АНГЕЛИН ЛАЛЕВ

ВЪВЕДЕНИЕ В ЛИНУКС

ТОМ I

2019

Въведение в Линукс, Том I

Онлайн издание.

Copyright© Ангелин Лалев; Автор: Ангелин Лалев

ISBN: ……

Настоящата книга е лицензирана под Creative Commons CC BY-NC-ND 4.0 лиценз. Пълният текст на лицензa се намира на следния адрес: https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode

СЪДЪРЖАНИЕ НА ТОМ I

ПРЕДГОВОР 7

Как да използваме тази книга? 13

Глава 1. Първи стъпки с Линукс 17

1. Достъп чрез системната конзола 17

2. Отдалечен достъп от Windows система 21

2.1. Отдалечен достъп чрез използване на PuTTY 22

2.2. Отдалечен достъп чрез използване на SSH 36

3. Сигурно прехвърляне на файлове между Windows и Линукс система 43

3.1. Прехвърляне на файлове с WinSCP 45

3.2. Прехвърляне на файлове със scp 50

4. Казус – рестартиране на системата 53

5. Упражнения 58

Глава 2. Работа с файлове и директории 63

1. Основни понятия при работата с файлове и директории 63

1.1. Файлове и директории 63

1.2. Текуща директория и относителни пътища 67

1.3. Основни детайли за организацията на файловата система при Линукс 74

2. Работа с директории 80

2.1. Създаване на директории 80

2.2. Премахване на директория 82
2.3. Показване на съдържанието на директория.....83
3. Работа с файлове 96
3.1. Създаване на празни файлове.....96
3.2. Копиране на файлове.....97
3.3. Преместване на файлове 104
3.4. Изтриване на файлове 106
4. Твърди и символични връзки към файлове 111
4.1. Твърди връзки 112
4.2. Символични връзки.....121
5. Търсене на файлове с Findutils 124
6. Казус - стартиране на програма в chroot среда 151
7. Упражнения 168
Глава 3. Контрол на достъпа до файловете 173
1. Базова система за контрол на достъпа до файловете в Линукс 173
1.1. Собственост върху файловете и разрешения за достъп до тях.....173
1.2. Потребители и групи в Линукс 178
1.3. Команди за смяна на собственика и групата на даден файл 182
1.4. Команди за задаване права за достъп до файловете 191
1.5. Задаване на права за достъп в осмична бройна система 194
1.6. Разрешения за достъп до директории.....202
1.7. Setuid, setgid и „sticky” битове.....210

2. Списъци за контрол на достъпа.....223

3. Казус – потребителски директории на Apache сървър 251

4. Упражнения.....264

Приложение А – Инсталиране на Ubuntu Server 18.04 LTS 269

Приложение В – Инсталиране на CentOS 7.6 305

ПРЕДГОВОР

Най-вероятно повечето читатели, които отварят страниците на настоящата книга, вече са чували за Линукс и имат представа за някои от неговите възможности. От самото си създаване през далечната 1991 година, операционната система Линукс се радва на култов статус сред онези информатици, които имат амбициите да придобият пълно и дълбоко разбиране за работата на компютърните системи и софтуера, който ги управлява. Също като преди 20 години, когато авторът за пръв път инсталира тази операционна система на компютъра си, и днес начинаещите информатици се насочват към изучаването на Линукс заради възможността да се докоснат до вълнуващи иновации, да допринесат за реализирането на справедливи каузи и не на последно място – за да впечатлят своите приятели, споделящи интереса към информатиката. Това допринася за почти „виралното“ разпространение на Линукс, което продължава с главозамайваща скорост и днес в епохата на терабитовите връзки, социалните мрежи и 4K видеото в Youtube.

За разлика от преди 20 години обаче, днес Линукс е нещо много повече от средство за обучение и среда за експерименти. Понастоящем той задвижва повечето от облачните услуги, които потребителите считат за неразделна част от Интернет, а неговото ядро е основна част от популярната операционна система за мобилни телефони Android. Това на практика означава, че всеки потребител на Интернет използва Линукс директно или индиректно всеки ден.

Въпреки всичко това, ако си представим, че читателят е напълно незапознат с Линукс и се опитаме да обясним задоволително неговата същност и значение в кратък обем като настоящото въведение, ние ще бъдем изправени пред сериозно предизвикателство. Въобще не е лесно да се обясни накратко как и защо точно Линукс успя да надмогне конкуренцията на много други

безплатни и комерсиални операционни системи, за да направи прехода от софтуер-играчка към водещата операционна система в света.

Отправна точка при подобен опит без съмнение би следвало да бъде това, което от самото начало направи Линукс много различен от останалите софтуерни проекти на времето си. Линукс бе една от първите операционни системи, която възприе принципите на отворения код, формулирани от т.нар. „GNU General Public License“ или накратко - GPL. GPL е лицензионен договор, който постановява нещо много просто – потребителите на софтуера получават същите права да го модифицират и надграждат, както и да разпространяват направените промени, каквито има и самият автор.

За разлика от комерсиалния софтуер, чийто изходен код представлява зорко пазена търговска тайна, софтуерът, разработван под GPL лиценз, може да бъде модифициран от всеки с нужните умения, а направените промени, при условие че те ще се използват и от други потребители, трябва да се споделят на ниво програмен код с тях, така че и те на свой ред да могат да направят свои модификации. Именно стриктното придържане към GPL лиценза осигури това, че разработката на Линукс, дори по най-модерните стандарти на днешния ден, е демократична и отворена за нови идеи и експерименти. Резултатите не закъсняха и днес Линукс може да се похвали с феноменалните 15 хиляди разработчици само на ядрото на операционната система, както и с немислимите 26 милиона реда програмен код (отново само на ядрото), публично достъпни под GPL лиценз.

Тези цифри са само бледо изражение на невероятния успех, постигнат от Линукс. Линукс не бе първият проект с отворен код, но той бързо се превърна в най-успешният първопроходник и символ на това, което днес наричаме „движение за отворения код“. Редица общности от разработчици, окуражавани от успеха на Линукс общността, емулираха нейните методи, за

да разработят други успешни софтуерни продукти с отворен код. Такива например са уеб сървърите Apache и NGINX, сървърите за бази от данни PostgreSQL, MySQL и MariaDB, езиците за програмиране PHP и Python, които заедно задвижват над половината уеб сайтове в Интернет. Операционната система, под чието управление най-често се внедряват тези и много други невероятни продукти с отворен код, остава Линукс.

Линукс се оказа и идеалната операционна система за облачни дейта центрове, като тези, които стоят зад услугите на Google, Facebook и Amazon AWS. Той помогна на тези компании да избягнат нуждата да заплащат лицензионна такса за всяко едно копие на операционната система, инсталирано на десетките хиляди сървърни възли, необходими за обработването на милиарди заявки на ден. Днес тези компании осъзнават ролята на Линукс за своя успех и на свой ред допринасят за разработката и подобряването му.

Казаното до тук може и да не е най-доброто описание на Линукс, предназначено за напълно незапознати с материята, но със сигурност би следвало да убеди и най-големите скептици в това, че Линукс е навсякъде – от евтините лаптопи за деца в Африка до дейтацентровете на огромните технологични компании. Това е абсолютна гаранция, че както в близко, така и в много по-далечно бъдеще ИТ индустрията ще продължи да се нуждае от подготвени специалисти и системни администратори, които да работят с Линукс.

