ТЕМА 6. УПРАВЛЕНИЕ НА ФАЙЛОВАТА СИСТЕМА В UNIX

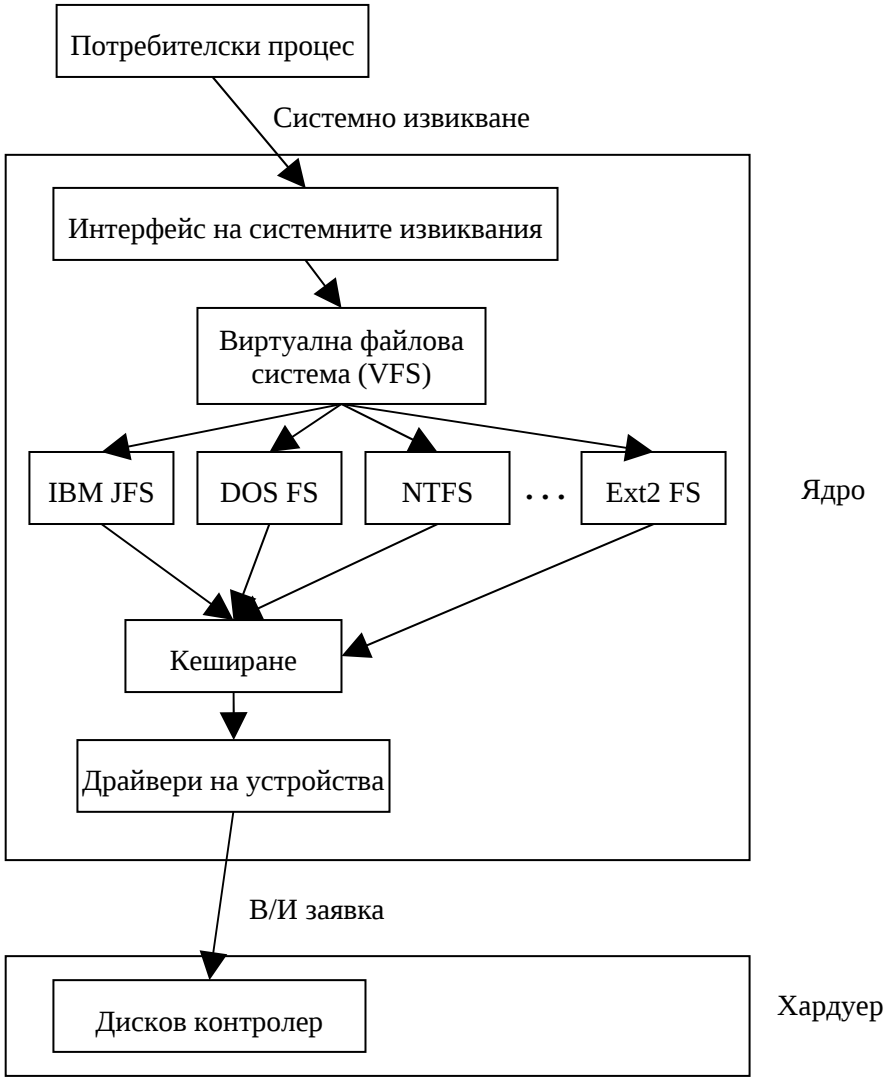
1. Виртуална файлова система

По настоящем съществуват много и различни файлови системи, всяка със своя собствена архитектура (различни методи за разполагане на файла на физическия носител, различна организация на директориите и т.н.). Съвременните версии на UNIX-подобните операционни системи са способни да използват множество файлови системи, различаващи се по своята структура и организация. Това се реализира посредством използване на една абстрактна файлова система, наречена виртуална файлова система (VFS), която предоставя на ядрото и потребителските процеси единен интерфейс за работа с файлове, независимо от конкретната файлова система върху дисковото устройство (фиг. 2.1).

