

Sistem Programlama

DR. ÖĞR. ÜYESİ ABDULLAH SEVİN

İçerik

a) Some Basic Terminology

b) Introduction to System Calls and I/O

- <http://web.eecs.utk.edu/~jplank/plank/classes/cs360/360/notes/Chap1/lecture.html>
- <http://web.eecs.utk.edu/~jplank/plank/classes/cs360/360/notes/Syscall-Intro/lecture.html>

Dosya Sistemi

- ❑ "Dosya sistemi"nin tanımı: Dizinlerin hiyerarşik bir düzenlemesi.
- ❑ Unix'te kök (root) dosya sistemi "/" ile başlar. Ancak, kökün parçası olan başka alt dosya sistemleri de vardır. Makinenizdeki dosya sistemlerini görmek için "df" yazın. Şunun gibi bir şey göreceksiniz:

```
abduallah@abduallah-VirtualBox: ~  
Dosya Düzenle Görünüm Ara Uçbirim Yardım  
abduallah@abduallah-VirtualBox:~$ df  
Dosyasistemi 1K-blok Dolu Boş Kull% Bağlanılan yer  
udev 1986140 0 1986140 0% /dev  
tmpfs 403916 1384 402532 1% /run  
/dev/sda1 20464208 16626588 2772764 86% /  
tmpfs 2019560 0 2019560 0% /dev/shm  
tmpfs 5120 4 5116 1% /run/lock  
tmpfs 2019560 0 2019560 0% /sys/fs/cgroup  
/dev/loop0 512 512 0 100% /snap/gnome-characters/781  
/dev/loop1 144128 144128 0 100% /snap/gnome-3-26-1604/104  
/dev/loop3 48000 48000 0 100% /snap/git-repo/18  
/dev/loop4 640 640 0 100% /snap/gnome-logs/106  
/dev/loop6 66816 66816 0 100% /snap/gtk-common-themes/1519  
/dev/loop8 471424 471424 0 100% /snap/gnome-42-2204/65  
/dev/loop9 1536 1536 0 100% /snap/gnome-system-monitor/181  
/dev/loop7 74752 74752 0 100% /snap/core22/583  
/dev/loop10 56960 56960 0 100% /snap/core18/2714  
/dev/loop11 119680 119680 0 100% /snap/core/14946  
/dev/loop5 223744 223744 0 100% /snap/gnome-3-34-1804/90  
/dev/loop2 168832 168832 0 100% /snap/gnome-3-28-1804/161  
/dev/loop12 768 768 0 100% /snap/gnome-characters/741  
/dev/loop15 64896 64896 0 100% /snap/core20/1828  
/dev/loop16 2176 2176 0 100% /snap/gnome-calculator/926  
/dev/loop21 56960 56960 0 100% /snap/core18/2721
```

Dosya Sistemi

- ❑ Her satır farklı bir dosya sistemini gösterir.
- ❑ Satırdaki ilk giriş, dosya sisteminin nerede olduğunu gösterir ve son giriş, makinenizde ona nasıl eriştiğinizi gösterir.
- ❑ Örneğin, /dev/sda1 dosya sistemi, makinedeki disklerden birinin bölümüdür. / (root dizini) dizininden alabilirim.
- ❑ Dosya sistemlerinin çalışma şekli yıldan yıla değişir, ancak genellikle df'nin çıktısını ayrıştırabilir ve neler olduğunu anlayabilirsiniz.

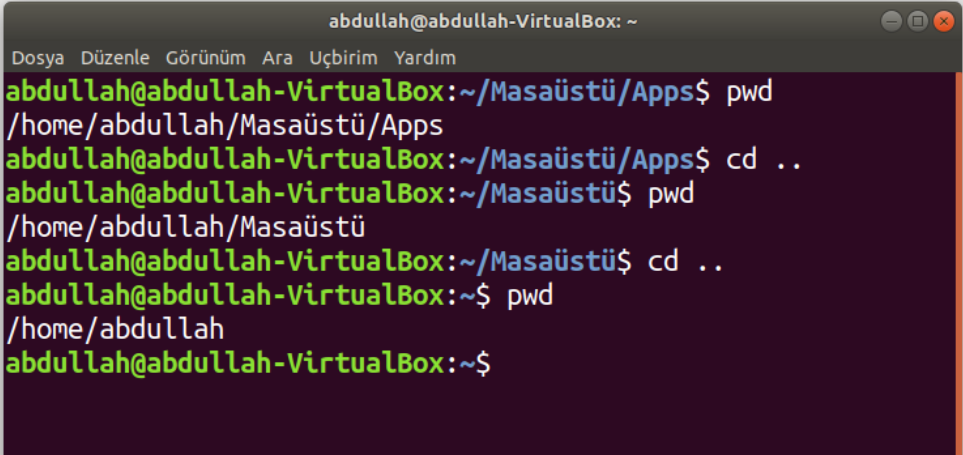
Adlar, Yollar, Dizinler

- ❑ Dosya adı (Filename): Bir dosyanın bir dizinde görüldüğü şekliyle adı.
- ❑ Yol adı (Pathname): Eğik çizgilerle ayrılmış sıfır veya daha fazla dosya adı dizisi.
- ❑ "ls -a" dediğinizde, geçerli dizindeki tüm dosya adlarını listeler:
- ❑ "." geçerli dizindir. ".." geçerli dizinin üst ögesidir.

```
abduallah@abduallah-VirtualBox: ~  
Dosya Düzenle Görünüm Ara Uçbirim Yardım  
abduallah@abduallah-VirtualBox:~$ ls  
Belgeler                               İndirilenler                        Şablonlar  
build-untitled1-Desktop-Debug          Masaüstü                          tfhe  
cs360-lecture-notes                   Müzik                             tfhe-1.0.1.zip  
examples.desktop                      netbeans-8.2                      tfhe_denemeler  
FELICS                                NetBeansProjects                 untitled1  
Genel                                 release-1.8.0.tar.gz             untitled2  
glassfish-4.1.1                       Resimler                         Videolar  
googletest-release-1.8.0              sist_prog  
abduallah@abduallah-VirtualBox:~$ clear  
  
abduallah@abduallah-VirtualBox:~$ ls  
Belgeler                               İndirilenler                        Şablonlar  
build-untitled1-Desktop-Debug          Masaüstü                          tfhe  
cs360-lecture-notes                   Müzik                             tfhe-1.0.1.zip  
examples.desktop                      netbeans-8.2                      tfhe_denemeler  
FELICS                                NetBeansProjects                 untitled1  
Genel                                 release-1.8.0.tar.gz             untitled2  
glassfish-4.1.1                       Resimler                         Videolar  
googletest-release-1.8.0              sist_prog  
abduallah@abduallah-VirtualBox:~$ ls -a  
.  
..  
.bash_history                         Müzik  
.bash_logout                         .nbi  
.bashrc                              .netbeans  
Belgeler                             netbeans-8.2  
build-untitled1-Desktop-Debug        NetBeansProjects  
.cache                               .oracle_jre_usage  
.config                             .profile  
cs360-lecture-notes                  release-1.8.0.tar.gz  
examples.desktop                    Resimler  
FELICS                              sist_prog  
Genel                               .ssh  
glassfish-4.1.1                     .sudo_as_admin_successful  
                                     Şablonlar  
                                     tfhe
```