Ето защо Линукс ще продължи да бъде тема на неелегантно дебели, скучни, твърде подробни и написани в академичен стил учебници, към която група (поне авторът ѝ се надява) може да бъде причислена и настоящата книга. В много отношения тя е експеримент и сигурно остава далеч от поставените цели, но за сметка на това тя е безплатна и подлежи на постоянно подобрение.

Последното най-актуално издание на тази книга винаги ще може да бъде намерено на адрес **https://github.com/lalev-angelin/linuxbook/** или на блога на авгора – **https://angelinlalev.blogspot.com**. Авторът приема коментари и корекции по електронната поща на адрес **lalev@uni-svishtov.bg**.

Настоящата книга има за цел да обхване онези аспекти на операционната система Линукс, които формират фундаменталните знания, които даден системен администратор трябва да притежава. Тези знания остават релевантни за всяка една дистрибуция на Линукс и читателят може да очаква, че наученото от книгата ще му служи дълги години.

Това, което читателите не трябва да очакват да научат от тази книга, касае използването на конкретен приложен софтуер. Книгата няма да ги научи например как да инсталират уеб-сървъра Apache на най-новата версия на Arch Linux. Процедурата по инсталацията на този и подобен софтуер се променя от версия до версия, а читателите, които решат да сменят дистрибуцията на Линукс, ще открият неочаквани разлики в инсталационните процедури. Информация за подобни процедури може да бъде намерена в онлайн документацията, форумите за поддръжка, блогове и дори онлайн видеоматериали, които са идеална медия за разпространение на такъв вид информация. Авторът се надява обаче, че когато се стигне до по-сериозни задачи като писането на правила за преписване на URL адреси с mod_rewrite на Apache например, читателите на тази книга ще се чувстват по-сигурни в регулярните изрази, които са стандартизирани и се използват както от този модул на Apache, така и от много други софтуерни продукти.

Настоящата книга е предназначена за използване от две различни групи читатели. Първата от тях са читателите, които вече са запознати до някаква степен с Линукс и търсят източник на информация, който има по-дълбок и фундаментален обхват от типичните ръководства и онлайн

видео материали. Тези читатели без проблеми могат да използват книгата самостоятелно и да я четат в ред, който отговаря на знанията и интересите им. Авторът се надява, че тези читатели ще намерят в книгата повече релевантна информация, събрана на едно място и оформена по начин, който да помогне при евентуалната подготовка за професионална сертификация.

Втората група читатели са тези с минимален опит в информационните технологии, които тепърва навлизат в материята чрез курсове, водени от инструктор. В настоящата книга едва ли има материал, който да е извън възможностите на самостоятелен читател, който разбира основните концепции, свързани с работата с компютър. За съжаление материалът е обемист и несъмнено ще породи в подобен читател най-разнородни въпроси и проблеми. Разрешаването на подобни проблеми без помощта на инструктор или по-опитен потребител просто би отнело значително време, което читателите вероятно биха желали да икономисат.

Независимо дали става въпрос за студенти по информатика, курсисти в специално организирани курсове или читатели, работещи самостоятелно, начинаещите без предишен опит с Линукс трябва да усвоят материала в следните точки, преди да продължат по-нататък с цялата книга:

- Глава 1 (всички точки)
- Глава 2 (точки 1,2,3 и 4)
- Глава 3 (точка 1)

Участниците в тези курсове вероятно ще намерят за полезни приложения А и Б към първи том, които описват съответно процедурите по инсталация на Ubuntu Server и CentOS.

Авторът би искал да благодари на многото хора, които през годините запалиха интереса му към Линукс и по различни начини допринесоха за осъществяването на този проект. Без да има претенции за изчерпателност, този

списък включва Юрий Кузнецов, Емил Цанов, Станислав Грозев, Кирил Стоянов, Борислав Борисов, Дани Гюрчев, Николай Ненков, Радослав Василев, Бранимир Борисов, Ангел Генчев, Иван Мирчев, Борис Стоев, Калин Пенчев, Емил Георгиев, Галин Грозев, Иван Станев, Илиян Маринов и Иван Витков. Авторът би желал да благодари на семейството си и специално на съпругата си Елена и сина си Виктор. В известен смисъл всеки от дългите часове, прекарани в работа по тази книга е откраднат от тях.

Приятно четене!

КАК ДА ИЗПОЛЗВАМЕ ТАЗИ КНИГА?

За да послужат на читатели с основни знания, на много места в текста са вмъкнати подробни разяснения. Тези разяснения най-често описват някое по-специфично понятие от областта на информатиката или разиграват даден хипотетичен сценарий, за да обяснят как представяните концепции се вписват в по-широката картина на нещата. Текстовете на тези разяснения са обозначени с по следния начин:

Това е разяснение към текста.

Ако читателят се чувства несигурен в смисъла на някое от тези разяснения след едно-две прочитания, вероятно е по-безопасно да прескочи разяснението и да се върне към него след време или пък да потърси помощта на инструктор.

Авторът би желал да препоръча и някои добри практики, в полезността на които се е убедил лично през годините:

1. **Отпечатайте тази книга на хартия!**

Хартията има едно главно предимство пред електронните носители и то е, че тя се прелиства бързо. Скролирането до точно място в PDF файл остава в пъти по бавно спрямо прелистването на книжно тяло, независимо на какво устройство се чете този файл. Оказва се, че това по-бавно прелистване подсъзнателно кара читателя да избягва връщания назад в текста за повторно прочитане, припомняне и изясняване на неясни концепции. Всичко това пречи на много на ученето с разбиране.

По същия начин, книжното тяло се прелиства много по-бързо при повторно четене, когато се налага да се прескачат цели пасажи с вече усвоен учебен материал.

За улеснение на читателите печатни копия на настоящата книга могат да бъдат поръчани от https://eacademy.bg/

2. Водете си бележки!

Без съмнение всеки прочит на книгата ще поражда у Вас най-различни въпроси. Намирането на задоволителен отговор на тези въпроси много пъти ще изисква провеждането на експерименти, осмислянето на концепции или допитвания до по-опитни потребители. След време всичко това ще бъде забравено и при евентуално повторно прочитане на материала работата по намиране на отговори на същите въпроси ще започне отначало.

Бележките помагат за избягване на подобни неприятни ситуации. Ако в даден момент решите, че имате коментар към написаното, поставете го направо в полето. Загубата на естетика ще бъде повече от компенсирана, когато след месеци се върнете на същия въпрос и осъзнаете, че коментарът, написан по-рано, Ви е спестил часове работа.

Поставянето на бележки директно в книгата има предимството, че по този начин те не могат да бъдат загубени и работи толкова по-добре, колкото повече място има за поставянето им. Авторът често си позволява да отпечатва книгите, които чете, едностранно, като оставя гърба на всеки лист за коментари.

3. Не настоявайте задължително да прочетете тази книга „линейно“ от край до край!

Тази книга и подобните на нея често не могат да се четат като роман или новела. Много обучаеми настояват да ги четат по подобен начин и когато срещнат проблем, те спират и чакат с дни за да разработят решение или да изяснят възникналите неясноти. Това е разочароващо за тях и доста често те губят инерция, което води до това, че книгата никога не бива прочетена до край. Ако се сблъскате с подобна ситуация помнете, че на война понякога е

по-добре да отстъпите и оцелеете, за да се биете друг ден. Прескочете тази част и продължете по-нататък.

