemlerin tamamı sıralanmıştır.

```
oc@olmak2:~/Documents/HEP_Okulu$ mathomatic
Mathomatic version 14.0.6 (www.mathomatic.org)
Copyright (C) 1987-2008 George Gesslein II.
This is free software with NO WARRANTY.
```

```
100 equation spaces available, 960 kilobytes per equation space.
ANSI color mode enabled.
```

```
1-> f=(a-b)*(c+d)/(2a-3c+5bd)
```

```

      (a - b)*(c + d)
#1: f = -----
      ((2*a) - (3*c) + (5*bd))
```

```
1-> b
```

```

      f*((2*a) - (3*c) + (5*bd))
#1: b = -1*(----- - a)
              (c + d)
```

```
1-> □
```

```
2-> help
```

```
Mathomatic is a small Computer Algebra System (CAS).
This help command is provided as a quick reference.
Type "help equations" for help with entering expressions and equations.
Type "help all" for a summary and syntax of all commands.
Type "help usage" to display the syntax of all commands.
Type "help geometry" for some commonly used geometric formulas.
"help" or "?" followed by command names will give info on those commands.
```

```
Available commands:
```

approximate	calculate	clear	code	compare
copy	derivative	display	divide	echo
edit	eliminate	extrema	factor	fraction
help	imaginary	integrate	laplace	limit
list	nintegrate	optimize	pause	product
push	quit	read	real	replace
roots	save	set	simplify	solve
sum	tally	taylor	unfactor	variables
version				

```
To select an equation space, type the equation number at the main prompt.
To solve the current equation, type the variable name at the main prompt.
```

```
2-> □
```

# Bilgisayarlı Cebir - II

## Güvenli Simgesel Hesap (devam)

- Diğer tüm uygulamalarda olduğu gibi CAS uygulamalarında da girilmek istenen harfler, matematik işlevler (fonksiyonlar) ve komutlar, daha sonra tekrar kullanılmak üzere kütüklerde saklanabilir. Aşağıdaki içerik “matGir” isimli kütüğe girilmiştir:

```
f=(1-(1-(y+1)/(x+y+1))/(1-x/(x+y+1)))/
((y+1)^2-x/(1+x/(y-x+1))*(x*(y+1)/(y-
x+1)-x))
code
simplify
factor
code
quit
```

