

- Dosya sistemi
- prosesler – ps komutu
- Hata yönetme
- Sinyaller

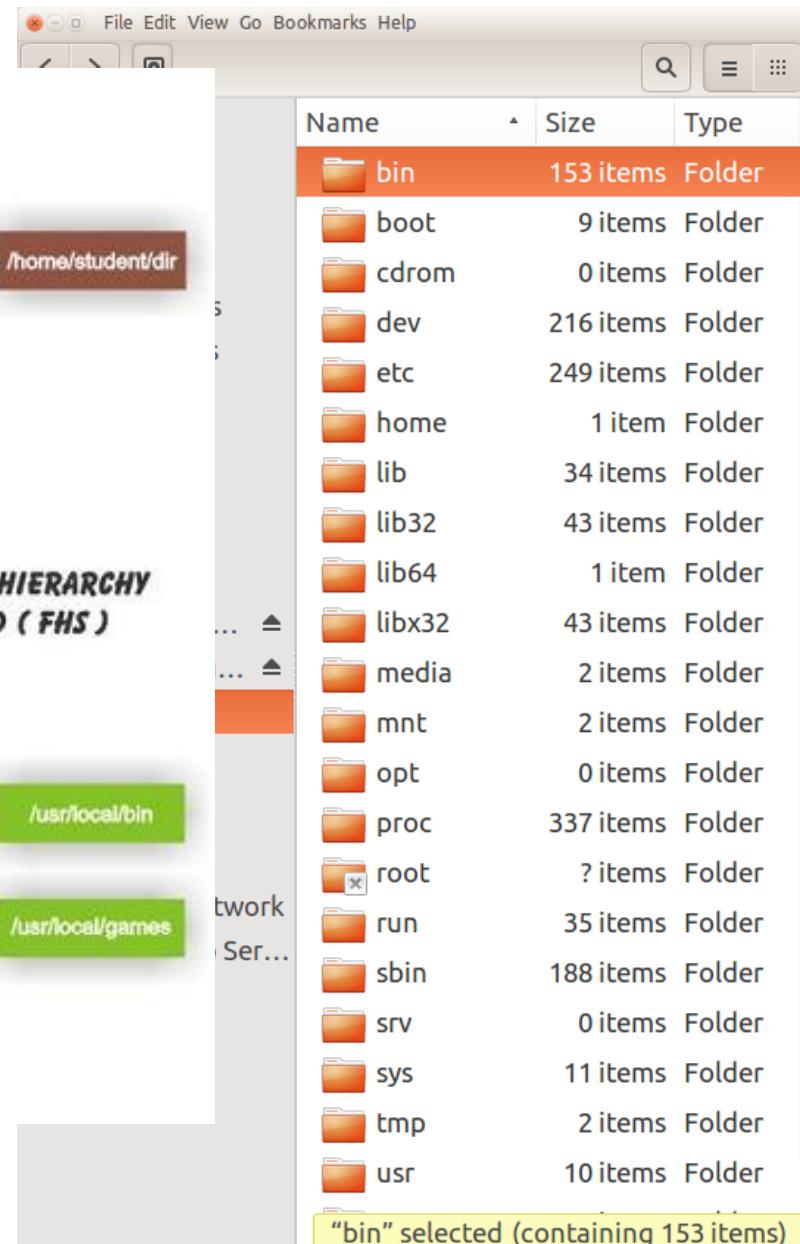
Dosya sistemi - File System

- Dosya sistemi klasörlerin (directory, dizin) hiyerarşik olarak düzenlenmesidir

```
File Edit View Search Terminal Help
bilg@bilg:~$ df
Filesystem      1K-blocks   Used Available Use% Mounted on
/dev/sda7        67153528 10474548  53244692  17% /
none             4          0          4          0% /sys/fs/cgroup
udev             8146692    4          8146688  1% /dev
tmpfs            1631476   1592     1629884  1% /run
none              5120     0          5120     0% /run/lock
none              8157368   420      8156948  1% /run/shm
none              102400   48       102352  1% /run/user
/dev/sda5        125653824 13946356 105301584 12% /home
bilg@bilg:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda7        65G  10G  51G  17% /
none             4,0K  0  4,0K  0% /sys/fs/cgroup
udev             7,8G  4,0K  7,8G  1% /dev
tmpfs            1,6G  1,6M  1,6G  1% /run
none              5,0M  0  5,0M  0% /run/lock
none              7,8G  420K  7,8G  1% /run/shm
none              100M  48K  100M  1% /run/user
/dev/sda5        120G  14G  101G 12% /home
bilg@bilg:~$
```

df - report file system disk
space usage

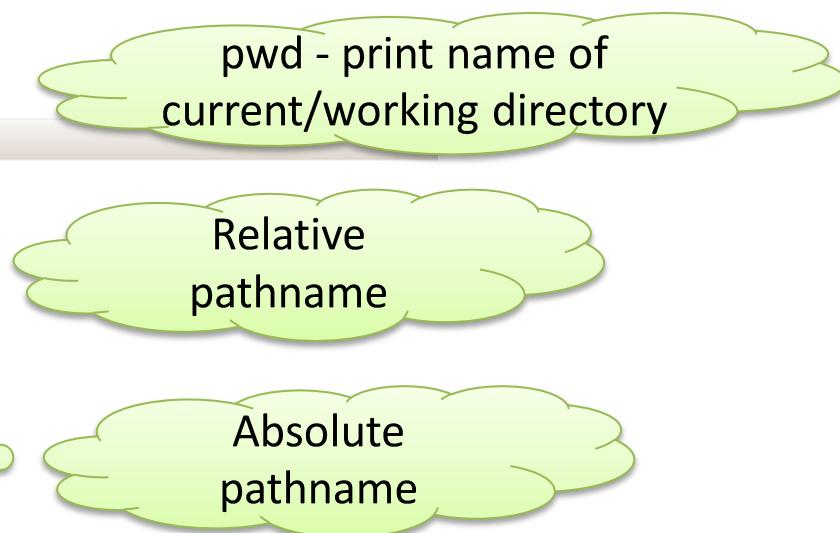
Dosya sistemi - File System



Dosya sistemi - File System

- **Filename:** dosya adı
- **Pathname:** yol adı. /home/bilg/Documents
- **Absolute Pathname:** A pathname starting with a slash.
/home/bilg/Documents
- **Relative Pathname:** Çalışılmakta olan klasöre göre yol.
- **Working Directory:** Çalışılmakta olan klasör
- **Home Directory:** kullanıcı oturum açtığı zaman
bulunduğu klasör

```
File Edit View Search Terminal Help
bilg@bilg:~$ pwd
/home/bilg
bilg@bilg:~$ cd Documents
bilg@bilg:~/Documents$ pwd
/home/bilg/Documents
bilg@bilg:~/Documents$ cd ..
bilg@bilg:~$ cd /home/bilg/Documents
bilg@bilg:~/Documents$ █
```

