SATBAYEV UNIVERSITY

# Компьютер өнімділігін арттыру әдістері. Қазіргі заманғы процессорлар архитектурасы және нұсқаулар жиыны

# Файлдардағы операциялар

# Файлдардағы операциялар


1. unlink() жүйелік шақыруы арқылы файлды жою.
2. Файл индекстері, каталогтар, сілтемелер.
3. Жүйелік қоңырау арқылы файлдың атын өзгерту және жылжыту
4. Қатты және символдық сілтемелер жасау
5. Каталогтарды жасау және жою
6. Рұқсаттар маскасы және umask() жүйелік қоңырауы.

# 16.1.unlink() жүйелік шақыруы арқылы файлды жою.


- Файлды жою үшін unistd.h тақырып файлында төмендегідей жарияланған unlink() жүйелік шақыруын пайдаланыңыз:
- int ажырату(const char * FNAME);
- FNAME аргументі - жойылатын файлдың аты. unlink() сәтті болған кезде 0 мәнін қайтарады
- аяқтау. Қате туралы -1 қайтарады

- ЕСКЕРТУ
- Файлдық жүйені іске асырудың орта деңгейінде каталогтар ретінде қарастырылады
- файлдар, бірақ оларды жою үшін rmdir() жүйелік шақыруы пайдаланылады, ол туралы
- бөлімінде талқыланады 16.5.

# unlink() жүйелік шақыруы арқылы файлды жою


- Бір қарағанда, бәрі қарапайым және түсінікті болып көрінеді. Бірақ неге деген сұрақ туындайды
- UNIX жасаушылар бұл жүйені unlink() деп атайды, емес
- мысалы, remove() немесе delete()? Мәселе мынада, бұл файл күрделі
- мынадай құрамдас бөліктерден тұратын тұжырымдама:
- ˆ деректер (деректер);
- ˆ индекстер (инодтар);
- ˆ сілтемелер.
- Нақты файлдық жүйелер әдетте блоктық құрылғыларда орналасады. Мұндай құрылғылардағы деректер шын мәнінде бекітілген блоктарда адрестеледі
- жеке байттарға қарағанда өлшем.

**Листинг 16.1. Программа unlink1.c**

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



```
int main (int argc, char ** argv)
{
        if (argc < 2) {
                fprintf (stderr, "Too few arguments\n");
                return 1;
        }

        if (unlink (argv[1]) == -1) {
                fprintf (stderr, "Cannot unlink file "
                                 "(%s)\n", argv[1]);
                return 1;
        }

        return 0;
}
```

- Сілтемелер - бұл файлдық жүйедегі файлдардың бізге таныс көрінісі. мүмкін
- шарт бойынша, сілтемелер файлдық жүйенің инодтарының атаулары болып табылады. Каталогтарды инодтарға сілтемелерді сақтайтын арнайы файлдар ретінде қарастыруға болады.
- -i жалауымен шақырылған ls бағдарламасы inode нөмірін көруге мүмкіндік береді
- бұл сілтемеге сілтеме жасайды. Кішкене эксперимент жасайық:

```
$ mkdir idemo
$ cd idemo
$ touch file1
$ touch file2
```

*Глава 16. Операции над файлами*

```
$ ls -i
952139 file1  952141 file2
```

Сонымен, жоғарыда келтірілген мысал file1 952139 инод нөміріне сілтеме екенін көрсетеді (сіздің нұсқаңыз басқаша болуы мүмкін). Ал файл2 сілтемесін көрсетеді
952141 саны бар басқа индекстік түйін.
Тәжірибені жалғастырайық:

```
$ ln -s file1 symlnkf1
$ ls -i
952139 file1  952141 file2  952190 symlnkf1
```

- ls бағдарламасының шығысы символдық сілтеме де 952190 инод нөміріне сілтеме екенін көрсетеді. Символдық сілтемелер инодты емес, файл атауын көрсететінін түсіну маңызды.
- Енді -s жалауынсыз ln командасын қолданайық:


Символдықпен қатар қиынның да барын білетін шығарсыз
сілтемелер. hardfile1 – файл1-ге қатты сілтеме.
ls пәрменінің шығысы
file1 және hardfile1 саны бар бір инодты көрсететінін көрсетеді
952139 Қатты сілтемелер (символдық сілтемелерден айырмашылығы) іс жүзінде емес
бағыныңқы кейіпкер. Сондықтан, файл1 және қатты файл1 аяқталды
бірдей инодқа сілтемелер

```
$ ln file1 hardfile1
$ ls -i
952139 file1  952141 file2  952139 hardfile1  952190 symlnkf1
```

- Индекстер – деректерге сілтемені байланыстыратын аралық сілтеме
- блоктау құрылғысы. Егер екі сілтеме бір индексті көрсетсе, онда олар бірдей деректерге қол жеткізеді деуге болады. Мұны тексеру оңай:

```
$ echo hello > file1
$ cat hardfile1
hello
```

Енді rm бағдарламасы келесідей жұмыс істейді деп айта аламыз:
ˆ егер жойылатын файл файлдық жүйедегі сәйкес инодтың соңғы сілтемесі болса, онда деректер мен инод босатылады;
ˆ файлдық жүйеде әлі де сәйкес инодқа сілтемелер болса
түйін, тек сілтеме жойылады.

```
$ rm file1
$ cat hardfile1
hello
```

Оны тексеріп көрейік:

```
$ rm file1
$ cat hardfile1
hello
```

Теперь обратите внимание на вывод программы ls с флагом -l:

```
$ ls -l
-rw-r--r-- 1 nnivanov nnivanov 0 2011-05-07 10:00 file2
-rw-r--r-- 1 nnivanov nnivanov 6 2011-05-07 10:00 hardfile1
lrwxrwxrwx 1 nnivanov nnivanov 5 2011-05-07 10:00 symlnkf1 -> file1
```

- Енді сектадан statvfsdemo бағдарламасына көшейік. 14.1 (14.1 листинг). Бұл бағдарлама орнатылған файлдық жүйелер туралы ақпаратты басып шығару үшін statvfs() функциясын пайдаланды. statvfs құрылымында келесі өрістер де бар:
- біз оны 14-тарауда қарастырмадық:
- ^ f_files – берілген файлдық жүйе үшін инодтардың жалпы саны;
- ^ f_free – бос файлдық жүйе инодтарының саны;
- ^ f_favail – файлдық жүйедегі қолжетімді инодтар саны.
- 16-2 листингінде statvfsdemo бағдарламасының жақсартылған нұсқасы бар
- сонымен қатар әрбір файлдық жүй үшін инодтар туралы ақпарат

Теперь во втором столбце вывода ls напротив `file2` и `hardfile2` находится число 2, символизирующее о том, что на соответствующий индексный узел указывают две ссылки.

Если вызвать команду `df` с флагом `-i`, то на экран будет выведена информация по индексным узлам смонтированных файловых систем:

```
$ df -i
Filesystem      Inodes   IUsed   IFree IUse% Mounted on
/dev/sda6      1311552  264492 1047060   21% /
udev             96028     487   95541    1% /dev
/dev/sda1        66264      58   66206    1% /boot
/dev/sda7      3407872  149600 3258272    5% /home
```

ЕСКЕРТУ

Жүйедегі кейбір бағдарлама файлдарды бір уақытта жасап немесе жойса, жоғарыдағы мысалдар дұрыс жұмыс істемеуі мүмкін екенін ескеріңіз.