Adlar, Yollar, Dizinler

- ❑ pwd komutu size geçerli dizinin tam yol adını söyler.
- ❑ cd komutu sizi dizinler arasında hareket ettirir:



```
abduallah@abduallah-VirtualBox: ~
Dosya  Düzenle  Görünüm  Ara  Uçbirim  Yardım
abduallah@abduallah-VirtualBox:~/Masaüstü/Apps$ pwd
/home/abduallah/Masaüstü/Apps
abduallah@abduallah-VirtualBox:~/Masaüstü/Apps$ cd ..
abduallah@abduallah-VirtualBox:~/Masaüstü$ pwd
/home/abduallah/Masaüstü
abduallah@abduallah-VirtualBox:~/Masaüstü$ cd ..
abduallah@abduallah-VirtualBox:~$ pwd
/home/abduallah
abduallah@abduallah-VirtualBox:~$
```

Adlar, Yollar, Dizinler

- ❑ Mutlak Yol Adı: Eğik çizgi ile başlayan bir yol adı.
- ❑ Göreceli Yol Adı: Eğik çizgi ile başlamayan bir yol adı.
- ❑ Çalışma Dizini: Göreli yol adlarının göreli olduğu dizin.

Çalışma dizininizi pwd ile görebilirsiniz.

- ❑ Ev Dizini: Bir kullanıcının ilk oturum açtığında çalışma dizini.

```
UNIX> pwd
/home/plank
UNIX> cd cs360/notes
UNIX> pwd
/home/plank/cs360/notes
UNIX> ls Chap1
bin lecture.html makefile src
UNIX> echo $HOME
/home/plank
UNIX> cd ~bvz
UNIX> pwd
/home/bvz
UNIX> cd ~
UNIX> pwd
/home/plank
UNIX>
```

Program ve prosesler

- ❑ Program: Doğrudan veya yorumlayıcılar, derleyiciler ve/veya bağlayıcılar yardımıyla yürütülebilen bir dosya.
- ❑ Proses: Bir programın yürütülmekte olan örneği.
- ❑ Proses ID-Kimliği: İşletim sistemi tarafından bir işleme verilen numara.
- ❑ `/bin/ls --` bu doğrudan çalıştırılabilen bir programdır.
- ❑ `/usr/bin/vim --` bu doğrudan çalıştırılabilen bir programdır.
- ❑ `/home/plank/cs360/notes/Chap1/src/ch1a.c --` bu, yürütmek için derlenmesi gereken bir programdır.
- ❑ `/home/plank/bin/calc --` bu bir kabuk komut dosyasıdır -- `/bin/sh` tarafından yorumlanması gereken bir programdır.