Също така, не се страхувайте да се връщате и препрочитате вече прочетеното, ако имате съмнения, че не сте схванали смисъла. Никой автор, най-малко този на настоящата книга, не може да се изразява по абсолютно ясен начин през цялото време. Свикнете с мисълта, че ще трябва да се потрудите, за да извлечете смисъла от често неясните бръщолевения на автора.

4. Търсете помощ!

Помощта от страна на по опитни потребители пести време, а ефективно използваното време е това, което прави разликата между успеха и провала в едно начинание!

Тези от Вас, които нямат приятели или познати с умения в областта на Линукс, могат да намерят помощ в редицата форуми и Фейсбук групи, посветени на темата. Авторът би посочил като най-важен форум в България общността „Линукс за българи“ - **http://www.linux-bg.org/**. Авторът не е особено сигурен дали този форум и групата във Фейсбук „Линукс за българи“ **https://www.facebook.com/groups/8050070055/** имат общо, но последната също е полезно място да намерите помощ. Преди да зададете въпроси в подобен форум или група, уверете се, че спазвате правилата и винаги бъдете възпитани. Хората, които отделят от времето си, за да оказват помощ, го правят от добра воля и не са длъжни на никого.

Помнете, че дори хора, които нямат опит и умения, могат да помогнат един на друг, тъй като един от тях ще схване нещо и ще го обясни на останалите, което ще се повтори при друг от тях и т.н. По този начин цялостният ефект от обучението ще бъде по-голям от приноса на отделните обучаеми.

Полезно е да се знае, че авторът организира, чрез своята фирма „Фомалхаут Компютинг“, курсове за начинаещи по Линукс. Участие може да бъде заявено на адрес **http://www.lalev.co.uk/linuxcourses/.**

ГЛАВА 1. ПЪРВИ СТЪПКИ С ЛИНУКС

1. Достъп чрез системната конзола

Под *системна конзола* в модерните Линукс системи най-общо се разбира комбинацията от клавиатурата и монитора на компютъра, които осъществяват непосредствен вход и изход.

Типичните дистрибуции на Линукс предлагат не една, а множество *виртуални* конзоли между които може да се превключва с клавишни комбинации. Първата виртуална конзола се активира с клавишната комбинация *Ctrl+Alt+F1*. Втората виртуална конзола се активира с *Ctrl+Alt+F2* и т.н.

Виртуалните конзоли изглеждат като показаната на долната разпечатка:

```
Ubuntu 18.04 LTS dragonfly tty2

dragonfly login:
```

Показаната в случая конзола е на машина, наречена ***dragonfly***, която работи под управлението на Линукс дистрибуция ***Ubuntu 18.04 LTS***. ***tty2*** означава, че това е втората виртуална конзола.

Редът ***dragonfly login:*** подканва потребителя да се идентифицира пред системата.

Идентификацията се извършва на две стъпки:

1. Потребителят въвежда потребителското си име и натиска клавиша <*Enter*>. При това конзолата извежда подкана за въвеждане на паролата – **Password**:

2. Потребителят въвежда паролата си и натиска <*Enter*>**.** При изписване на паролата, конзолата не извежда символи (дори звездички или точки), за да

предпази потребителя от ситуация, в която стоящ зад него човек би преброил от колко символа се състои тя.

Следващата разпечатка показва виртуална конзола след успешно влизане в системата на потребител с потребителско име ***lalev***.

```
Ubuntu 18.04 LTS dragonfly tty1

dragonfly login: lalev
Password:
Last login: Tue Apr  3 13:22:57 EEST 2018 on tty1
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-20-generic x86_64)

 * Documentation   https://help.ubuntu.com
 * Management      https://landscape.canonical.com
 * Support         https://ubuntu.com/advantage

 * Meltdown, Spectre and Ubuntu: What are the attack vectors,
   how the fixes work, and everything else you need to know
   - https://ubu.one/u2Know

 * Cannonical Livepatch is available for installation
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

0 packages can be updated.
0 updates are security updates.

lalev@dragonfly:~$
```

Повечето редове на горната фигура не се нуждаят от описание. Редът

```
Last login: Tue Apr  3 13:22:57 EEST 2018 on tty1
```

дава информация на потребителя за това кога последно някой с неговото име и парола е осъществявал достъп до системата. ***EEST*** в случая означава „източноевропейско лятно време” и идва от настройките на системата, които указват датите и времената да се показват в този часови пояс.

Редът също показва на коя конзола е бил извършен достъпът. В този случай, това е първата виртуална конзола.

Редът

```
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.15.0-20-generic x86_64)
```

показва версията на дистрибуцията **(*Ubuntu 18.04 LTS*),** вида на операционната система **(*GNU/Linux*),** версията на ядрото ***(4.15.0-20-generic)***, както и вида на хардуера (**x86_64**, което идентифицира 64 битова машина**)**.

Следващите редове информират потребителя къде може да намери помощ и поддръжка за дистрибуцията на операционната система. Те са част от така нареченото „съобщение на деня“. Администраторът може да постави в съобщението на деня всякаква информация, която е от полза за потребителите.

Редът

```
0 packages can be updated.
```

информира потребителя, че няма софтуерни пакети, които могат да бъдат обновени.

Софтуерните пакети в Линукс са начинът, по който се инсталират допълнителни (обикновено безплатни) приложения, като например текстови редактори, компилатори и пр., като работата с тях донякъде наподобява софтуерните магазини на Android и Windows. Софтуерните пакети обикновено се обновяват поради необходимостта да се поправят открити грешки[1] в тях. Има обаче особен вид грешки, които могат да се експлоатират от злонамерени лица за заобикаляне на мерките за сигурност и осъществяване на нерегламентиран достъп до системата.

Редът

```
0 updates are security updates.
```

информира потребителя, че 0 от тези пакети са обновления, наложени поради критични грешки, които компрометират информационната сигурност на системата.

[1] „Грешките“ в софтуера са плод на недомислици и недоглеждания на програмистите, които водят до проблеми с функционалността на софтуера.

Грешки в софтуера, излагащи сигурността на системата на риск, са особено нежелани от системните администратори. Ето защо те се стараят да инсталират възможно най-бързо обновленията в софтуера, които премахват подобни грешки.

Текстът на последния ред - ***lalev@dragonfly:~$*** - представлява т.нар. „промпт“, чието предназначение е да напомни на потребителя, че системата е готова да приема команди от него.

Освен тази основна функция, промптът показва и друга важна информация. Така например горният промпт напомня, че сме влезли в системата като потребител ***lalev***, а самата система се казва ***dragonfly***. Тилдата (~) указва, че работим в домашната директория на потребителя ***lalev***. Знакът ***$*** указва, че потребител ***lalev*** не е администратор на системата.

Към момента на показване на промпта, входът в системата е осъществен успешно. На професионален език може да наречем взаимодействието на потребителя със системата „сесия“. Сесията включва осъществяване на вход, изпълнение на команди и получаване на отговор на командите.

Програмата, която се стартира след успешния вход и отговаря за получаването на командите и изпълнението им, се нарича „обвивка“. Промптът, дискутиран по-горе, се извежда именно от нея.