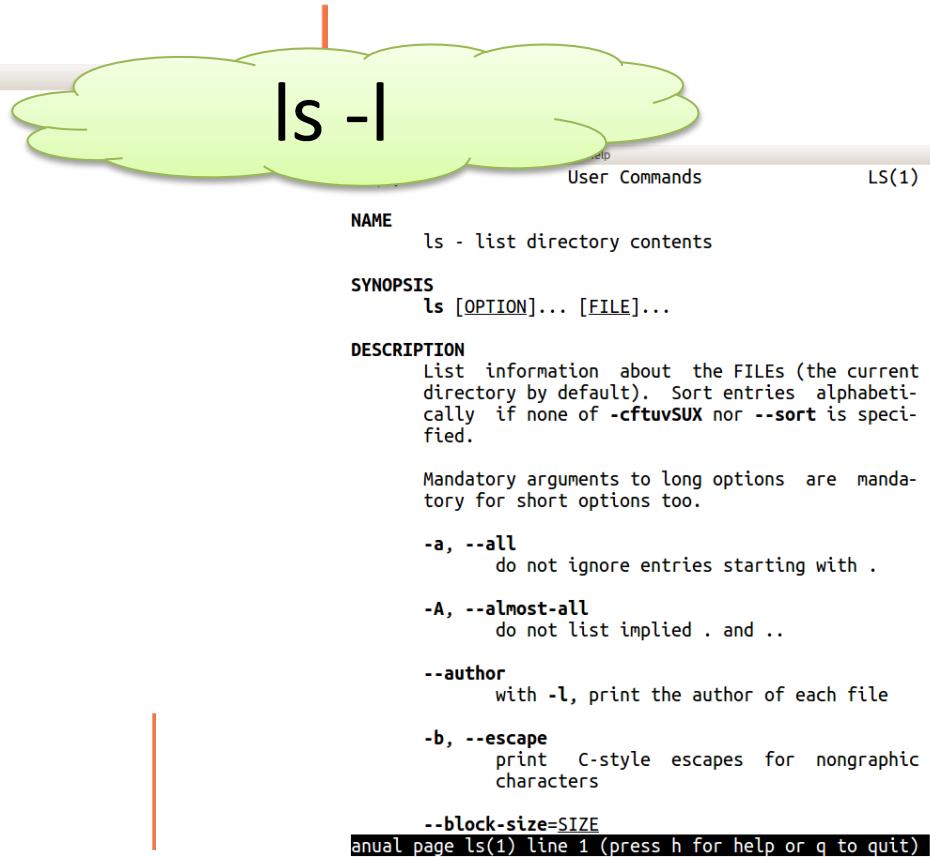


Dosya sistemi - File System

```
File Edit View Search Terminal Help  
bilg@bilg:~/Documents/ders1a/h2$ ls  
 endian.c    id3.cpp   id6.cpp      p1.c   p5.c   pc.c  test  
 endian.jpg  id5a.c   lecture.html  p2.c   p8.c   pd.c  
 id1.c       id5b.c   p12.c      p3.c   p9.c   pm  
 id2.c       id5.c    p13.c      p4.c   pa.c   pm.c  
bilg@bilg:~/Documents/ders1a/h2$
```

ls - list directory contents

```
File Edit View Search Terminal Help  
bilg@bilg:~/Documents/ders1a/h2$ ls -l  
total 340  
-rw----- 1 bilg bilg 1798 Oca 18 2015 endian.c  
-rw----- 1 bilg bilg 196427 Oca 18 2015 endian.jpg  
-rw----- 1 bilg bilg 188 Oca 14 2013 id1.c  
-rw----- 1 bilg bilg 188 Oca 13 2015 id2.c  
-rw----- 1 bilg bilg 292 Oca 14 2013 id3.cpp  
-rw----- 1 bilg bilg 293 Şub 23 10:36 id5a.c  
-rw----- 1 bilg bilg 359 Şub 23 10:46 id5b.c  
-rw----- 1 bilg bilg 313 Oca 14 2013 id5.c  
-rw----- 1 bilg bilg 320 Oca 14 2013 id6.cpp  
-rw----- 1 bilg bilg 38060 Oca 18 2015 lecture.html  
-rw-rw-r-- 1 bilg bilg 151 Şub 23 09:52 p12.c  
-rw-rw-r-- 1 bilg bilg 212 Şub 23 10:15 p13.c  
-rw----- 1 bilg bilg 141 Şub 23 09:43 p1.c  
-rw----- 1 bilg bilg 120 Şub 23 10:20 p2.c  
-rw----- 1 bilg bilg 193 Şub 23 11:15 p3.c  
-rw----- 1 bilg bilg 168 Şub 23 11:12 p4.c  
-rw----- 1 bilg bilg 187 Oca 25 2011 p5.c  
-rw----- 1 bilg bilg 473 Şub 23 11:27 p8.c  
-rw----- 1 bilg bilg 470 Oca 25 2011 p9.c  
-rw----- 1 bilg bilg 80 Ağu 23 1996 pa.c  
-rw----- 1 bilg bilg 203 Oca 25 2011 pc.c  
-rw----- 1 bilg bilg 353 Oca 25 2011 pd.c  
-rwxrwxr-x 1 bilg bilg 8987 Şub 23 11:41 pm  
-rw----- 1 bilg bilg 669 Şub 23 11:41 pm.c  
-rwxrwxr-x 1 bilg bilg 8657 Şub 23 11:45 test  
bilg@bilg:~/Documents/ders1a/h2$
```



Programs and processes

- **Program:** Doğrudan veya interpreters, compilers, linkers yardımıyla çalıştırılabilen dosyalar.
- **Process:** Bir programın çalışmakta olan bir örneği proses olarak adlandırılır.
- **Process ID:** Bir prosese işletim sistemi tarafından verilen tanımlayıcı sayı.

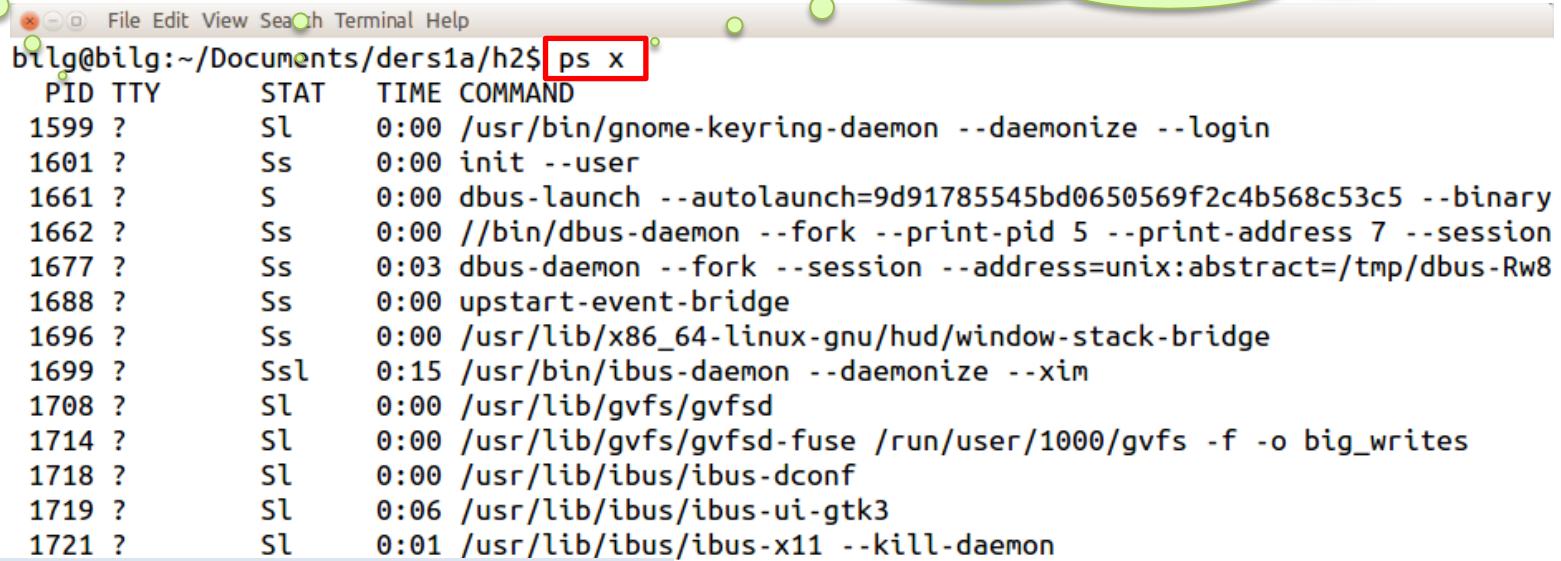
Programs and processes

PID:

