

Dosya ve Dosya Sistemi Durumu

Kaynak: <http://web.eecs.utk.edu/~plank/plank/classes/cs360/>

stat

- stat - display file or file system status
- stat() sistem çağrısı dosyalar hakkında inode içeriğini kullanarak bilgi sağlar. Bunun için dosyası içeren klasöre izin olması gerekir.
- /usr/include/sys/stat.h

```
File Edit View Search Terminal Help
[bi1g 10_stat]$ stat ls1.c
  File: 'ls1.c'
  Size: 628          Blocks: 8          IO Block: 4096   regular file
Device: 805h/2053d  Inode: 1447112      Links: 1
Access: (0600/-rw-----)  Uid: ( 1000/      bi1g)   Gid: ( 1000/      bi1g)
Access: 2016-03-13 17:49:26.868669156 +0200
Modify: 2016-03-13 17:32:42.880689643 +0200
Change: 2016-03-13 17:32:42.940689642 +0200
 Birth: -
[bi1g 10_stat]$
```

Stat yapısı

- Terminal ekranından `man 2 stat` yazılarak
- `int stat(const char *path, struct stat *buf);`
- `int fstat(int fd, struct stat *buf);` (fd -file descriptor ile çalışır)
- `int lstat(const char *path, struct stat *buf);` (l-symbolic link)
- `lstat()` sembolik (soft) link durumunda dosyaya ait bilgiler yerine link bilgilerini getirir.

Örnek 1: Dosya boyutu ölçen program

- Bu örnekte dosya adların ve içeriğini listeleyen stat fonksiyonu benzeri bir program yazılmıştır.
- Dosya boyutunu ölçmek için dosya içerisinde istenilen konuma ulaşmayı sağlayan *lseek()* fonksiyonu kullanılmıştır.
- Program komut satırından belirtilen dosya veya dosyaları açtıktan sonra *lseek()* ile işaretçiyi dosya sonuna konumlandırarak geriye döndürülen karakter adedinden dosya boyutunu ölçer.

Dosyanın sonuna kadar gidip karakterleri sayar. Böylece byte cinsinden büyüklüğü verir.

```
File Edit View Search Tools Documents Help
Open Save Undo
ls1.c x
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <unistd.h>
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("%10ld %s\n", size, argv[i]);
            close(fd);
        }
    }
}
```

Komut satırı parametresi olarak belirtilen dosyalardan sıradakini aç

C Tab Width: 8 Ln 29, Col 1 INS

Örnek: Dosya boyutu ölçen program

```
File Edit View Search Terminal Help
[bilg 10_stat]$ ./ls1 lecture.html
12347 lecture.html
[bilg 10_stat]$ ls -l lecture.html
-rw----- 1 bilg bilg 12347 Şub  6 2012 lecture.html
[bilg 10_stat]$
```

Örnek olarak seçilen
lecture.html
dosyasının boyutu
ölçülmüştür.

```
File Edit View Search Terminal Help
[bilg 10_stat]$ ./ls1 *.c
625 ls1.c
659 ls2.c
754 ls3.c
1031 ls4.c
1215 ls5a.c
952 ls5.c
1303 ls6.c
[bilg 10_stat]$
```

.c ile biten dosya
büyüklükleri
listelenmiştir.

Örnek: Dosya boyutu ölçen program

```
File Edit View Search Terminal Help
[bilg 10_stat]$ echo "bu bir test dosyası" > test
[bilg 10_stat]$ ./ls1 test
      21 test
[bilg 10_stat]$ chmod 0 test
[bilg 10_stat]$ ./ls1 test
Couldn't open test
[bilg 10_stat]$ ls -l test
----- 1 bilg bilg 21 Mar 14 00:55 test
[bilg 10_stat]$
```

- Örnek program korumalı dosyalara ulaşamıyor.
- stat() sistem çağrısını kullanan ls korumalı dosya hakkında bilgi veriyor

