

# Sistem Programlama

---

DR. ÖĞR. ÜYESİ ABDULLAH SEVİN

# İçerik

---

a) Stat and Opendir/Readdir/Closedir

b) Prsize: recursive directory traversal

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

# Stat

ls1

- ❑ Stat, dosyalar hakkında - dosyaların inode'larında bulunan bilgiler - bilgi almak için kullanabileceğiniz bir sistem çağrısıdır.
- ❑ Burada basit bir motive edici örneğin üzerinden geçelim.
- ❑ Diyelim ki stat sistem çağrınız yok ve bir dosya olan her bağımsız değişken için dosyanın **boyutunu ve adını** listeleyen bir program yazmak istiyorsunuz.
- ❑ Yandaki gibi bir şey (src/ls1.c) işe yarayacaktır:

```
/* This program lists each program on its command line together with
its size. It does this by opening each file, lseeking to the
end, and printing the value of the file pointer. */

#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    int i;
    int fd;
    off_t size;

    for (i = 1; i < argc; i++) {
        fd = open(argv[i], O_RDONLY);
        if (fd < 0) {
            printf("Couldn't open %s\n", argv[i]);
        } else {
            size = lseek(fd, (off_t) 0, SEEK_END);
            printf("%10lld %s\n", size, argv[i]);
            close(fd);
        }
    }
    return 0;
}
```

girilen parametreler

↑  
↓  
yok



# Stat

- ❑ Yapılan şey, her dosyayı açmaya çalışmak ve ardından boyutunu anlamak için dosyanın sonuna kadar aramaktır.

```
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Stat$ ls -l txt/input*.txt
-rw-rw-r-- 1 abdullah abdullah 185 Mar 12 17:54 txt/input1.txt
-rw-rw-r-- 1 abdullah abdullah 179 Mar 12 17:54 txt/input2.txt
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Stat$ bin/ls1 txt/input*.txt
    185 txt/input1.txt
    179 txt/input2.txt
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Stat$
```

- ❑ Yine de burada bir sorun var:

```
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Stat$ rm -rf txt/myfile.txt
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Stat$ make txt/myfile.txt
echo "Hi" > txt/myfile.txt
chmod 0 txt/myfile.txt
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Stat$ ls -l txt/myfile.txt
----- 1 abdullah abdullah 3 May 15 06:05 txt/myfile.txt
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Stat$ bin/ls1 txt/myfile.txt
Couldn't open txt/myfile.txt
```

yazdiramaz izin yok

# Stat

---

- ❑ `ls1 "txt/myfile.txt"` dosyasını açamadığı için boyutunu yazdıramadı.
- ❑ Bu talihsiz bir durum ama aynı zamanda "**stat**" işlevine neden ihtiyacımız olduğuna da işaret ediyor -- bir dosya hakkında, dosyanın kendisine erişmemize izin verilmese bile bilmek güzel olacak şeyler vardır.
- ❑ Yinelemek gerekirse, **stat** sistem çağırısı size bir dosyanın inode'u hakkında bilgi verir.
- ❑ Kullanıcının dosyayı içeren dizine erişim izni olduğu sürece bunu yapabilir.
- ❑ Detaylar için man sayfasını okuyun.

# Stat

## stat yapisinda ne var

□ stat yapısı /usr/include/sys/stat.h içinde tanımlanır ve kabaca şöyledir:

```
struct stat {  
    mode_t    st_mode;    /* File mode (see mknod(2)) */  
    ino_t      st_ino;     /* Inode number */  
    dev_t      st_dev;     /* ID of device containing */  
                  /* a directory entry for this file */  
    dev_t      st_rdev;    /* ID of device */  
                  /* This entry is defined only for */  
                  /* char special or block special files */  
    nlink_t    st_nlink;   /* Number of links */  
    uid_t      st_uid;     /* User ID of the file's owner */  
    gid_t      st_gid;     /* Group ID of the file's group */  
    off_t      st_size;    /* File size in bytes */  
    time_t     st_atime;    /* Time of last access */  
    time_t     st_mtime;    /* Time of last data modification */  
    time_t     st_ctime;    /* Time of last file status change */  
                  /* Times measured in seconds since */  
    long       st_blksize; /* Preferred I/O block size */  
    long       st_blocks;  /* Number of 512 byte blocks allocated*/  
};
```

# Stat

---

- ❑ Kafa karıştırıcı türler çoğunlukla ints, longs ve shorts'tur . yani /usr/include/sys/types.h'den:

```
typedef unsigned long    ino_t;  
typedef short           dev_t;  
typedef long            off_t;  
typedef unsigned short  uid_t;  
typedef unsigned short  gid_t;
```