Process id

Durum

ps - report a snapshot of the current processes



```
bilg@bilg:~/Documents/ders1a/h2$ ps x
  PID TTY      STAT   TIME COMMAND
 1599 ?        Sl     0:00 /usr/bin/gnome-keyring-daemon --daemonize --login
 1601 ?        Ss     0:00 init --user
 1661 ?        S      0:00 dbus-launch --autolaunch=9d91785545bd0650569f2c4b568c53c5 --binary
 1662 ?        Ss     0:00 //bin/dbus-daemon --fork --print-pid 5 --print-address 7 --session
 1677 ?        Ss     0:03 dbus-daemon --fork --session --address=unix:abstract=/tmp/dbus-Rw8
 1688 ?        Ss     0:00 upstart-event-bridge
 1696 ?        Ss     0:00 /usr/lib/x86_64-linux-gnu/hud/window-stack-bridge
 1699 ?        Ssl    0:15 /usr/bin/ibus-daemon --daemonize --xim
 1708 ?        Sl     0:00 /usr/lib/gvfs/gvfsd
 1714 ?        Sl     0:00 /usr/lib/gvfs/gvfsd-fuse /run/user/1000/gvfs -f -o big_writes
 1718 ?        Sl     0:00 /usr/lib/ibus/ibus-dconf
 1719 ?        Sl     0:06 /usr/lib/ibus/ibus-ui-gtk3
 1721 ?        Sl     0:01 /usr/lib/ibus/ibus-x11 --kill-daemon
pi2-core/at-spi-bus-launcher
mon --config-file=/etc/at-spi2/accessibility.conf --n
pi2-core/at-spi2-registryd --use-gnome-session
y-settings-daemon/unity-settings-daemon
64-linux-gnu/hud/hud-service
bridge --daemon --session --user --bus-name session
bridge --daemon --user
bridge --daemon --system --user --bus-name system
--session=ubuntu
y/unity-panel-service
/ibus-engine-simple
64-linux-gnu/bamf/bamfdaemon
f/dconf-service
1.0 -t -K -R
```

Process state codes

The codes used are:

| Code | Meaning |
|------|--|
| D | Uninterruptible sleep (usually IO) |
| R | Running or runnable (on run queue) |
| S | Interruptible sleep (waiting for an event to complete) |
| T | Stopped, either by a job control signal or because it is being traced. |
| W | paging (not valid since the 2.6.xx kernel) |
| X | dead (should never be seen) |
| Z | Defunct ("zombie") process, terminated but not reaped by its parent. |

For BSD formats and when the stat keyword is used, additional characters may be displayed:

| Code | Meaning |
|------|---|
| < | high-priority (not nice to other users) |
| N | low-priority (nice to other users) |
| L | has pages locked into memory (for real-time and custom IO) |
| s | is a session leader |
| I | is multi-threaded (using CLONE_THREAD, like NPTL pthreads do) |
| + | is in the foreground process group |

Programs and processes

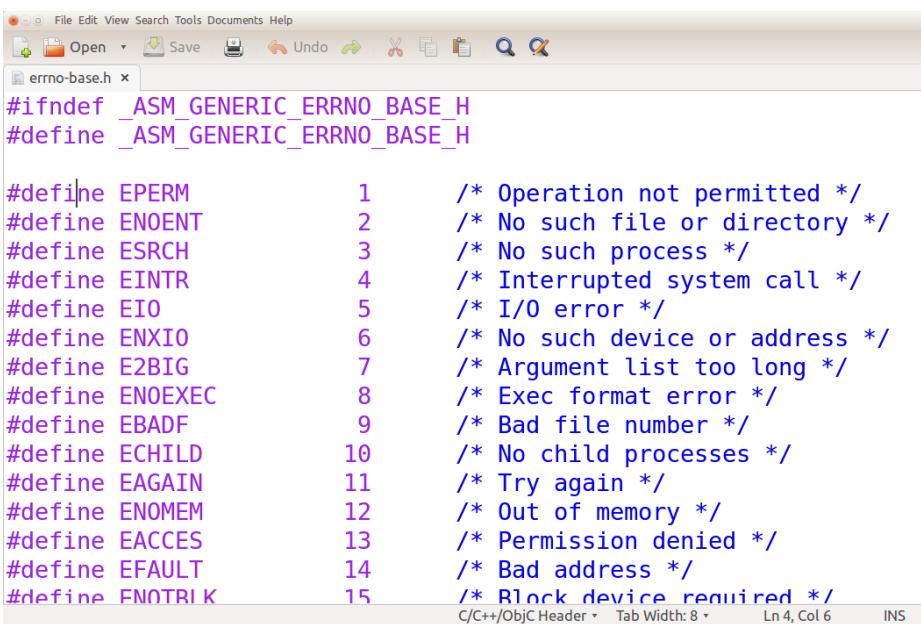
Printer ile ilgili prosesler

```
File Edit View Search Terminal Help
bilg@bilg:~$ ps x|grep printer
 1955 ?      Ssl    0:00 /usr/lib/x86_64-linux-gnu/indicator-printers/indicator-
rvice
12267 pts/1    S+     0:00 grep --color=auto printer
bilg@bilg:~$
```

```
File Edit View Search Terminal Help
bilg@bilg:~$ ps x|grep usr
 1599 ?      Sl    0:00 /usr/bin/gnome-keyring-daemon --daemonize --login
 1696 ?      Ss    0:00 /usr/lib/x86_64-linux-gnu/hud/window-stack-bridge
 1699 ?      Ssl   0:21 /usr/bin/ibus-daemon --daemonize --xim
 1708 ?      Sl    0:00 /usr/lib/gvfs/gvfsd
 1714 ?      Sl    0:00 /usr/lib/gvfs/gvfsd-fuse /run/user/1000/gvfs -f -o big_
writes
 1718 ?      Sl    0:00 /usr/lib/ibus/ibus-dconf
 1719 ?      Sl    0:07 /usr/lib/ibus/ibus-ui-gtk3
 1721 ?      Sl    0:01 /usr/lib/ibus/ibus-x11 --kill-daemon
 1727 ?      Sl    0:00 /usr/lib/at-spi2-core/at-spi-bus-launcher
 1737 ?      Sl    0:01 /usr/lib/at-spi2-core/at-spi2-registryd --use-gnome-ses
```

