ТЕМА 1. ЗАПОЗНАВАНЕ С ОПЕРАЦИОННАТА СИСТЕМА UNIX. СИСТЕМНИ ИЗВИКВАНИЯ

История на операционата система UNIX. Структура. Системни извиквания и библиотека libc. Понятия login и password. Устройство на файловата система в UNIX. Пълно име на файл. Текуща директория. Команда pwd. Относително име на файл. Домашна директория на потребителя. Команда man – универсален справочник. Команда cd – смяна на текущата директориа и ls – преглед съдържанието на директория. Команда cat. Създаване на файл, посредством cat. Пренасочване на входа и изхода. Стандартни команди за работа с файлове – cp, rm, mkdir, mv. Редактиране на файлове – текстови редактори в UNIX. Потребители и групи. Команди chown и chgrp. Права за достъпа до файл. Команда ls с опции -al. Използване на командите chmod и umask. Системни извиквания getuid и getgid. Компилиране и стартиране на програми на C в UNIX.

1. История на операционата система UNIX. Структура

Тази операционна система е достигнала до своя съвременен вариант вследствие на дълга еволюция на UNIX-образните операционни системи. Историята на развитието на UNIX е подробно описана в множество книги посветени на операционните системи и изчислителната техника [x, y, z]. Едно кратко описание на историята на развитие на UNIX може да намерите в приложение 1 на настоящото ръководство. Най-важното в историята на UNIX е съществуването на две линии на развитие – System V и BSD. В процеса на обучение, ще бъдат изяснени различията между тези два варианта на операционната система UNIX.

Операционната система Linux е създадена и развита съобразно принципите залегнали в UNIX (Приложение 2) [x]. Макар, нейният програмен код да не е копие на този на UNIX, тя спазва UNIX стандартите и интерфейса на системните извикваня, което я прави подходяща за изучаване на принципите на работа на UNIX-образните операционни системи.

Ядрото на операционната система Linux представлява монолитна система. Въпреки това, при компилирането му, може да се разреши

динамично зареждане и премахване на различни негови компоненти – така наречените модули. В момента на зареждане на модула, неговия код започва да се изпълнява в привилигерован режим и се свързва с останалата част на ядрото. Вътре в модула може да се използва всяка една от функциите на ядрото.

2. Системни извиквания и библиотека libc

Основната и постоянно работеща част на ОС UNIX е нейното ядро. Останалите програми (системни или потребителски) могат да се обръщат към ядрото, посредством системни извиквания, които се явяват точки за вход на програмите в ядрото. При изпълнение на системно извикване програмата на потребителя времено преминава в привилигерован режим. Така, тя получава достъп до данни или устройства, които са недостъпни при работа в потребителски режим.

Машините команди, необходими за активиране на системно извикване, наред със способите за предаване на параметри и резултати между извикващата програма и ядрото са различни за различните процесори. От гледна точка на програмирането на езика C, обаче, използването на системни извиквания по нищо не се отличава от използването на другите функции от стандартната ANSI библиотека, например, функциите за работа с низове, като *strlen()*, *strcpy()* и др. Стандартната за UNIX библиотека – *libc* – обезпечава връзката на C с всяко едно системно извикване. От това произтича, че за програмистта системното извикване изглежда като функция на езика. Освен това, при изпълнение в ОС UNIX, много от известните стандартни функции, например функциите за работа с файлове: *fopen()*, *fread()*, *fwrite()* използват различни системни извиквания. В хода на настоящите упражнения се разглеждат множество системни извиквания и техния C-интерфейс.

По-голяма част от системните извиквания, които връщат целочислен резултат, използват стойност *-1*, за да известят за възникването на грешка и стойност по-голяма или равна на *0* – при успешно завършване. Системните извикания, които връщат указатели, обикновено връщат стойност NULL, за да укажат възникването на грешка. C-интерфейсът

предоставя глобалната променлива *errno*, за конкретно определяне на причината за възникналата грешка. Тя е дефинирана във файла *<errno.h>* заедно с нейните възможни стойности и кратки техни описания. Променливата *errno* е необходимо да се анализира, едва след възникване на грешка, тъй като след успешно завършване, системното извикване не променя нейното значение. Стандартната UNIX-функция *perror()* служи за извеждане на текстово съобщение относно възникналата грешка на стандартния изход на програмата (по подразбиране екрана на терминала).

Функция perror()

Прототип на функцията

```
#include <stdio.h>
void perror(char *str);
```

Описание на функцията

Функцията perror() е предназначена за извеждане на текстови съобщения, описващи възниналата грешка. Функцията отпечатва съдържанието на низът str (ако параметърът str не е равен на NULL), двоеточие, интервал и текста на сообщението, съответстващо на възникналта грешка, следван от символа за нов ред ('\n').