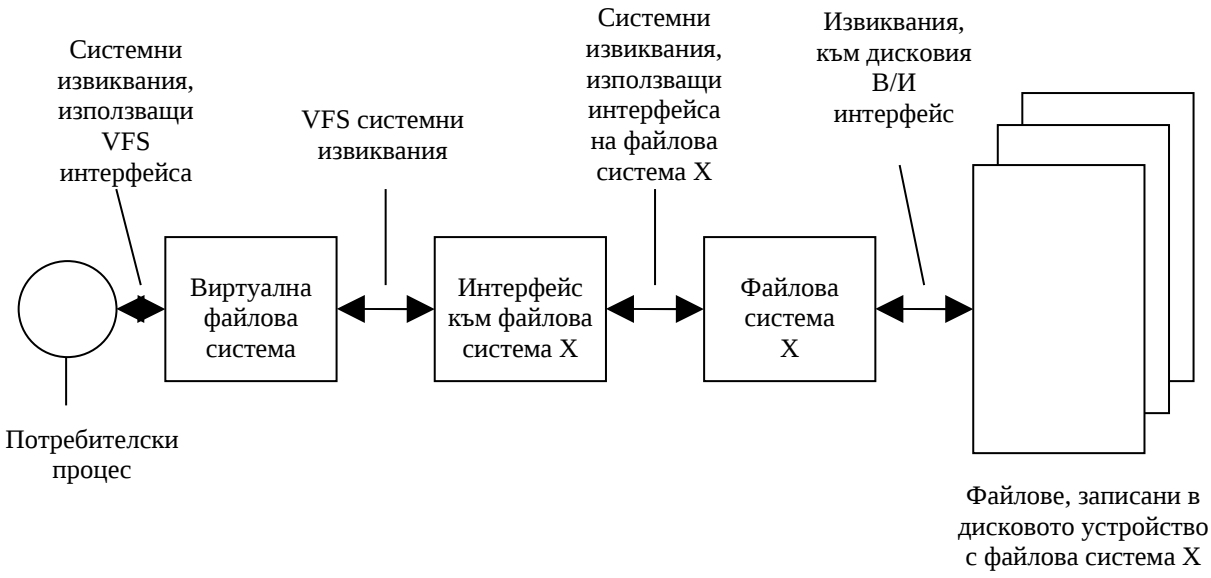
Виртуалната файлова система дефинира един обобщен файлов модел, който представя файловете като обекти, разпложени в който и да е дисков дял, коит притежават общи базови характеристики и поведение, независимо от конкретната файлова система или запомнящ хардуер.

В този модел файловете се идентифицират посредством символни имена, които са уникални за определяна директория във файловата система. Файлът, също така има собственик, защита срещу неоторизиран достъп или модифициране и мнжество други характеристики. Един файл може да бъде създаен, да бъде четено от него, да се записва във файла или да бъде изтрит. За всяка конкретна файлова система е необходимо в ядрото да се зареди интерфейсен модул, който да преобразува характеристиките на реалната файлова система в характеристиките, които се очакват от виртуалната файлова система (фиг. 2.2).

Интерфейсните модули за различните файлови системи могат да се зареждта в ядрото на етапа на компилацята или да се добавят динамично при необходимост без перекомпиляция на системата (както в система с микроядерна архитектура).



Фиг. 2.1. Място на виртуалната файлова система в Linux ядрото



Фиг. 2.2. Роля на виртуалната файлова система в ядрото на Linux

Във виртуалната файлова система данните за физическото разположение на отворените файлове и техните атрибутри са представени в операционната система в структура от данни, наречена таблица на виртуалните индексни възли на отворените файлове. Всеки виртуален индексен възел в тази таблица характеризира даден файл, както и индексня възел в традиционните UNIX файлови системи. Заедно с таблицата на индексните възли на отворените файлове в операционната система се подържа и таблица на виртуалните индексни възлли на отворените файлове. При отваряне на файл за него се добавя информация в елемент от тази таблица (ако тази информация не е била добавена преди това). Информацията в еин такъв елемент съдържа поне: типа на файла, брояч за броя на отваряниятa на файла, указател към реалните физически данни на файла и задължително, указател към таблица системните извиквания, чрез които се извършват операции над файла – таблица на операците. Реалните данни на файла на диска и системните изивквания, които реално изпълняват операциите над файла, не са елементи на виртуалната файлова система. Те са част от конкретната файлова система и интерфейсния модул за работа с нея.