Hata yönetme - Error Handling

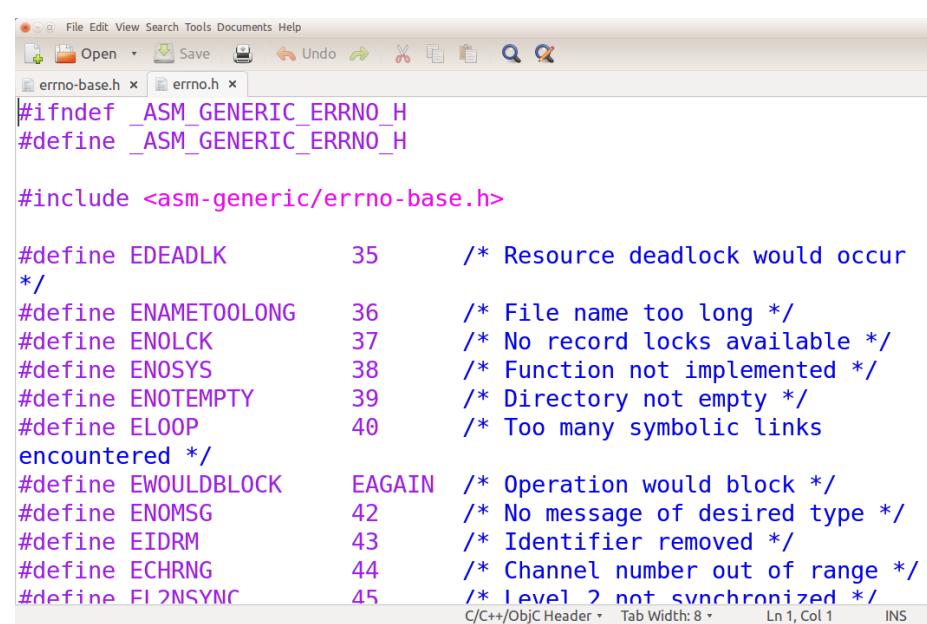
- Linux sistemde hata oluştugu zaman errno isimli bir global tamsayı değişkende hatayı tanımlayan bir numara saklanır.
- errno : sistem değişkeni. Son hatayı tanımlayan numarayı tutar
- /usr/include/asm-generic



```
#ifndef __ASM_GENERIC_ERRNO_BASE_H
#define __ASM_GENERIC_ERRNO_BASE_H

#define EPERM      1      /* Operation not permitted */
#define ENOENT     2      /* No such file or directory */
#define ESRCH      3      /* No such process */
#define EINTR      4      /* Interrupted system call */
#define EIO        5      /* I/O error */
#define ENXIO      6      /* No such device or address */
#define E2BIG       7      /* Argument list too long */
#define ENOEXEC    8      /* Exec format error */
#define EBADF      9      /* Bad file number */
#define ECHILD    10      /* No child processes */
#define EAGAIN     11      /* Try again */
#define ENOMEM     12      /* Out of memory */
#define EACCES     13      /* Permission denied */
#defineEFAULT    14      /* Bad address */
#define ENOTRIK   15      /* Block device required */

C/C++/ObjC Header Tab Width: 8 Ln 4, Col 6 INS
```



```
#ifndef __ASM_GENERIC_ERRNO_H
#define __ASM_GENERIC_ERRNO_H

#include <asm-generic/errno-base.h>

#define EDEADLK    35      /* Resource deadlock would occur */
#define ENAMETOOLONG 36      /* File name too long */
#define ENOLCK     37      /* No record locks available */
#define ENOSYS      38      /* Function not implemented */
#define ENOTEMPTY   39      /* Directory not empty */
#define ELOOP      40      /* Too many symbolic links
encountered */
#define EWOULDBLOCK 41      /* Operation would block */
#define ENOMSG      42      /* No message of desired type */
#define EIDRM      43      /* Identifier removed */
#define ECHRNG     44      /* Channel number out of range */
#define EL2NSYNC    45      /* Level 2 not synchronized */

C/C++/ObjC Header Tab Width: 8 Ln 1, Col 1 INS
```

Hata yönetme- Error Handling

A screenshot of a C IDE showing a code editor and a terminal window. The code editor displays a C program named 'ch1a.c' with the following content:

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

main()
{
    int i;
    FILE *f;

    f = fopen("/home/bilg/yok", "r");
    if (f == NULL) {
        printf("f = null. errno = %d\n", errno);
        perror("/home/bilg/yok");
    }
}
```

The terminal window below shows the output of running the program:

```
bilg:05_Chap1$ ./ch1a
f = null. errno = 2
hata nedeni: No such file or directory
bilg:05_Chap1$
```

Dosya olmadığı için
f=NULL

errno=2: olmayan dosya
hatası

perror: Hata
açıklamasını yazdır

Hata yönetme - Error Handling

```
File Edit View Search Terminal Help  
bilg:05_Chap1$ echo ""> /home/bilg/yok  
bilg:05_Chap1$ chmod 0 /home/bilg/yok  
bilg:05_Chap1$ ./ch1a  
f = null. errno = 13  
hata nedeni: Permission denied  
bilg:05_Chap1$ █
```

yok dosyasını oluştur

Dosya erişimini engelle

#define EACCES

13

/* Permission denied */

chmod - change file mode bits

| # | Permission | |
|---|-------------------------|-----|
| 7 | read, write and execute | rwx |
| 6 | read and write | rw- |
| 5 | read and execute | r-x |
| 4 | read only | r-- |
| 3 | write and execute | -wx |
| 2 | write only | -w- |
| 1 | execute only | --x |
| 0 | none | --- |

Assert

- Assert(koşul), tanımlanan koşul yanlışsa programın akışını durdurur.

The screenshot shows a terminal window with a dark theme. The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. Below the menu is an 'Open' button with a dropdown arrow, a '+' icon for creating new files, and a 'Save' button. The main area contains the following C code:

```
#include <stdio.h>
#include <assert.h>
int main(int argc, char **argv)
{
    assert(argc==2);
    printf("argc=%d\n",argc);

    assert(strcmp("dosya.txt",argv[1])==0);
    printf("argv[1]=%s\n",argv[1]);
    return 0;
}
```

At the bottom, a status bar displays "Saving file '/home/bilg/Documents/h6/assert1.c'" and "C Tab Width: 8 Ln 11, Col 2 INS".

Assert

```
Terminal File Edit View Search Terminal Help  
bilg:h6$ ./assert1 dosya.txt  
argc=2  
argv[1]=dosya.txt  
bilg:h6$
```

Tüm assert fonksiyonları içeriği true olduğu için program doğru bir şekilde çalıştı.

```
Terminal File Edit View Search Terminal Help  
bilg:h6$ ./assert1  
assert1: assert1.c:5: main: Assertion `argc==2' failed.  
Aborted (core dumped)  
bilg:h6$
```

```
Terminal File Edit View Search Terminal Help  
bilg:h6$ ./assert1 dosya.tx  
argc=2  
assert1: assert1.c:8: main: Assertion `strcmp("dosya.txt", argv[1])==0' failed.  
Aborted (core dumped)  
bilg:h6$
```

Assert

The screenshot shows a code editor window with a dark theme. The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. Below the menu is a toolbar with Open, Save, and other icons. The main area contains the following C code:

```
#define NDEBUG /*eklendiği zaman programdaki
tüm assert() fonksiyonları iptal edilir.
#include <assert.h> satırından önce
eklenmelidir/