- “matGir” isimli kütüğe yazılan yukarıdaki mathomatic komutları, yanda görüldüğü gibi işleme sokulur:
  - Göreceli olarak karmaşık bir “f” matematik işlevi girildikten sonra, daha sonra kendisiyle karşılaştırma yapılmak üzere “code” komutu ile f’ in C dilindeki ifadesi üretilir. Code komutu öntanımlı olarak C dilinde olmakla birlikte, python (code python) ve java (code java) dillerinde de çıktı üretebilir.
  - Bunu takiben “simplify” komutu ile ifadenin sadeleştirilmesi ve ardından verilen “factor” komutu ile de ifadenin çarpım biçiminde yeniden yazılması sağlanır.
  - Son olarak yine “code” komutu ile ifade C dilinde yeniden yazılır.
- Verilen iki “code” komutunun çıktılarını karşılaştırın.

```
oc@olmak2:~/Documents/HEP_Okulu/workDir$ mathomatic matGir
Mathomatic version 14.0.6 (www.mathomatic.org)
Copyright (C) 1987-2008 George Gesslein II.
This is free software with NO WARRANTY.

100 equation spaces available, 960 kilobytes per equation space.
ANSI color mode enabled.
1-> f=(1-(1-(y+1)/(x+y+1))/(1-x/(x+y+1)))/((y+1)^2-x/(1+x/(y-x+1))*(x*(y+1)/(y-x+1)-x))

          (y + 1)
          (1 - -----)
          (x + y + 1)
(1 - -----)
          x
          (1 - -----)
          (x + y + 1)
#1: f = -----
          x*(y + 1)
          x*(------ - x)
          (y - x + 1)
          ((y + 1)^2) - -----
          x
          (1 + -----)
          (y - x + 1)

1-> code
f = ((1.0 - ((1.0 - ((y + 1.0)/(x + y + 1.0)))/(1.0 - (x/(x + y + 1.0)))))/(pow((y + 1.0), 2.0) - (x*((x*(y + 1.0)/(y - x + 1.0)) - x)/(1.0 + (x/(y - x + 1.0))))));
1-> simplify
Division simplified with polynomial GCD.

          1
#1: f = -----
          (1 + (y^2) + (y*(2 + x)) + x + (x^2))

1-> factor

          1
#1: f = -----
          (1 + (y*(y + 2 + x)) + (x*(1 + x)))

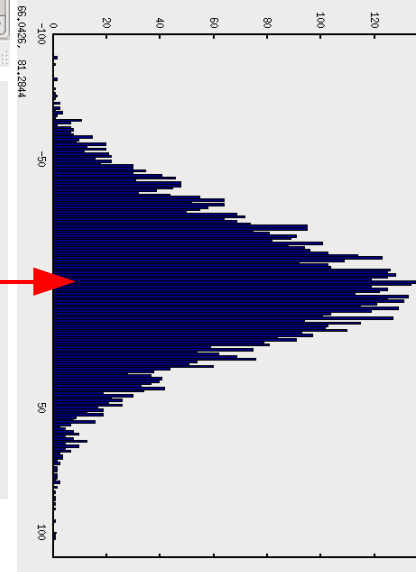
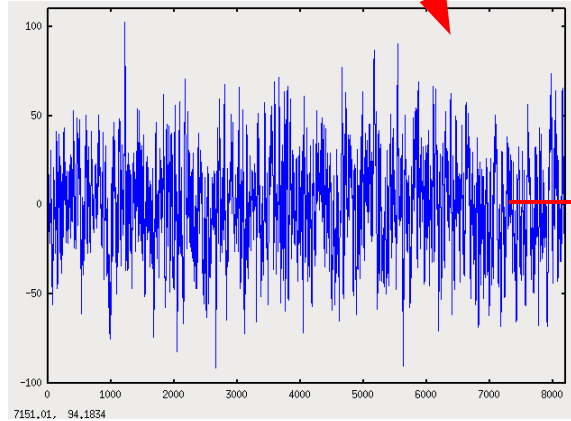
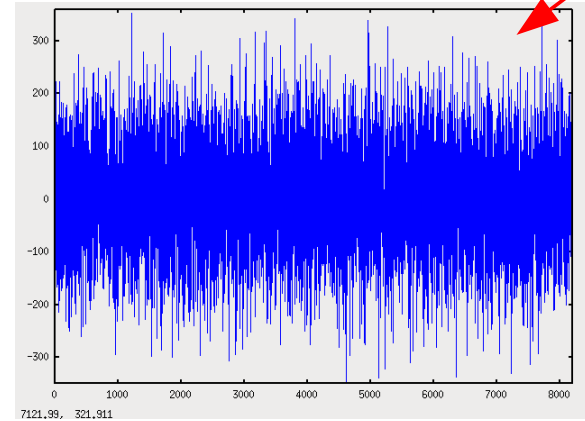
1-> code
f = (1.0/(1.0 + (y*(y + 2.0 + x)) + (x*(1.0 + x))));
1-> quit
oc@olmak2:~/Documents/HEP_Okulu/workDir$
```

# Tüm Mühendislik Hesapları İçin - I

## Octave

```
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut
runningAverage.m rnd.dat
1 % calculates the running average of a time series
2
3 clear;
4
5 load rnd.dat;
6 offset=8;
7 length=length(rnd);
8 i=offset;
9
10 while(i<length-offset)
11     j=-7;
12     sum=0;
13     while(j<8)
14         sum=sum+rnd(i+j);
15         j=j+1;
16     endwhile
17     runningAverage(i)=sum/15;
18     i=i+1;
19 endwhile
20
21 save runningAverage.dat runningAverage;
22
23 plot(rnd); figure;
24 plot(runningAverage); figure;
25 hist(rnd, 200); figure;
26 hist(runningAverage, 200); figure;
27
28 pause;
```

```
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut
runningAverage.m rnd.dat
1 1.969035e+01
2 -5.497029e+01
3 3.709430e+01
4 4.045590e-01
5 -1.099759e+02
6 9.326301e+01
7 2.225967e+02
8 -1.950402e+01
9 -4.286325e+01
10 4.225486e+01
11 -8.823909e+01
12 -7.340794e+01
13 5.040934e+01
14 -1.196590e+01
15 1.916279e+02
16 -1.338167e+02
17 -7.349054e+01
18 1.453011e+02
19 -1.681152e+02
20 -1.448187e+01
21 2.850741e+00
22 4.081624e+01
23 -6.777451e+01
24 -8.666743e+01
25 1.034120e+02
26 -1.465114e+02
27 -1.060710e+01
28 1.461387e+02
```