За да се изпълни операция над отворен файл се използва таблицата на операциите, чийто адрес се съдържа във *vnode*, за да се определи системното извикване, което в действителност ще бъде изпълнено над реалните физически данни на файла, адреса на тези данно също се намира в *vnode*. Всички файлове принаделжащи на една файлова система имат обща таблицата на операциите.

2. Операции над файловите системи. Монтиране на файлови системи

Наличието на няколко файлови системи в рамките на една операционна система поставя въпрост за логическото обединение на техните структури. Първоначално в опреационната система е достъпна само една файлова система, наречена коренова файлова система. Преди да може да се работи с файловете, намиращи се в някоя друга файлова система е необходимо нейното присъединявне към вече същесвуващия ацикличен граф на файловете (дървото на директориите). Това

присъединяване на файлова система към съществуващата директориина структура се нарича монтиране на файлова система (mount).

За монтиране на файлова система в съществуващото дърво на директориите е необходимо да се използва подходяща празна директория – точката на монтиране, към която се присъединява корена (кореновата директория) на монтираната файлова система. При монтирането в ядрото се създава структура от данни, описваща файловата система, а в *vnode* се помества информация за точката на монтиране на файловата система.

Монтирането на файлови системи се извършва или ръчно с командата *mount*, или автоматично при стартиране на операционната система. Ако тази команда се използва без параметри потребителя ще получи информация за всички монтирани файлови системи и съответстващите им физически устройства. Потребителите, обикновено имат права за монтиране на файлови системи разположени върху дискетим оптични дискове и флаш памети. Например, командата за монтиране на файловта система на дискета поставена във флопидисковото устройстово дискета е следната:

mount /dev/fd0 <име на празна директория>

където <име на празна директория> описва точката на монтиране, а /dev/fd0 е спеиален файл на устройството, съответстващо на първото флопидисково устройство.

ИМЕ

mount

СИНТАКСИС

```
mount [-hV]
mount [-rw] [-t fstype] device dir
```

ОПИСАНИЕ

Командата mount е предназначена за изпълнение на операцията монтиране на файлова система, както и за получаване на информация за текущо монтираните файлови системи.
Опциите -h, -V се прилагат при извикване на командата без параметри и служат за следното:
-h – извежда кратка информация за използването на командата;
-V – извежда информация за версията на командата mount;
Командата mount без опции и без параметри извежда информация за всички текущо монтирани файлови системи.
Параметърът device задава името на специалния файл на устройството, съдържащо файловата система.
Параметърът dir задава името на точката на монтиране. Точката на монтиране е съществуваща директория, която трябва да е празна. При монтирането могат да се използват следните опции:
-r — монтиране на файловата система само за четене (read only);
-w — монтиране на файловата система за четене и за запис (read/write).

```
Използва се по подразбиране;
-t fstype – задава типа на монтираната файлова система, чрез fstype.
Поддържаните типове файлови системи в операционната система Linux са: adfs,
affs, autofs, coda, coherent, cramfs, devpts, efs, ext, ext2, ext3, hfs,
hpfs, iso9660 (для CD), minix, msdos, ncpfs, nfs, ntfs, proc, qnx4, reiserfs,
romfs, smbfs, sysv, udf, ufs, umsdos, vfat, xenix, xfs, xiafs. При липса на
явно зададен тип файлова система, командата е способна сама автоматично да
разпознае повечето от изборените типове.
```

Операцията по логическо разединяване на монтирана файлова система се нарича демонтиране и се извършва с командта umount. Тази команда, както и командата *mount* може се изпълява в диалогов (ръчен) режим или автоматично при завершване на работата на операционната система. Командата за демонтиране, обикновено се използва във вида

umount <име на точката на монтиране>

където <име на точката на монтиране> е <име на празна директория>, използвано при монтирането с командата *mount*, или във вида

umount /dev/fd0

където /dev/fd0 е специален файл на устройство. В случая съответстващо на първот флопидисково устройстово.