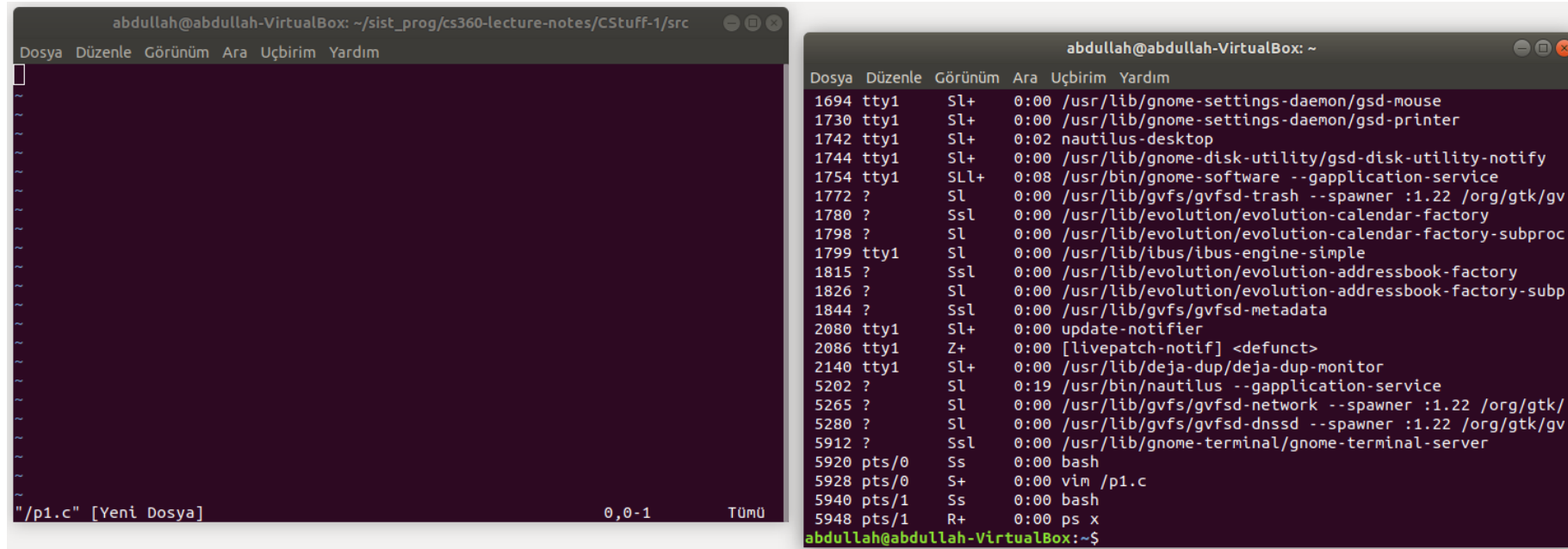
prosesler

❑ "ps x", şu anda yürütmekte olduğunuz tüm işlemleri listeleyecektir:

```
abduallah@abduallah-VirtualBox: ~  
Dosya Düzenle Görünüm Ara Uçbirim Yardım  
abduallah@abduallah-VirtualBox:~$ ps x  
PID TTY      STAT   TIME COMMAND  
883 ?        Ss      0:00 /lib/systemd/systemd --user  
894 ?        S        0:00 (sd-pam)  
932 ?        SLL     0:00 /usr/bin/gnome-keyring-daemon --daemonize --login  
960 tty1     Ssl+    0:00 /usr/lib/gdm3/gdm-x-session --run-script env GNOME_SH  
968 tty1     Sl+     0:17 /usr/lib/xorg/Xorg vt1 -displayfd 3 -auth /run/user/1  
1021 ?        Ss      0:00 /usr/bin/dbus-daemon --session --address=systemd: --n  
1024 tty1     Sl+     0:00 /usr/lib/gnome-session/gnome-session-binary --session  
1163 ?        Ss      0:00 /usr/bin/ssh-agent /usr/bin/im-launch env GNOME_SHELL  
1181 ?        Ssl     0:00 /usr/lib/at-spi2-core/at-spi-bus-launcher  
1191 ?        S        0:00 /usr/bin/dbus-daemon --config-file=/usr/share/default  
1200 ?        Sl      0:00 /usr/lib/at-spi2-core/at-spi2-registryd --use-gnome-s  
1339 tty1     Sl+     0:45 /usr/bin/gnome-shell  
1523 ?        Ssl     0:00 /usr/lib/gvfs/gvfsd  
1528 ?        Sl      0:00 /usr/lib/gvfs/gvfsd-fuse /run/user/1000/gvfs -f -o bi  
1539 ?        S<l     0:00 /usr/bin/pulseaudio --start --log-target=syslog  
1555 tty1     Sl      0:00 ibus-daemon --xim --panel disable  
1559 tty1     Sl      0:00 /usr/lib/ibus/ibus-dconf  
1561 tty1     Sl      0:00 /usr/lib/ibus/ibus-x11 --kill-daemon  
1565 ?        Sl      0:00 /usr/lib/ibus/ibus-portal  
1574 ?        Sl      0:00 /usr/lib/gnome-shell/gnome-shell-calendar-server  
1578 ?        Ssl     0:00 /usr/lib/evolution/evolution-source-registry  
1586 ?        Sl      0:00 /usr/lib/gnome-online-accounts/goa-daemon
```

prosesler

- ❑ Vim Editörü Terminalden kullanılabilen ve komut alabilen yaygın kullanıma sahip bir editördür.
- ❑ Aynı anda birden fazla vim işlemi çalıştırabileceğimizi unutmayın. Başka bir pencereye gidin ve "vim \$pwd/p1.c" yazın. Şimdi "ps x" yazdığınızda ikinci işlemi göreceksiniz.



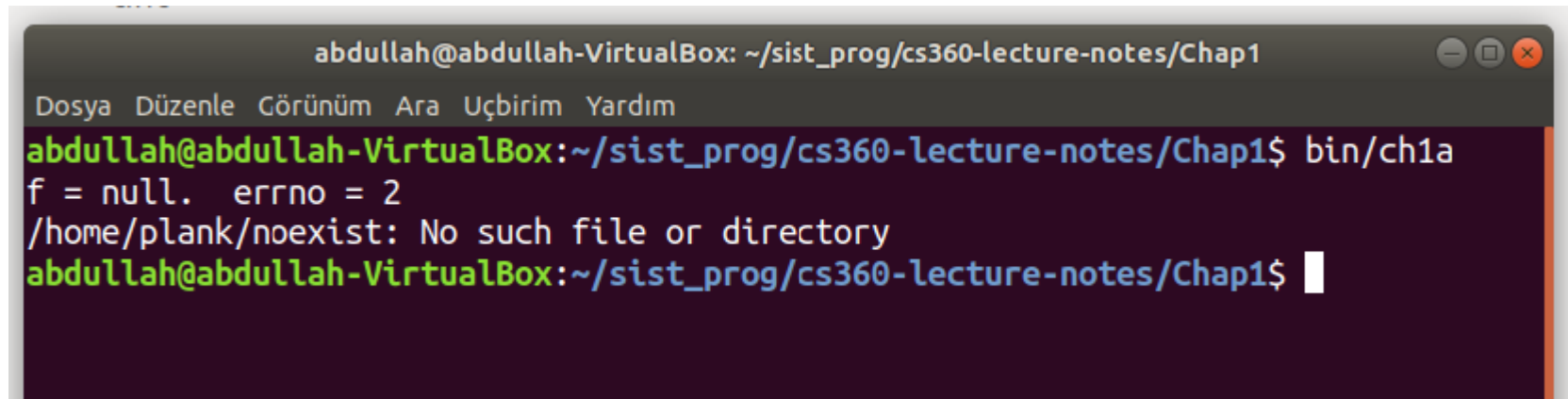
The image shows two terminal windows from a VirtualBox environment. The left window is a Vim editor editing a file named p1.c. The right window shows the output of the 'ps x' command, listing running processes. The process list includes various system services and user processes, with the last line showing the current user's bash shell.

```
abduallah@abduallah-VirtualBox: ~/sist_prog/cs360-lecture-notes/CStuff-1/src
Dosya Düzenle Görünüm Ara Uçbirim Yardım
~/p1.c" [Yeni Dosya] 0,0-1 Tümü

abduallah@abduallah-VirtualBox: ~
Dosya Düzenle Görünüm Ara Uçbirim Yardım
1694 tty1 SL+ 0:00 /usr/lib/gnome-settings-daemon/gsd-mouse
1730 tty1 SL+ 0:00 /usr/lib/gnome-settings-daemon/gsd-printer
1742 tty1 SL+ 0:02 nautilus-desktop
1744 tty1 SL+ 0:00 /usr/lib/gnome-disk-utility/gsd-disk-utility-notify
1754 tty1 SLL+ 0:08 /usr/bin/gnome-software --gapplication-service
1772 ? SL 0:00 /usr/lib/gvfs/gvfsd-trash --spawnner :1.22 /org/gtk/gv
1780 ? Ssl 0:00 /usr/lib/evolution/evolution-calendar-factory
1798 ? SL 0:00 /usr/lib/evolution/evolution-calendar-factory-subproc
1799 tty1 SL 0:00 /usr/lib/ibus/ibus-engine-simple
1815 ? Ssl 0:00 /usr/lib/evolution/evolution-addressbook-factory
1826 ? SL 0:00 /usr/lib/evolution/evolution-addressbook-factory-subp
1844 ? Ssl 0:00 /usr/lib/gvfs/gvfsd-metadata
2080 tty1 SL+ 0:00 update-notifier
2086 tty1 Z+ 0:00 [livepatch-notif] <defunct>
2140 tty1 SL+ 0:00 /usr/lib/deja-dup/deja-dup-monitor
5202 ? SL 0:19 /usr/bin/nautilus --gapplication-service
5265 ? SL 0:00 /usr/lib/gvfs/gvfsd-network --spawnner :1.22 /org/gtk/
5280 ? SL 0:00 /usr/lib/gvfs/gvfsd-dnssd --spawnner :1.22 /org/gtk/gv
5912 ? Ssl 0:00 /usr/lib/gnome-terminal/gnome-terminal-server
5920 pts/0 Ss 0:00 bash
5928 pts/0 S+ 0:00 vim /p1.c
5940 pts/1 Ss 0:00 bash
5948 pts/1 R+ 0:00 ps x
abduallah@abduallah-VirtualBox:~$
```