- ❑ Ve /usr/include/sys/stdtypes.h'den:

```
typedef unsigned short  mode_t;           /* file mode bits */  
typedef short          nlink_t;          /* links to a file */  
typedef long           time_t;           /* value = secs since epoch */
```

boyutunu aliyor ve dosya ismini aliyor  
dosyayı ve boyutunu yazdırır

# Stat

- ❑ Bu kılavuz sayfasını okuduktan sonra, ls1.c'yi open/lseek yerine stat kullanarak düzgün çalışacak şekilde değiştirmek basit olmalıdır.
- ❑ Bu src/ls2.c'de:

```
UNIX> bin/ls2 txt/*
185 txt/input1.txt
179 txt/input2.txt
 3 txt/myfile.txt
UNIX>
```

```
/* This is a program which lists files and their sizes to standard output.
   The files are specified on the command line arguments. It uses stat to
   see if the files exist, and to determine the files' sizes. */

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>

int main(int argc, char **argv)
{
    int i;
    struct stat buf;
    int exists;

    for (i = 1; i < argc; i++) {
        exists = stat(argv[i], &buf);
        if (exists < 0) {
            fprintf(stderr, "%s not found\n", argv[i]);
        } else {
            printf("%10lld %s\n", buf.st_size, argv[i]);
        }
    }
    return 0;
}
```



mesela degisim zamanlarini yazdirmek istiyorum

# Stat()-Zamanlar

---

- ❑ Geçmiş olarak, `st_atime`, `st_mtime` ve `st_ctime` dosya erişim, değiştirme ve oluşturma sürelerini saniye cinsinden kaydeder.
- ❑ İşletim sistemleri ilerledikçe, bunlar daha yüksek hassasiyet süreleriyle değiştirildi.
- ❑ Neyse ki, `stat.h`'de genellikle `st_atime`, `st_mtime` ve `st_ctime` ile ikinci-kesinlik zamanlarına erişebilmeniz için tanımlanmış makrolar vardır.
- ❑ Bu güzel, çünkü yalnızca ikinci-kesinliği önemsiyorsanız, kodunuz taşınabilir kalacaktır.

# Stat-Zamanlar

- ❑ Örneğin src/mtime.c programı komut satırında verilen dosyaların değiştirilme sürelerini saniye cinsinden yazdırır. Çoğu makinede src/stat.h'de tanımlanan bir makro olan st\_mtime'ı kullanır:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>

int main(int argc, char **argv)
{
    int i;
    struct stat buf;

    for (i = 1; i < argc; i++) {
        if (stat(argv[i], &buf) != 0) {
            printf("Couldn't stat %s\n", argv[i]);
        } else {
            printf("%s %ld\n", argv[i], buf.st_mtime);
        }
    }
    return 0;
}
```

```
UNIX> echo "Fred" > fred.txt      # Create fred.txt
UNIX> bin/mtime fred.txt          # Show its modification time.
fred.txt 1645028284 ✓
UNIX> touch fred.txt              # I waited 5 seconds before doing this
UNIX> bin/mtime fred.txt          # This is reflected in its modification time.
fred.txt 1645028289 ✓
UNIX> rm fred.txt
UNIX>
```

# Stat

---

- ❑ Ardından, "ls"mizin gerçek "ls" gibi çalışmasını istiyoruz -- argüman kabul etmemesini ve geçerli dizindeki tüm dosyaları listelemesini.
- ❑ Bunun için "opendir/readdir/writer" çağrılarını kullanmamız gerekiyor.
- ❑ Bunların C kitaplığı çağrıları olduğunu ve **sistem çağrıları** olmadığını unutmayın.
- ❑ Bu, sizin için aç/kapat/oku/yaz deyip dizin dosyalarının biçimini yorumladıkları anlamına gelir.
- ❑ "struct dirent" yapısı /usr/include/sys/dirent.h dosyasında tanımlanmıştır:

```
struct dirent {  
    off_t          d_off;          /* offset of next disk dir entry */  
    unsigned long  d_fileno;       /* file number of entry */  
    unsigned short d_reclen;       /* length of this record */  
    char           *d_name;        /* name */  
};
```

# Stat

□ src/ls3.c, ls2.c'yi geçerli dizinden (".") okumak ve tüm dosyaları ve boyutlarını yazdırmak için ayarlar:

```
UNIX> ( cd txt ; ../bin/ls3 )
. 192
.. 320
input1.txt 185
input2.txt 179
.keep 0
myfile.txt 3
UNIX>
```

```
int main(int argc, char **argv)
{
    struct stat buf;
    int exists;
    DIR *d;
    struct dirent *de;

    d = opendir(".");
    if (d == NULL) {
        fprintf(stderr, "Couldn't open \".\"\\n");
        exit(1);
    }

    for (de = readdir(d); de != NULL; de = readdir(d)) {
        exists = stat(de->d_name, &buf);
        if (exists < 0) {
            fprintf(stderr, "%s not found\\n", de->d_name);
        } else {
            printf("%s %lld\\n", de->d_name, buf.st_size);
        }
    }
    closedir(d);
    return 0;
}
```

okudugum dosyann boyutunu  
yazdiriyorum

# Stat

---

- ❑ Şimdi, ls3'ü çalıştırdığınızda iki şeye dikkat edin –
- ❑ ilk olarak, çıktı biçimlendirilmemiştir.
- ❑ ikincisi, dosyalar sıralanmamıştır.
- ❑ Bunun nedeni, `readdir()`'in bir dizindeki dosyaların sıralaması hakkında hiçbir garanti vermemesidir.
- sadece okur
- ❑ Sonraki iki program bu sorunların her birini çözer.
- ❑ İlk olarak, çıktıyı biçimlendirme. Görmek istediğimiz şey şuna benzer:

```
.                192
..               352
input1.txt       185
input2.txt       179
.keep            0
myfile.txt       3
```

# Stat

---

- ❑ Bunu yapabilmek için en uzun dosya adının ne kadar uzun olduğunu bilmemiz gerekiyor.
- ❑ Herhangi bir dosya adını yazdırmadan önce bunu bilmemiz gerekiyor.
- ❑ Yani, yaptığımız şey, tüm dizin girişlerini bağlantılı bir listeye okumak ve yol boyunca maksimum uzunluğu hesaplamak.
- ❑ Bunu yaptıktan sonra listeyi dolaşıyoruz ve çıktıyı güzel bir formatta yazdırıyoruz.
- ❑ `printf()` deyimine yakından bakın ve nasıl çalıştığını anlayabilmek için `printf()` üzerindeki man sayfasını okuyun.



# Stat

---

Bu src/ls4.c'dir:

```
int main(int argc, char **argv)
{
    struct stat buf;
    int exists;
    DIR *d;
    struct dirent *de;
    Dllist files, tmp;
    int maxlen;

    d = opendir(".");
    if (d == NULL) {
        fprintf(stderr, "Couldn't open \".\"\\n");
        exit(1);
    }

    maxlen = 0;
    files = new_dllist();
```

# Stat

## çift yönlü bağlı listeleri okuyoruz

```
for (de = readdir(d); de != NULL; de = readdir(d)) { /* List all fo the files and store in a linked list. */
    dll_append(files, new_jval_s(strdup(de->d_name)));
    if (strlen(de->d_name) > maxlen) maxlen = strlen(de->d_name); /* Maintain the size of the longers filename. */
}
closedir(d);

dll_traverse(tmp, files) { /* Now traverse the list and call stat() on each file to determine its size. */
    exists = stat(tmp->val.s, &buf);
    if (exists < 0) {
        fprintf(stderr, "%s not found\n", tmp->val.s);
    } else {
        printf("%*s %10lld\n", -maxlen, tmp->val.s, buf.st_size);
    }
}
return 0;
}
```

# Stat

---

- ❑ `dll_append()` çağrısında `de->d_name` yerine neden `strdup` kullandık?
- ❑ Kılavuz sayfası, `readdir()` işlevinin döndürdüğü yapının nasıl tahsis edildiği hakkında size hiçbir şey söylemez.
- ❑ Gerçekten varsayabileceğiniz tek şey, bir sonraki `readdir()` veya `closedir()` çağrısını yapana kadar, `readdir()` öğesinin döndürdüğü değerin normal olduğudur.
- ❑ Eğer `readdir()`'in döndürdüğü "struct dirent" için alan `malloc` ettiğini ve kullanıcı `free()`'yi çağırana kadar bu alanın boş olmadığını bilseydik, `de->d_name`'yi kolayca `dlist`'imize koyabilirdik,
- ❑ Ancak, man sayfasından böyle bir güvence gelmediği için `strdup()`'u çağırmamız gerekir.

# Stat

---

❑ Örneğin, opendir/readdir/closedir şu şekilde uygulanabilir:

1. opendir() dizin dosyasını açar ve bir "struct dirent" mallocs yapar.
2. readdir() bu "struct dirent" içindeki bir sonraki dizin girişini okur ve ona bir işaretçi döndürür.,
3. closedir() dizin dosyasını kapatır ve "struct dirent" serbest kalır.

❑ Böyle bir uygulamanın neden dll\_append() ifadesinde strdup() olarak adlandırmamızı gerektirdiğini anlayabilmelisiniz.

❑ Buraya de->d\_name koyarsak, o zaman bellekle ilgili her türlü sorunu yaşayacağız.

❑ Bu ince ama önemli bir nokta.

# Stat

- ❑ Şimdi, sıralanmış dizin dosyalarını yazdırmak için, girdileri bir dlist yerine red-black trees içine eklemeniz yeterlidir. Kod src/ls5.c'dedir. Bu çok basit bir değişikliktir.
- ❑ Ardından, printf ifadesindeki %10d'den kurtulmak istiyoruz.
- ❑ Yani dosya adlarının son sütunu ile dosya boyutlarının ilk sütunu arasında bir boşluk olmasını istiyoruz.
- ❑ Bunu, dizinde gezilirken dosya boyutunun maksimum boyutunu bularak ve bunu printf deyiminde kullanarak yaparız. Bu bir sprintf() ve bir strlen() alır -- bkz. src/ls5a.c.

```
abduallah@abduallah-VirtualBox: ~/sist_prog/cs360-lecture-notes/Stat
Dosya  Düzenle  Görünüm  Ara  Uçbirim  Yardım
abduallah@abduallah-VirtualBox:~/sist_prog/cs360-lecture-notes/Stat$ bin/ls5 txt/*
.                4096
..               4096
.gitignore       51
PN.html          607
README.md        344
bin              4096
clicker-a-e8a0c.html 3100
clicker.html     1963
img              4096
lec_soft.html    17393
lecture.html     17393
makefile         1042
src              4096
txt              4096
abduallah@abduallah-VirtualBox:~/sist_prog/cs360-lecture-notes/Stat$ bin/ls5a txt/*
.                4096
..               4096
.gitignore       51
PN.html          607
README.md        344
bin              4096
clicker-a-e8a0c.html 3100
clicker.html     1963
img              4096
lec_soft.html    17393
```

# Stat

---

- ❑ Son olarak, src/ls6.c, "ls -F" ile aynı işlevi gerçekleştirir.
- ❑ Yani, sonunda "/" olan dizinleri, "@" ile sembolik (hafif) bağlantıları ve "\*" ile yürütülebilir dosyaları yazdırır.
- ❑ Bunu "struct buf"un "st\_mode" alanını yorumlayarak yapabiliyoruz.
- ❑ Kodu gözden geçirin, çünkü jtar yazarken çok işinize yarayacaktır!