- Sayısal hesaplamaları hedef alan yüksek seviyeli (insan diline yakın) kullanıcı ile etkileşimli (interactive) bir dil.
- *Matlab* uygulamasının açık kaynak karşılığı
- Örnek olarak bir veri setinin değişen ortalamasını (running average) hesaplayan bir işlev yazılmıştır
- Aşağıda sıralanan işlevlerin ne kadar kolay yapıldığına dikkat edilmesi gerekir:
  - ➔ rnd.dat kütüğünün okunuşu (load)
  - ➔ okunan kütüğün boyutunun hesaplanması (length)
  - ➔ sonuçta elde edilen yeni verinin kaydedilişi (save)
  - ➔ bu verinin ve histogramının çizdirilmesi (plot & hist)

Octave, gnuplot' u da kullanabilir grafikler için

# Tüm Mühendislik Hesapları İçin - II

Octave

- ➔ Sistem tasarımı ile ilgili genel amaçlı işlevleri sunar
- ➔ Hızlandırıcı ve ilgili pek çok alt sistemin tasarımında, davranış (behavioural) ve sayısal benzetimlerde (simülasyon) kullanılır
- ➔ Örneğimizde bir geri besleme sisteminin davranışını istenilen hale getirmek için bir model oluşturuyor (daha doğrusu oluşturduğumuz modeli octav betiğine giriyoruz) ve model değişkenlerinin en uygun değerlerini bulmaya çalışıyoruz.
- ➔ Sonuç olarak girdiğimiz sistemin ve değişkenlerinin girdiğimiz değerlerinin nasıl bir davranışa yol açacağını sayısal olarak hesaplamak (benzetmek, simülasyonunu yapmak) mümkün.
- ➔ Örnekte görülen sistem kararlı olmasına rağmen kararlılık noktasına salınarak yaklaşan bir davranış biçimine sahip. Root locus, Bode grafikleri ve sistemin atkı (impulse) cevabı (response) ya da delta işlevine verdiği cevap görülmekte.

```
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
cppll.m
1 #####
2 #
3 # OCTAVE script to evaluate the closed loop parameters of a CP-PLL. #
4 # Loop parameters are calculated by CaPpELLo and "condition" #
5 # variable corresponds to 16 different Loop parameter sets. #
6 #
7 # http://www.cern.ch/ocobanog, Ozgur.Cobanoğlu@cern.ch #
8 #
9 #####
10 #
11 #      Kp      Icp      Tau      Kvco      #
12 # +-----+ +-----+ +-----+ +-----+ #
13 # Wi -->| PFD |-->| CP |-->| LPF |-->| VCO |-->| Wo #
14 # +-----+ +-----+ +-----+ +-----+ | #
15 # | #
16 # | Wo/N      +-----+      N*Wi #
17 # +-----+ | %N |<-----+ #
18 # #
19 # #
20 # [System under consideration] #
21 #####
22
23 clear
24
25 condition = -1;
26
27 #####
28 # Quick parameter set for CP-PLL - by CaPpELLo # (Deferine bak)
29 #####
30
31 if (condition == -1)
32 # CaPpELLo - Kvco 22.0e9 -N 120 -Wi 40.0e6 -Ksi 4.236 -Wn 1.498e6 -Icp 145.0e-6
33 wn = 9.412212e6;
34 N = 120.000000;
35 Ksi = 4.236000;
36 Ko = 138.230087e9;
37 Icp = 144.999998e-6;
38 C = 300.072418e-12;
39 R = 2999.633545;
40 K = 79740264.000000;
41 Tau = 0.900107e-6;
42 KTau2 = 71.774789;
43 WiTau2 = 226.221634;
44 endif
45
```