Ако в даден момент потребителят реши да прекъсне сесията, той може да направи това по много начини. Така например, ако потребителят напише командата ***exit*** и натисне клавиша *<Enter>*, който обозначава край на въвеждането на команда и начало на изпълнението ѝ, това ще накара обвивката да прекъсне сесията. Същото ще се случи, ако потребителят натисне комбинацията *Ctrl+d* когато обвивката очаква въвеждане на нова команда.

2. Отдалечен достъп от Windows система

Много често се налага сървърите да бъдат управлявани отдалечено. Това е типично не само за Linux сървърите, но и за всички сървъри по принцип, тъй като сървърите обикновено се намират в специализирани помещения. Тези помещения често могат да бъдат отдалечени физически от мястото, на което се намират администраторите.

Стандартен начин за организиране на отдалечения достъп в Linux е чрез използване на криптографския протокол SSH. Този протокол позволява на системния администратор да се свърже с отдалечената машина и да въвежда команди по начин, който е идентичен с достъпа от системната конзола. При това, SSH използва криптографски средства, които предотвратяват евентуалното подслушване на връзката между администратора и сървъра с цел неоторизирано извличане на информация.

За да може да се осъществи достъп до Linux система от Windows, е необходим специален SSH клиент. Съществуват множество SSH клиенти, но в настоящото изложение ние ще разгледаме два. Първият от тях – PuTTY е утвърден и безплатен софтуер, който обаче трябва да бъде инсталиран допълнително на Windows системата. PuTTY работи под всички версии на Windows.

Вторият клиент е достъпен само под управление на Windows 10 (Fall Creators Update и по-късни) и неговото предимство е, че той идва като част от операционната система. За да се използва този клиент, той все още трябва да бъде активиран като част от операционната система. Веднъж активиран обаче, грижите по обновяването на клиента се поемат от Windows Update. По този начин потребителят не трябва сам да следи кога има нужда да обнови клиента си и не трябва да изпълнява операциите по обновяването ръчно.

За да се осъществи достъп е нужен специален софтуер и на сървъра, към който SSH клиентът иска да осъществи връзка. Този софтуер се нарича „SSH сървър“. Повечето Linux машини, които се използват отдалечено, имат инсталиран и работещ SSH сървър от момента на инсталацията си. От друга страна, Linux се използва и за обикновени „десктоп“ машини, на които хората използват графичен потребителски интерфейс и софтуер като текстообработващи програми, браузъри и т.н. Рядко има нужда тези машини да бъдат управлявани отдалечено, поради което при тях SSH сървърът не е инсталиран по подразбиране.

За да инсталирате SSH сървър на Вашата Ubuntu, Debian или Linux Mint машина като администратор трябва да изпълните командата ***apt-get install openssh-server.*** За дистрибуциите от родословното дърво на Fedora, командата е ***yum install openssh-server***.

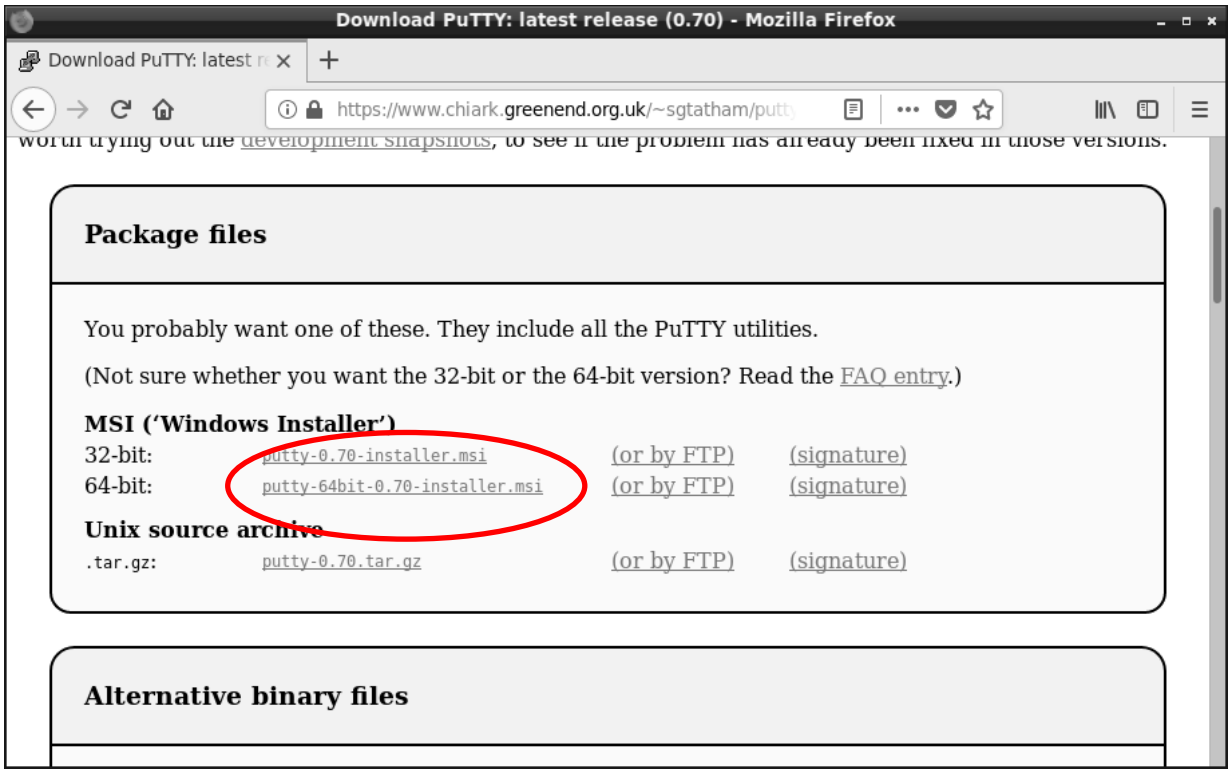
В контекста на софтуера, който комуникира по Интернет, думите „клиент“ и „сървър“ обозначават двете крайни точки на връзката. Софтуерът, по чиято инициатива се осъществява връзката, се нарича винаги „клиент“, докато софтуерът, който изчаква в готовност клиентите да се свържат към него, се нарича „сървър“. Така например потребителските браузъри като Edge и Google Chrome са клиенти, които разговарят по Интернет с други програми, наречени “уеб сървъри“ чрез протоколите HTTP или HTTPS.

Понякога администраторите разбират под сървър целия компютър, докато друг път имат предвид конкретно софтуерните програми с такава роля. Така например, ако компютърът се използва основно за сервиране на уеб страници, той е уеб сървър, независимо, че освен софтуер за реализиране на уеб сървър, на него ще работят и други сървърни приложения като например сървър за електронна поща, SSH сървър и т.н. Това двусмислие може да затрудни начинаещите, но на практика е винаги ясно от контекста дали става дума за целия компютър или за конкретно софтуерно приложение.