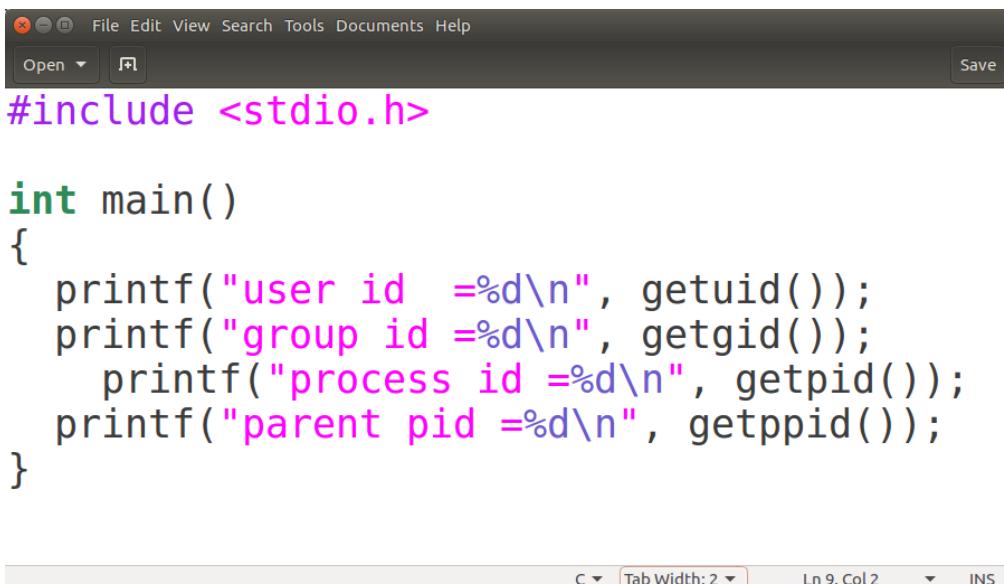
#include <stdio.h>
#include <assert.h>
int main(int argc,char **argv)
{
    assert(argc==2);
    printf("argc=%d\n",argc);

    assert(strcmp("dosya.txt",argv[1])==0);
    printf("argv[1]=%s\n",argv[1]);
    return 0;
}
```

The status bar at the bottom shows "Saving file '/home/bilg/Documents/h6/assert1.c'" and "Ln 2, Col 58".

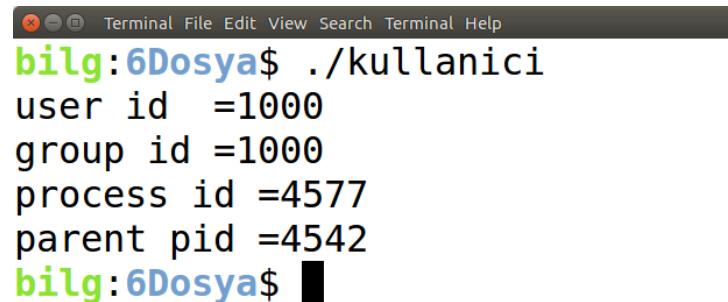
getuid(), getgid(), getpid(), getppid()

- User ID: sistem tarafından her kullanıcıya verilen numara.
- Group ID: kullanıcı gruplarına verilen numara
- Process ID: Bir prosese işletim sistemi tarafından verilen tanımlayıcı sayı.
- Parent Process ID: prosesi oluşturan proses



```
#include <stdio.h>

int main()
{
    printf("user id =%d\n", getuid());
    printf("group id =%d\n", getgid());
    printf("process id =%d\n", getpid());
    printf("parent pid =%d\n", getppid());
}
```



```
bilg:6Dosya$ ./kullanici
user id  =1000
group id =1000
process id =4577
parent pid =4542
bilg:6Dosya$ █
```

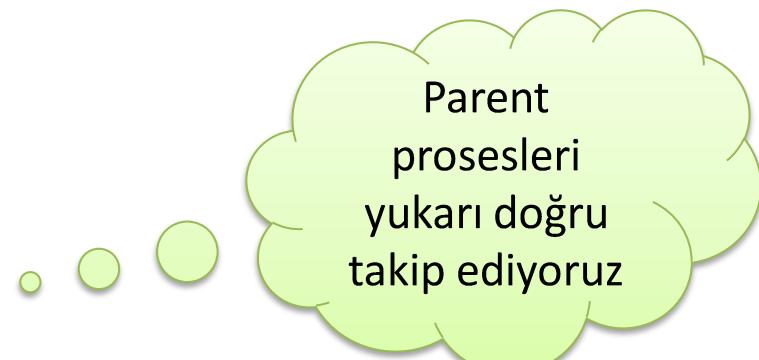
ps -o ppid processid

```
Terminal File Edit View Search Terminal Help
3915 ? Ssl 0:00 C:\windows\system32\service
3919 ? Sl 0:00 C:\windows\system32\winedev
3927 ? Sl 0:00 C:\windows\system32\plugpla
3934 ? Ssl 0:00 C:\windows\system32\explore
3944 ? Sl 0:00 C:\Program Files\Common Fil
4029 ? Ssl 0:00 C:\windows\system32\rpcss.e
4493 ? Sl 0:01 gedit /home/bilg/Documents/
4542 pts/19 Ss 0:00 bash
5777 pts/19 R+ 0:00 ps x
bilg:6Dosya$ ps 4542
  PID TTY      STAT      TIME COMMAND
 4542 pts/19    Ss      0:00 bash
bilg:6Dosya$
```

4512 bash'in
parent prosesi
2972

```
Terminal File Edit View Search Terminal Help
bilg:6Dosya$ ps -o ppid 4542
PPID
2972
bilg:6Dosya$ ps 2972
  PID TTY      STAT      TIME COMMAND
 2972 ?        Sl      0:03 /usr/lib/gnome-terminal/gno
bilg:6Dosya$
```

```
Terminal File Edit View Search Terminal Help
PID TTY      STAT      TIME COMMAND
2972 ?        Sl       0:03 /usr/lib/gnome-terminal
/gno
bilg:6Dosya$ ps -o ppid 2972
PPID
1919
bilg:6Dosya$ ps -o ppid 1919
PPID
1757
bilg:6Dosya$ ps -o ppid 1757
PPID
1290
bilg:6Dosya$ ps -o ppid 1290
PPID
1
bilg:6Dosya$ ps 1
PID TTY      STAT      TIME COMMAND
1 ?        Ss       0:01 /sbin/init splash
bilg:6Dosya$
```



```
Terminal File Edit View Search Terminal Help
SYSTEMD(1)          systemd          SYSTEMD(1)
)
NAME
    systemd, init - systemd system and
    service manager
SYNOPSIS
    systemd [OPTIONS...]
    init [OPTIONS...] {COMMAND}
DESCRIPTION
    systemd is a system and service manager
    for Linux operating systems. When run as
    first process on boot (as PID 1), it act
    s
    nit(1) line 1 (press h for help or q to quit)
```

Sinyaller

- Sinyaller programa gönderilen bir çeşit kesmedir (interrupt).
- Sinyal oluşması durumundan programın nasıl davranışacağı belirlenebilir. Belirlenmezse daha önceden belirlenmiş (default) davranışlar gerçekleşir.
- programa SIGINT sinyali gönderen CRTL+C ile bir programı sonlandırabiliriz. SIGINT için önceden tanımlanmış işlem budur.
- CTRL-\ programa SIGQUIT sinyali gönderir
- Bir bölüm ihlali (segmentation violation) oluştüğü zaman programa SIGSEGV sinyali gönderilir.

Sinyaller (İşaretler) - Signals

Proses sinyalleri temel olarak üç şekilde değerlendirir:

- Sinyali ihmal etmek
- Default bir işlemi gerçekleştirmek. Örneğin sıfıra bölme için default işlem prosesi sonlandırmaktır.
- Sinyali değerlendiren bir fonksiyon yazmak.

Sinyaller listesi