За коректна работа на операционната система е задължително преди премахване на физическия носител на информация да се демонтира неговата файлова система.

ИМЕ

```
umount
```

СИНТАКСИС

```
umount [-hV]
umount device
umount dir
```

ОПИСАНИЕ

```
Командата umount е предназначена за изпълнение на операцията логическо
разединение на монтирани файлови системи.
Опциите -h, -V се прилагат при извикване на командата без параметри и служат
за следното:
-h – извежда кратка информация за използването на командата;
-V – извежда информация за версията на командата umount;
Като параметър на командата се задава или името на запомнящото устройство,
съдържащо файловата система – device, или името на точката на монтиране (т.е.
името на директорията, в която е монтирана файловата система) – dir.
Файловата система не може да бъде демонтиране докато, тя се използва
(състояние busy) – например, когато в нея съществуват отворени файлове, или
който и да е процес използва като работна директория директория от тази
файлова система.
```

3. Операции за промяна на логическта структура на файловата система

Логическата структура на файловата система представлява

ацикличен насочен граф, възлите на който са директориите и файловете, а ребрата са връзки към тях (упражнение 2, т. 2.2). Промяната на логическата структура на файловата система (добавяне/изтриване на възли и ребра към/от ацикличния граф) се извършва, посредством следните операции: създаване и изтриване на файлове, създаване и изтриване на директории, създаване на символни връзки, създаване и премахване на твърди връзки. При създаване на нов файл или директория се създава нов възел в логическата структура на файловата система и ново именовано ребро, което да сочи към него. Създаването на файл от тип „символна връзка“ добавя нов възел в графа със съответставщото му именувано и неименувано ребро. При създаване на твърда връзка в съществуващата структура на файловата система се добавя ново именувано ребро, без да се създава нов възел в нея. Изтриването на файлове, директории и твърди връзкаи премахва съответните възли и ребра от ацикличния граф, прдставящ файловата система

Използването на твърди връзки е един подход да се реши задачата по синхронизиране на промените в един файл, когато няколко потребителя работят съвместно с него. Твърдите връзки позволяват файловете да имат няколко имена. Тогава на един физически екземпляр с данни на диска могат да съответстват различни файлови имена, които се намират в една или няколко директории. Така, всеки потребител, който работи с общия файл използва различно име, но борави с данните на общия файл. В другия случай, когато всеки поребител работят със собствено копие на файла, което се намира в собствената му директория, задачата по синхронизиране на промените и замяна на всички копия на файла с новите версии е значително по-сложна за реализация.

От гледна точка на логическата структура на файловата система създавнето на твърда връзка (hard link) съответства на създаване на ново именувано ребро от възел, съответстващ на директория към възел, съответстващ на файл. Така файла получава допълнително име. От гледна точка на стуктурата на данните, описващи файловата система, създаването на твърда връзка води до добавяне на запис в директорията, който съдържа допълнително име за съществуващ файл и номер на неговия индексен

възел, който също е наличен. При този подход и новото и старото име на файла са абсолютно равноправни за операционната система и са взаимозаменяеми при осъществяването на всички операции с него.