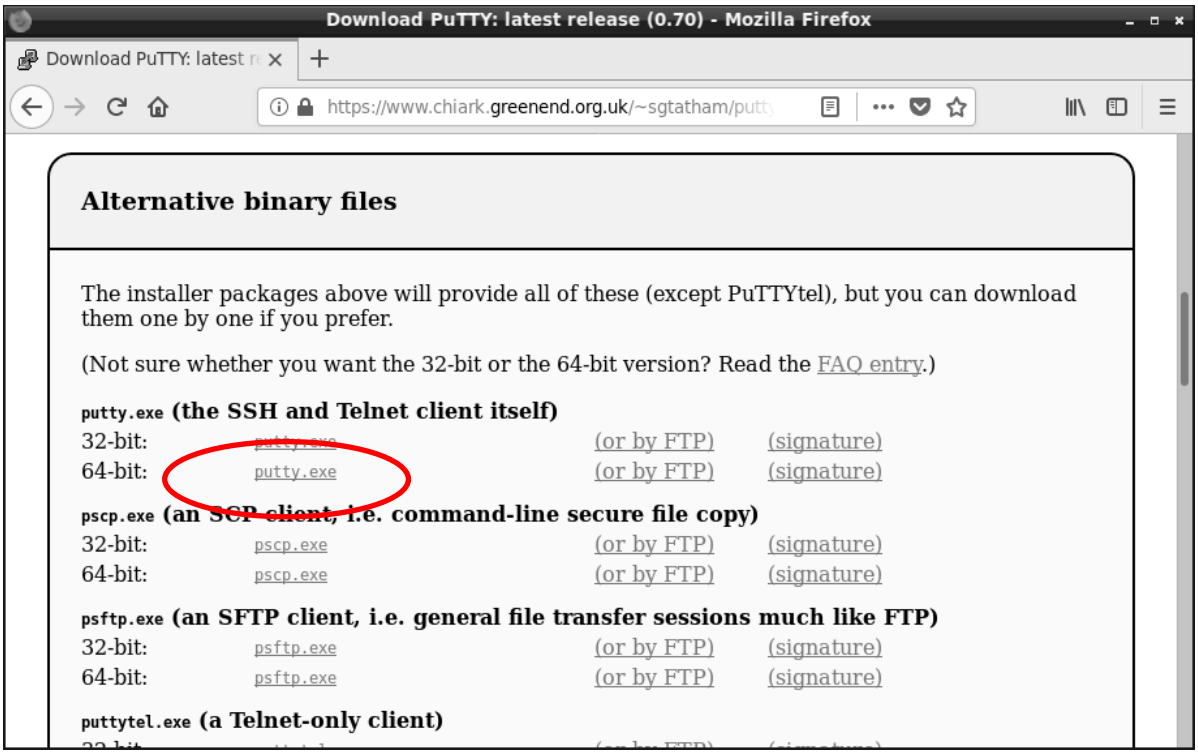
2.1. Отдалечен достъп чрез използване на PuTTY

За да се използва ***PuTTY***, клиентът първо трябва да бъде изтеглен от Интернет. ***PuTTY*** може да бъде изтеглен от адрес **https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html**

Страницата на този адрес предлага ***PuTTY*** като инсталационен **.msi** пакет или като готова за изпълнение програма, която не се нуждае от инсталация. Въпрос на предпочитания е кой от двата варианта ще бъде избран.



Фигура 1.1. Изтегляне на PuTTY от официалния сайт



Фигура 1.2. Изтегляне на алтернативен преносим изпълним файл

Фиг. 1.1. и фиг. 2.1 показват точното местоположение в страницата, от което следва да се изтегли ***PuTTY***. Ако искаме инсталационна програма, ние ще я намерим в секция ***Package Files*** в два варианта – за по-стари 32-битови компютри и за по-новите 64-битови компютри.

Ако се интересуваме само от изпълнимия файл, който не се нуждае от инсталация, той може да бъде намерен по-надолу в страницата – ***putty.exe*** в секция ***Alternative binary files***, като отново идва в два варианта – за 32 битови и 64-битови компютри.

Инсталацията на ***.msi*** файла е тривиална. В нейния ход ще бъде създадени елемент в стартовото меню и (евентуално) икона на десктопа. Що се отнася до ***putty.exe***, веднъж свалена, програмата просто трябва да бъде стартирана (например с двойно щракване на мишката върху ***putty.exe***).

При стартиране, ***PuTTY*** извежда основния диалог за конфигурация и връзка (вж. фиг. 1.3.), който е организиран по секции. Имената на секциите се показват като дървовиден изглед в лявата част на прозореца, докато съдържанието на активната секция се показва вдясно.

При стартиране на ***PuTTY***, активната секция винаги е ***Session***, тъй като основните данни, необходими за осъществяване на връзката, се намират в нея.

За да осъществи връзка с отдалечен сървър, потребителят трябва да знае IP адреса или домейн името на сървъра, което се въвежда в полето ***Hostname (or IP address)***.

IP адресът е последователност от числа (например 195.20.24.18 или 2a00:1450:4017:807::2004), която уникално идентифицира всеки компютър в Интернет. IP адресите изпълняват същата роля в Интернет, каквато играят телефонните номера в телефонната мрежа. Те позволяват да укажем с коя машина искаме да осъществим връзка по Интернет.

Свързването към отдалечена машина в Интернет не може да се извърши без да имаме IP адреса и, но тъй като числата в IP адресите са неудобни за запомняне от обикновените потребители, от много години в Интернет вместо тях се използват домейн имена като ***bimoodle.uni-svishtov.bg*** или ***www.google.com***. „Преводът“ от домейн имена към IP адреси се

извършва автоматично и в реално време от система, наречена Domain Name System (DNS), която прилича в много отношения на електронен телефонен указател.

Също така потребителят трябва да знае и номера на порта, под който работи програмата – SSH сървър на отдалечената машина.

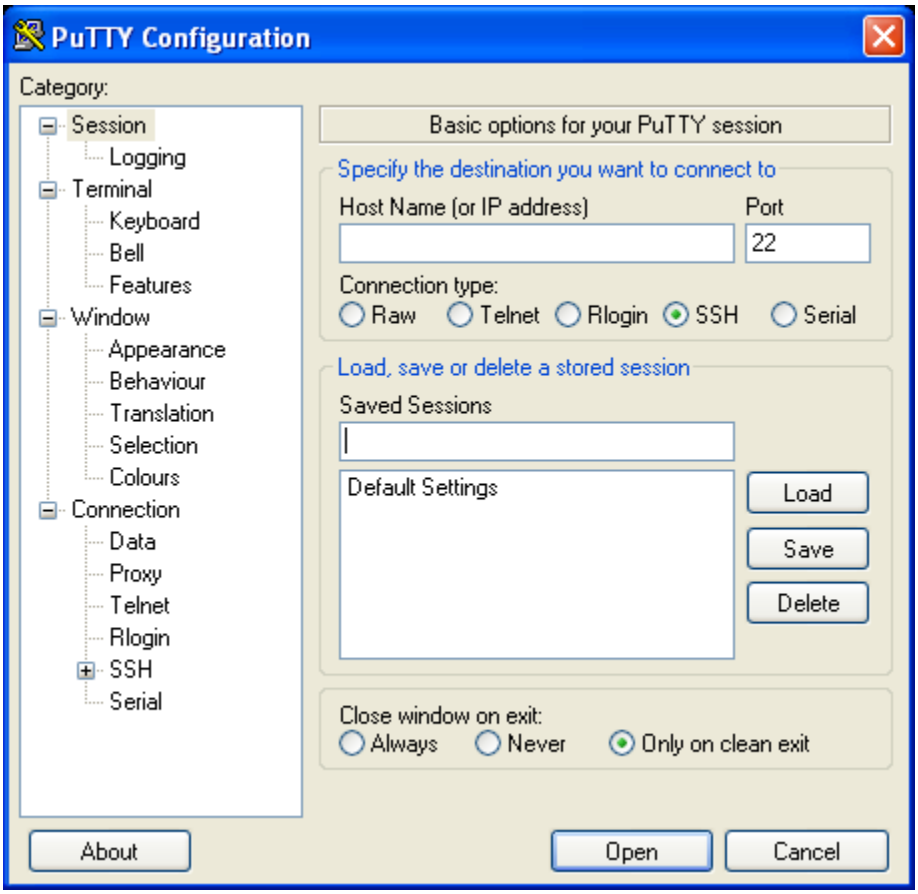
Портовете са цели положителни числа и най-общо служат за уникална идентификация на тези работещи програми на даден компютър, които реализират мрежова функционалност. Наличието на портове позволява на дадена физическа машина да работят едновременно много и различни софтуерни приложения, които реализират различна сървърна функционалност.