Stat struct yapısı

■ stat() sistem çağrısı stat yapısında veri döndürür:

```
struct stat {  
    dev_t    st_dev;        /* ID of device containing file */  
    ino_t    st_ino;        /* inode number */  
    mode_t   st_mode;      /* protection */  
    nlink_t  st_nlink;     /* number of hard links */  
    uid_t    st_uid;       /* user ID of owner */  
    gid_t    st_gid;       /* group ID of owner */  
    dev_t    st_rdev;      /* device ID (if special file) */  
    off_t    st_size;      /* total size, in bytes */  
    blksize_t st_blksize;  /* blocksize for filesystem I/O */  
    blkcnt_t st_blocks;    /* number of 512B blocks allocated */  
    time_t   st_atime;     /* time of last access */  
    time_t   st_mtime;     /* time of last modification */  
    time_t   st_ctime;     /* time of last status change */  
};
```

Stat yapısı

■ typedef unsigned long	ino_t;
■ typedef short	dev_t;
■ typedef long	off_t;
■ typedef unsigned short	uid_t;
■ typedef unsigned short	gid_t;
■ typedef unsigned short	mode_t;
■ typedef short	nlink_t;
■ typedef long	time_t;

Örnek 2: stat sistem çağrısı

- Bir önceki programın doğru çalışması için open ve lseek yerine stat sistem çağrısı kullanılabilir.

```
File Edit View Search Tools Documents Help
*ls2.c x
*/
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <time.h>
main(int argc, char **argv)
{
    int i;
    struct stat buf;
    int exists;

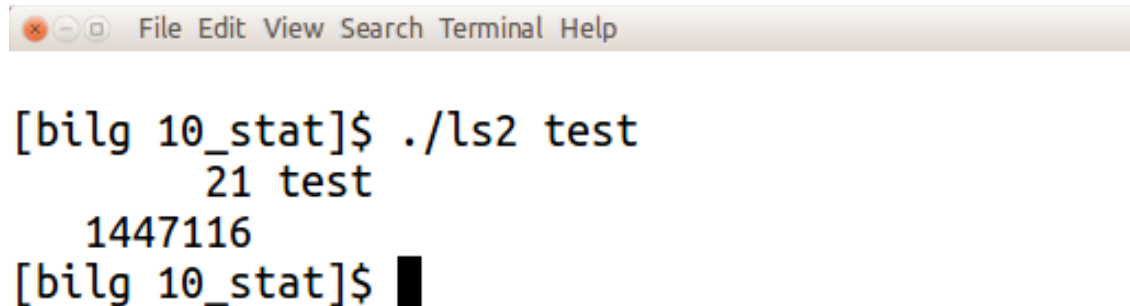
    for (i = 1; i < argc; i++) {
        exists = stat(argv[i], &buf);
        if (exists < 0) {
            fprintf(stderr, "%s not found\n", argv[i]);
        } else {
            printf("%10ld %20s %10ld\n",
                buf.st_size, argv[i], buf.st_ino);
        }
    }
}
```

stat çağrısı ile dosya bilgilerini al

st_size : dosya boyutu
st_ino : inode numarası

stat structure ile dosya bilgileri döndürülüyor

Örnek: stat sistem çağrısı



```
File Edit View Search Terminal Help

[bilg 10_stat]$ ./ls2 test
      21 test
    1447116
[bilg 10_stat]$
```

- ls2 programının ls gibi parametre kabul etmeden çalışabilmesi için çalışma klasöründeki tüm dosyaların listelenmesi gerekir.

opendir/readdir

- Çalışma klasöründeki dosyaların listelenmesi için **opendir**, **readdir**, **writerdir**, **closedir** gibi çağrılarının kullanılması gerekir. Bunlar arka planda open, read, write, close gibi çağrılar kullanır.
- Bunun için dirent yapısının kullanılması gerekir
- `struct dirent *readdir(DIR *dirp);`

```
struct dirent {  
    ino_t      d_ino;    /* inode number */  
    off_t      d_off;    /* not an offset; see NOTES */  
    unsigned short d_reclen; /* length of this record */  
    unsigned char d_type;  /* type of file; not supported  
                           by all filesystem types */  
    char        d_name[256]; /* filename */  
};
```

Örnek 3: klasördeki dosyaların listelenmesi

```
File Edit View Search Tools Documents Help
Open Save Undo
ls3.c x
#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <dirent.h>
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 %ld\\n", de->d_name, buf.st_size);
        }
    }
    closedir(d);
}
```