Използването на твърди връзки води до възникване на два проблема:

Първият е свързан с операцията изтриване на файл. Изтриването на записа/записите за един файл в данена директория не позволява да бъде освободен логическия блок заеман от файла и неговия индексен възел, докато не стане ясно, че файла няма допълнителни имена (към неговия индексен възел не сочат връзки от тази или друга директория. В противен случай ще се наруши целостта на файловата система. За преодоляването на този проблем файлът получава още един атрибут – брояч на твърдите връзки (именованите ребра), сочещи към него, който както и оснаналите атрибути се намира в индексния възел. При създаване на файла неговия брояч получава стойност 1. При създаване на всяка следваща нова твърда връзка, сочеща към файла, брояча се увеличава с 1. Когато от дадена директория бъде изтрит някой файл, то от нейното съдържание се изтрива записа за този файл и брояча на твърдите връзки се намалява с 1. Едва след като стойността на този брояч стане равна на 0, следва освобождаване на логическите блокове и индексния възел заделени за този файл.

Вторият проблем е свързан с опасността логическата структура на файловата система да се превърне от ацикличен в цикличен граф с възможна неопределеност при тълкуването на записа в директорията с име "..". За предотвратяване на това в ОС UNIX е забранено създаването на твърди връзки към вече съществуващи директории. Така към възел от тип „директория“, не може да сочи повече от едно именувано ребро. В ОС Linux, ограниченията са още по стриктни и допълнително е забранено създаванто на твърди връзки към специалните файлове на устройства.

4. Създаване на твърди и символни връзки. Системни извиквания link() и symlink()

За създаване на твърда връзка се използва командата на ОС ln без опции и системното извикване link().

Системното извикване link() работи с файла без да е необходимо файлът предварително да бъде отворен, тъй като се работи със

съдържанието на индексния възел, който е отделен от съдържанието на файла.

ИМЕ

```
ln
```

СИНТАКСИС

```
ln [options] source [dest]
ln [options] source ... directory
```

ОПИСАНИЕ

Командата ln е предназначена за реализация на операцията създаване на твърда или символна връзка във файловата система. Изпълнена без опции, командата създава твърда връзка. За създаване на символна връзка се добавя опцията –s.
Синтаксиса на командата има два варианта. При първият вариант чрез параметъра source се задава името на файла към който да сочи връзката. Параметърът dest задава името на връзката (пълно или относително). Ако параметъра dest отсъства новата връзка се създава в текущата директория със същото име като на указания, чрез параметъра source файл.
При вторият вариант на командата, чрез параметра source се задава името на един или няколко файла, разделени с интервали, а чрез параметъра directory се задава име на съществуваща във файловата система директория. Така се създават връзки към всички файлове, изброени в параметъра source, в директорията directory. Имената на връзките съвпадат с имента на изборените файлове.

Забележка: Във всички версии на ОС UNIX е забранено създаването на твърди връзки към директории. В операционната система Linux създаването на твърди връзки е забранено и към специалните файлове на устройства.

ИМЕ

```
link
```

ПРОТОТИП

```
#include <unistd.h>
int link(char *pathname, char *linkpathname);
```

ОПИСАНИЕ

Системното изивкване link служи за създаване на твърда връзка към файл зададен с параметъра pathname. Името на създаваната връзка се задава с параметъра linkpathname (абсолютно или относително име на връзката). Във всички съствуващи реализации на ОС UNIX е забранено създаването ан твърди връзки към директории. В ОС Linux допълнително е забранено създаването на твърди връзки към файлове на устройства.

ВРЪЩАНИ СТОЙНОСТИ

Системното извикване връща стойност 0 при нормално завършване и стойност -1 при възникване на грешка.

Докато твърдата връзка на файла се явява аналог на използването на указател в съвременните езици за програмиране, символната връзка (symbolic link или soft link) до известна степен напомня указател на указател. При създаване на символна връзка с име symlink в дадена директория към файл, зададен посредством абсолютен или относителен път linkpath, действително в директорият се създава нов файл от тип