Hata Yönetimi

- ❑ Genellikle bir Unix sisteminde veya kitaplık çağrısında bir hata oluştuğunda, özel bir dönüş değeri geri gelir ve hatanın ne olduğunu söylemek için genel bir "errno" değişkeni ayarlanır.
- ❑ Örneğin, var olmayan bir dosyayı açmaya çalıştığınızı varsayalım:

A screenshot of a terminal window titled 'abdullah@abdullah-VirtualBox: ~/sist_prog/cs360-lecture-notes/Chap1'. The terminal shows a command 'bin/ch1a' being executed, which results in an error: 'f = null. errno = 2' and '/home/plank/noexist: No such file or directory'. The prompt returns to 'abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Chap1\$'.

```
abdullah@abdullah-VirtualBox: ~/sist_prog/cs360-lecture-notes/Chap1
Dosya Düzenle Görünüm Ara Uçbirim Yardım
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Chap1$ bin/ch1a
f = null.  errno = 2
/home/plank/noexist: No such file or directory
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Chap1$
```

Hata yönetimi

- ❑ Diyelim ki /home/plank/noexist oluşturduğum ve okumak için açamayacağım şekilde üzerinde chmod yaptım.
- ❑ Ardından bin/ch1a farklı bir hata yazdıracaktır:

```
UNIX> echo "" > /home/plank/noexist
UNIX> chmod 0 /home/plank/noexist
UNIX> bin/ch1a
f = null.  errno = 13
/home/plank/noexist: Permission denied
UNIX> rm -f /home/plank/noexist
UNIX> bin/ch1a
f = null.  errno = 2
/home/plank/noexist: No such file or directory
UNIX>
```

Kullanıcı Kimliği

- ❑ Kullanıcı Kimliği (User ID): Sistem yöneticisi tarafından her kullanıcıya verilen numara.

```
/* This program prints out your user id.  
  
#include <stdio.h>  
#include <unistd.h>  
  
int main()  
{  
    printf("%d\n", getuid());  
    return 0;  
}
```

```
UNIX> bin/ch1b  
503  
UNIX>
```

Sinyaller

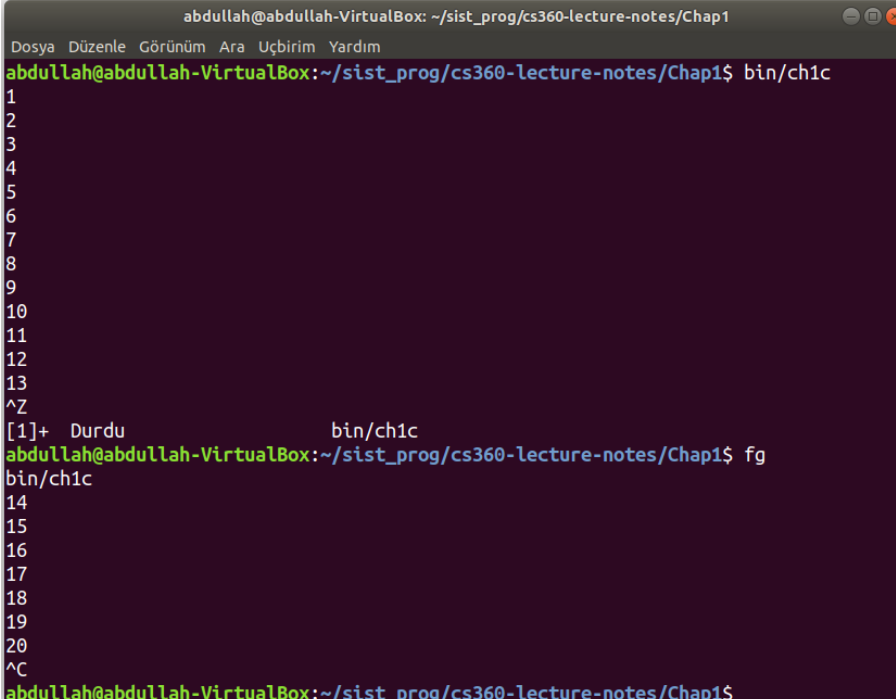
- ❑ Sinyal: Programdaki bir kesme
- ❑ Sinyal İşleyici (Signal Handler): Programın sinyallerle başa çıkabileceği mekanizma.
- ❑ Bu program her saniye artan bir sayaç yazdırır. SIGSTOP sinyalini programa gönderen <CNTL-Z> yazarak geçici olarak durdurursunuz.
- ❑ fg yazarak tekrar çalıştırabilirsiniz. Ardından, SIGINT sinyalini gönderen <CNTL-C> ile programı sonlandırabilirsiniz.

```
/* This program prints out a c
   You stop it temporarily by
   the program. You can run i
   program with <CNTL-C>, whic

#include <stdio.h>
#include <unistd.h>

int main()
{
    int i;

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