Итак, каждый раз при создании файла в файловой системе выделяется индекс:

```
$ df -i .
Filesystem      Inodes   IUsed   IFree IUse% Mounted on
/dev/sda7      3407872  149586 3258286    5% /home
$ touch file3
$ df -i .
Filesystem      Inodes   IUsed   IFree IUse% Mounted on
/dev/sda7      3407872  149587 3258285    5% /home
```

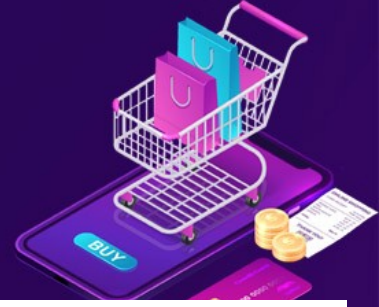
А создание жесткой ссылки не приводит к появлению в файловой системе нового индекса:

```
$ df -i .
Filesystem      Inodes   IUsed   IFree IUse% Mounted on
/dev/sda7      3407872  149587 3258285    5% /home
$ ln file3 hardfile3
```



```
$ df -i .
Filesystem      Inodes   IUsed   IFree IUse% Mounted on
/dev/sda7      3407872  149587 3258285    5% /home
```

# Листинг 16.2. Программа statvfsinode.c




- Секцияның stat1 бағдарламасын да қарастырыңыз. 15.5 (Листинг 15.5). статистикалық құрылым
- құрамында инод түйінінің нөмірін қамтитын басқа st_ino өрісі бар
- ол файлға сілтеме жасайды. 16.3 листингінде бағдарламаның жетілдірілген нұсқасы бар
- stat1.

```
#include <sys/statvfs.h>
#include <mntent.h>
#include <stdio.h>

int main (void)
{
        struct mntent * entry;
        struct statvfs fs;
        FILE * file;

        file = setmntent ("/etc/mtab", "r");
        if (file == NULL) {
                fprintf (stderr, "Cannot open /etc/mtab\n");
                return 1;
        }

        while ((entry = getmntent (file)) != NULL) {
                if (statvfs (entry->mnt_dir, &fs) == -1) {
                        fprintf (stderr, "statvfs() error\n");
                        return 1;
                }

                printf ("Filesystem: %s\n", entry->mnt_fsname);
                printf ("Mountpoint: %s\n", entry->mnt_dir);
                printf ("Block size: %ld\n", (unsigned long int)
                                fs.f_bsize);
```

222 Часть IV. Файловая систе

```
                printf ("Blocks: %ld\n", (unsigned long int)
                                fs.f_blocks);
                printf ("Blocks free: %ld\n", (unsigned long int)
                                fs.f_bfree);
                printf ("Blocks available: %ld\n",
                                (unsigned long int) fs.f_bavail);
                printf ("Max. filename length: %ld\n",
                                (unsigned long int) fs.f_namemax);

                printf ("Inodes: %ld\n",
                                (unsigned long int) fs.f_files);
                printf ("Inodes free: %ld\n",
                                (unsigned long int) fs.f_ffree);
                printf ("Inodes available: %ld\n",
                                (unsigned long int) fs.f_favail);

                printf ("---\n");
        }

        endmntent (file);
        return 0;
}
```

# Листинг 16.3. Программа statinode.c

- Unlink1 бағдарламасына оралайық (16.1 листингті қараңыз). Енді біз мұны айта аламыз
- unlink() жүйелік шақыруы инодқа сілтемені жояды. Егер бұл сілтеме соңғы болса, индекс босатылады. Осы тұжырымдаманы көрсететін тағы бір мысалды қарастырайық (16.4-тізім).
- Листинг 16.4. ажырату2 мысалы.