“.” çalışma klasörü

stat çağrısı ile dosya bilgilerini al

Klasörü kapat

d ile belirtilen klasörden sıradaki içeriği oku

Örnek 3: klasördeki dosyaların listelenmesi

```
10_stat:./ls3 *
makefile 64
ls5.o 2864
. 4096
derle.txt 57
ls5 19546
ls1 8703
ls6.o 2984
ls5a.c 1169
makefile~ 64
ls4 14812
ls4.c 1051
```

- Listeleme yandaki gibi hizalama yapılarak yazdırılabilir.
- Doğru hizalama için dosya isimlerinin kaç karakter uzunluğunda olduğunu tespit etmek gerekir.

```
10_stat:./ls3 *
    makefile          64
    ls5.o             2864
    .                 4096
    derle.txt         57
    ls5               19546
    ls1               8703
    ls6.o             2984
    ls5a.c            1169
    makefile~         64
    ls4               14812
    ls4.c             1051
```

Örnek 4

- Bu örnekte en uzun dosya adı belirlenip ona göre hizalama yapılıyor.
- Önceki örnekten farklı olarak dosya isimleri çift yönlü bağlı listeye eklenmiştir. strdup () string duplication dosya ismi için hafızada yer ayırır ve böylece bağlı listeye ekler.
- Bağlı liste kullanmadan da ls3.c üzerinde değişiklik yaparak aynı işlem gerçekleştirilebilir.
- printf satırında kullanılan *kaç karakter kullanılacağı ile belirtilir. Ör: makslen=10 ise %10s .

Dosya isimlerini listeye ekle.
En uzun string boyutunu belirle.

Çift yönlü bağlı listenin elemanları dll_traverse makro fonksiyonu ile dolaşılıyor.

```
#include <sys/stat.h>
#include <dirent.h>
#include "dllist.h"//doubly linked list

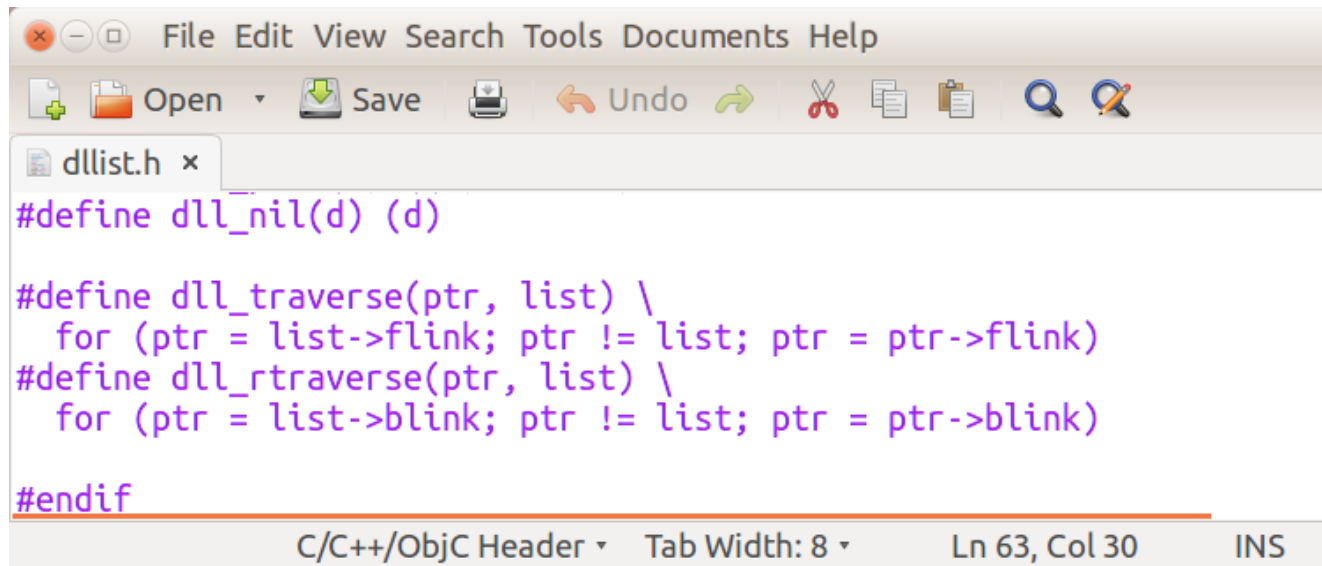
main(int argc, char **argv)
{
    struct stat buf;
    int exists;
    DIR *d;
    struct dirent *de;
    Dllist files, tmp;
    int maxlen;
    d = opendir(".");
    if (d == NULL) {
        fprintf(stderr, "Couldn't open \".\"\\n");
        exit(1);
    }

    maxlen = 0;
    files = new_dllist();//doubly linked list
```

```
for (de = readdir(d); de != NULL; de = readdir(d)) {
    dll_append(files, new_jval_s(strdup(de->d_name)));
    if (strlen(de->d_name) > maxlen) maxlen = strlen(de->d_name);
}
closedir(d);
```

```
dll_traverse(tmp, files) {
    exists = stat(tmp->val.s, &buf);
    if (exists < 0) {
        fprintf(stderr, "%s not found\\n", tmp->val.s);
    } else {
        printf("%*s %10ld\\n", -maxlen, tmp->val.s, buf.st_size);
    }
}
```