4. Понятия login и password

Операционната система UNIX е многопотребителска операционна система. За обезпечаване на безопасната работа на потребителите и целостта на системата, достъпът до нея трябва да бъде защитен. Всеки потребител в UNIX притежава регистрационно име – *username* или *login* и парола – *password*, която съответства на това име. Обикновено при създаването на нов потребител в системата, ситемния администратор назначава служебна парола. При първият вход в ситемата потребителят е длъжен да промени паролата си с помощта на съотетната команда. В последствие, той може да смени паролата си във всеки един момент.

5. Вход в системата

При първоначалния вход в системата на екрана се появява надпис, известяващ за въвеждане на потребителско (регистрационно) име, обикновенно, това е надписа "*login:*". След въвеждане на потребителското име се натиска калвиша *<Enter>*. След това системата известява за въвеждане на парола, съответстваща на въведеното име, като обикновено се извежда надписа "*Password:*". След въвеждане на паролата се натиска

калвиша *<Enter>*. Въвеждането на паролата не се отразява на екрана, поради което е необходимо внимателното и набиране.

> *Поздравление:* Вече сте пълноценен потребител на операционната система UNIX.

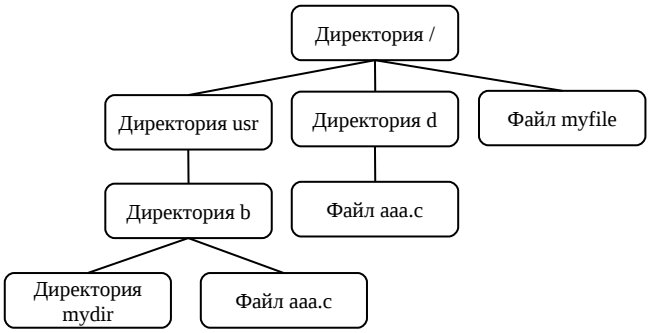
6. Устройство на файловата система в UNIX. Пълно и относително име на файл

В операционната система UNIX съществуват три базови понятия: "процес", "файл" и "потребител". За понятието "потребител" вече стана въпрос, то се обсъжда и при по-нататъшното изучаване на работата на операционната система UNIX. Понятието "процес" характеризира динамичната страна на процесите протичащи в изчислителните системи, то е подробно разгледано в следващите упражнения. Понятието "файл" характеризира статичната страна на изчислителните системи.

Всеки има известен опит с изчислителната техника и най-вероятно има изградена представа за файла, като за именован набор от данни, съхраняван на някакъв вид запомнящо устройство. За целите на настоящото упражнение това разбиране за файл е достатъчно, за да може да се разгледа, как е организирана работата с файлове в операционната система UNIX. Цялостно, подробно разглеждане на понятието "файл" и организацията на файловите системи в UNIX е направено в третата част на настоящото ръководство.

Всички файлове в операционната система UNIX, както и останалите известни операционни системи са обединени в дървовидна логическа структура. Файловете могат да се обединяват в каталози или директории. Не съществуват файлове, които да не влизат в състава на някоя директория. Директорията от своя страна може да влиза в състава на друга директория. Допуска се съществуването на празни директории, в които не влиза нито един файл и нито една друга директория (фиг. 1.1). Измежду всички директории съществува само една директория, която не влиза в състава на никоя друга – прието е, тя да се нарича коренова. На този етап може да се заключи, че във файловата система на UNIX присъстват два типа файлове: 1) обикновенни файлове, които могат да съдържат текст, програмен код, изпълним код, данни и т.н. – прието е, те да се наричат

регулярни файлове и 2) директории.



Фиг.1.1. Примерна структура на файлова система

На всеки файл (бил той регулярен или директория) трябва да му бъде присвоено име. В различните версии на операционната система UNIX съществуват различни ограничения за избора на файлови имена. В стандарта POSIX за интерфейса на системните изиквания за операционната система UNIX се съдържат само три явни ограничения:

• Не трябва да се създават имена с по-голяма дължина от предвиденото в операционната система (за Linux – 255 символа).

• Не трябва да се използва символът NUL (да не се бърка с указателя NULL!) – това е символът с код нула, той е знак за края на низ в езика C.

• Не трябва да се използва символът '/'.

Трябва да се добави, също така, че е нежелателно да се използват символите "звезда" – "*", "въпросителен знак" – "?", "кавичка" – "\"", "апостроф" – "\'", "интервал" – " " и "обратен слаш" – "\\"(символите са представени, поредством формата за символни константи в езика C).

Единственото изключение се явява корновата директория, която винаги има име "/". Тази директория по напълно понятни причини представява единственият файл, който трябва да има уникално име във всички файлови системи. Всички останали файлове трябва да имат уникални имена единствено в рамките на тази директория, в състава на която се намират. По какво се отличават двата файла с имена "*aaa.c*", намиращи се в директориите "*b*" и "*d*" на фиг. 1.1, така че да е ясно с кой от тях се работи? Тук на помощ идва понятието за пълно име на файл.