```
abduallah@abduallah-VirtualBox: ~/sist_prog/cs360-lecture-notes/Chap1
Dosya Düzenle Görünüm Ara Uçbirim Yardım
abduallah@abduallah-VirtualBox:~/sist_prog/cs360-lecture-notes/Chap1$ bin/ch1c
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
^Z
[1]+  Durdu                               bin/ch1c
abduallah@abduallah-VirtualBox:~/sist_prog/cs360-lecture-notes/Chap1$ fg
bin/ch1c
14
15
16
17
18
19
20
^C
abduallah@abduallah-VirtualBox:~/sist_prog/cs360-lecture-notes/Chap1$
```


Sinyaller

- ❑ Bu program, kendisini her saniye artıran bir sayaç uygular.
- ❑ Birkaç saniye çalışmasına izin verin ve ardından < CNTL-Z > yazın. Bu, onu durduran programa "DUR" sinyali gönderir. Şimdi kabuğuna geri döneceksin. "ps" yazarsanız, şöyle bir şey görürsünüz:
- ❑ 2483 p5 T 0:00 ch1c
- ❑ "T", işlemin çalışmadığı anlamına gelir - durdurulmuştur.
- ❑ Başlatmak için, "BAŞLAT" sinyalini gönderecek olan "fg" yazabilirsiniz.
- ❑ Şimdi, çalışırken, programı sonlandırmak için < CNTL-C > yazın -- bu ona "INT" sinyalini gönderir ve bu onu garanti altına alır. Segmentasyon hataları da sinyallerdir.

Sistem Çağrıları

- ❑ Bir bilgisayar açıldığında, ilk çalıştırılan programa "işletim sistemi" denir.
- ❑ Bilgisayardaki hemen hemen tüm faaliyetleri kontrol eder.
- ❑ Bu, kimin oturum açtığını, disklerin nasıl kullanıldığını, belleğin nasıl kullanıldığını, CPU'nun nasıl kullanıldığını ve diğer bilgisayarlarla nasıl konuştuğunuzu içerir.
- ❑ Programların işletim sistemiyle konuşma şekli `` sistem çağrıları " yoluyla.
- ❑ Bir sistem çağrısı bir prosedür çağrısı gibi görünür, ancak farklıdır -- işletim sisteminden bazı faaliyetler gerçekleştirmesi için bir **taleptir**.

Sistem çağrıları

- ❑ Sistem çağrıları pahalıdır. Bir fonksiyon çağrısı genellikle birkaç makine talimatıyla gerçekleştirilebilirken, bir sistem çağrısı bilgisayarın durumunu kaydetmesini, işletim sisteminin CPU'nun kontrolünü ele geçirmesini, işletim sisteminin bazı işlevleri gerçekleştirmesini, işletim sisteminin durumunu kaydetmesini gerektirir.
- ❑ ve ardından işletim sisteminin CPU'nun kontrolünü size geri vermesini sağlar.(User mode, Kernel mode)

Sistem çağrıları

□ Unix'in dosya G/Ç için sağladığı 5 temel sistem çağrısı vardır;

1. `int open(const char *path, int flags [, int mode]);`
2. `int close(int fd);`
3. `ssize_t read(int fd, void *buf, size_t count);`
4. `ssize_t write(int fd, const void *buf, size_t count);`
5. `off_t lseek(int fd, off_t offset, int whence);`

Sistem çağrıları

- ❑ Normal prosedür çağrıları gibi göründüklerini fark edeceksiniz.
- ❑ Onlarla bu şekilde programlama yaparsınız -- normal prosedür çağrıları gibi.
- ❑ Ancak bunların farklı olduğunu bilmelisiniz:
- ❑ **Bir sistem çağrısı**, işletim sistemine bir istekte bulunur.
- ❑ **Bir prosedür çağrısı**, programınızın başka bir yerinde tanımlanan bir prosedüre atlar.
- ❑ Bu prosedür çağrısının kendisi bir sistem çağrısı yapabilir (örneğin, **fopen()** **open()** çağrısını yapar), ancak bu farklı bir çağrıdır.

Sistem çağrıları

- ❑ İşletim sisteminin G/Ç'yi kontrol etmesinin nedeni güvenlik içindir
- ❑ -- bilgisayar, programımda bir hata varsa, sistemi çökertmemesini ve başka insanların programlarını bozmamasını sağlamalıdır.
- ❑ aynı anda veya daha sonra çalışır.
- ❑ Bu nedenle, disk veya ekran veya ağ I/O yaptığınızda, işletim sisteminden geçmeli ve sistem çağrılarını kullanmalısınız.
- ❑ Bu beş sistem çağrısı, kılavuz sayfalarında tam olarak tanımlanmıştır (do 'man -s 2 open', 'man -s 2 close', vb.).
- ❑ ssize_t ve off_t gibi tüm bu sinir bozucu tipler ints ve longs'lardır. Eskiden hepsi int idi, ancak makineler ve dosyalar büyüdükçe onlar da büyüdü.

Open

- ❑ **Open**, işletim sisteminden bir dosyayı kullanması için istekte bulunur.
- ❑ 'Yol-Path' bağımsız değişkeni hangi dosyayı kullanmak istediğinizi belirtir ve 'flags' ve 'mode' bağımsız değişkenleri onu nasıl kullanmak istediğinizi belirtir.
- ❑ İşletim sistemi isteğinizi onaylarsa, size bir ``dosya tanıtıcı-file descriptor" döndürür.
- ❑ Bu, negatif olmayan bir tam sayıdır. -1 döndürürse, erişiminiz reddedildi ve nedenini belirlemek için "errno" değişkeninin değerini kontrol etmeniz gerekiyor.
- ❑ Dosyalar üzerinde yapacağınız tüm işlemler işletim sistemi üzerinden yapılacaktır.
- ❑ Doğrudan işletim sistemiyle dosya G/Ç yapmak istediğinizde, dosyayı dosya tanıtıcısı ile belirtirsiniz.
- ❑ Bu nedenle, belirli bir dosya üzerinde dosya G/Ç yapmak istediğinizde, bir dosya tanıtıcısı almak için önce o dosyayı açmalısınız.

Open

- ❑ Örnek: src/o1.c, txt/in1.txt dosyasını okumak için açar ve dosya tanıtıcısının değerini yazdırır.
- ❑ txt/in1.txt yoksa veya onu açma izniniz yoksa, open() çağrısı başarısız olduğundan -1 yazdırır.
- ❑ txt/in1.txt varsa, 3 yazdırır, bu da open() isteğinin verildiği anlamına gelir