За да се свърже със сървъра, клиентското приложение трябва да посочи както IP адрес, така и порт, тъй като IP адресът указва уникално машина в Интернет, а портът указва уникално кое приложение – сървър на машината ще обслужва нашия клиент. За да се избегне обаче обременяването на потребителя с нуждата да помни повече информация, съществуват стандарти, които указват фиксирани номера на портове за различните видове сървърни приложения.

Така например се очаква уеб сървърите да работят на порт 80. По този начин, ако потребителят не е указал специфично порт, браузърът се опитва да се свърже с уеб сървъра на отдалечената машина на порт 80 по подразбиране, докато клиентите за електронна поща, ако не е указано друго, очакват да се свържат с отдалечените пощенски сървъри на портове 25, 110 и 143.

Понякога администраторите преместват портовете на услугите, които не се използват от много клиенти, с намерението да попречат на хакерите, които се опитват да определят какви сървърни приложения работят на различни машини от мрежата чрез автоматизирани приложения, способни да „сканират“ десетки хиляди IP адреси в час.

Ако администраторът на отдалечената машина не е променил стандартния порт, на която работи софтуерът, реализиращ SSH сървър, този порт ще бъде 22. Тази стойност е попълнена и в полето ***Port*** на секция ***Session***.



Фигура 1.3. Секция „Session”

Секция ***Connection type*** указва какъв е видът на отдалечения сървър, с който ще се свързваме. ***Connection type*** почти винаги трябва да бъде ***SSH***, понеже другите алтернативи са или стари и несигурни протоколи за връзка или – алтернативно - са специални режими на комуникация, които се използват рядко.

След попълването на адрес (и евентуално порт), осъществяването на връзката може да започне чрез натискането на бутон ***Open***.

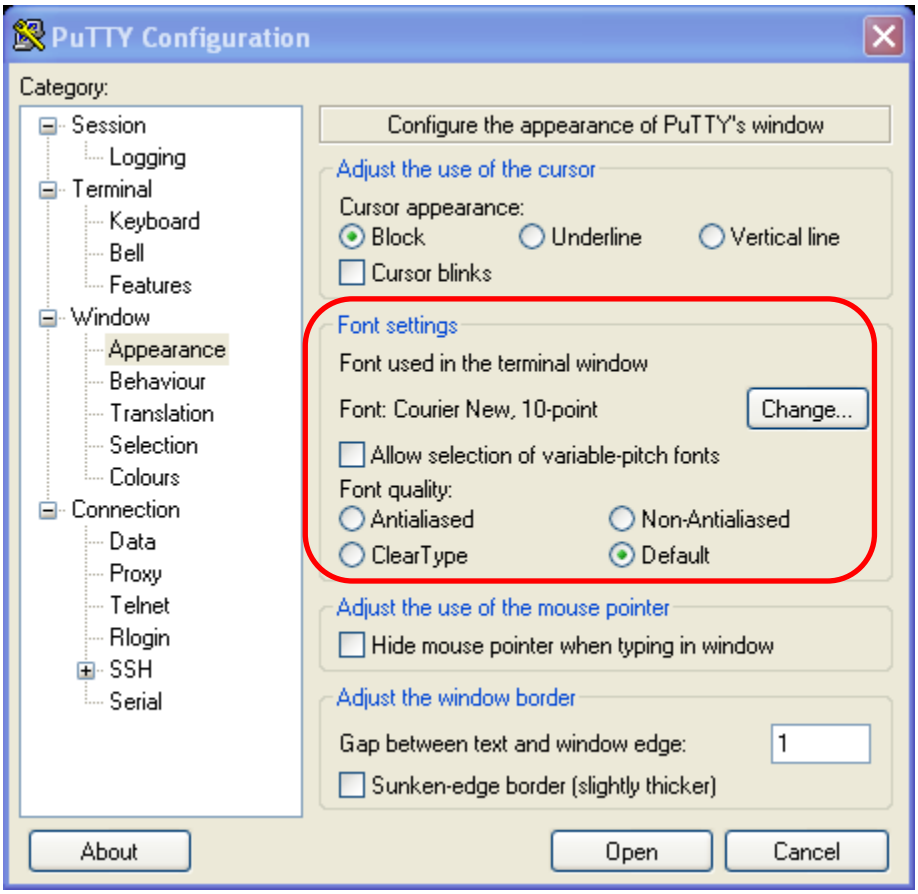
На пръв поглед, свързването с отдалечена система изисква въвеждането на малко на брой детайли. Това е така обаче само в най-типичните случаи. Трябва да се отбележи, че всички секции, разположени вляво, съдържат

буквално стотици специални настройки. Въвеждането на тези настройки всеки път, когато се осъществява връзка, може да се превърне в тежко бреме.

За да се избегне това, ***PuTTY*** предлага прост, макар и не толкова интуитивен начин за записване на вече въведените настройки. След като настройките бъдат въведени във всяка от нужните секции и преди да бъде осъществена връзка с бутона ***Open***, потребителят може да се върне в секция ***Session*** и да запише уникално име в поле ***Saved Sessions*** (най-често това е някакво описателно име на сървъра, към който се свързваме). След като запише име, потребителят може да щракне върху бутона ***Save***, което ще доведе до записване на всички настройки под това име, което ще се появи в списъчното поле под ***Saved Sessions***.

При повторно стартиране на ***PuTTY***, така записаните настройки могат да бъдат заредени чрез щракване върху името им и натискане на бутона ***Load***.

От множеството разширени настройки на ***PuTTY***, най-голям интерес за начинаещите вероятно представлява секция ***Appearance***, която позволява промяна на размера на шрифта, използван за виртуалната конзола, която ***PuTTY*** ще визуализира след осъществяването на връзката. Фиг. 1.4. показва настройките и бутона ***Change***, от който се осъществява промяната на шрифта.



Фигура 1.4. Секция ***Appearance*** *с оградени настройки на шрифта*

След като потребителят е въвел всички нужни настройки, той може да започне процеса на свързване с отдалечения сървър с натискането на бутона ***Open***.

Осъществяването на връзка протича на два етапа:

На първия етап от осъществяването на връзката, ***PuTTY*** клиентът проверява самоличността на сървъра и осъществява шифрирана връзка с него. За да провери самоличността на сървъра, клиентът сравнява криптографския отпечатък от публичния ключ на сървъра с този, съхранен на локалната система.

На втория етап сървърът проверява самоличността на клиента, като обикновено използва за целта напълно същия механизъм като при директния достъп от системната конзола, тоест чрез въвеждането на потребителско име и парола. Този метод е допустим, тъй като връзката на този етап е вече шифрирана и защитена от вмешателства.