Ако мислено се построи път от кореновия връх на дървото (фиг. 1.1) до друг негов връх (файл) и се впишат всички имена на файлове (т.е.

върхове на дървото), срещащи се по този път, например, "*/ usr b aaa.c*". В тази последователност, първо винаги ще стои името на корневата директория, а последно – името на търсения файл. Ако имената на възлите в този запис се отделят не с интервал, а със символа "/", с изключение на името на кореновата директория и следващото след нея име ("*/usr/b/aaa.c*") се получава запис, който еднозначно идентифицира файла във всички логически конструкции на файловите системи. Този начин на записване (означаване) на файловете се нарича пълно име на файл.

7. Текуща директория. Команда pwd. Относително име на файл

Пълните имена на файлове могат да включват в себе си достатъчно много имена на директории, което да ги направи твърде дълги и неудобни за работа. В същото време, съществуват понятия като текуща или работна директория и относително име на файл, които помагат да се съкрати изписването на файлови имена

За всяка работеща програма в операционната система, включително и за командния интерпретатор (shell), който обработва въвежданите команди, една от директориите в логическата структура на файловата система се назначава за текуща или работна за дадената програма. С помоща на командата *pwd* може да се установи, коя е текущата директория за използвания в даден момент команден интерпретатор.

Команда pwd

Синтаксис на командата

```
pwd
```

Описание на командата

```
Командата pwd извежда пълното име на текущата директория за работещия
команден интерпретатор.
```

Знаейки текущата директория може да бъде положен път по графа на файловете от текущата директория към търсения файл. За записването на последователността от възли, намиращи се в този път, се спазва следното правило: възелът, съответстващ на текущата директория не се включва в записа; при движение по направление към кореновия каталог, всеки възел се означава с две последователни "точки" – "*..*", а при движение по направление от корневия каталог се записва името на срещнатия възел. Обозначенията, отнасящи се до различните възли в този запис се разделят,

посредством символа "/". Полученият символен низ е прието да се нарича относително име на файл. Относителното име на файл се променя при смяна на работния каталог. В примера от фиг. 1.1, ако работния каталог е директорията "*/d*", то относителното име за файла "*/usr/b/aaa.c*" е "*../usr/b/aaa.c*", а ако работния каталог е директорията "*/usr/b*", то неговото относително име става "*aaa.c*".

За пълнота, може да се каже също, че в относителното име може да участва и текущия каталог, като се означи с "точка" – ".". Тогава за горния пример относителните имена стават съответно следните "*./../usr/b/aaa.c*" и "*./aaa.c*".

Работната директория на програми стартирани с помоща на командния интерпретатор е същата, както и на командния интерпретатор. Освен, ако вътрешно, такава програма не си промени разположението с помоща на специално системно извикване.

8. Домашна директория на потребителя

За всеки нов потребител в системата се създава специална директория, която става негова текуща директория при влизане на потребителя в ОС. Тази директория се нарича домашна директория на потребителя.

Задача: Използвайте командата *pwd*, за да определите своята домашна директория.

9. Команда man – универсален справочник

При работа с операционната система UNIX често е необходимо да се направи справка и да се потърси информация за синтаксиса и/или действието на една или друга команда или системно извикване, за предназанчението на някои системни файлове, както и техния формат и т.н. За получаване на тази информация, потребителя може да използва интерактивното ръководство на UNIX (UNIX Manual) с помоща на програмата *man*.

За използване на тази програма е достатъчно да се въведе:

```
man име
```

където *име* е търсената команда, програма, системно извикване, библиотечна функциа или файл.

Задача: Опитайте с помоща на *man* да разгледате информация за командта *pwd*.

Ако полученото описание не се събира на екрана, може да се премине на следващата екранна страница с клавиша *<интервал>*. За преминаване на следващ ред се използва *<Enter>*. За връщане на една екранна страница назад се използва *<Ctrl> + <b>*. За изход от програмата се изпозлва *<q>*.

Понякога имената на командите въвеждани от командния интерпретатор и имента на системните извиквания или още някакви имена съвпадат. Тогава за намиране на точната информация в *man* се задава и категорията (номер на раздел) към която се отнася търсената информация. Разделянето на информацията на категории може да се отличава в малка степен в различните версии на UNIX. В Linux, например е прието следното разделение:

1. Изпълними файлове или команди на комндния интерпретатор.
2. Системни извиквания.
3. Библиотечни функции.
4. Специални файлове (обикновенно файлове на устройства).
5. Формат на системните файлове.
6. Игри (обикновенно отсъстват).
7. Макропакети и помощни програми (такава програма е *man*).
8. Команди за системно администриране.
9. Подпрограми на ядрото (нестандартен раздел).