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

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

        if (argc < 2) {
                fprintf (stderr, "Too few arguments\n");
                return 1;
        }

        if (stat (argv[1], &st) == -1) {
                fprintf (stderr, "stat() error\n");
```

*Глава 16. Операции над файлами*

```
                return 1;
        }

        printf ("FILE:\t\t%s\n", argv[1]);
        printf ("UID:\t\t%d\n", (int) st.st_uid);
        printf ("GID:\t\t%d\n", (int) st.st_gid);
        printf ("SIZE:\t\t%ld\n", (long int) st.st_size);
        printf ("AT:\t\t%s", ctime (&st.st_atime));
        printf ("MT:\t\t%s", ctime (&st.st_mtime));

        printf ("INODE:\t\t%ld\n", (long int) st.st_ino);

        return 0;
}
```

# 16.4. Пример unlink2.c

- Бұл мысалды іске қосқаннан кейін не болды:

```
$ touch file1
$ ln file1 file2
$ ls -i file1 file2
1048740 file1  1048740 file2
$ ./unlink2 file1
$ ls file1
/bin/ls: file1: No such file or directory
$ cat file2
Hello World
```

Проведем еще один эксперимент:

```
$ ./unlink2 file2
$ ls file2
/bin/ls: file2: No such file or directory
```

Бұл бағдарлама енгізу-шығару операцияларын ашық файлда сәтті орындауға болатынын көрсетеді, тіпті егер сол файлға соңғы сілтеме unlink() жүйелік шақыруы арқылы жойылса да. Біз бұл тұжырымдамаға қайта ораламыз. 18 тарауда.

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

#define HELLO_MSG       "Hello World\n"

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

        if (argc < 2) {
                fprintf (stderr, "Too few arguments\n");
                return 1;
        }

        fd = open (argv[1], O_WRONLY | O_TRUNC, 0644);
        if (fd == -1) {
                fprintf (stderr, "Cannot open file "
                                 "(%s)\n", argv[1]);
                return 1;
        }

        if (unlink (argv[1]) == -1) {
                fprintf (stderr, "Cannot unlink file "
                                 "(%s)\n", argv[1]);
```



```
                return 1;
        }

        if (write (fd, HELLO_MSG,
                   strlen (HELLO_MSG)) == -1) {
                fprintf (stderr, "write() error\n");
                return 1;
        }

        if (close (fd) == -1) {
                fprintf (stderr, "close() error\n");
                return 1;
        }

        return 0;
}
```

# 16.2. Файлды жылжыту: rename()


- rename() жүйелік шақыруы файлдың атын өзгертуге немесе жылжытуға мүмкіндік береді
- бір файлдық жүйе ішінде. stdio.h тақырып файлында ол келесідей жарияланады:
- int атын өзгерту(const char * OLDF, const char * NEWF);
- Сәтті болған кезде rename() 0 қайтарады. Қатеде -1 қайтарылады.

ЕСКЕРТУ
rename() функциясы файлда жарияланғанымен, Linux жүйелік шақыруы болып табылады
stdio.h әдетте кітапхана механизмдері орналасқан жер.
rename() іске асыру
fs/namei.c файлындағы Linux ядросының көздерінен табуға болады.
Келесі мысал (16.5 тізімі) жүйелік шақырудың қалай жұмыс істейтінін көрсетеді
атын өзгерту().

- Енді жылжытылатын файл ашық болса, rename() функциясының не істейтінін көрейік.
- Ол үшін басқа программа құрайық (16.6 листинг)


**Листинг 16.5. Пример rename1.c**

```
#include <stdio.h>

int main (int argc, char ** argv)
{
        if (argc < 3) {
                fprintf (stderr, "Too few arguments\n");
                return 1;
        }

        if (rename (argv[1], argv[2]) == -1) {
                fprintf (stderr, "rename() error\n");
                return 1;
        }

        return 0;
}
```

- Енді кішкене эксперимент жасайық:

```
$ touch file1
$ cat file1
$ ./rename2 file1 file2
$ ls file1
/bin/ls: file1: No such file or directory
$ cat file2
Hello World
```

Осылайша, біз ашық файлды жылжыту файлға әсер етпейтініне көз жеткіздік Енгізу/шығару операциялары.