```
File Edit View Search Terminal Help
9_Signals\> kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
21) SIGTTIN     22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO       30) SIGPWR
31) SIGSYS      34) SIGRTMIN    35) SIGRTMIN+1  36) SIGRTMIN+2  37) SIGRTMIN+3
38) SIGRTMIN+4  39) SIGRTMIN+5  40) SIGRTMIN+6  41) SIGRTMIN+7  42) SIGRTMIN+8
43) SIGRTMIN+9  44) SIGRTMIN+10 45) SIGRTMIN+11 46) SIGRTMIN+12 47) SIGRTMIN+13
48) SIGRTMIN+14 49) SIGRTMIN+15 50) SIGRTMAX-14 51) SIGRTMAX-13 52) SIGRTMAX-12
53) SIGRTMAX-11 54) SIGRTMAX-10 55) SIGRTMAX-9  56) SIGRTMAX-8  57) SIGRTMAX-7
58) SIGRTMAX-6  59) SIGRTMAX-5  60) SIGRTMAX-4  61) SIGRTMAX-3  62) SIGRTMAX-2
63) SIGRTMAX-1  64) SIGRTMAX
9_Signals\> █
```

Örnek:



```
*ch1c.c x
#include <stdio.h>
main()
{
    int i;

    i = 0;
    while (1) {
        i++;
        printf("%d\n", i);
        fflush(stdout);
        sleep(1);
    }
}
```

- Yandaki program 1sn aralıklarla ekrana mesaj yazdırır.

Ekrana yazmaya zorlar

Sinyaller (İşaretler) - Signals

```
Terminal File Edit View Search Terminal Help
```

```
bilg:~$ ps x
```

./ch1c programını çalıştırıldıkten sonra başka bir terminal ekranı açıp ps x yazarsak sağdaki gibi ./ch1c programını görürüz. Bu prosese kill ile sinyal gönderebiliriz.

```
Terminal File Edit View Search Terminal Help
```

```
6876 pts/21 Ss 0:00 bash  
6918 pts/22 Ss 0:00 bash  
6954 pts/21 S+ 0:00 ./ch1c  
6974 pts/22 R+ 0:00 ps x
```

```
bilg:~$
```

```
Terminal File Edit View Search Terminal
```

```
bilg:6Dosya$ ./ch1c
```

```
1  
2  
3
```

```
Terminal File Edit View Search Terminal Help
```

```
6918 pts/22 Ss 0:00 bash  
6954 pts/21 S+ 0:00 ./ch1c  
6974 pts/22 R+ 0:00 ps x
```

```
bilg:~$ kill -9 6954
```

```
bilg:~$
```



```
Terminal File Edit View Search Terminal
```

```
49  
50  
51  
52  
53  
54  
55  
56  
57  
Killed
```

```
bilg:6Dosya$
```

Sinyaller (işaretler) - Signals

The diagram illustrates the flow of signals between two terminal windows. A blue arrow points from the first terminal window to the second, indicating the transmission of a signal.

Terminal Window 1 (Left):

```
Terminal File Edit View Search Terminal Help
6918 pts/22 Ss 0:00 bash
7008 pts/21 S+ 0:00 ./ch1c
7044 pts/22 R+ 0:00 ps x
bilg:~$ kill -20 7008
bilg:~$
```

A pink thought bubble contains the text:

20 ile SIGSTOP
gönderdik.
18 ile SIGCONT
gönderdik.

Terminal Window 2 (Right):

```
Terminal File Edit View Search Terminal
73
74
75
[1]+ Stopped
./ch1c
bilg:6Dosya$
```


Terminal Window 3 (Bottom Left):

```
Terminal File Edit View Search Terminal Help
7008 pts/21 S+ 0:00 ./ch1c
7044 pts/22 R+ 0:00 ps x
bilg:~$ kill -20 7008
bilg:~$ kill -18 7008
bilg:~$
```

A blue arrow points from the third terminal window to the right, indicating the transmission of a signal.

Terminal Window 4 (Bottom Right):

```
Terminal File Edit View Search Terminal
77
78
79
[1]+ Stopped
./ch1c
bilg:6Dosya$ 76
```

Sinyaller

- Bir sinyal olduğu zaman yapılacak işlemleri tanımlamak için signal() fonksiyonu kullanılır.

```
void sinyal_fonk (int signum){  
}
```

```
void fonk(){  
...  
signal( sinyal_tipi, sinyal_fonk)  
...  
}
```

Örnek: ctrl+c

- Örnekte ctrl+c tuşlarına basıldığında programı sonlandırma yerine tanımlanmış fonksiyon çalıştırılmaktadır.
- Bu amaçla SIGINT sinyali için bir interrupt handler yazılmıştır.
- Program 1 saniyede bir ekrana 10 kez mesaj yazdırır. Bu sırada ctrl+c tuşlarına basılırsa fonksiyon çalıştırılır.

```
File Edit View Search Terminal Help  
9_Signals> ./sh1  
10  
9  
8  
^Cctrl-c  
7  
6  
5  
4  
3  
^Cctrl-c  
2  
1  
9_Signals> █
```

```
*sh1.c x  
#include <signal.h>  
#include <stdio.h>  
#include <stdlib.h>  
  
void ctrl_c_handler(int dummy)  
{  
    signal(SIGINT, ctrl_c_handler);  
    printf("ctrl-c\n");  
}  
  
main()  
{  
    int i, j;  
  
    signal(SIGINT, ctrl_c_handler);  
  
    for (j=10;j>0;j--) {  
        printf("\t%d \n", j);  
        sleep(1);  
        fflush(stdout);  
    }  
}
```

Örnek: Birden fazla sinyal

```
sh1a.c x
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
int j;
void ctrl_c_handler(int signum)
{
    printf("ctrl-c.  j=%d\n", j);
}

void ctrl_bs_handler(int signum)
{
    printf("ctrl-\\".  j=%d\n", j);
}

main()
{
    signal(SIGINT, ctrl_c_handler);
    signal(SIGQUIT, ctrl_bs_handler);

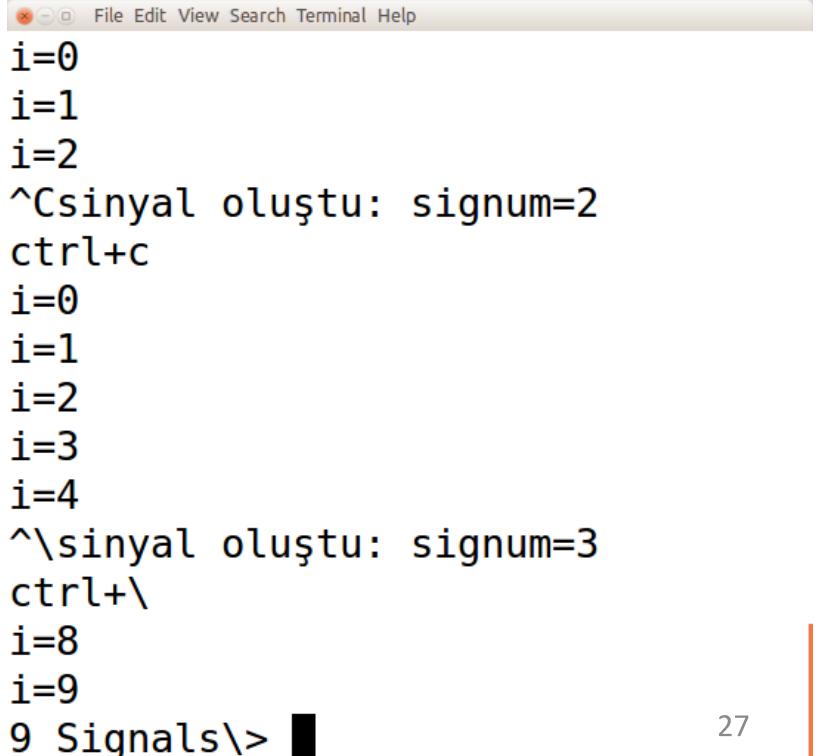
    for (j=0;j<10;j++){
        printf("\t%d \n",j);
        sleep(1);
    }
}
```

C Tab Width: 8 Ln 26, Col 1 INS

- Örnekte SIGINT yanında SIGQUIT sinyali içinde bir bir interrupt handler yazılmıştır.
- Ctrl+\ ile SIGQUIT üretiliyor.