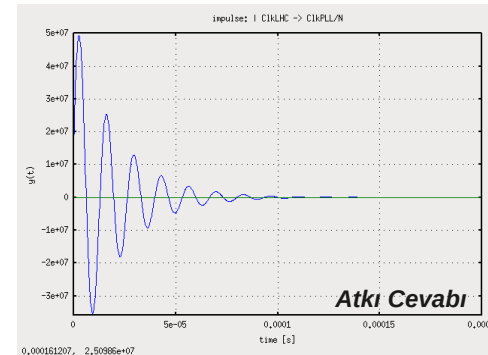
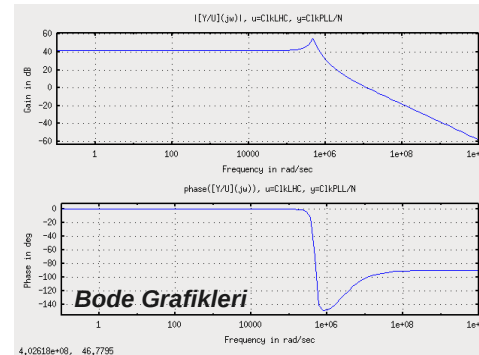
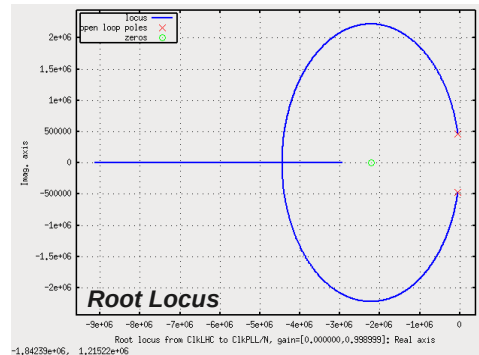
```
File Edit View Search Tools Documents Help
New Open Save Print... Undo Redo Cut Copy Paste Find Replace
cppll.m
369 #####
370 # Enter the transfer function #
371 #####
372
373 #C3 = 0.1e-12;
374 #b = 1+C/C3;
375
376 num = [(R*C*wn^2) (wn^2)];
377 den = [(1/N) (2*Ksi*wn/N) (wn*wn/N)];
378 #num = [(K*N*(b-1)/b) (K*N*(b-1)/(b*Tau))];
379 #den = [(Tau/b) (1) (K*(b-1)/b) (K*(b-1)/(b*Tau))];
380 T = tf(num, den, 0, "clkLHC", "clkPLL");
381
382 # See what you entered
383
384 sysout(T)
385
386 # Extract some parameters to cross check
387
388 damp(T)
389
390 # Do I have a manageable system (Terminology from Control Theory)
391
392 is_observable (T)
393 is_controllable (T)
394 is_stabilizable (T)
395 is_detectable (T)
396 is_stable (T)
397
398 #####
399 # Simulate the system #
400 #####
401
402 wrange = logspace(log10(0.1),log10(10^10),100);
403 #impulse(T, 1, 1.0*10^-6, 1000); figure;
404 #impulse(T, 1, 1.0*10^-6, 1000); figure;
405 #step(T, 1, 1.0*10^-6, 1000); figure;
406 #step(T, 1, 50*10^-6, 1000); figure;
407 bode(T, wrange); figure;
408 rlocus(T, 0.1, 0.0, 0.0); figure;
409 #rlocus(T, 0.0001, 0.0, 1.0); figure;
410
411 #####
412 # Noise performance of the system #
413 #####
```

$$T(s) = \frac{\omega_n^2(\tau s + 1)}{s^2 + 2\xi s \frac{\omega_n}{N} + \frac{\omega_n^2}{N}}$$

$$\omega_n = \sqrt{\frac{K_o I_p}{2\pi C_1 N}}$$

$$\xi = \frac{\tau \omega_n}{2}$$

$$K = \frac{K_o I_p R}{2\pi N}$$



# Erişim, Aktarım ve Yedeklemek

## Güvenli Uzak Erişim, Kütük Aktarımı ve Veri Yedeklemek

- Uzak bir bilgisayara güvenli erişim için “ssh <bilgisayar> -l <kullanici>” veya “ssh kullanıcı@bilgisayar” komutları sıkça kullanılır:

```
ssh lxplus.cern.ch -l ocobanog ya da  
ssh ocobanog@lxplus.cern.ch
```