**Листинг 16.6. Программа rename2.c**

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

#define HELLO_MSG       "Hello World\n"

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

        if (argc < 3) {
                fprintf (stderr, "Too few arguments\n");
                return 1;
        }
```



```
        fd = open (argv[1], O_WRONLY | O_TRUNC, 0644);
        if (fd == -1) {
                fprintf (stderr, "Cannot open file "
                                 "(%s)\n", argv[1]);
                return 1;
        }

        if (rename (argv[1], argv[2]) == -1) {
                fprintf (stderr, "rename() error\n");
                return 1;
        }

        if (write (fd, HELLO_MSG,
                        strlen (HELLO_MSG)) == -1) {
                fprintf (stderr, "write() error\n");
                return 1;
        }

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

# 16.3. Сілтемелерді жасау: link()


- Linux файлдық жүйесінде сілтемелердің екі түрі бар екенін білдік:
- ˆ символдық сілтемелер;
- ˆ қатты (тікелей) сілтемелер (қатты сілтемелер).
- Пайдаланушы ln бағдарламасы арқылы сілтемелер жасай алады. Ал бағдарламашыда link() және symlink() жүйелік шақырулары бар. Алдымен жасайды
- тікелей сілтемелер, екіншісі - символдық. Бұл жүйелік шақырулар unistd.h тақырып файлында келесідей жарияланады:
- int сілтемесі (const char * FROM, const char * TO);
- int symlink(const char * FROM, const char * TO);

Екеуі де сәттілікке 0 және сәтсіздікке -1 қайтарады. FROM (from) және TO (to) аргументтері ln бағдарламасының сәйкес бос аргументтерімен бірдей. Келесі бағдарлама (16.7 тізімі) жүйелік шақырудың қалай жұмыс істейтінін көрсетеді.
сілтеме().

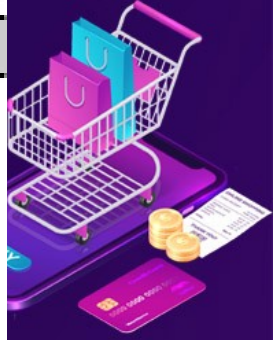
- тексеру:

```
$ touch file1
$ ./linkdemo file1 file2
$ ls -i file1 file2
1048776 file1  1048776 file2
```

**Листинг 16.7. Программа linkdemo.c**

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

int main (int argc, char ** argv)
{
        if (argc < 3) {
                fprintf (stderr, "Too few arguments\n");
                return 1;
        }

        if (link (argv[1], argv[2]) == -1) {
                fprintf (stderr, "link() error\n");
                return 1;
        }

        return 0;
}
```

symlink() жүйелік шақыруын көрсететін басқа бағдарламаны (16-8 тізім) жасайық.



**Листинг 16.8. Программа symlinkdemo.c**

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

int main (int argc, char ** argv)
{
        if (argc < 3) {
                fprintf (stderr, "Too few arguments\n");
                return 1;
        }
```



```
        if (symlink (argv[1], argv[2]) == -1) {
                fprintf (stderr, "link() error\n");
                return 1;
        }

        return 0;
}
```

- нәтиже:

```
$ touch file1
$ ./symlinkdemo file1 file2
$ ls -li file1 file2
1048779 -rw-r--r-- 1 nnivanov nnivanov 0 2011-05-07 10:03 file1
1048780 lrwxrwxrwx 1 nnivanov nnivanov 5 2011-05-07 10:03 file2 -> file1
```

# 16.4. Каталог жасау: mkdir()

- Каталогты құру үшін жарияланған mkdir() жүйелік шақыруы қолданылады
- sys/stat.h тақырып файлында келесідей:
- int mkdir(const char * NAME, mode_t MODE);
- mkdir() жүйелік шақыруы MODE режимі бар NAME атты каталогты жасайды. Сағат
- Сәтті болса, mkdir() 0 қайтарады. Қатеде -1 қайтарылады.
- Келесі мысал (16.9 тізімі) жүйелік шақырудың қалай жұмыс істейтінін көрсетеді.
- mkdir().