Докато вторият етап е напълно подобен на гореописаната процедура по осъществяване на достъп чрез системната конзола, първият етап се нуждае от малко по-подробно обяснение.

Както бе вече обяснено, SSH използва криптографски средства за защита на връзката от подслушване. Тези криптографски средства са изключително ефективни при спиране на т.нар. „*пасивно*“ подслушване, но се нуждаят от малко помощ за да спрат и по-изтънчените атаки, които могат да бъдат определени като „*активно*“ подслушване.

Пасивното подслушване се изразява в това, че някоя от междинните системи, които препращат трафика между клиента и сървъра в Интернет, може да го записва с цел да извлече пароли за достъп (както впрочем и друга информация от интерес), които да бъдат използвани по-късно за осъществяване на неоторизиран достъп.

Съществуването на такива междинни системи не може да бъде избегнато, тъй като те са залегнали дълбоко в начина на изграждане и работата на Интернет. Тези междинни системи често се намират далеч извън контрола на потребителя (друга организация, друг град, друга държава), поради което той няма гаранции за тяхната сигурна работа.

Пасивното подслушване от междинните системи не оставя забележими следи и за крайните страни е много трудно, ако не и невъзможно, да установят дали са обект на подслушване. За щастие, вградените в ***PuTTY*** алгоритми за шифриране правят пасивното подслушване напълно непрактично. Така

например Дифи-Хелман обменът на тайни ключове, използван от SSH, позволява клиент и сървър, които предварително не знаят нищо един за друг и никога не са обменяли тайна информация (например кодови книги), да договорят тайна „парола“ дори при условие, че евентуален подслушвач може да подслуша абсолютно всички разменени между страните съобщения. Получената тайна парола може да се използва за шифриране на връзката между клиента и сървъра по начин не особено по-различен[2] от възприетия при немската шифровъчна машина „Енигма“ от Втората Световна Война.

Активното подслушване се изразява в промяна на данните, които преминават по линията. То е по-специфично за Интернет, където междинните системи преповтарят получените данни по пътя им между клиента и сървъра.

Активното подслушване крие много по-сериозен риск от разкриване, поради което то се практикува значително по-рядко и по-целенасочено. От своя страна възможността за промяна на данните позволява много по-изтънчени и ефективни атаки срещу сигурността на криптографските протоколи, които нормално биха предотвратили пасивното подслушване без проблем. Атакуващ, извършващ активно подслушване, може да се представи пред едната страна за другата и обратното (известно атака от тип „Човек по средата“), което обезсилва иначе гениални алгоритми като споменатия по-горе Дифи-Хелман обмен.

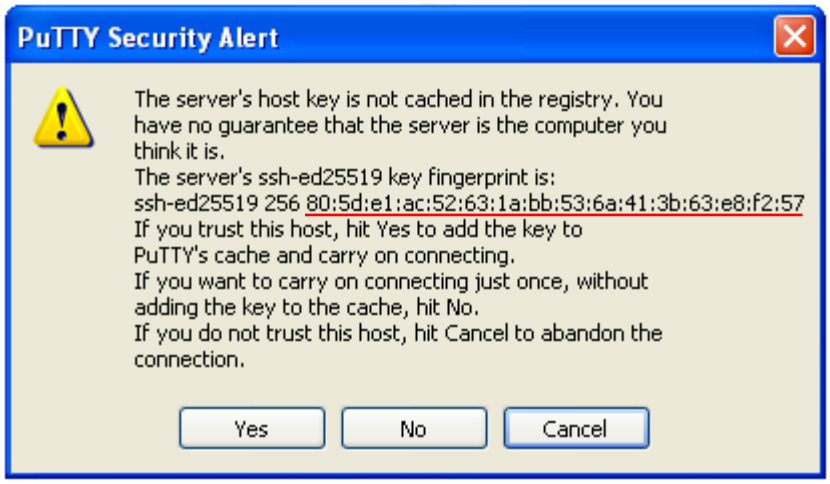
За да се избегне активното подслушване, при всяко свързване към даден сървър, SSH клиентът трябва да провери дали криптографският ключ на сървъра не е подменен от евентуален активен подслушвач, който се опитва да се представи за истинския сървър.

[2] Трябва да се отбележи, че модерните симетрични шифри, чиито прародител е “Енигма“, са милиони пъти по-устойчиви на криптографски анализ, когато са използвани по правилния начин.

Тъй като ключът на сървъра обикновено не се променя (освен ако сървърът не е основно преинсталиран), той може да се запише на диска на клиента още първия път, когато клиентът се свърже към сървъра, след което да бъде сверяван всеки следващ път, когато клиентът се свързва към същия сървър. Евентуално разминаване между записаната версия и получената по мрежата е много силен индикатор за активно подслушване, което се случва в дадения момент.

За съжаление, за да бъде записан ключа на диска на клиента, той трябва да дойде от сървъра, което поражда своеобразен проблем. Ако потребителят разчита само на Интернет, за да получи ключа на сървъра, евентуален атакуващ може да успее да прихване още първото свързване на клиента и да подмени ключа при предаването му от сървъра, което ще доведе до това, че фалшивият ключ ще бъде записан на диска на клиента.

За да се справи с проблема, ***PuTTY*** извежда съобщение първия път, когато потребителят се свързва с нов сървър. Съобщението съдържа споменатия криптографски *отпечатък* на ключа на сървъра (фиг. 1.5.).



Фигура 1.5. Съобщение за проверка на криптографския отпечатък на публичния ключ на сървъра при първо свързване с него

Отпечатъкът е 256 битово число, представено чрез поредица от шестнадесетични цифри, разделени с двоеточия за по-голяма четливост (вж. подчертаното на диаграмата).

Системните администратори, които държат на максимална сигурност, следва да сверят този отпечатък с отпечатъка, който е генериран присъствено на отдалечения сървър. Така например администратор, който има физически достъп до системната конзола на сървъра ще извика

```
sudo ssh-keygen -l -E md5 -f /etc/ssh/ssh_host_ed25519_key.pub
```

на конзолата и ще запише (например на лист хартия) изхода от командата, който за показания сървър изглежда по следния начин

```
256     MD5:80:5d:e1:ac:52:63:1a:bb:53:6a:41:3b:63:e8:f2:57     root@bi
(ED25519)
```

При първото отдалечено свързване, администраторът ще свери показания в диалоговия прозорец на PuTTY отпечатък със записания. Съвпадението на двата отпечатъка гарантира, че никой не се опитва да се представи за сървъра.

Ако не е възможно за един администратор да генерира отпечатък присъствено на сървъра преди първото отдалечено свързване към него, е възможно отпечатъкът да се предаде по друг информационен канал между администратор, който има локален достъп до сървъра, и този, който няма. Достатъчно е каналът да бъде различен от Интернет, което ще осигури че евентуален подслушващ няма да може лесно да прихване и промени информацията и по двата канала. Такъв алтернативен канал може да бъде телефонен разговор или дори куриер с дипломатическо куфарче, в зависимост от това каква сигурност е нужна.

В много случаи дори и такова ниво на сигурност не нужно, тъй като описаният протокол по сигурността е доста обезкуражаващ за евентуални атакуващи, които искат да останат *незабелязани*.