В Linux помощната програма *man* може да се извика с допълнителен параметър, указващ номера на раздела, в който се предполага, че се намира търсената информация:

```
man номер_на_раздела име
```

Задача: За да получите точна информация за боря и вида на използваните раздели, както и за допълнителни възможности на *man* въведете комадата: *man man*

10. Команди cd – за смяна на текущата директория и ls – за преглед съдържанието на директория

За смяна на текущата директория на командния интерпретатор може

да се използва командата *cd* (change directory). Тя се въвежда във вида

```
cd име_на_директория
```

където *име_на_директория* е пълното или относителното име на новата текуща директория.

> *Съвет:* Използвайте командата *cd* без параметри, за да направите домашната си директория текуща.

Разглеждане на съдържанието на текущата директория или която и да било друга е възможно с помощта на командата *ls* (от list). Ако се въведе без параметри, тази команда извежда на екрана списък на файловете, намиращи се в текущата директория. Ако в качеството на параметър се зададе пълното или относъстелно име на директория:

```
ls име_на_директория
```

то се извежда списък на файловете в указаната директория. Трябва да се отбележи, че в получения списък не влизат файловете, чийто имена започват със символа "точка" – ".". Такива файлове, обикновенно се създават от различни системни програми за настройка и конфигуриране. За да се види пълния списък на файловете към командата *ls* се добавя опцията *-a* във вида

```
ls -a
```

или

```
ls -a име_на_директория
```

Командата *ls* притежава и много други опции, част от които се разглеждат в следващите точки от упражнението.

> *Съвет:* За да видите пълна информация за командата *ls* използвайте помощната програма *man*.

11. Преглед на структурата на файловата система

> *Задача:* Използвайте командите *cd*, *ls* и *pwd*, за да прегледате структурата и съдържанието на файловата система. Възможно е преминаването или разглеждането на някой директории да е забранено от механизма за защита на файлове и директории, който ще бъде разгледан в последствие. Не забравяйте в края отново да се върнете в своята домашна директория.

12. Команда cat. Създаване на файл, посредством cat. Пренасочване на входа и изхода

Освен преместване в логическта структура на файловите системи и разглеждане на нейното съдържание е необходимо потребителят да умее

да разглежда съдържанието на файлове и да ги създава. За извеждане на съдържанието на неголям текстов файл на екрана може да се използва командата *cat* във вида

```
cat име__на_файл
```

> *Внимание!* Тази комада не може да се използва за извеждане на екрана на съдържанието на директории, а също така на бинарни файлове. Извеждането на екрана на бинрен файл може да доведе до непредсказуемо поведение на терминала.

Ако файлът е текстов, но голям, то на екрана се извежда само последната му страница. Големите текстови файлове е удобно да се разглеждат, посредством помощната програма *more* (описание на нейното използване може да се намери в интерактвното ръководство на UNIX с програмата *man*). Командата *cat* е интересна от друга гледна точка.

Ако в качеството на параметри за командата *cat* се зададе не едно, а няколко файлови имена

```
cat файл1 файл2 ... файлN
```

то системата ще изведе на екрана тяхното съдържание в указания ред. Изхода от комадата *cat* може да се пренасочи от екран на терминала към някой файл, възползвайки се от символа за пренасочване на изходния поток на данните – знак "по-голямо" – "*>*". Командата придобива вида

```
cat файл1 файл2 ... файлN > файл_резултат
```

и слива съдържанието на всички файлове, чийто имена стоят перед знака "*>*", в едно във файла *файл_резултат*, т.е. конкатенира ги (от анлг. concatenate). Възможността за пренасочване на изходните данни от стандартния изходен поток (екрана) във файл може да се прилага за всички команди, изпълнявани от командния интерпретатор. Така, например може да се получи файл, съдържащ списък на всички файлове в текущата директория, ако се изпълни командта *ls -a* с пренсочване на изходните данни

```
ls -a > нов_файл
```

Ако не се зададат имена на входни файлове за командата *cat*, то тя ще използва в качество на входни данни информацията, която се въвежда от клавиатурата, до момента, в който не се натисне клавишната комбинаця *<CTRL> + <d>*.

По този начин, командата

```
cat > нов_файл
```

позволява да се създаде нов текстов файле с име *нов_файл* и съдържание, което потребителя въвежда от клавиатурата. Командата *cat* притежава множество различни опции, които могат да се разгледат в UNIX Manual, посредством *man*.

Трябва да се отбележи, че наред с пренасочването на изходните данни съществува способ за пренасочване и на входните. Ако във времето на своето изпълнение някоя команда трябва да получи данни от клавиатурата, те могат предварително да се поставят във файл, а след това да се пренасочи стандартния вход за тази команда с помоща на символа "по-малко" – "<" последван от името на файла с входните данни. В UNIX Manual могат да се видят и други варианти за пренасочване на потоците от данни за командния интерпретатор.