**Листинг 16.9. Пример mkdir1.c**

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

int main (int argc, char ** argv)
{
        if (argc < 2) {
                fprintf (stderr, "Too few arguments\n");
                return 1;
        }

        if (mkdir (argv[1], 0755) == -1) {
                fprintf (stderr, "mkdir() error\n");
                return 0;
        }

        return 0;
}
```

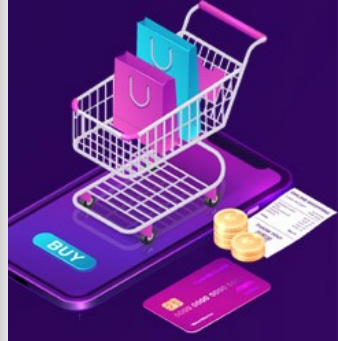
Жоғарыда келтірілген мысал mkdir бағдарламасының жеңілдетілген нұсқасы болып табылады

```
$ ./mkdir1 mydir
$ ls -l | grep mydir
drwx------ 2 nnivanov nnivanov 4096 2011-05-07 10:05 mydir
```

**Листинг 16.10. Пример mkdir2.c**

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

int main (int argc, char ** argv)
{
        mode_t mode = 0777;

        if (argc < 2) {
                fprintf (stderr, "Too few arguments\n");
                return 1;
        }

        if (mkdir (argv[1], mode) == -1) {
                fprintf (stderr, "mkdir() error\n");
                return 0;
        }
        return
}
```

```
$ ./mkdir2 mydir
$ ls -l | grep mydir
drwxr-xr-x 2 nnivanov nnivanov 4096 2011-05-07 10:09 mydir
```




- Бұл бағдарлама біз күткендей жұмыс істемейді:

```
$ ./mkdir2 mydir
$ ls -l | grep mydir
drwxr-xr-x 2 nnivanov nnivanov 4096 2011-05-07 10:09 mydir
```


- Режим аргументі mkdir() жүйелік шақыруына жіберілді, онда базаның барлық биттері бар.
- рұқсаттар біреуге орнатылған. Бірақ ls бағдарламасының шығысы жасалған каталогта 0755 рұқсаты бар екенін көрсетеді (сізде басқа нәрсе болуы мүмкін). Мұның неліктен болғанын қарастырайық.
- Linux жүйесіндегі әрбір процесте мұраланған рұқсаттар маскасы бар.
- ата-аналық процестегі бала (ағымдағы каталогқа, ортаға ұқсас
- және т.б.). Рұқсат маскасы рұқсат биттерінің жиынын көрсететін сан болып табылады
- процесс жасау үшін ешқашан орнатылмайтын рұқсаттар
- файлдар немесе каталогтар. Umask пәрмені қабықтың ағымдағы рұқсат маскасын білуге мүмкіндік береді:
- $ umask
- 0022

Сонымен, 0022 рұқсат маскасы файлдарды немесе каталогтарды жасау кезінде иесіне кез келген рұқсаттарды орнатуға мүмкіндік береді, бірақ жазу рұқсаттарын бермейді.

топтар және басқа пайдаланушылар.

ЕСКЕРТУ

umask - ішкі bash қабықшасының пәрмені. Көптеген басқа қабықтар (csh, ksh т.б.) олардың арсеналында осы пәрменді де қамтиды.

Ағымдағы процесс рұқсат маскасының көшірмесін өзгертуге тегін. Мысалға, bash қабығы мұны бірдей umask пәрменімен жасайды:

```
$ umask 0044
$ umask
0044
```

Еншілес процестер ата-ананың рұқсат маскасының көшірмесін иеленеді. Мұны тексеру оңай:

```
$ umask 000
$ umask
0000
$ bash
$ umask
0000
$ exit
exit
```


Енді мысалға оралайық (16.10 листинг). Жүгіруге тырысайық mkdir2 бағдарламасы қабық рұқсаттарының маскасы өзгертілген:

```
$ umask 0000
$ umask
0000
$ ./mkdir2 mydir
$ ls -l | grep mydir
drwxrwxrwx 2 nnivanov nnivanov 4096 2011-05-07 10:15 mydir
```