```
/* This program opens the file "txt/in1.txt" and prints the
   return value of the open() system call. The return value is a
   non-negative integer (three). If "txt/in1.txt" does not
   exist, the return value is -1. */

#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int fd;

    fd = open("txt/in1.txt", O_RDONLY);
    printf("%d\n", fd);
    return 0;
}
```

```
abdullah@abdullah-VirtualBox: ~/sist_prog/cs360-lecture-notes/S
Dosya Düzenle Görünüm Ara Uçbirim Yardım
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes
c1
Opened the file txt/in1.txt twice: Fd's are 3 and 4.
Closed both fd's.
Reopened txt/in1.txt into fd2: 3.
Closed fd2. Now, calling close(fd2) again.
This should cause an error.

c1: Bad file descriptor
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes
```

Open

❑ flags'ın değerine dikkat edin -- open() için kılavuz sayfası size bayrakların ve nasıl çalıştıklarının açıklamasını verecektir. bu nedenle O_RDONLY'nin ve tümünün gerçekten ne anlama geldiğini bilmemiz gerek).

❑ Bin/o1'i çağırmanın birkaç örneğini burada bulabilirsiniz.

❑ Başlangıçta, dizinimde txt/in1.txt adlı bir dosyam var, bu nedenle open() çağrısı başarılı oldu ve 3'ü döndürdü.

❑ Daha sonra onu tmp.txt olarak yeniden adlandırdım ve şimdi open() çağrısı başarısız oldu, -1 döndür. Yeniden adlandırdım ve open() çağrısı tekrar başarılı oldu ve 3'ü döndürdü:

❑ *mv [kaynak dosya] [hedef dizin] (dosya/dizin taşıma)

```
abduallah@abduallah-VirtualBox: ~/sist_prog/cs360-lecture-notes/Syscall-Intro
Dosya Düzenle Görünüm Ara Uçbirim Yardım
abduallah@abduallah-VirtualBox:~/sist_prog/cs360-lecture-notes/Syscall-Intro$ bin/c1
Opened the file txt/in1.txt twice: Fd's are 3 and 4.
Closed both fd's.
Reopened txt/in1.txt into fd2: 3.
Closed fd2. Now, calling close(fd2) again.
This should cause an error.

c1: Bad file descriptor
abduallah@abduallah-VirtualBox:~/sist_prog/cs360-lecture-notes/Syscall-Intro$ ls -l
txt/in1.txt
-rw-rw-r-- 1 abduallah abduallah 22 Mar 12 17:54 txt/in1.txt
abduallah@abduallah-VirtualBox:~/sist_prog/cs360-lecture-notes/Syscall-Intro$ bin/o1
3
abduallah@abduallah-VirtualBox:~/sist_prog/cs360-lecture-notes/Syscall-Intro$ mv txt
/in1.txt tmp.txt
abduallah@abduallah-VirtualBox:~/sist_prog/cs360-lecture-notes/Syscall-Intro$ bin/o1
-1
abduallah@abduallah-VirtualBox:~/sist_prog/cs360-lecture-notes/Syscall-Intro$ mv tmp
.txt txt/in1.txt
abduallah@abduallah-VirtualBox:~/sist_prog/cs360-lecture-notes/Syscall-Intro$ bin/o1
3
abduallah@abduallah-VirtualBox:~/sist_prog/cs360-lecture-notes/Syscall-Intro$
```

Open

- ❑ İkinci örnek: src/o2.c "txt/out1.txt" dosyasını yazmak için açmaya çalışır.
- ❑ Bu başarısız olur çünkü txt/out1.txt zaten mevcut değil. İşte kod -- hatanın neden oluştuğunu yazdırmak için perror() kullandığını not edin.

```
/* This program attempts to open the file
   directory. Note that this fails becaus
   See src/o3.c for an example of opening
```

```
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
```

```
int main()
{
    int fd;

    fd = open("txt/out1.txt", O_WRONLY);
    if (fd < 0) {
        perror("txt/out1.txt");
        exit(1);
    }
    return 0;
}
```

```
UNIX> ls -l txt
total 8
-rw-r--r--  1 plank  staff  22 Jan 30  2018 in1.txt
-rw-r--r--  1 plank  staff   0 Jan 30  2018 out2.txt
UNIX> bin/o2
txt/out1.txt: No such file or directory
UNIX> echo Hi > txt/out1.txt
UNIX> bin/o2
UNIX> cat txt/out1.txt
Hi
UNIX> chmod 0400 txt/out1.txt
UNIX> bin/o2
txt/out1.txt: Permission denied
UNIX> chmod 0644 txt/out1.txt
UNIX> rm txt/out1.txt
UNIX> bin/o2
txt/out1.txt: No such file or directory
UNIX>
```

```
# As you can see, there's no txt/out1.txt
```

```
# Accordingly, then open() call fails.
```

```
# I create txt/out1.txt
```

```
# And now the open() call succeeds
```

```
# The program did not change the file.
```

```
# Here I change the permissions so that I can't open for writing.
```

```
# And the open() call fails.
```

```
# I remove the file
```

```
# And the open() call fails again.
```

Open

- ❑ Yazmak üzere yeni bir dosya açmak için onu flags argümanı olarak (O_WRONLY | O_CREAT | O_TRUNC) ile açmalısınız.
- ❑ O_CREAT, dosya zaten yoksa oluşturmanızı söylüyor.
- ❑ O_TRUNC, dosya varsa, onu sıfır bayta "kırpmasını-truncate" ve orada olanı silmesini söylüyor.