Örnek 4



```
File Edit View Search Tools Documents Help
Open Save Undo
dllist.h x
#define dll_nil(d) (d)

#define dll_traverse(ptr, list) \
    for (ptr = list->fblink; ptr != list; ptr = ptr->fblink)
#define dll_rtraverse(ptr, list) \
    for (ptr = list->blink; ptr != list; ptr = ptr->blink)

#endif
C/C++/ObjC Header Tab Width: 8 Ln 63, Col 30 INS
```

- `dll_traverse` makrosundaki for döngüsü ile `ls4.c` içerisinde kullanılan aşağıdaki parça tekrarlı olarak çağırılır. Bağlı listeye eklenen her eleman için `stat` ile dosya durum bilgileri alınır ve ekrana yazdırılır.

```
dll_traverse(tmp, files) {
    exists = stat(tmp->val.s, &buf);
    if (exists < 0) {
        fprintf(stderr, "%s not found\n", tmp->val.s);
    } else {
        printf("%*s %10ld\n", -maxlen, tmp->val.s, buf.st_size);
    }
}
```

Örnek 4:

- ls4.c programını derlerken dlist.h dosyasını kullanmak için derleme anında libfdr.a arşiv dosyasının kullanılması gerekir.
- ls4.c'yi derlemek için make ile libfdr.a arşiv dosyası oluşturulur
- ls4.c için object dosya ls4.o oluşturulur
- **gcc -c ls4.c -I libfdr**

ls4.o ve libfdr.a birlikte derlenir

- **gcc -o ls4 ls4.o libfdr/libfdr.a**

Örnek 5:

- Bu örnekte bağlı liste yerine red-black tree kullanılmıştır.
- RB tree ile ağaç elemanları sıralanarak kaydedilir.

```
#include "jrb.h"
main(int argc, char **argv)
{
    struct stat buf;
    int exists;
    DIR *d;
    struct dirent *de;
    JRB files, tmp;
    int maxlen;

    d = opendir(".");
    if (d == NULL) {
        fprintf(stderr, "Couldn't open \".\"\\n");
        exit(1);
    }
    maxlen = 0;
    files = make_jrb();

    for (de = readdir(d); de != NULL; de = readdir(d)) {
        jrb_insert_str(files, strdup(de->d_name), JNULL);
        if (strlen(de->d_name) > maxlen) maxlen = strlen(de->d_name);
    }
    closedir(d);

    jrb_traverse(tmp, files) {
        exists = stat(tmp->key.s, &buf);
        if (exists < 0) {
            fprintf(stderr, "%s not found\\n", tmp->key.s);
        } else {
            printf("%*s %10ld\\n", -maxlen, tmp->key.s, buf.st_size);
        }
    }
}
```

Örnek 5:

Dosya adına benzer şekilde, dosya boyutunu ekrana kaç rakam ile yazdırılacağı da max size ile belirlenmiştir.