"символна връзка" с име symlink със свои собствени индексен възел и логически блокове. При щателно разглеждане на неговото съдържание може да се открие, че той се състои само от символния запис linkpath. Операцията по отварянето на файла от тип „символна връзка” е устроена по начин, че в действителност се отваря не самият файл, а онзи файл, чието име се съдържа в него (при необходимост, това се извършва рекурсивно!). Ето защо операциите над файловете, за които е необходимо предварително да се отвори файла (каквито са повечето команди на ОС, извършващи действия над файлове, операцията отваряне на файл се извършва, но е скрита от потребителя), в действителност се извършва не над файла от тип "символна връзка", а над онзи файл, чието име се съдържа в него (или над онзи файл, който се отвая в крайна сметка при рекурсивните връзки). Оттук следва, че опитите да се прочете реалното съдържание на файл от тип символна връзка с помоща на системното извикване read() са обречени на неуспех. Както е видно, създаването на мека връзка, от гледна точка на изменение на логическата структура на файловата система е еквивалентно на добавянето на именувано ребро към съществуващ възел, чрез файл от тип "символна връзка" и неименувано ребро.

Създаването на символни връзки не води до проблемите, свързани с изтриване на файлове. Ако файл, към който има създадени символни връзки, бъде изтрит от физическия носител, то опита да се отвори символната връзка (а следователно и изтрития файл) ще доведе до грешка от вида "Файл с такова име не съществува", което може да бъде обработено акуратно от приложната програма. В този смисъл, изтриването на свързания обект и неговата липса, не е фатално и няма да наруши целостта на файловата система.

Произволното и неакуратно прилагане на символни връзки от потребителите на ОС, могат да превърнат логическтат структура на файловата система от ацикличен в цикличен граф. Това, разбира се е нежелателно, но няма разрушителен характер, както циклите, които могат да бъдат създадени от твърдите връзки, ако не е въведена забрана за създаването на такива към директории. Тъй като символните връзки се

отличават принципно от твърдите и връзките възникващи между директориите и файловете при тяхното създаване, меките връзки могат лесно да бъдат идентифицирани от ОС или приложните програми. За предотвратяване на зацикляния в програмите, изпълняваши операции над файлове, обикновенно се ограничава дълбочината на рекурсията, по пътя на символните връзки. Превишаването на тази дълбочина, води до възникването на грешка от типа "Прекалено много символни връзки", която може лесно да бъде обработена от приложенията. Ето зашо в ОС UNIX не е предвидено ограничение на типа файлове, за които могат да се създават символни връзки.

За създаване на символна връзка се използва командата на ОС ln с опция -s и системното извикване symlink(). При използване на системното извикване symlink(), също не е необходимо предварително отваряне на файла за който се създава символната връзка, тъй като неговото съдържание не е необходимо за извършването на тази операция.

ИМЕ

symlink

ПРОТОТИП

```
#include <unistd.h>
int symlink(char *pathname, char *linkpathname);
```

ОПИСАНИЕ

Системното извикване symlink служи за създаване на символна (мека) връзка към файл с имен указано от парапетъра pathname. Указателят на името на създаваната връзка се задава с параметърът linkpathname (пълно или относително име на връзката).
Системното извикване не извършва проверка, дали реално съществува файл с име pathname.

ВРЪЩАНИ СТОЙНОСТИ

Системното извикване връща стойност 0 при нормално завършване или стойност -1 при възникване на грешка.

5. Премахване на твърди и символни връзки. Системно извикване unlink()

За премахване (изтриване) на връзки и файлове се използва командата на ОС rm или системно извикване unlink().

Необходимото, но недостатъчно условие за премахване на един регулярен файл от диска е да се премахнат всички негови твърди връзки (брояча на твърдите връзки в i-възела на стане 0). Ако съществуват

процеси, които използват файла, то файла няма да може да бъде премахнат, докато всички процеси, които го използват не завършат своето изпълнение и не затворят файла.

За да се изтрие един файл, процесът не е нужно да го отваря. Тези операции не влияят върху съдържанието на засегнатия файл, те влияят върху съдържанието на една или повече директории.