- Bilgisayarlar arası kütük ve izin değiş tokuşu için “scp” komutu sıklıkla kullanılır. Bu komut “ssh” komutunun sağladığı güvenliği sağlar. Örnek olarak “sehirler.txt” kütüğünü “cern.ch” alanındaki “lxplus” adlı bilgisayarın üzerindeki kullanıcılardan biri olan “ocobanog”un “home” dizinine (“.” ile sağlanır) çoğaltmak (kopyalamak) için:

```
scp sehirler.txt ocobanog@lxplus.cern.ch:
```

Aynı işlemi “home” dizini içindeki “public” dizinini kullanarak yapmak için ise:

```
scp sehirler.txt ocobanog@lxplus.cern.ch:~/public
```

komutları kullanılabilir. Belirtmeye gerek dahi yok, özel işaretler scp gibi komutlarla kullanılabilir, örneğin:

```
scp *.dat ocobanog@lxplus.cern.ch:~/public
```

komutu “.dat” ile biten tüm kütükleri uzak bilgisayara çoğaltacaktır (kopyalayacaktır). Bu komut ile sadece kendi bilgisayarımızdan veya ssh ile bağlandığımız başka bir bilgisayardan değil, uzak iki bilgisayar arasında da kütük ve izin aktarımı yapılabilir:

```
scp ocobanog@lxplus.cern.ch:~/public/sehirler.txt cobanoglu@proton.istanbul.edu.tr:~/pub
```

yukarıdaki komut ile, *cern.ch* alanında *ocobanog* ve *proton.istanbul.edu.tr* alanında *cobanoglu* olarak bilinen kullanıcı, CERN'deki hesabından kendi üniversitesindeki hesabına *sehirler.txt* kütüğünü gönderiyor. Dizinlerin de gönderilebilmesi için scp komutuna -r (recursive) anahtarının verilmesi yeterlidir.

```
scp -r ocobanog@lxplus.cern.ch:~/public/sehirler.txt cobanoglu@proton.istanbul.edu.tr:~/pub
```

- Kullanıcıların düzenli olarak veri yedeklemesi çoğunlukla önemsenmeyen ve yeterince sık yapılmayan ama hayati önem taşıması nedeni ile edinilmesi gerekli olan bir alışkanlıktır. Pek çok veri yedekleme yazılımı vardır. Örneğimiz linuxun standart komutlarından biri olan “rsync” ile verilecektir.

```
rsync [seçenekler] /yedeklenmek/istenen/dizin /yedeğin/saklanacağı/dizin  
rsync -avz /home/ozgur /dev/sdb3/yanginda/ilk/kurtarilacak
```

Bu komut esnek ve güvenilirdir. Ağ üzerinden sadece değişikliğe uğramış (kütük tarihine ve büyüklüğüne bakılarak) kütükleri ve bu kütüklerin de sadece değişikliğe uğramış kısımlarını aktarması (delta-transfer algorithm) nedeniyle çok hızlıdır. Önerilen, içinde tez/makale gibi önemli metinlerin bulunduğu veya analiz yapılan dizinlerin *her gün* ya da *haftada bir* yedeklenmesidir.

- Daha ayrıntılı bilgi için yukarıdaki komutların *man* ve *info* sayfalarına bakılabilir.