Bu amaçla sprintf ile dosya boyutu ssize ile belirtilen değişkene string olarak yazdırılıp strlen(ssize) ile string uzunluğu ölçülmüştür.

```
main(int argc, char **argv)
{
    struct stat buf;
    int exists;
    DIR *d;
    struct dirent *de;
    JRB files, tmp;
    int maxlen;
    int maxsize;
    char ssize[20];

    d = opendir(".");
    if (d == NULL) {
        fprintf(stderr, "Couldn't open \".\"\\n");
        exit(1);
    }
    maxlen = 0;
    maxsize = 0;
    files = make_jrb();
    for (de = readdir(d); de != NULL; de = readdir(d)) {
        if (strlen(de->d_name) > maxlen) maxlen = strlen(de->d_name);
        exists = stat(de->d_name, &buf);
        if (exists < 0) {
            fprintf(stderr, "%s not found\\n", de->d_name);
        } else {
            sprintf(ssize, "%ld", buf.st_size);
            if (strlen(ssize) > maxsize) maxsize = strlen(ssize);
            jrb_insert_str(files, strdup(de->d_name), new_jval_s(strdup(ssize)));
        }
    }
    closedir(d);
    jrb_traverse(tmp, files) {
        printf("%s %s\\n", tmp->key.s, tmp->val.s);
    }
}
```

Örnek 5:

```
File Edit View Search Terminal Help

10_stat: ./ls5a
.          4096
..         4096
compile.txt~ 0
deneme      14
deneme2     30
derle.txt   57
f1          7
f2          7
lecture.html 12347
libfdr      4096
ls1         8703
ls1.c       617
```

Örnek 6:

- The following POSIX macros are defined to check the file type using the `st_mode` field:
- `S_ISREG(m)` is it a regular file?
- `S_ISDIR(m)` directory?
- `S_ISCHR(m)` character device?
- `S_ISBLK(m)` block device?
- `S_ISFIFO(m)` FIFO (named pipe)?
- `S_ISLNK(m)` symbolic link? (Not in POSIX.1-1996.)
- `S_ISSOCK(m)` socket? (Not in POSIX.1-1996.)

Örnek 6:

■ The following flags are defined for the `st_mode` field:

■	<code>S_IFMT</code>	<code>0170000</code>	bit mask for the file type bit fields
■	<code>S_IFSOCK</code>	<code>0140000</code>	socket
■	<code>S_IFLNK</code>	<code>0120000</code>	symbolic link
■	<code>S_IFREG</code>	<code>0100000</code>	regular file
■	<code>S_IFBLK</code>	<code>0060000</code>	block device
■	<code>S_IFDIR</code>	<code>0040000</code>	directory
■	<code>S_IFCHR</code>	<code>0020000</code>	character device
■	<code>S_IFIFO</code>	<code>0010000</code>	FIFO
■	<code>S_ISUID</code>	<code>0004000</code>	set-user-ID bit
■	<code>S_ISGID</code>	<code>0002000</code>	set-group-ID bit (see below)
■	<code>S_ISVTX</code>	<code>0001000</code>	sticky bit (see below)
■	<code>S_IRWXU</code>	<code>00700</code>	mask for file owner permissions
■	<code>S_IRUSR</code>	<code>00400</code>	owner has read permission
■	<code>S_IWUSR</code>	<code>00200</code>	owner has write permission
■	<code>S_IXUSR</code>	<code>00100</code>	owner has execute permission
■	<code>S_IRWXG</code>	<code>00070</code>	mask for group permissions
■	<code>S_IRGRP</code>	<code>00040</code>	group has read permission
■	<code>S_IWGRP</code>	<code>00020</code>	group has write permission
■	<code>S_IXGRP</code>	<code>00010</code>	group has execute permission
■	<code>S_IRWXO</code>	<code>00007</code>	mask for permissions for others (not in group)
■	<code>S_IROTH</code>	<code>00004</code>	others have read permission
■	<code>S_IWOTH</code>	<code>00002</code>	others have write permission
■	<code>S_IXOTH</code>	<code>00001</code>	others have execute permission

Örnek 6:

- Bu örnekte ls -F ile benzer çalışacak şekilde bir listeleme yapılmıştır.
- Bunun için listelenen elemanın sonuna semboller eklenmiştir.
- Klasörlerin sonuna: "/"
- symbolic (soft) link sonuna: "@"
- Executable sonuna: "*"