```
abdu1lah@abdu1lah-VirtualBox
./
../
.gitignore
PN.html
README.md
bin/
clicker-a-e8a0c.html
clicker.html
img/
lec_soft.html@
lecture.html
makefile
src/
txt/
```



# Soru

---

□ Standart girdiden gelen karaktere göre dizindeki dosyanın ismi ve özelliğini yazan kod (ls3);

s - boyut

t – erişim zamanı

u – kullanıcı ID

l – link sayısı

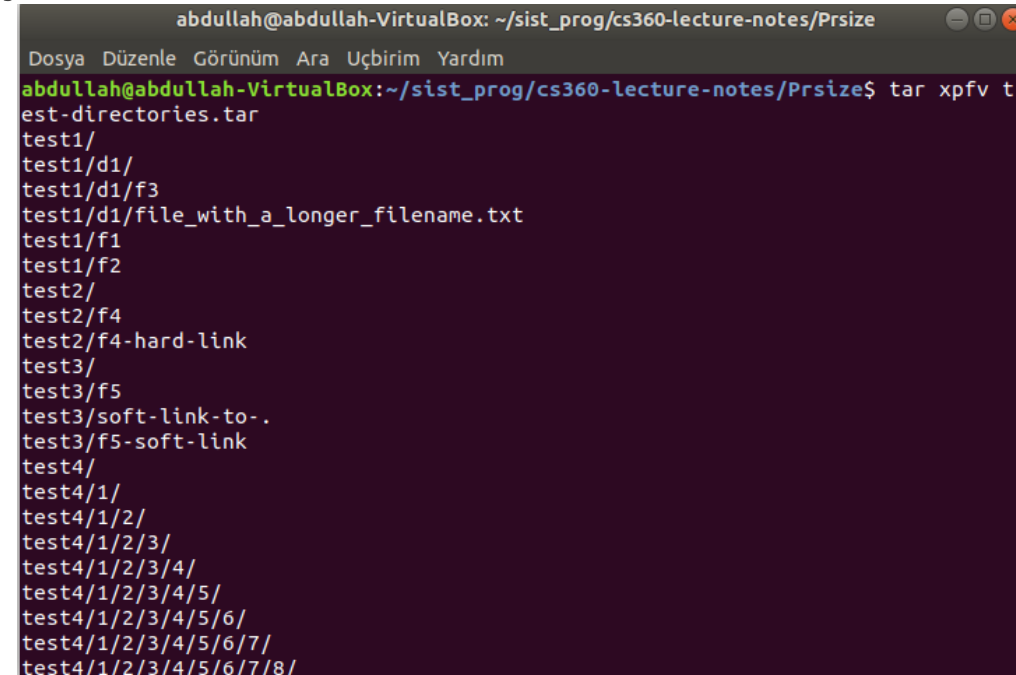
# Prsize

- ❑ Kendi makinenizdeyseniz, şu talimatları izleyin:
- ❑ Prsize dizinini konsoldan açalım. (../Libfdr dizinine gitmeniz ve önce make yazmanız gerekebilir.)
- ❑ tar xpfv test-directories.tar yazalım. Sıkıştırılmış dosyayı açar.

```
UNIX> tar xpfv test-directories.tar önce aciliyor
test1/
test1/d1/
test1/d1/f3
test1/d1/file_with_a_longer_filename.txt
test1/f1
test1/f2
test2/
test2/f4-soft
.....
UNIX>
```

Finally:

```
UNIX> cd test1
```



```
abduallah@abduallah-VirtualBox: ~/sist_prog/cs360-lecture-notes/Prsize
Dosya Düzenle Görünüm Ara Uçbirim Yardım
abduallah@abduallah-VirtualBox:~/sist_prog/cs360-lecture-notes/Prsize$ tar xpfv test-directories.tar
test1/
test1/d1/
test1/d1/f3
test1/d1/file_with_a_longer_filename.txt
test1/f1
test1/f2
test2/
test2/f4
test2/f4-hard-link
test3/
test3/f5
test3/soft-link-to-.
test3/f5-soft-link
test4/
test4/1/
test4/1/2/
test4/1/2/3/
test4/1/2/3/4/
test4/1/2/3/4/5/
test4/1/2/3/4/5/6/
test4/1/2/3/4/5/6/7/
test4/1/2/3/4/5/6/7/8/
```

# Prsize

---

- ❑ Başlangıç: Geçerli dizindeki tüm dosyaların boyutunu hesaplama.
  - ❑ Prsize'nin yaptığı, geçerli dizinden erişilebilen tüm dosyalar tarafından kaplanan bayt sayısını döndürmek (yazılım bağlantıları hariç).
  - ❑ opendir/readdir/closedir, stat, recursive kullanmayı, yol adları oluşturmayı ve sabit bağlantılar bulmayı gösterdiği için iyi bir programdır.
  - ❑ Önce src/prsize1.c yazdım.
  - ❑ Bu, geçerli dizindeki tüm dosyaların toplam boyutunu yazdırır.
  - ❑ Bu, stat ve opendir/readdir/closedir'in basit bir kullanımudur:
- ```
UNIX> pwd
/home/jplank/cs360/notes/Prsize/test1
UNIX> ../bin/prsize1
357
UNIX>
```

# Prsize

---