# Ararken...

## google'arken, altavista'rken, vikipedya'rken v.b.

- ◆ Varolan tüm komutlar ve uygulamaları bilmek mümkün değildir. Ayrıca yapmakla yükümlü olduğumuz her işe karşılık o işi yapan bir uygulamanın varolması da gerekli değildir. Yine de açık kaynak uygulamalar çok çeşitlidir ve kapsadıkları alan her geçen gün genişlemektedir.
- ◆ Bildiğimiz uygulamalar ve yollar ile göreceli olarak uzun bir sürede çözülebileceğini düşündüğümüz bir sorun (problem) ile ilgili olarak internet üzerinde daha önce başkasının ilgili bir uygulama ya da komut geliştirip geliştirmedicine bakmak zaman kazandırabilir.
- ◆ Bu durumda en yaygın olarak kullanılan **arama motorlarından** birini seçmeli (google, altavista v.b.) ve sorunumuza yönelik çözümü aramalıyız.
- ◆ Tam bu noktada **“uygun kelimeyi bilmek”** sorunsalı ile yüzleşiriz: hangi kelime ya da kelime toplulukları bizi arama motorlarında ihtiyacımız olan çözüme ulaştırabilir ?
- ◆ Bu soruya pek çok farklı cevap verilebilir; biz konumuzun ismi nedeniyle uygulama aramaya yönelik aslında daha önce verilmiş bir önermeyi tekrarlayacağız burada: **uygulamaların karşılık geldiği teknolojinin ya da fikrin adını bilmek** arama işlemini kolaylaştırır.
- ◆ Örnek olarak, “xmgrace” uygulamasını hiç bilmediğimizi ve linuxun en iyi veri çizici/ayrıştırıcısını (analizer) aradığınızı varsayalım. Xmgrace ismi kolay hatırlanır ya da kendini açıklayan bir isim değil ayrıca uluslararası ağ (internet) üzerindeki yaygın dil İngilizce, dolayısı ile “veri çizici” gibi Türkçe aramalar kısıtlı sonuç döndürecektir.
- ◆ Çözüme ulaşmanın en kestirme yolu çoğunlukla, aradığımız işlevin (fonksiyonun) karşılık geldiği teknolojinin ya da kavramın İngilizce'de hangi kelime ya da kelimelerle karşılandığını bilmek olabilir. Aşağıda buna örnek olabilecek bazı karşılıklar kısaca sıralanmıştır:
  - ➔ Metin düzenleyici – text editor, ascii file editor, source code editor
  - ➔ Derleyici/yorumlayıcı – compiler/interpreter
  - ➔ Yorumlanan dilden çalıştırılabilir kütük üreten – bit byte compiler
  - ➔ Farklı bir platform için derlemek – cross compiling
  - ➔ Matematik işlev çizici – function plotter
  - ➔ Veri çizici – data plotter
  - ➔ Komut satırında çalışan bir yardımcı program – command line utility, batch mode utility
  - ➔ Tablolu hesap makinası – spreadsheet (MS Excel, OpenOffice Calculator v.b.)
  - ➔ Sonsuz hassaslıkta sayısal işlem yapan hesap makinası/kütüphane – arbitrary precision calculator/library
  - ➔ Kelime işlemci – word processor, WYSIWYG editor, WYSIWYM editor
  - ➔ Herhangi bir konu (latex, c/c++ v.b.) ile ilgili tüm ilgili ihtiyaçları içinde barındıran çalışma ortamı – IDE, integrated development environment
  - ➔ Yazılan c/c++ programından mantık ya da akış diyagramı üreten program – flow chart generator
  - ➔ Resim işleme yazılımı – image editing tool, image processing application
  - ➔ Bir şeyden başka bir şeye çeviren yazılım – <bir şey> to <başka bir şey> converter
  - ➔ Eşitlikleri simgesel olarak türeten yazılım – CAS, computer algebra system
  - ➔ Excel gibi bir uygulama – Excel-like application, app like Excel
  - ➔ v.b.