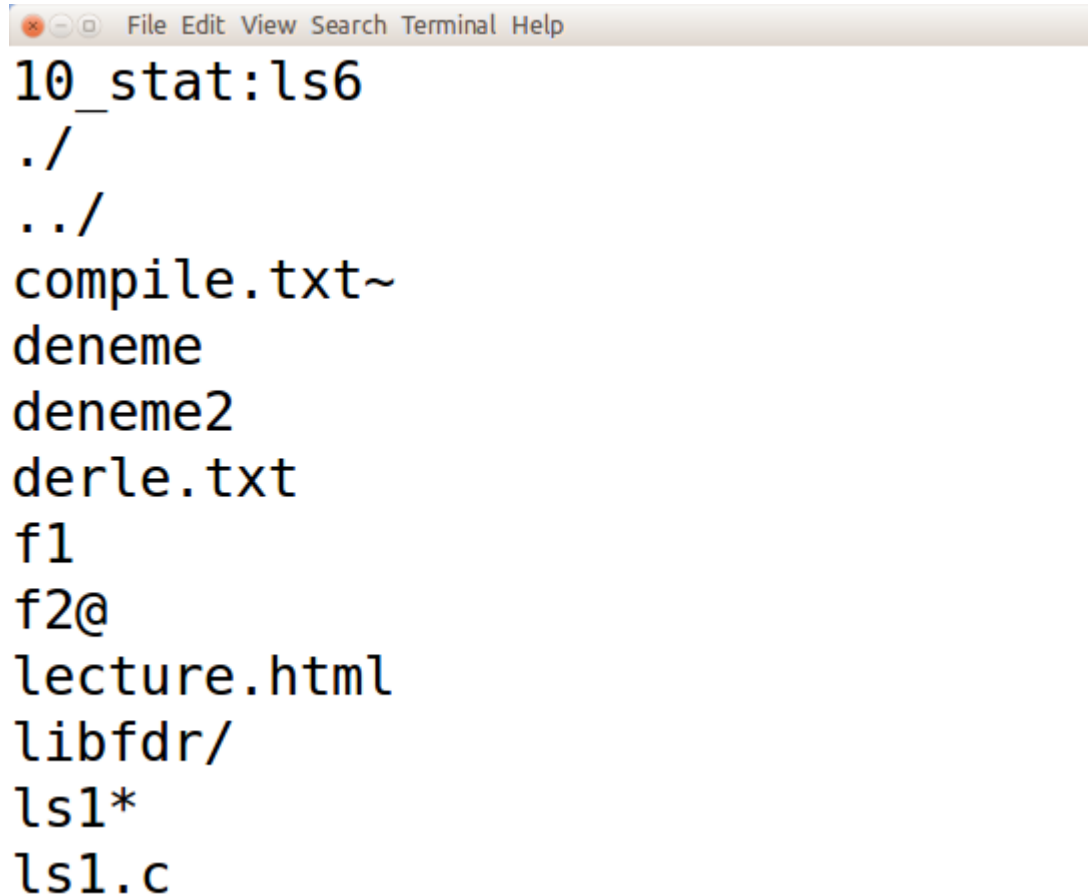
```
main(int argc, char **argv)
{
    struct stat buf;
    int exists;
    DIR *d;
    struct dirent *de;
    JRB r, tmp;
    char *fn;

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

    for (de = readdir(d); de != NULL; de = readdir(d)) {
        fn = strdup(de->d_name);
        jrb_insert_str(r, fn, JNULL);
    }
    closedir(d);

    jrb_traverse(tmp, r) {
        exists = lstat(tmp->key.s, &buf);
        if (exists < 0) {
            fprintf(stderr, "%s not found\\n", tmp->key.s);
        } else if (S_ISDIR(buf.st_mode)) {
            printf("%s/\\n", tmp->key.s);
        } else if (S_ISLNK(buf.st_mode)) {
            printf("%s@\\n", tmp->key.s);
        } else if (buf.st_mode & (S_IXUSR | S_IXGRP | S_IXOTH)) {
            printf("%s*\\n", tmp->key.s);
        } else {
            printf("%s\\n", tmp->key.s);
        }
    }
}
```

Örnek 6:



A screenshot of a terminal window with a title bar containing 'File Edit View Search Terminal Help'. The terminal displays the output of the command 'ls' executed in a directory named '10_stat'. The output lists the following files and directories: '.', '..', 'compile.txt~', 'deneme', 'deneme2', 'derle.txt', 'f1', 'f2@', 'lecture.html', 'libfdr/', 'ls1*', and 'ls1.c'. A vertical orange line is positioned to the right of the terminal output.

```
10_stat:ls6
./
../
compile.txt~
deneme
deneme2
derle.txt
f1
f2@
lecture.html
libfdr/
ls1*
ls1.c
```