Бағдарлама ағымдағы процестің рұқсат маскасын өзгерте алады umask() жүйелік шақыруы, ол /sys/stat.h тақырып файлында төмендегідей жарияланады:

```
mode_t umask (mode_t MASK);
```

Бұл жүйелік қоңырау ағымдағы рұқсаттар маскасын өзгертеді және алдыңғы бүркеніш мәнін қайтарады. Енді 0777 рұқсатымен каталог жасау үшін umask() жүйелік шақыруын пайдалану мысалын қарастырайық (16-11 тізім).

- Міне, осы мысалды іске қосудың нәтижесі:

```
$ umask 0022
$ umask
0022
$ ./mkdir3 mydir
$ ls -l | grep mydir
drwxrwxrwx 2 nnivanov nnivanov 4096 2011-05-07 10:17 mydir
```

ЕСКЕРТУ
mydir каталогы үшін ls пәрменінің шығысында сілтеме саны екенін ескеріңіз (екінші бағаң) 2. Бұл екінші сілтеме mydir каталогының өзінде,
нүктемен (.) белгіленеді және ағымдағы каталогты көрсетеді. Бүкіл құпия осында

**Листинг 16.11. Пример mkdir3.c**

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

*Глава 16. Операции над файлами*

```
#include <sys/types.h>

int main (int argc, char ** argv)
{
        mode_t mode = 0777;

        if (argc < 2) {
                fprintf (stderr, "Too few arguments\n");
                return 1;
        }

        umask (0000);
        if (mkdir (argv[1], mode) == -1) {
                fprintf (stderr, "mkdir() error\n");
                return 0;
        }

        return 0;
}
```

Кеңейтілген рұқсаттарды каталогтарды жасау кезінде де пайдалануға болады.
Келесі мысал (16-12 тізім) жабысқақ каталогты қалай жасау керектігін көрсетеді

**Листинг 16.12. Пример mkdir4.c**

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

int main (int argc, char ** argv)
{
        mode_t mode = 0777 | S_ISVTX;
```

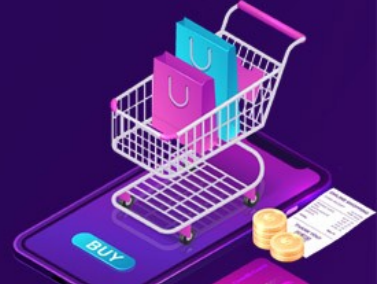


```
        if (argc < 2) {
                fprintf (stderr, "Too few arguments\n");
                return 1;
        }

        umask (0000);
        if (mkdir (argv[1], mode) == -1) {
                fprintf (stderr, "mkdir() error\n");
                return 0;
        }

        return 0;
}
```


- тексеру

```
$ ./mkdir4 mydir
$ ls -l | grep mydir
drwxrwxrwt 2 nnivanov nnivanov 4096 2011-05-07 10:20 mydir
```

# 6.5. Каталогты жою: rmdir()




- rmdir() жүйелік шақыруы бос каталогтарды ғана жояды.
- Егер каталог бос болмаса, rmdir() сәтсіз болады. Оны тексеріп көрейік:

**Листинг 16.13. Пример rmdirdemo.c**

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

int main (int argc, char ** argv)
{
        if (argc < 2) {
                fprintf (stderr, "Too few arguments\n");
                return 1;
        }

        if (rmdir (argv[1]) == -1) {
                fprintf (stderr, "rmdir() error\n");
                return 1;
        }

        return 0;
}
```

```
$ mkdir mydir
$ touch mydir/myfile
$ ./rmdirdemo mydir
rmdir() error
$ rm mydir/myfile
$ ./rmdirdemo mydir
$ ls mydir
/bin/ls: mydir: No such file or directory
```