13. Създаване на файл с помоща на командата cat

Задача: Убедете се, че се намирате в домашната си директория и създайте с помощта на командата *cat* нов текстов файл. Разгледайте неговото съдържание.

14. Стандартни команди за работа с файлове – cp, rm, mkdir, mv

За пълноценна работа с файлове в ОС UNIX също така е необходимо и умения за създаване на собствени поддиректории, копиране и изтриване на файлове, както и тяхното преименуване.

За създаване на нова поддиректория се използва командата *mkdir* (съкращение от make directory). В най-простия си вариант командата изглежда по следния начин:

```
mkdir име_на_директория
```

където *име_на_директория* е пълното или относителното име на създаваната директория. Командата *mkdir* разполага с набор от опции, описанието, на които може да бъде разгледано с помощтната програма man.

Команда cp

Синтаксис на командата

```
cp файл_източник файл_назначение
cp файл1 файл2 ... файлN дир_назначение
cp -r дир_източник дир_назначение
cp -r дир1 дир2 ... дирN дир_назначение
```

Описание на командата

```
Настоящото описание е непълно и е само въведение в използването на
командата cp. За пълно описание на командата се обърнете към UNIX
```

```
Manual.
Командата cp във вида
cp файл_източник файл_назначение
служи за копиране на един файл с име файл_източник във файл с име
файл_назначение.
Командата cp във вида
cp файл1 файл2 ... файлN дир_назначение
служи за копириане на файла или файловете с имена файл1, файл2, ...
файлN в съществуваща директориа с име дир_назначение под своите си
имена. Вместо имената на копираните файлове могат да се използват
техни шаблони.
Командата cp във вида
cp -r дир_източник дир_назначение
служи за рекурсивно копиране на една директория с име дир_източник в
нова директория с име дир_назначение. Ако директорията дир_назначение
вече съществува, то се получава командата cp в следния вид
cp -r дир1 дир2 ... дирN дир_назначение
Тази команда служи за рекурсивно копиране на директория или директории
с имена дир1, дир2, ... дирN във вече съществуваща директория с име
дир_назначение под своите собствени имена. Вместо имената на
копираните директори могат да се използват техните шаблони.
```

За копиране на файлове може да се използва командата *cp* (съкращение от copy). Командата cp може да копира не само отделни файлове, но и набор от файлове, и даже цели директории заедно със всички поддиректории и файлове в тях (рекурсивно копиране). За да се зададе набор от файлове може да се използва шаблон за техните имена. Такива шаблони могат да се използват и в командите за преименуване и изтриване на файлове, както е разгледано по-долу.

Шаблон за имената на файлове

```
Шаблонът, задаващ група от файлови имена може да се използва като
параметър, заместващ набор от имена на файлове в много команди на
операционната система. При използване на шаблон се преглеждат всички
имена на фсйлове, намиращи се във файловата система, и тези имена,
които удовлетворяват шаблона се включват в набора. В общия случай
шаблони могат да се задават, посредством следващите метасимволи:
* – съответства/замества всякакви последователности от символи,
включително интервал;
? – съответства/замества единичен символ;
[...] – съответства/замества, който и да е символ, затворен в скобите.
Диапзаон от символи се задават като се разделят с тире.
Така, например, чрез шаблона *.c се указват всички файлове в текущата
директория, чийто имена завършват на .c. Шаблонът [a-d]* указва всички
файлове в текущата директория, чийто имена започват с буква a, b, c
или d. Съществува едно ограничение при използването на метасимвола * в
началото на именто на файл, например, в случая с шаблона *c. За такива
шаблони имената на файловете, започващи с точка се считат за не
удовлетворяващи шаблона.
```

За изтриване на файлове или директории се използва командата *rm* (съкращение от remove). За да се изтрият един или няколко регулярни файла, то командата *rm* може да изглежда по следния начин:

```
rm файл1 файл2 ... файлN
```

където *файл1*, *файл2*, ... *файлN* – са пълните или относителните имена на регулярни файлове, които следва да се изтрият. Вместо имена могат да се използват шаблони. За изтриване на една или няколко

директории заедно с тяхното съдържание (рекурсивно изтриване), то към командата се добавя опцията *-r*:

```
rm -r дир1 дир2 ... дирN
```

където *дир1*, *дир2*, ... *дирN* – са пълните или относителните имена на директориите, които следва да се изтрият. Вместо непосредствено да се задават имена, също могат да се използват шаблони. Командата *rm*, също има полезен набор от опции, които са подробно описани в UNIX Manual. В действителност процесът на изтриване на файлове не е така прост, както изглежда на пръв поглед. По-подробно този процес ще бъде разгледан в третата част на това ръководство, при разглеждане на операциите над файлове в операционната система UNIX.