# Ararken...

*google'arken, altavista'rken, wikipedya'rken v.b.*



***Doğru kelimeleri bilmek, doğru büyüleri bilmektir !..***

# Linux ve Uygulamalar - I

## İşletim sistemi ve uygulamalarından oluşan bilimsel çalışma ortamına giriş

- ◆ Kabuk ve Komut Satırı
  - ◆ Giriş: Linux kabuktur
  - ◆ Bazı Çevre Değişkenleri
  - ◆ Alias (Komut İsimlerini Değiştirmek)
- ◆ Kabuk İşlemleri
  - ◆ Standart giriş, çıkış ve hata' nın yönlendirilmesi
  - ◆ Süzgeç komutları
- ◆ Kabuk ve Komut Satırı
  - ◆ Özel İşaretler
  - ◆ Kütük ve Dizin Bulmak
- ◆ Kabukta Süreç Yönetimi
  - ◆ Süreçleri İzlemek ve Sonlandırmak
  - ◆ Süreç Öncelikleri ve Ardalanda Çalıştırma
- ◆ Kabuk Programlama
  - ◆ Betik Genel Yapısı ve Çalıştırma
  - ◆ Neden var ?
  - ◆ Bir Örnek
  - ◆ Bilmediğimi Biliyorum
- ◆ Uygulamalar
  - ◆ Ayrımlar
- ◆ Metin Düzenleyiciler
  - ◆ 8 Bitlik Gözlüklerimiz
- ◆ Kütüklerin Anlamlılık Ölçekleri
  - ◆ Ölçek Seçimi Sonuçtur
  - ◆ Ölçek Seçimi Anlamdır
- ◆ Matematik İşlev ve Veri Çiziciler
  - ◆ Verinin Okunabilirliğini Artırmak
- ◆ Bilgisayarlı Cebir
  - ◆ Güvenli Simgesel Hesap
- ◆ Tüm Mühendislik Hesapları
  - ◆ Octave
- ◆ Güvenli Uzak Erişim, Kütük Aktarımı ve Veri Yedeklemek
- ◆ Ararken
- ◆ Akşam Sefası
  - ◆ Ertesi günün sabahına hazırlanması beklenen aksamlık ödevler

# Aksam Sefası

## Ertesi günün sabahına hazırlanması beklenen akşamlik ödevler

- Uptime komutunun çıktısında sonda bulunan sayılar ne anlama gelmektedir ?  
oc@olmak2:~\$ uptime  
23:56:28 up 1 day, 2:45, 4 users, load average: 0.06, 0.07, 0.07  
oc@olmak2:~\$ \_
- Aile soy ağacı tutan açık-kaynak bir veri tabanı uygulaması arıyorsunuz. Bu veri tabanı uygulaması MySQL gibi standart bir veritabanı uygulamasından farklı olmalı çünkü bu uygulama girilen bireylerin ve verilen ailevi bağlılıkların (evli olmak, birinin oğlu veya dedesi olmak vb) mantıklı olup olmadığını da denetleyebilmelidir. Böyle bir uygulamayı google'da hangi kelimelerle aradınız ? Kullandığınız arama kelimelerinin sizi ulaştırdığı uygulama hangisi ? (En az bir uygulama bulunuz)
- Konu içinde “yedekle” ismiyle verilen kabuk betiğinin, yedekleme gerçekleştirirken yalnızca kütükleri değil aynı zamanda dizinleri de içermesini sağlayınız. Başka geliştirme fikirleriniz varsa savununuz ve ekleyiniz.
- Belirli bir işlevi yerine getiren aşağıdaki c/c++ kaynağının sonuç vermesi uzun zaman almaktadır. İşlev değişmeden kalmak şartıyla bu zamanı kısaltınız. (c/c++ dilinde “math.h” içinde, pi sayısı  $M\_PI$  ve  $a^b = \text{pow}(a, b)$  olarak karşılık bulur.)

```
#include <stdio.h>
#include <math.h>
int main() {
    float T, C1=300e-12, Ip=10e-9, N=120.0, R=1500.0, toplam = 0.0;
    int Ko, s;
    for (Ko=1 ; Ko<10000 ; Ko=Ko+1) {
        for (s=1 ; s<10000 ; s=s+1) {
            T = (pow((pow(((Ko*Ip)/(2*M_PI*C1*N)),0.5)),2)*((R*C1)*s+1))/
                (pow(s,2)/N+2*
                 ((R*C1)*(pow(((Ko*Ip)/(2*M_PI*C1*N)),0.5)))/2) *
                 s*(pow(((Ko*Ip)/(2*M_PI*C1*N)),0.5))/N +
                 pow((pow(((Ko*Ip)/(2*M_PI*C1*N)),0.5)),2)/N);
            toplam = toplam + T;
        }
    }
    printf("Toplam = %f \n", toplam);
    return 0;
}
```