```
File Edit View Search Terminal Help
9_Signals\> ./sh1a
          0
          1
          2
^Cctrl-c.  j=2
          3
          4
          5
^\"ctrl-\\". j=5
          6
          7
          8
          9
9_Signals\> █
```

Örnek: Birden fazla sinyal



```
#include<stdio.h>
#include<signal.h>
int i;
void fonk(int signum)
{
    printf("sinyal oluştu: signum=%d\n",signum);
    if (signum==SIGINT)
    {
        printf("ctrl+c\n");
        i=0;
    }
    else if (signum==SIGQUIT)
    {
        i=8;
        printf("ctrl+\n");
    }
}
main()
{
    signal(SIGINT,fonk);
    signal(SIGQUIT,fonk);
    for(i=0;i<10;i++)
    {
        sleep(1);
        printf("i=%d \n",i);
    }
}
```

Sinyaller için tanımlanmış işlemler aynı fonksiyon içerisinde gerçekleştirilebilir. Örnekte sinyal numarasına bakılarak main() içindeki döngü değişkeni *i* değiştirilmiştir.

```
i=0
i=1
i=2
^Csinyal oluştu: signum=2
ctrl+c
i=0
i=1
i=2
i=3
i=4
^\\sinyal oluştu: signum=3
ctrl+\
i=8
i=9
9_Signals\>
```

Sinyalleri ihmali etmek

The screenshot shows a C code editor window with the file `*testsinal1.c`. The code demonstrates how to ignore specific signals (SIGINT and SIG_DFL) using the `signal` function.

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

main()
{
    int i;

    signal(SIGINT, SIG_IGN);
    for(i=0;i<5;i++)
    {
        printf("1. döngü: i=%d\n", i);
        sleep(1);
    }

    signal(SIGINT, SIG_DFL);
    for(i=0;i<5;i++)
    {
        printf("2. döngü: i=%d\n", i);
        sleep(1);
    }
}
```

Annotations in the code:

- A green thought bubble above the first `for` loop states: "Fonksiyon ismi yerine SIG_IGN yazarsak sinyali ihmali etmiş oluruz."
- A green thought bubble below the second `for` loop states: "SIG_DFL ile default tanımlanmış işleme geri dönebiliriz."

Bottom status bar: C Tab Width: 8 Ln 20, Col 1 INS

The screenshot shows a terminal window running the command `./testsinal1`.

```
9_Signals> ./testsinal1
1. döngü: i=0
1. döngü: i=1
1. döngü: i=2
^C1. döngü: i=3
1. döngü: i=4
2. döngü: i=0
2. döngü: i=1
^C
9_Signals>
```

Annotations in the terminal output:

- A red thought bubble next to the first three lines of output states: "Birinci döngü çalışırken i=3 iterasyonunda ctrl+c ile gönderilen sinyal ihmali ediliyor."
- A red thought bubble next to the last two lines of output states: "İkinci döngüde ise i=1 iterasyonunda ctrl+c ile default tanımlanmış programı sonlandırma işlemi gerçekleştiriyor."

Sinyal içerisinde sinyal

```
#include <stdio.h>
#include <signal.h>
void sigint_handler(int signum)
{
    int i;
    for(i=0;i<8;i++)
    {
        printf("sigint-ctrl+c: i=%d \n",i);
        sleep(1);
    }
}
void sigquit_handler(int signum)
{
    int i;
    for(i=0;i<4;i++)
    {
        printf("sigquit-ctrl+\\" : i=%d \n",i);
        sleep(1);
    }
}
main()
{
int i;
signal(SIGINT,sigint_handler);
signal(SIGQUIT,sigquit_handler);
for(i=0;i<10;i++)
{
    printf("main: i=%d \n",i);
    sleep(1);
}
```

- Yandaki programda ctrl+c ile oluşturulan SIGINT ve ctrl+\ ile oluşturulan SIGQUIT sinyalleri tanımlanmıştır.
- Sinyallerden biri uygulandığı zaman main fonksiyonu çalışmayı durdurarak sinyale ait fonksiyonu çalıştırır.
- Bu esnada ikinci bir sinyal uygulanırsa ilgili fonksiyon durur ve uygulanan sinyale ait fonksiyon çalıştırılır.

Sinyal içerisinde sinyal

File Edit View Search Terminal Help

9_Signals\> ./testsinyal4

```
main: i=0
main: i=1
main: i=2
main: i=3
main: i=4
^Csigint-ctrl+c: i=0
sigint-ctrl+c: i=1
sigint-ctrl+c: i=2
sigint-ctrl+c: i=3
sigint-ctrl+c: i=4
^\\sigquit-ctrl+\\: i=0
sigquit-ctrl+\\: i=1
sigquit-ctrl+\\: i=2
sigquit-ctrl+\\: i=3
sigint-ctrl+c: i=5
sigint-ctrl+c: i=6
sigint-ctrl+c: i=7
main: i=5
main: i=6
main: i=7
main: i=8
main: i=9
```

Ctrl+c ile SIGINT oluşturuldu. main() çalışmasını durdurdu. sigint_handler() fonksiyonu çalışıyor.

Ctrl+\ ile SIGQUIT oluşturuldu. Şimdi sigquit_handler() çalışıyor.

sigquit_handler() sonlandı. sigint_handler() kaldığı yerden devam ediyor.

sigint_handler() sonlandı. main() kaldığı yerden devam ediyor.

Bir sinyal işlenirken başka bir sinyali ihmal etmek

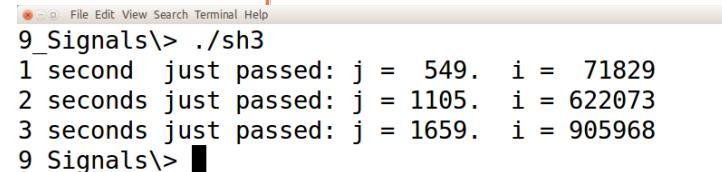
The screenshot shows a code editor window with the file 'testsinal4.c' open. The code demonstrates signal handling in C. It includes two signal handlers: 'sigint_handler' and 'sigquit_handler'. The 'sigint_handler' prints 'sigint-ctrl+c: i=%d \n' for each integer from 0 to 7, with a one-second delay between each print. The 'sigquit_handler' prints 'sigquit-ctrl+\\" i=%d \n' for each integer from 0 to 3, with a one-second delay between each print. It also enables the 'sigint_handler' for the SIGQUIT signal. Both sections of code are highlighted with red boxes.

```
#include <stdio.h>
#include <signal.h>
void sigint_handler(int signum)
{
    int i;
    for(i=0;i<8;i++)
    {
        printf("sigint-ctrl+c: i=%d \n",i);
        sleep(1);
    }
}
void sigquit_handler(int signum)
{
    int i;
    signal(SIGINT,SIG_IGN); // sinyali ihmal et
    for(i=0;i<4;i++)
    {
        printf("sigquit-ctrl+\\" i=%d \n",i);
        sleep(1);
    }
    signal(SIGINT,sigint_handler); //sinyali etkinleştir
}
main()
{
int i;
signal(SIGINT,sigint_handler);
signal(SIGQUIT,sigquit_handler);
```

Yandaki örnekte SIGQUIT sinyali için işlem yapılırken SIGINT sinyali ihmal edilmiştir. Fonksiyondan çıkışmadan önce SIGINT tekrar aktif duruma getirilmiştir.