Команда mv

Синтаксис на командата

```
mv име_на_източника име_на_назначението
mv име1 име2 ... имеN дир_назначение
```

Описание на командата

```
Настоящото описание е непълно и е само въведение в използването на
командата mv. За пълно описание на командата се обърнете към UNIX
Manual.
Командата mv във вида
mv име_на_източника име_на_назначението
служи за преименуване или преместване на един файл (регулярен или
директория) с име име_на_източника във файл с име име_на_назначението.
При това преди изпълнение на командата не е задължително да съществува
файла с име име_на_назначението.
Командата mv в формата
mv име1 име2 ... имеN дир_назначение
служи за преместване на файла или файловете (регулярни файлов или
директории) с имена име1, име2, ... имеN във вече съществуваща
директория с име дир_назначение под техните собствени имена. Вместо
имената на преместваните файлове могат да се  използват, заместващи ги
шаблони.
```

Командата за изтриване на файлове и директории следва да се използва с повишено внимание, тъй като изтритата информация е невъзможно да бъде възстановена.

Съвет: Ако вие се системния администратор и вашата текща директория е корновата, моля, не изпълнявайте командата *rm -r* *!

За преименуване или преместване на файл в друг каталог се използва командата *mv* (съкращение от move). За задаване на имената на преместваните файлове в нея, също могат да се използват шаблони.

15. Редактиране на файлове – текстови редактори в UNIX

В периода, когато се появяват първите варианти на операционната система UNIX, устройствата за въвеждане и извеждане на информация

съществено се отличават от днес съществуващите такива. Тогавашните дисплеи работят единствено в буквено-цифров режим, а на клавиатурите отсъстват клавиши със стрелки за преместване на курсора. Поради тази причина първият текстов редактор за операционната система UNIX – редакторът *ed* е редактор с командноредов интерфейс и изисква от потребителя, точно да укаже с помоща на специални команди, какво и как да се редактира. Така например, за да се замени първото срещнато съчетание на символите "*ti*" на "*to*" на единайсти ред от редактирания файл, трябва да се въведе командта

```
11 s/ra/ru
```

Впоследствие се появява екранният редактор *vi*, който обаче също зависи от точните инструкции, указващи какво и как да се направи в текущата позиция на екрана или по какъв начин да се промени текущата позиция. За тази цел се използват специални команди, съответстващи на ограничените възможности на тогаващните буквено-цфровите клавиатури. От днешна гледна точка тези редактори изглеждат архайчни, но те и до днес влизат в състава на всички варианти на UNIX и понякога (например, при работа с отдалечена машина) са единственото налично средство, позовляващо редактиране на файлове.

Работата в UNIX, осъществяваща се изключително на ниво команден интерпретатор и екранни тексови редактори е далеч от съвременните удобства, налични в други ОС. Все пак съществуват разнообразни пакети, облекчаващи задачите на потребителя в UNIX. Това са различните графични работни среди (по настоящем най-популярни от тях са KDE и GNOME), които осигуряват аналог на графичния потребителски интерфейс на Windows и които имат свои вградени графични текстови редактори.

В режим на команден ред, освен споменатите вече текстови редактори могат да се използват и някои по дружелюбни към потребителя редактори като екранният редактор *joe*. Информация за него може да се намери в UNIX Manual. Също така с много големи възможности е многофункционалният текстов редактор *emacs*.

16. Потребители и групи. Команди chown и chgrp. Права за достъп до файл

В операционната система UNIX всяко потребителско име се представя с числова стойност, която се нарича потребителски идентификатор – UID (user identificator).

Всички потребители в системата се делят на групи. Например, студентите от една учебна група могат да съставят отделна потребителска група в UNIX. Потребителските групи, също получават свои имена и съответстващи им идентификациони номера – GID (group identificator). В някои от версиите на UNIX, всеки потребител може да влиза в състава само на една група, докато при други – в няколко.

Команда chown

Синтаксис на командата

```
chown owner файл1 файл2 ... файлN
```

Описание на командата

```
Командата chown е предназначена за промяна на собствеността върху
файлове. Настоящото описание не е пълно описание на командата, а
адаптирано към нуждите на това ръководство. За пълно описание на
командата се обърнете към UNIX Manual.
Единствено, предишният собственик или системният администратор могат
да променят собственика на даден файл.
Параметърът owner задава новия собственик на файла, посредством
неговото потребителско име – username, или посредством неговитя
потребителски идентификатор - UID.
Параметрите файл1, файл2, ... файлN са имената на файловете, за които
следва да се промени собственика. Вместо имена могат да се използват
заместващи ги шаблони.
```

За всеки създаден файл във файловата система се запомнят имената на неговия създадел и групата на създателите. Трябва да се отбележи, че тази група не е обезателно групата, в която влиза собственика. Впоследствие собственика на файла или системният администратор могат да сменят неговия собственик с друг потребител или да променят групата на собствениците с помоща на командите *chown* и *chgrp*, чието подробно описание може да се намери в UNIX Manual.