```
/* This program opens the file "out2.txt" for writing in the curre
   uses O_CREAT to create the file if it does not exist already, a
   truncate the file to zero bytes if it does exist. */

#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int fd;

    fd = open("txt/out2.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fd < 0) {
        perror("txt/out2.txt");
        exit(1);
    }
    return 0;
}
```

Open

```
UNIX> ls -l txt/out2.txt
-rw-r--r-- 1 plank staff 0 Jan 30 2018 txt/out2.txt
UNIX> bin/o3
UNIX> ls -l txt/out2.txt
-rw-r--r-- 1 plank staff 0 Feb 3 14:56 txt/out2.txt
UNIX> rm txt/out2.txt
UNIX> bin/o3
UNIX> ls -l txt/out2.txt
-rw-r--r-- 1 plank staff 0 Feb 3 14:57 txt/out2.txt
UNIX> echo "Hi" > txt/out2.txt
UNIX> ls -l txt/out2.txt
-rw-r--r-- 1 plank staff 3 Feb 3 14:57 txt/out2.txt
UNIX> bin/o3
UNIX> ls -l txt/out2.txt
-rw-r--r-- 1 plank staff 0 Feb 3 14:57 txt/out2.txt
UNIX> echo "Hi Again" > txt/out2.txt
UNIX> chmod 0400 txt/out2.txt
UNIX> ls -l txt/out2.txt
-r----- 1 plank staff 9 Feb 3 14:57 txt/out2.txt
UNIX> bin/o3
txt/out2.txt: Permission denied
UNIX> ls -l txt/out2.txt
-r----- 1 plank staff 9 Feb 3 14:57 txt/out2.txt
UNIX> chmod 0644 txt/out2.txt
UNIX> bin/o3
UNIX> ls -l txt/out2.txt
-rw-r--r-- 1 plank staff 0 Feb 3 14:58 txt/out2.txt
.....
```

txt/out2.txt has zero bytes and was last changed in 2018

It still has zero bytes, but the modification time has updated.

Now it created the file anew.

The echo command has put "Hi" and a newline into the file.

bin/o3 has truncated the file.

I have put 9 bytes into the file using echo, but the permission is read-only.

As such, bin/o3 fails to open the file.

And the file is unchanged.

When I change the permissions back to R/W, bin/o3 truncates the file again.

Close

- ❑ `close()`, işletim sistemine bir dosya tanıttıcıyla-file descriptor işinizin bittiğini söyler.
- ❑ İşletim sistemi daha sonra bu dosya tanıttıcıyı yeniden kullanabilir.
- ❑ `src/c1.c` programı, `txt/in1.txt` dosyasının açılıp kapanmasıyla ilgili bazı örnekler gösterir.
- ❑ Dosyayı kapatmadan birden çok kez açtığı için dikkatli bir şekilde bakmalısınız ki bu Unix'te tamamen yasaldır.

❑ Örnek: `src/c1.c`

```
UNIX> bin/c1
Opened the file txt/in1.txt twice:  Fd's are 3 and 4.
Closed both fd's.
Reopened txt/in1.txt into fd2: 3.
Closed fd2.  Now, calling close(fd2) again.
This should cause an error.

c1: Bad file descriptor
UNIX>
```

Read

- ❑ Read(), işletim sistemine "fd" dosya tanıtcısında açılan dosyadan "boyut" baytlarını okumasını ve bu baytları "buf" ile işaret edilen konuma koymasını söyler.
- ❑ Kaç baytın gerçekten okunduğunu döndürür.
- ❑ [Örnek: src/r1.c](#)

```
UNIX> bin/r1  
called read(3, c, 10).  returned that 10 bytes  were read.  
Those bytes are as follows: Jim Plank
```

```
called read(3, c, 99).  returned that 12 bytes  were read.  
Those bytes are as follows: Claxton 221
```

```
UNIX>
```

Read

- ❑ Bu program hakkında dikkat edilmesi gereken birkaç şey var. İlk olarak, **buf** geçerli hafızayı göstermelidir.
- ❑ src/r1.c'de bu, c için malloc()-ing boşlukla sağlanır. Alternatif olarak, c'yi 100 karakterlik statik bir dizi olarak ilan edebilirdim: `char c[100];`
- ❑ **İkinci olarak**, printf()'in anlayacağından emin olmak için read() çağrılarından sonra c'yi sonlandırırım.
- ❑ Bu önemlidir -- metin dosyalarında NULL karakter yoktur. read() bunları okuduğunda, NULL sonlandırmaz.
- ❑ Karakterleri C'de string olarak kullanacaksanız, **onları NULL olarak sonlandırmanız gerekir.**

Read

- ❑ **Üçüncüsü**, `read()` 0 değerini döndürdüğünde, dosyanın sonuna ulaşılmıştır.
- ❑ Bir dosyadan okurken, `read()` istediğinizden daha az bayt döndürürse, dosyanın da sonuna gelmişsiniz demektir. Bu, `src/r1.c`'deki ikinci okumada() olan şeydir.
- ❑ **Dördüncü olarak**, ilk `read()` çağrısındaki 10. karakterin ve ikincideki 12. karakterin her ikisinin de yeni satır karakterleri olduğuna dikkat edin.
- ❑ Bu nedenle `printf()` ifadesinde iki yeni satır alırsınız. Biri `c`'de, diğeri ise `printf()` deyiminde.
- ❑ Yinelemek gerekirse, okuma çağrısı bir NULL karakteri okumaz. Yalnızca dosyadan bayt okur ve dosya herhangi bir NULL karakter içermez. Bu nedenle, NULL karakterini açıkça dizginize koymanız gerekir.

Read

❑ NULL sonlandırmayan (src/r2.c) benzer bir programa bakalım:

```
/* Showing what happens when you don't NULL terminate. */

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

int main()
{
    char c[100];
    int fd;

    strcpy(c, "ABCDEFGHIJKLMNOPQRSTUVWXYZ");
    fd = open("txt/in1.txt", O_RDONLY);
    if (fd < 0) { perror("r1"); exit(1); }

    read(fd, c, 10);           /* I read 10 bytes, but I don't null terminate. */
    printf("%s\n", c);         /* So this printf() will print the characters from K to Z. */

    read(fd, c, 99);           /* This reads 12 bytes, so it prints M to Z. */
    printf("%s\n", c);

    return 0;
}
```

```
UNIX> bin/r2
Jim Plank
KLMNOPQRSTUVWXYZ
Claxton 221
MNOPQRSTUVWXYZ
UNIX>
```

Write

- Write() tıpkı read() gibidir, yalnızca baytları okumak yerine yazar. Gerçekte yazılan ve her zaman "boyut" olan bayt sayısını döndürür.