```
/* This program prints the size of all files in the current directory. */

#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>
#include <sys/stat.h>

int main()
{
    DIR *d;
    struct dirent *de;
    struct stat buf;
    int exists;
    long total_size;

    d = opendir(".");
    if (d == NULL) {
        perror(".");
        exit(1);
    }

    total_size = 0;

    /* Return value of opendir(). */
    /* Return value of each readdir() call. */
    /* The information about each file returned by stat() */
    /* Return value of stat on each file. */
    /* The total size of all files. */

    /* Open "." to list all the files. */
```

# Prsize

---

```
total_size = 0;

/* Run through the directory and run stat() on each file,
   keeping track of the total size of all of the files. */

for (de = readdir(d); de != NULL; de = readdir(d)) {
    exists = stat(de->d_name, &buf);
    if (exists < 0) {
        fprintf(stderr, "Couldn't stat %s\n", de->d_name);
    } else {
        total_size += buf.st_size;
    }
}

/* Although the closedir call isn't necessary, it will be later... */

closedir(d);
printf("%ld\n", total_size);
return 0;
}
```

# Prsize

- ❑ Geçerli dizininizde test edin (test1):
- ❑ Farklı bir değer elde edebilirsiniz (örneğin, ev makinemde 12314 aldım), ancak tüm dosyaların uzun bir listesini yaparsanız, değeriniz tüm dosya boyutlarının toplamına eşit olmalıdır. Örneğin:
- ❑ bc (basic mathematical calculations): temel hesaplama komutu

```
abdullah@abdullah-VirtualBox: ~/sist_prog/cs360-lecture-notes/Prsize/test1
Dosya Düzenle Görünüm Ara Uçbirim Yardım
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Prsize/test1$ pwd
/home/abdullah/sist_prog/cs360-lecture-notes/Prsize/test1
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Prsize/test1$ ls -l
a
toplam 20
drwxr-xr-x 3 abdullah abdullah 4096 Şub 18 2021 .
drwxrwxr-x 9 abdullah abdullah 4096 May 15 07:34 ..
drwxr-xr-x 2 abdullah abdullah 4096 Şub 18 2021 d1
-rw-r--r-- 1 abdullah abdullah 11 Eyl 23 1994 f1
-rw-r--r-- 1 abdullah abdullah 15 Eyl 23 1994 f2
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Prsize/test1$ ../bin/prsize1
12314
abdullah@abdullah-VirtualBox:~/sist_prog/cs360-lecture-notes/Prsize/test1$ echo
4096 + 4096 + 4096 + 11 + 15 | bc
12314
```



# Prsize

---

- ❑ Alt dizinlere de erişmek için programı özyinelemeli yapalım;
- ❑ Şimdi, atacağımız bir sonraki adım, programın geçerli dizinden erişilebilen tüm dosyaların boyutlarını özetlemesini sağlamaktır.
- ❑ Bunu yapmak için programı özyinelemeli yapmalıyız.
- ❑ Tüm kodumuzu `main()` yordamına koymak yerine, onu bir fonksiyon içine gömeceğiz ve bu fonksiyonu çağıracağız. `src/prsize2.c` bunu yapar.
- ❑ Boyutu bulmak için `get_size()` çağrısı yapması dışında `src/prsize1.c` ile aynı işlevselliği sağlar.

# Prsize

---

- ❑ Şimdi bin/prsize2'yi özyinelemeli yapmak istiyoruz.
- ❑ Ne zaman bir dizinle karşılaşsak, o dizindeki her şeyin boyutunu öğrenmek isteriz, bu yüzden o dizinde yinelemeli olarak `get_size()`'yi çağırırız.
- ❑ Bu, `src/prsize3.c`'de yapılır İşte `get_line()` içindeki ilgili kod

```
/* Run through the directory and run stat() on each file,
   keeping track of the total size of all of the files. */

for (de = readdir(d); de != NULL; de = readdir(d)) {
    exists = stat(de->d_name, &buf);
    if (exists < 0) {
        fprintf(stderr, "Couldn't stat %s\n", de->d_name);
        exit(1);
    }
    total_size += buf.st_size;

    /* If the file is a directory, make a recursive call to get_size(): */
    if (S_ISDIR(buf.st_mode)) {
        total_size += get_size(de->d_name);
    }
}
```

cagiracagizz

# Prsize

---

When we try it, we get an odd error:

```
UNIX> ../bin/prsize3
.: Too many open files
UNIX>
```

❑ Peki, neler oldu? Pekala, kontrol etmek için, özyinelemeli çağrılar ne zaman yaptığını görmek için src/prsize3a.c'ye bir print ifadesi koyduk:

```
UNIX> ../bin/prsize3a
Making a recursive call to .
Making a recursive call to .
Making a recursive call to .
Making a recursive call to .
.....
Making a recursive call to .
.: Too many open files
UNIX>
```

# Prsize

---

- ❑ Şimdi neler olduğunu görebilirsiniz. "." içindeki dosyaları sıralarken "." dosyasıyla karşılaşılırsınız.
- ❑ Bu bir dizindir, bu nedenle üzerinde özyinelemeli bir arama yaparsak problem oluşur.
- ❑ Bu, açık dosya tanımlayıcılarınız bitene kadar sonsuz bir döngüye girer ve bu noktada opendir() başarısız olur.
- ❑ Bunu düzeltmek için, "." dizin. ".." için de kontrol etmeniz gerekir. Bunu bir sonraki programda yapacağız:

```
/* If the file is a directory, and not . or .. make a recursive call to get_size(): */  
  
if (S_ISDIR(buf.st_mode) && strcmp(de->d_name, ".") != 0 && strcmp(de->d_name, "..") != 0) {  
    total_size += get_size(de->d_name);  
}  
}
```

src/prsize4.c

# Prsize

---

- ❑ Çalıştırdığımızda sonsuz döngü hatası düzeliyor ama başka bir hatamız daha var!!

```
UNIX> ../bin/prsize4  
Couldn't stat f3  
UNIX>
```

- ❑ Soruna bakalım: . Dizininde (mevcut dizin) f3 ismindeki dosyaları bul yazdır (-print)

```
UNIX> find . -name f3 -print          # find is a super-helpful command.  Read the man page.  
./d1/f3  
UNIX>
```

## sirayla arama yapıyor

# Prsize

---

- ❑ Bu bize şu yanıtı verir: program d1 dizininde f3'ü stat etmeye çalışıyor, ancak d1 dizininde çalışmaz.
- ❑ Başka bir deyişle, prsize4, Prsize/test1 dizininden çağrılır ve "exists = stat("f3", &buf)" çağrısını yapar.
- ❑ Tabii ki stat -1 döndürecek, çünkü dizinde f3 dosyası yok. Bunun yerine "d1/f3"ü aramamız gerekiyor.
- ❑ Başka bir deyişle, kodumuzda bir hata var -- sadece de->d\_name'yi değil, get\_size() içinde fn/de->d\_name'yi aramamız gerekiyor.



# Prsize

---

- ❑ Dosya adlarını oluşturmak bir sorunu çözerken diğer bir sorunu ortaya çıkarır
- ❑ Dosya adlarını src/prsize5.c içinde oluşturuyoruz. İşte değişen kod. C++ string'leri keşke olsaydı burada, değil mi?
- ❑ Çalıştırdığımızda oldukça iyi görünüyor. realloc() kodunun test edilebilmesi için d1'e daha uzun bir dosya adı koydum:

# Prsize

---

```
UNIX> ../bin/prsize5
1322
UNIX> ls d1
f3 file_with_a_longer_filename.txt
UNIX>
```

Before moving on, we should sanity check our output:

```
UNIX> ../bin/prsize5|
1322
UNIX> ls -la
total 8
drwxr-xr-x. 3 jplank jplank 36 Feb 18 15:00 .
drwxr-xr-x. 9 jplank jplank 240 Feb 18 15:45 ..
drwxr-xr-x. 2 jplank jplank 55 Feb 18 15:44 d1
-rw-r--r--. 1 jplank jplank 11 Sep 23 1994 f1
-rw-r--r--. 1 jplank jplank 15 Sep 23 1994 f2
UNIX> ls -l d1
total 8
-rw-r--r--. 1 jplank jplank 17 Sep 23 1994 f3
-rw-r--r--. 1 jplank jplank 857 Feb 18 15:44 file_with_a_longer_filename.txt
UNIX> echo 36 + 240 + 55 + 11 + 15 + 17 + 857 | bc
1231
UNIX>
```

# Prsize

---

- ❑ Bekle -- 1322, 1231'e eşit değildir. stat() çağrısının önüne bir print ifadesi koymak iyi bir fikir olabilir.
- ❑ Bunu yapmayacağız çünkü sorunu biliyorum -- "d1/"'yi dikkate almadım.
- ❑ ve "d1/.." yukarıda 1231'i hesapladığımda:

```
UNIX> ls -la d1
total 8
drwxr-xr-x. 2 jplank jplank  55 Feb 18 15:44 .
drwxr-xr-x. 3 jplank jplank  36 Feb 18 15:00 ..
-rw-r--r--. 1 jplank jplank  17 Sep 23  1994 f3
-rw-r--r--. 1 jplank jplank 857 Feb 18 15:44 file_with_a_longer_filename.txt
UNIX> echo 36 + 240 + 55 + 11 + 15 + 55 + 36 + 17 + 857 | bc
1322
UNIX>
```

# Prsize

---

❑ Sizce hem "./d1" hem de "d1/." boyutunu saymak doğru mu?

❑ Ya da her ikisi de "." ve "d1/.."? **Yani alt dizinde iken üst dizini ve üst dizinde iken alt dizini**

❑ Yapmıyorum. ../test2'ye geçerseniz, ilgili bir sorunu ortaya çıkaracağız:

```
UNIX> cd ../test2
UNIX> ls
f4  f4-hard-link
UNIX> ls -lai
total 8
    7876 drwxr-xr-x. 2 jplank jplank  36 Feb 11  2018 .
402660801 drwxr-xr-x. 9 jplank jplank 240 Feb 18 15:45 ..
    7877 -rw-r--r--. 2 jplank jplank  11 Sep 23  1994 f4
    7877 -rw-r--r--. 2 jplank jplank  11 Sep 23  1994 f4-hard-link
UNIX>
```

# Prsize

---

- Gördüğünüz gibi, f4 ve f4-hard-link aynı dosyanın linkleridir. bin/prsize5'i çalıştırdığımızda, elbette ikisini de sayar:

```
UNIX> ../bin/prsize5
298
UNIX> echo 36 + 240 + 11 + 11 | bc
298
UNIX>
```

# Prsize

---

- ❑ inode'ları takip edebiliriz, böylece her dosyayı yalnızca **bir kez** sayarız.
- ❑ Prsize'ın sabit bağlantıları tanıyabilmesi ve bunları yalnızca bir kez sayabilmesi için ihtiyacımız olan şey.
- ❑ İki dosyanın aynı disk dosyasına bağlantı olup olmadığını nasıl anlarsınız? Inode numarasını kullanırsınız. Bu, `buf.st_ino`'da tutulur.
- ❑ Duplicate edilen inode'ları kontrol etme yöntemimiz, şimdiye kadar gördüğümüz bir rb-ağaç yapısını korumaktır.
- ❑ Herhangi bir dosyanın boyutunu eklemekten önce, inode'unun rb ağacında olup olmadığını kontrol ederiz. Eğer öyleyse, başka hiçbir şey yapmıyoruz.
- ❑ Aksi takdirde, boyutu ekleriz ve inode'u rb-tree'ye koyarız.

# Prsize

---

- ❑ Bazı sistemlerde inode'ların int yerine long olması talihsiz bir durumdur, bu yüzden onları bir JRB'de depolamak için jval'in ".l" alanını kullanırız ve özel bir karşılaştırma işlevi ekleriz.
- ❑ Kod src/prsize6.c'de ve her zamanki gibi değişiklikleri vurgulayacağız.

```
int compare(Jval v1, Jval v2)           /* Adding a comparison function for inodes. */
{
    if (v1.l < v2.l) return -1;
    if (v1.l > v2.l) return 1;
    return 0;
}

long get_size(const char *fn, JRB inodes) /* get_size now passes the tree of inodes. */
{
    /* A lot of code deleted. */

    /* Check the inodes tree to check if we've seen this file before.
       If so, ignore. If not, then add in its size. */

    if (jrb_find_gen(inodes, new_jval_l(buf.st_ino), compare) == NULL) {
        jrb_insert_gen(inodes, new_jval_l(buf.st_ino), new_jval_i(0), compare);
        total_size += buf.st_size;
    }

    /* If the file is a directory, and not . or .. make a recursive call to get_size(): */

    if (S_ISDIR(buf.st_mode) && strcmp(de->d_name, ".") != 0 && strcmp(de->d_name, "..") != 0) {
        total_size += get_size(dir_fn, inodes); /* I add the inode tree to get recursion. */
    }
}
```

# Prsize

---

```
    closedir(d);
    free(dir_fn);
    return total_size;
}

int main()
{
    long total_size;
    JRB inodes;                /* I create the inode tree in main and pass it to the initial get_size() call. */

    inodes = make_jrb();
    total_size = get_size(".", inodes);
    printf("%ld\n", total_size);
    return 0;
}
```

This fixes our previous problems:

```
UNIX> ../bin/prsize6
287                                # This is 11 less than before, so it's correct.
UNIX> cd ../test1
UNIX> ../bin/prsize6
1231                               # This matches our first calculation above, so it's correct.
UNIX>
```



## Prsize

- ❑ Hafif (Soft) bağlantılar için endişelenmeli miyiz?

```
UNIX> cd ../test3
```

```
UNIX> ls -lai
```

```
total 4
```

```
134269623 drwxr-xr-x. 2 jplank jplank 58 Sep 24 1996 .
```

```
402660801 drwxr-xr-x. 9 jplank jplank 240 Feb 18 16:16 ..
```

```
134345832 -rw-r--r--. 1 jplank jplank 11 Sep 23 1994 f5
```

```
134345834 lrwxrwxrwx. 1 jplank jplank    2 Aug  1  2014 f5-soft-link -> f5
```

```
134345833 lrwxrwxrwx. 1 jplank jplank 1 Aug 1 2014 soft-link-to-. -> .
```

UNIX>

- 📌Burada birkaç hafif bağlantı var -- bin/prsize6'ya ne yaptıklarına bakalım:

```
UNIX> ../bin/prsize6
```

```
Couldn't stat ./soft-link-to-./soft-link-to-./soft-link-to-./soft-link-to-./soft-link-to-./soft-link-to-./soft-link-to-...

```

UNIX>

# Prsize

---

- ❑ Peki ne oldu? stat() kullandığımız için, bin/prsize6 hafif bağlantıları tanımıyor ve bu nedenle daha önce olduğu gibi aynı sonsuz döngü sorununuz var.
- ❑ Ne istediğimiz açık olmalı -- bağlantıyı "." konumuna geçmek yerine prsize'in bağlantının boyutunu saymasını istiyoruz (f5-soft-link için 2 bayt ve soft-link-to- için 1 bayt ).
- ❑ Bu nedenle, prsize7.c'de tek yapmamız gereken stat() yerine lstat() kullanmaktır.
- ❑ Bu, bağlantının işaret ettiği dosya yerine geçici bağlantının kendisi hakkında bilgi verir:

```
UNIX> ../bin/prsize7
312
UNIX> echo 58 + 240 + 11 + 2 + 1 | bc
312
UNIX>
```

# Prsize

---

- ❑ Henüz işimiz bitmedi -- açık dosyalarla ilgili **ince bir hata** var.
- ❑ Son olarak, bu programda bir hata daha var ve bu gerçekten detay.
- ❑ Açık dosya tanımlayıcıları ile ilgisi var. İlk olarak, test4 dizinine gidin.
- ❑ Aşağıda, iç içe geçmiş 10 dizinden oluştuğunu göstermek için `find` komutunu kullanıyorum.
- ❑ `Bin/prsize7`'nin üzerinde gayet iyi çalıştığını görebilirsiniz:

```
UNIX> cd ../test4
UNIX> find . -print
.
./1
./1/2
./1/2/3
./1/2/3/4
./1/2/3/4/5
./1/2/3/4/5/6
./1/2/3/4/5/6/7
./1/2/3/4/5/6/7/8
./1/2/3/4/5/6/7/8/9
UNIX> ../bin/prsize7
381
UNIX>
```

# Prsize

---

- ❑ Çalışmasının nedeni, varsayılan değerlerimizin genellikle işlem başına bir ton açık dosyaya izin vermesidir.
- ❑ Bunu BASH **ulimit** komutunu çalıştırarak görebiliriz (kabuğunuz bu komutu tanımıyorsa, bunun yerine limiti deneyin):

```
UNIX> ulimit -a | grep open
open files                (-n) 1024
```

**grep:** tanımlanmış bir kelime veya dize kriteri için metin aramaları gerçekleştirin

- ❑ Bu sayıyı 1024 yerine on olarak ayarlayalım. Şimdi bin/prsize7, çok fazla açık dosya nedeniyle başarısız oluyor:

```
UNIX> ulimit -n 10
UNIX> ulimit -a | grep open
open files                (-n) 10
UNIX> ../bin/prsize7
./1/2/3/4/5/6/7: Too many open files
UNIX>
```

# Prsize

---

- ❑ Olan şu ki, `get_size()`'ye tekrarlanan çağrılar, `opendir()` ve `closedir()` çağrıları arasında yapılıyor.
- ❑ Bu, her özyinelemeli çağrı yaptığımızda, açık dosya sayısına bir tane eklediğimiz anlamına gelir.
- ❑ Yalnızca on açık dosyayla (ve işlemi başlatmak için üç açık dosyayla), `"/1/2/3/4/5/6/7"` dosyasını açmaya çalıştığımızda dosya tanıtıcılarımız tükeniyor.
- ❑ Bunun çözümü, özyinelemeli çağrı yaptığımızda açık dosya olmadığından emin olmaktır.
- ❑ Bunu nasıl yapabiliriz? Bir dizindeki dosyaları numaralandırırken, tüm dizinleri bir `dllist`'e koyuyoruz ve ardından dizin dosyasını kapattıktan sonra, listeyi dolaşıp recursive çağrılar yapıyoruz.
- ❑ Dizinleri `dllist`'e koyduğumuzda bir `strdup()` yapmamız gerekiyor. Neden? Bir düşünün ya da yapmayınca ne oluyor görün... `prsize`'nin doğru ve son hali `src/prsize8.c`'de. İşte değişiklikler:

# Prsize

---

```
long get_size(const char *fn, JRB inodes)
{
    /* Other variable declarations are deleted. */

    Dllist directories, tmp; /* Dllist of directory names, for doing recursion after closing. */
    /* Initialize (other code deleted). */

    directories = new_dllist();

    for (de = readdir(d); de != NULL; de = readdir(d)) {

        /* Other code deleted */

        /* Don't make the recursive call, but instead put the directory into the dllist. */

        if (S_ISDIR(buf.st_mode) && strcmp(de->d_name, ".") != 0 && strcmp(de->d_name, "..") != 0) {
            dll_append(directories, new_jval_s(strdup(dir_fn)));
        }

        /* Make the recursive calls after you've closed the directory. */

        closedir(d);

        dll_traverse(tmp, directories) {
            total_size += get_size(tmp->val.s, inodes);
        }

        /* Clean up. You need to free the strings inside the dllist, because you
           allocated them with strdup(), and they'll be a memory leak otherwise. */

        dll_traverse(tmp, directories) free(tmp->val.s);
        free_dllist(directories);
        free(dir_fn);

        return total_size;
    }
}
```

# Prsize

---

```
UNIX> ulimit -n 10
UNIX> ../bin/prsize8
381
UNIX>
```

As an aside, it's 2021, and **find** still has the same bug as **prsize7.c**:

```
UNIX> ulimit -a | grep open
open files                (-n) 10
UNIX> find . -print
.
./1
./1/2
./1/2/3
./1/2/3/4
find: â€˜./1/2/3/4â€™: Too many open files
UNIX>
```

On the flip side, **tar** handles it correctly:

```
UNIX> tar cvf ~/junk.tar .
./
./1/
./1/2/
./1/2/3/
./1/2/3/4/
./1/2/3/4/5/
./1/2/3/4/5/6/
./1/2/3/4/5/6/7/
./1/2/3/4/5/6/7/8/
./1/2/3/4/5/6/7/8/9/
UNIX>
```