ИМЕ

unlink

ПРОТОТИП

```
#include <unistd.h>
int unlink(char *pathname);
```

ОПИСАНИЕ

Системното извикване unlink служи за изтриване на име от файловата система, указано от параметъра pathname.
Ако след изтриването брояча на твърдите връзки за даения файл е станал равен на 0, то са възможни следните ситуации.
* Ако в ОС няма процеси, които да държат дадения файл отворен, то файлът напълно се изтрива от физическия носител.
* Ако изтритото име е последната твърда връзка на регулярен файл, но съществува процес, който го държи отворен, то файла продължава да съществува до тогава докато не бъде затворен и последния файлов дескриптор, указващ дадения файл.
* Ако името се отнася за файл от тип socket, FIFO или за специалне файл на устройство, то файла се изтрива независимо от наличния процес, който го държи открит, но процесът отворил даденият файл може да продължи да го използва.
* Ако името се отнася до файл от тип „символна връзка”, то той се изтрива и мекатa връзка се разкъсва.

ВРЪЩАНИ СТОЙНОСТИ

Системното извикване връща стойност 0 при нормално завършване и стойност -1 при възникване на грешка.

Задача1: Създайте твърди и символни връзки в избрана директория към избрани файлове. Разгледайте съдържанието на директорията с помощта на командата ls -al. Обърнете внимание на разликата между символните и твърдите връзки в изведения на екрана списък. Определете допустимата дълбочина на рекурсивно създаване на символни връзки за използваната операционна система.

Създаване и премахване на директории. Системни извиквания mkdir() и rmdir()

За създаване на нова директория, респ. нов възел в логическата структура на файловата система съответното му именувано ребро се използва системното извикване mkdir(). Новосъздадената директория е празна, т.е. съдържа само двата стандартни записа с имена ″.″ и ″..″, които съответстват на самата директория и нейната родителска директория. За да създаде нова директория процесът трябва да има права за изпълнение за

всички директории по пътя в пълното име на новосъздаваната директории и право за запис за родителския каталог.

ИМЕ

mkdir

ПРОТОТИП

```
#include <sys/stat.h>
#include <sys/types.h>
int mkdir(const char *pathname, mode_t mode);
```

ОПИСАНИЕ

Системното извиквне mkdir() създава директория с име pathname.
Аргументът mode участва във формирането на правата за достъп до новосъздадента директория. Неговата стойност се модифицира, чрез маската на процеса (umask) по следния начин: Правата за достъп на създадената директория са (mode & ~umask & 0777).
Собствеността върху създаваната директория се определя от ефективния потребителски идентификатор на процеса. Ако родителската директория има установен бит set-group-ID, то новосъздадената ще наследи от нея групата собственик. В противен случай групата собственик ще се определи от ефективния идентификатор на групата на процеса.
Ако за родителската директория е установен бита set-group-ID, то така ще бъде и за новосъздадената.

ВРЪЩАНИ СТОЙНОСТИ

Системното извикване връща стойност 0 при успешно завършване и -1 ако възникне грешка.

За изтриване на директория, респ. премахване на възел в логическата структура на файловата система и съответното му именувано ребро се използва системното извикване rmdir(). Изтриваната директория трябва да е празна, т.е. съдържа само двата стандартни записа с имена ″.″ и ″..″, които съответстват на самата директория и нейната родителска директория. За да се изтрие директорията процесът трябва да има права за изпълнение за всички директории по пътя в пълното име на новосъздаваната директории и право за запис за родителския каталог.

ИМЕ

rmdir

ПРОТОТИП

```
#include <unistd.h>
int rmdir(const char *pathname);
```

ОПИСАНИЕ

Системното извикване изтрива директория, която трябва да е празна.

ВРЪЩАНИ СТОЙНОСТИ

При успех системното извикване връща 0. При възникване на грешка - -1.

4. Функции за работа със съдържанието на директории

Стандартните системни извиквания `open()`, `read()` и `close()` не могат да се използват от програмиста за разглеждане на съдържанието на

файла от тип "директория". За анализ на съдържанието на директории се използва набор от функции от стандартната библиотека на езика C.