Команда chgrp

Синтаксис на командата

```
chgrp group файл1 файл2 ... файлN
```

Описание на командата

```
Командата chgrp е предназначена за промяна на групите на собствениците
на файлове. Настоящото описание не е пълно описание на командата, а
адаптирано за нуждите на това ръководство. За пълно описание на
командата се обърнете към UNIX Manual.
Единствено собственика или сиатемния администратор могат да променят
групата на собственците на файла.
Параметърът group задава новата група на собствениците на файла,
```

```
посредством името на групата или посредством нейния идентификатор –
GID.
Параметрите файл1, файл2, ... файлN са имената на файловете за които
следва да се промени групата на собствениците. Вместо имена могат да
се използват, заместващи ги шаблони.
```

За всеки файл съществуват три категории потребители:

• Потребител, който е собственик на файла;

• Потребители, отнасящи се към групата на собственика на файла;

• Всички останали потребители.

За всяка от тези категории, собственика на файла може да задава различни права за достъп до него. Различават се три вида права за достъп: право на четене – *r* (от *r*ead), право запис (промяна) – *w* (от *w*rite) и право на изпълнение – *x* (от e*x*ecute). За регулярните файлове смисълът на тези права съвпада с указаното по-горе. За директории този смисъл е по-различен. Правото за четене за директория позволява да се четат само и единствено имената на файловете, намиращи се в нея. Доколкото "изпълнение" на директория е безмислено (както, впрочем и "изпълнение" на регулярен файл), правото за достъп "изпълнение" за директория има следния смисъл: наличието на това право позволява да се получи допълнителна информация за файловете, намиращи се в директорията (размер, собственик, дата на създаване и т.н.). Без това право не могат да се четат съдържащите се в директорията файлове, нито да бъдат модифицирани или изпълнявани. За да бъде направена една директория текуща, също е необходимо наличието на правото изпълнение, а също и за всички директории по пътя до нея. Правото за запис, отнесено към директория позволява да се променя нейното съдържание: да се създават и изтриват файлове в нея, както и да бъдат преименувани. Трябва да се отбележи, че за изтриване на файл е достатъчно наличието на права за запис и изпълнение за директорията, в която се намира дадения файл, независимо от правата за достъп до самия файл.

17. Команда ls с опции -al. Използване на командите chmod и umask

За получаване на подробна информация за файловете в дадена директория, включително име на собственик, група на собстеника и права за достъп, може да се използва вече известната команда *ls* с опции ***-al***. В

извежданата от командата информация третата колона съдържа името на собственика на файла, а четвъртата – името на групата на собственика. В първата колона се съдържа информация за типа на файла и правата за достъп до него. Типа на файла се определя от първия символ в набора от символи в колоната. Ако това е символът '*d*', то типа на файла е директория, ако там се намира символа '**-**', то това е регулярен файл. Следващите три символа определят правата за достъп, отнасящи се до собственика на файла, следващите три – за потребителите влизащи в групата на собстеника на файла, а последние три – за всички останали потребители. Наличието на буква (*r*, *w* или *x*), означава, че дадената категория потребители притежава това право.

Задача: Изпълнете командата *ls -al* за своята домашна директория и анализирайте резултата от нея.

Команда chmod

Синтаксис на командата

```
chmod [who] { + | – | = } [perm]
    файл1 файл2 ... файлN
```

Описание на командата

Командата chmod е предназначена за промяна на правата за достъп до един или няколко файла. Настоящото описание не е пълно описание на командата, а адаптирано за нуждите на това ръководство. За пълно описание на командата се обърнете към UNIX Manual.
Единствено собственика или системния администратор могат да променят правата за достъп до файл.
Параметърът who определя, категорията потребители за които се променят правата за достъп. Този параметър може да има една от следните стойности:
a – съответсва на всички категории потребители. Ако този параметър бъде изпуснат по подразбиране се приема стойност a. В тази ситуация правата за достъп се определят с отчитане на стойността на маската за създаваните файлове;
u – съответства на собственика на файла;
g – съответства на потребителите, влизащи в групата на собственика на файла;
O – съответства на всички останали потребители.
С един от следващите символи се определя операцията над правата за достъп:
+ – добавяне на право за достъп;
- – отмяна на право за достъп;
= – замяна на право за достъп, т.е. отмяна на всички съществуващи и добяне на преизчислени права.
Ако параметърът perm не зададен, то всички съществуващи права за достъп се отменят.
Параметърът perm определя правата за достъпа, които следва да бъдат добавени, отменени или заменени. Стойността на този параметър е комбинация от следните символи или един от тях:
r – право за четене;
w – право за модифициране (запис);
x – право за изпълнение.
Параметрите файл1, файл2, ... файлN са имента на файловете за които следва да се променят правата за достъп. Вместо имена могат да се използват, заместващи ги шаблони.