Zamanlanmış sinyal: alarm() ve SIGALRM

- **alarm(int saniye)** fonksiyonu ile saniye cinsinden belirtilen zaman sonra SIGALRM sinyali oluşturulur.
- Aşağıdaki örnekte main() içerisindeki alarm() fonksiyonunun SIGALRM işaretini üretmesiyle birlikte signal içerisinde tanımlanmış alarm_handler() çağrıılır. Burada ana programdaki i ve j değişkenlerinin sinyal geldiği andaki değerleri yazdırılır.
- Ayrıca alarm() tekrar tanımlandığı için main() içerisindeki döngü devam ettiği sürece 1 saniye aralıklarla alarm sinyali üretilir. Eğer alarm_handler() içindeki alarm(1) kaldırılırsa sadece bir kez alarm sinyali üretilir.



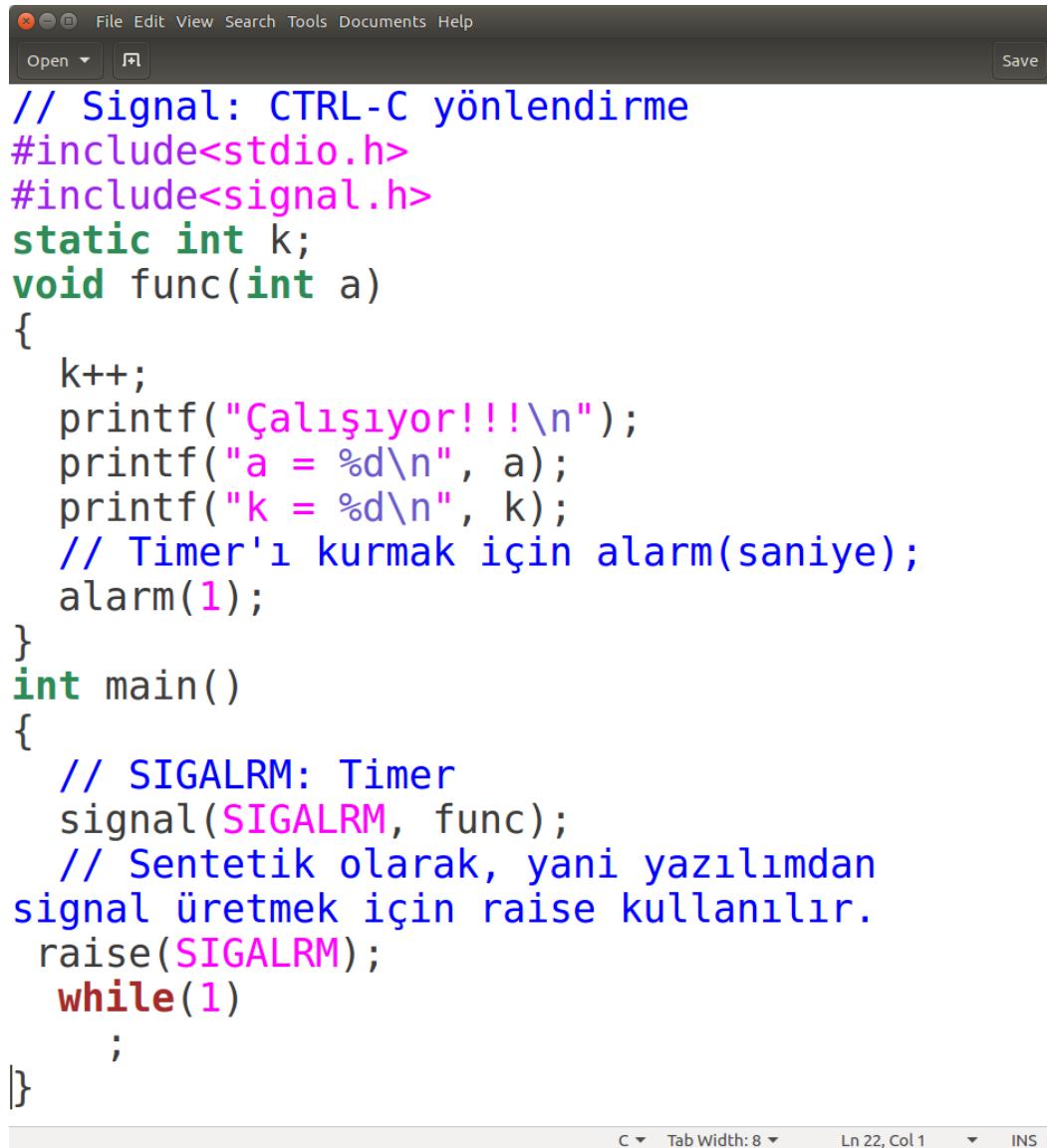
```
File Edit View Search Tools Documents Help
Open Save Undo Redo Cut Copy Paste Find Replace
sh3.c x
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
int i, j, seconds;
void alarm_handler(int dummy)
{
    seconds++;
    printf("%d second%s just passed: j = %4d.  i = %6d\n", seconds,
           (seconds == 1) ? " " : "s", j, i);
    alarm(1);
}

main()
{
    seconds = 0;
    signal(SIGALRM, alarm_handler);
    alarm(1);
    for (j = 0; j < 2000; j++) {
        for (i = 0; i < 1000000; i++);
    }
}
```

C Tab Width: 8 Ln 22, Col 1 INS

```
File Edit View Search Terminal Help
9_Signals\> ./sh3
1 second just passed: j =  549.  i =  71829
2 seconds just passed: j = 1105.  i = 622073
3 seconds just passed: j = 1659.  i = 905968
9_Signals\> █
```

Alarm



```
// Signal: CTRL-C yönlendirme
#include<stdio.h>
#include<signal.h>
static int k;
void func(int a)
{
    k++;
    printf("Çalışıyor!!!\n");
    printf("a = %d\n", a);
    printf("k = %d\n", k);
    // Timer'ı kurmak için alarm(saniye);
    alarm(1);
}
int main()
{
    // SIGALRM: Timer
    signal(SIGALRM, func);
    // Sentetik olarak, yani yazılımdan
    // signal üretmek için raise kullanılır.
    raise(SIGALRM);
    while(1)
        ;
}
```

The screenshot shows a terminal window with the following text:

```
C ▾ Tab Width: 8 ▾ Ln 22, Col 1 ▾ INS
```

```
// Signal: Multiple signals

#include<stdio.h>
#include<signal.h>

void func(int a)
{
    if(a == SIGINT) // veya if(a == 2)
        printf("CTR-C'ye basıldı\n");
    else if(a == SIGALRM){ // veya if(a == 14)
        printf("Timer signal\n");
        alarm(1);
    }
}

int main()
{
    signal(SIGINT, func);
    signal(SIGALRM, func);
    raise(SIGALRM);
    while(1)
    ;
}
```

The code editor window shows two tabs: "p2.c" (active) and "*p3.c". The "p2.c" tab contains C code demonstrating signal handling. It includes headers for stdio.h and signal.h, defines a function func that prints different messages based on the signal type (SIGINT or SIGALRM), and sets up signal handlers for both types in the main function. The code uses printf for output and alarm for timer functionality.