ФУНКЦИЯ

```
opendir
```

ПРОТОТИП

```
#include <sys/types.h>
#include <dirent.h>
DIR *opendir(char *name);
```

ОПИСАНИЕ

Функцията opendir служи за отваряне на директория, името на която е указано в параметъра name. Типът данни DIR представлява указател към структура от данни, която описва информацията съдържаща се в директорията. Функция opendir подготвя почвата за функционирането на останалите функции, изълняваши операции над директории и ни позиционира на първия запис от директорията.

ВРЪЩАНИ СТОЙНОСТИ

При успешно завършване, функцията връща указател към структурата на отворената директория, който в последствие ще се предава като параметър на останалите функции, работещи с тази директория. При неуспех, функцията връща стойност NULL.

От гледна точка на програмистта, използващ този интерфейс, директорията представлява файл с последователен достъп, над който могат да се извършват операции четене, запис и позициониране в началото на файла. Преди изпълнение на тези операции е нобходимо директори тята да бъде отворена, а след това окончателно затворена. За отваряне на директорията се използва функцията opendir(), която подготвя почвата за извършване на операциите и ни позициониа в началото на файла. Четене на поредния апис от директорията се извършва с функцията readdir(), която едновременно с това ни позиционира в началото на следващия запис (ако, разбира се, такъв съществува). За ново позициониране в началото на директорията се прилага функцията rewinddir(). След приключване на работата с директорията е необходимо тя да се затвори с помоща на функцията closedir().

ФУНКЦИЯ

```
readdir
```

ПРОТОТИП

```
#include <sys/types.h>
#include <dirent.h>
struct dirent *readdir(DIR *dir);
```

ОПИСАНИЕ

Функция readdir служи за четене на пореден запис от отворена директория.
Параметърът dir е указател към структура, описваща отворената директория, който връща функцията opendir().
Типът данни struct dirent представлява структура от данни, описваща един запис в директорията. Броя и вида на полетата в този запис зависят от

използваната файлова система, но едно от полетата винаги присъства. Това е полето char d_name[ ] с дължина не превишаваща стойността NAME_MAX+1, която съдържа символното име на файла, като завършва със символ за край на ред.

ВРЪЩАНИ СТОЙНОСТИ

При успешно завършване функцията връща указател към структура, съдържаща поредния запис в директорията. Пре неуспех или при достигане края на директорията се връща стойност NULL.

ФУНКЦИЯ

rewinddir

ПРОТОТИП

```
#include <sys/types.h>
#include <dirent.h>
void rewinddir(DIR *dir);
```

ОПИСАНИЕ

Функция rewinddir служи за позициониране в отворента директория, свързана с указателя dir (т.е. с върната стойност от функцията opendir()), на първия запис (или в началото) от директорията.

ФУНКЦИЯ

closedir

ПРОТОТИП

```
#include <sys/types.h>
#include <dirent.h>
int closedir(DIR *dir);
```

ОПИСАНИЕ

Функцията closedir служи за затваряне на директорията, ассоциирана с указателя dir (т.е. с върната стойност от функцията opendir()). След затваряне директорията става недостъпна за последващо използване.

ВРЪЩАНИ СТОЙНОСТИ

При успешно завършване на функцията, тя връща стойност 0, а при неуспешно, стойност -1.

Задача 2: Напишете, компилирайте и изпълнете програма, извеждаща на екрана списък на файловете, намиращи се в избрана директория. В извеждания списък да се обозначава типа на всеки от файловете. Името на директорията да се задава като параметър в командния ред. Ако то отсъства да се избира текущата директория.

Задача 3 (с повишена сложност): Напишете програма, разпечатваща съдържанието на зададена директория в формат, подобен на този използван в командата ls -al. За да тази програма е необходимо самостоятелно да изучите функцията ctime, информацията за която се намира в раздел 3 от UNIX Manual и системните изивквания time, и readlink, информацията за които се намира в раздел 2 от UNIX Manual.