Собственика на файл може да промени правата за достъп до него, използвайки командата *chmod*.

Създайте нов файл и разгледайте правата за достъп до него, зададени от системата при неговото създаване. За определяне на тези права операционнта система се ръководи от маската за създаваните файлове, която използва програмата създаваща файла. Първоначално тази маска има определена стойност по подразбиране, която може да се променя в последствие от програмите.

Маска за създаваните файлове от текущия процес

Маската за създаваните файлове от текущия процес (umask) се използва от системните извиквания open() и mknod() при първоначално определяне на правата за достъп за новосъздавани файлове или FIFO. Младшите 9 бита от тази маска съответстват на правата за достъп на потребителя, създаващ файла, групата, на която принадлежи той и всички останали потребители така, както е описано по-долу с нейните осмични стойности:
0400 – право за четене за потребителя, създаващ файла;
0200 – право за запис за потребителя, създаващ файла;
0100 – право за изпълнение за потребителя, създаващ файла;
0040 – право за четене за групата на потребителя, създаващ файла;
0020 – право за запис за групата на потребителя, създаващ файла;
0010 – право за изпълнение за групата на потребителя, създаващ файла;
0004 – право за четене за всички останали потребители;
0002 – право за запис за всички останали потребители;
0001 – право за изпълнение за всички останали потребители.
Установяването на някой от битовете в 1 забранява инициализирането на съответното право за достъп за новосъздавания файл. Стойността на тази маска може да се променя с помоща на системното извикване umask() или с командата umask. Маската за създаваните файлове се наследява от процеса-дете при пораждане на нов процес със системното извикване fork() и влиза в състава на неизменящата се част от системния контекст на процеса при системното извикване exec(). В резултат от това наследяване промяната на маската с помоща на командата umask оказва влияние на атрибутите за достъп до новосъздаваните файлове за всички процеси, породени впоследствие от командния интерпретатор.

С помоща на командата *umask* може да се извърши промяната на текущата стойност на маската за командния интерпретатор.

Команда umask

Синтаксис на командата

```
umask [value]
```

Описание на командата

Командата umask е предназначена за промяна на маската за създаваните файлове от командния интерпретатор или разглеждане на нейната текуща стойнсот. При отсъствие на параметър командта извежда на екрана текущата стойност на маската. Новата стойност се задава, посредством параметъра value в осмичен вид.

Внимание: Маската за създаваните файлове не се съхранява между работните сесии в системата. При нов вход в системата стойността на маската отново се установява по подразбиране.

18. Системни извиквания getuid и getgid

Използвайки системните извиквания *getuid()* и *getgid()*, вътре в една програма, може да се намерят идентификаторите на потребителя, изпълнил (стартирал) програмата – UID и идентификатора на групата към която принадлежи – GID.

Системни извиквания getuid() и getgid()

Прототипи на системните извиквания

```
#include <sys/types.h>
#include <unistd.h>
uid_t getuid(void);
gid_t getgid(void);
```

Описание на системните извиквания

```
Системното извикване getuid връща като резулат идентификатора на
потребителя за текущия процес.
Системното извикване getgid връща като резулат идентификатора на
групата на потребителя за текущия процес.
Типовете данни uid_t и gid_t са синоними на един от целочислените
типове в езика C.
```

19. Компилиране и стартиране на програми на C в UNIX

За компилиране на програма на езика C в Linux може да се използва компилатора *gcc*.

За неговата нормална работа е необходимо изходния файл, съдържащ текста на програмата да има име завършващо на *.c*.

В най-простия случай компилирането е възможно с командата

```
gcc име_на_изходния_файл
```

Ако програмата е написана без грешки, компилаторът създава изпълним файл с име *a.out*. Промяната на името на създавания изпълним файл може да се извърши с помоща на опцията ***-o***:

```
gcc име_на_изходния_файл -o име_на_изпълнимия_файл
```

Компилаторът *gcc* има няколко стотин възможни опции. Информация за тях може да се намери в UNIX Manual.

За стартиране на компилираната програма е необходимо в командния интерпретатор да се въведе нейното име предшествано от *./*, ако се намира в текущата директория и се натисне *<Enter>*.

20. Създаване, компилиране и стартиране на програма с използване на системното извикване getuid() и getgid()

Задача: Напишете, компилирайте и изпълнете програма, която отпечатва на екрана идентификатора на потребителя, стартирал програмата и идентификатора на неговата група.