```
/* This program opens the file "out3.txt" in the current directory
   for writing, and writes the string "cs360\n" to it. */

#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>

int main()
{
    int fd, sz;

    fd = open("txt/out3.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (fd < 0) { perror("txt/out3.txt"); exit(1); }

    sz = write(fd, "cs360\n", strlen("cs360\n"));

    printf("called write(%d, \"cs360\\n\\", %ld).  it returned %d\n",
          fd, strlen("cs360\n"), sz);

    close(fd);
    return 0;
}
```

```
UNIX> bin/w1
called write(3, "cs360\n", 6).  it returned 6
UNIX> cat txt/out3.txt
cs360
UNIX>
```


Write

- ❑ Farklı O_CREAT ve O_TRUNC kombinasyonlarını ve bunların yazma üzerindeki etkilerini düşünmelisiniz.
- ❑ Özellikle src/w2.c'ye bakın. Bu, open() çağrınızda kullandığınız O_WRONLY, O_CREAT ve O_TRUNC kombinasyonunu belirtmenizi sağlar:
- ❑ [src/w2.c](#)

Write

```
UNIX> bin/w2
usage: w2 w|wc|wt|wct input-word
UNIX> rm -f txt/out3.txt
UNIX> ls -l txt/out*
-rw-r--r--  1 plank  staff  0 Feb  3 14:58 txt/out2.txt
UNIX> bin/w2 w Hi
txt/out3.txt: No such file or directory
UNIX> ls txt/out*
txt/out2.txt
UNIX> bin/w2 wc ABCDEFG
called write(3, "ABCDEFG", 7).  It returned 7
UNIX> ls -l txt/out*.txt
-rw-r--r--  1 plank  staff  0 Feb  3 14:58 txt/out2.txt
-rw-r--r--  1 plank  staff  7 Feb  4 17:14 txt/out3.txt
UNIX> cat txt/out3.txt
ABCDEFGUNIX>
UNIX> bin/w2 w XYZ
called write(3, "XYZ", 3).  It returned 3
UNIX> ls -l txt/out3.txt
-rw-r--r--  1 plank  staff  7 Feb  4 17:14 txt/out3.txt
UNIX> cat txt/out3.txt
XYZDEFUNIX>
```

```
# Make sure there's no txt/out3.txt
```

```
# The open() fails because the file doesn't exist, and we didn't specify O_CREAT
```

```
# Because of O_CREAT, the file is created.
```

```
# It's 7 bytes because of the write().
```

```
# We didn't write a newline, so it doesn't print one.
```

```
# I type ENTER to get the prompt looking nice,
# and I write three bytes
```

```
# The file is still 7 bytes, because I didn't call with O_TRUNC
```

```
# It overwrote the "ABC" with "XYZ".
```

Write

```
UNIX> bin/w2 wc ---
called write(3, "---", 3). It returned 3
UNIX> ls -l txt/out3.txt
-rw-r--r-- 1 plank staff 7 Feb 4 17:15 txt/out3.txt
UNIX> cat txt/out3.txt
---DEFGUNIX>
UNIX> bin/w2 wt abcde
called write(3, "abcde", 5). It returned 5
UNIX> ls -l txt/out3.txt
-rw-r--r-- 1 plank staff 5 Feb 4 17:16 txt/out3.txt
UNIX> cat txt/out3.txt
abcdeUNIX>
UNIX> rm txt/out3.txt
UNIX> bin/w2 wt fghij
txt/out3.txt: No such file or directory
UNIX>
```

O_CREAT is specified, but the file exists, so it does nothing. I didn't truncate.

Still 7 bytes.

And the "XYZ" is replaced with "---".

Now, I specify O_TRUNC

And the file is 5 bytes now, rather than 7

Still no newline.

This fails because the file doesn't exist, and I didn't specify O_CREAT.

Lseek

Tüm açık dosyaların kendileriyle ilişkilendirilmiş bir "dosya işaretçisi-file pointer" vardır.

Dosya açıldığında, dosya işaretçisi dosyanın başlangıcını gösterir.

Dosya okunurken veya yazılırken, dosya işaretçisi hareket eder.

Örneğin r1.c'de ilk okumadan sonra dosya işaretçisi txt/in1.txt'de 11. baytı gösterir.

Dosya işaretçisini lseek() ile manuel olarak hareket ettirebilirsiniz.

Lseek

- ❑ lseek'in '-whence-nereden' değişkeni, dosyanın başından, işaretçinin geçerli değerinden ve dosyanın sonundan başlayarak aramanın nasıl yapılacağını belirtir.
- ❑ Dönüş değeri, işaretçinin lseek'ten sonraki konumudur.
- ❑ src/l1.c'ye bakın. txt/in1.txt dosyasında bir sürü arama yapar.
- ❑ İzleyin ve her şeyin mantıklı olduğundan emin olun.
- ❑ sys/types.h ve unistd.h'yi ekleyeceğimi nasıl bildim? "man -s 2 lseek" yazdım.
- ❑ [src/l1.c](#)

Standart Giriş, Çıkış ve Hata

- ❑ Artık Unix'teki her işlem, önceden tanımlanmış ve açık olan üç dosya tanıtıcısıyla başlar:
 - ❑ Dosya tanıtıcı 0 standart girdidir.
 - ❑ Dosya tanıtıcı 1 standart çıktıdır.
 - ❑ Dosya tanıtıcı 2 standart hatadır.
- ❑ Böylece, bir program yazarken standart girdiden `read(0, ...)` kullanarak okuyabilir ve `write(1, ...)` kullanarak standart çıktıya yazabilirsiniz.

Standart Giriş, Çıkış ve Hata

□ Bu bilgilerle donanmış olarak, tek satırlık çok basit bir cat programı (standart girdiyi standart çıktıya kopyalayan) yazabiliriz: (bu src/simpcat.c'dedir):

```
#include <unistd.h>

int main()
{
    char c;

    while (read(0, &c, 1) == 1) write(1, &c, 1);
    return 0;
}
```

```
UNIX> bin/simpcat < txt/in1.txt
Jim Plank
Claxton 221
UNIX>
```