Атакуващ, който иска да остане незабелязан, трябва да успее да подмени ключа на сървъра при първото и всяко последващо свързване. Дори атакуващият да е уверен във възможностите си да извърши подобен технологичен подвиг, това не му гарантира незабележимост, тъй като той никога няма как да знае дали потребителят няма да избере да валидира получения ключ (например по телефона). Именно тези обстоятелства правят активното подслушване проблематично и то се използва много по-рядко.

Ако двата отпечатъка съвпадат, нормалният избор е да се натисне бутона ***Yes*** в диалога за проверка. Натискането на бутона ***Yes*** ще доведе до

продължаване на свързването и записването на криптографския ключ на сървъра на диска на клиента. Оттук нататък ***PuTTY*** автоматично ще сверява подадения от сървъра ключ със записаната версия при всяко следващо свързване.

Ако двата отпечатъка не съвпадат, потребителят трябва да натисне ***Cancel***, което моментално ще прекрати опита за свързване. Несъвпадението в този случай е почти сигурен индикатор за активно подслушване или системна грешка, като и в двата случая най-мъдрото решение е да се избягва свързването до отстраняването на проблема.

Начинаещите администратори, които сравняват криптографските отпечатъци, трябва да помнят, че тези отпечатъци често представят едно и също число в различни формати, които изглеждат различно.

Натискането на бутона ***No*** ще доведе до продължаване на свързването, но ключът на сървъра няма да бъде записан на диска. Тази опция би била логичен избор само при наличието на много специфични обстоятелства, поради което начинаещите администратори следва да я избягват.

Интересен предмет на дискусия е и какво ще се случи, ако ключът, който ***PuTTY*** е записал при първо свързване със сървъра се промени на някое от последващите свързвания? В тази ситуация ***PuTTY*** ще изведе предупредителен екран (фиг. 1.6.).

Фигура 1.6. Предупредително съобщение при разминаване между получения и записания отпечатък на сървъра

Екранът подсказва на потребителя, че съществува неприемливо голям шанс да е (или да е бил в миналото) жертва на атака с подслушване, поради което трябва да направи сериозни усилия да свери полученият отпечатък с този на истинския сървър.

Ако се окаже, че има валидна причина новият отпечатък да съвпада с този на истинския сървър, а в същото време да се разминава със записания при клиента (защото може би сървърът е бил преинсталиран и старият криптографски ключ, записан при клиента, се разминава с новия), потребителят следва да натисне бутона ***Yes***, което ще запише новия ключ вместо стария.

Ако потребителят не е 100% уверен, че новият отпечатък е на сървъра, той трябва да прекрати връзката като натисне бутона ***Cancel*** и да потърси причините за този инцидент.

След безпроблемното приключване на първия етап, потребителят ще бъде запитан от сървъра за потребителско име и парола, с която потребителят доказва своята самоличност. Както бе споменато, този втори етап е аналогичен на свързването през конзолата. Фигура 1.7. показва екран на PuTTY след осъществяване на връзка и удостоверяване.

```
owner@bi: ~
login as: owner
owner@195.20.24.18's password:
Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.4.0-45-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue May 22 16:44:11 EEST 2018

  System load:  0.0                Processes:             172
  Usage of /:   39.0% of 3.81GB    Users logged in:       0
  Memory usage: 32%                IP address for ens33: 195.20.24.18
  Swap usage:   0%

  => /var is using 91.1% of 13.66GB

 * Meltdown, Spectre and Ubuntu: What are the attack vectors,
   how the fixes work, and everything else you need to know
   - https://ubu.one/u2Know

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

2 packages can be updated.
0 updates are security updates.


No mail.
Last login: Tue May 22 14:45:49 2018 from 91.222.23.68
owner@bi:~$
```

Фигура 1.7. Екран на PuTTY след свързване

Видно е, че потребителят получава информация, аналогична на тази, която получава при локално свързване от конзолата.

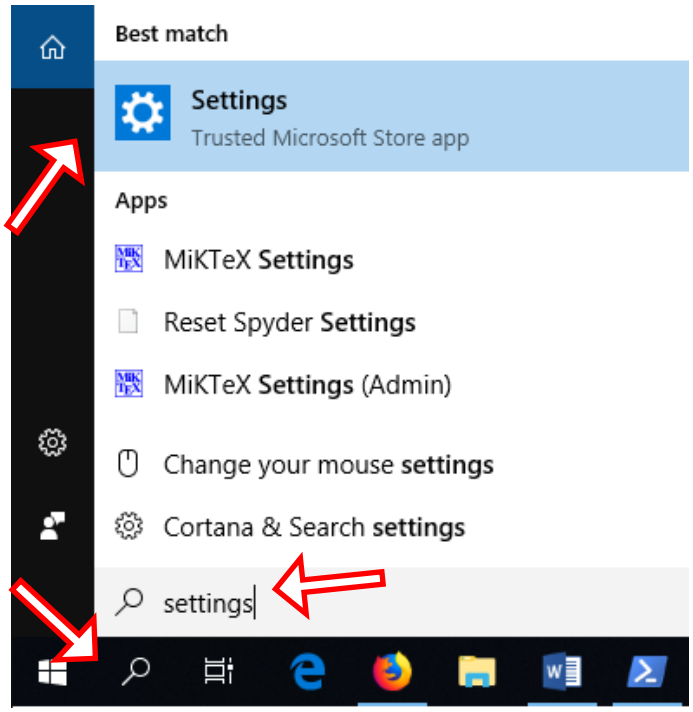
2.2. Отдалечен достъп чрез използване на SSH

Както вече бе споменато, използването на SSH под Windows 10 не изисква инсталиране на допълнителен, външен за операционната система софтуер. Вместо това, клиентът на Windows трябва да бъде „активиран“.

Клиентът, който е вграден в Windows всъщност е базиран на същия клиент, който се използва под Linux от дълги години – OpenSSH. „Пренасянето“ на кода на OpenSSH под Windows е резултат от усилията на Microsoft да направят своята операционна система по-приятелски настроена към Линукс и неговите потребители, като в същото време копират най-добрите практики в администрацията на сървъри и в своята система.

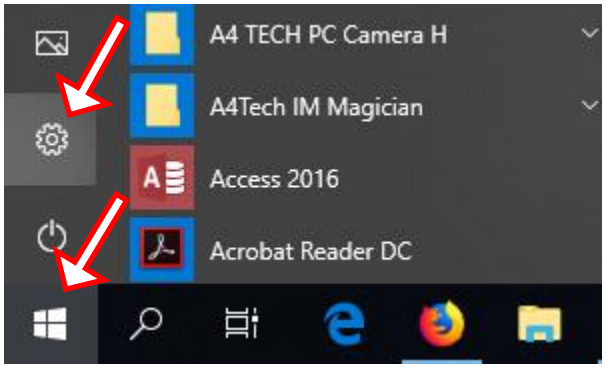
По същество, „активирането“ представлява своеобразна инсталация. При нея обаче изтеглянето и конфигурирането на софтуера става автоматично. Което е по-важно, след активирането потребителят не трябва да се тревожи за ръчното инсталиране на обновления, Windows ще върши това вместо него.

Активирането на ***OpenSSH*** клиента става от контролния панел на Windows 10, който може да се стартира чрез въвеждането на ***Settings*** в полето за търсене на приложения и стартирането на показания елемент от менюто (вж. фиг. 1.8).



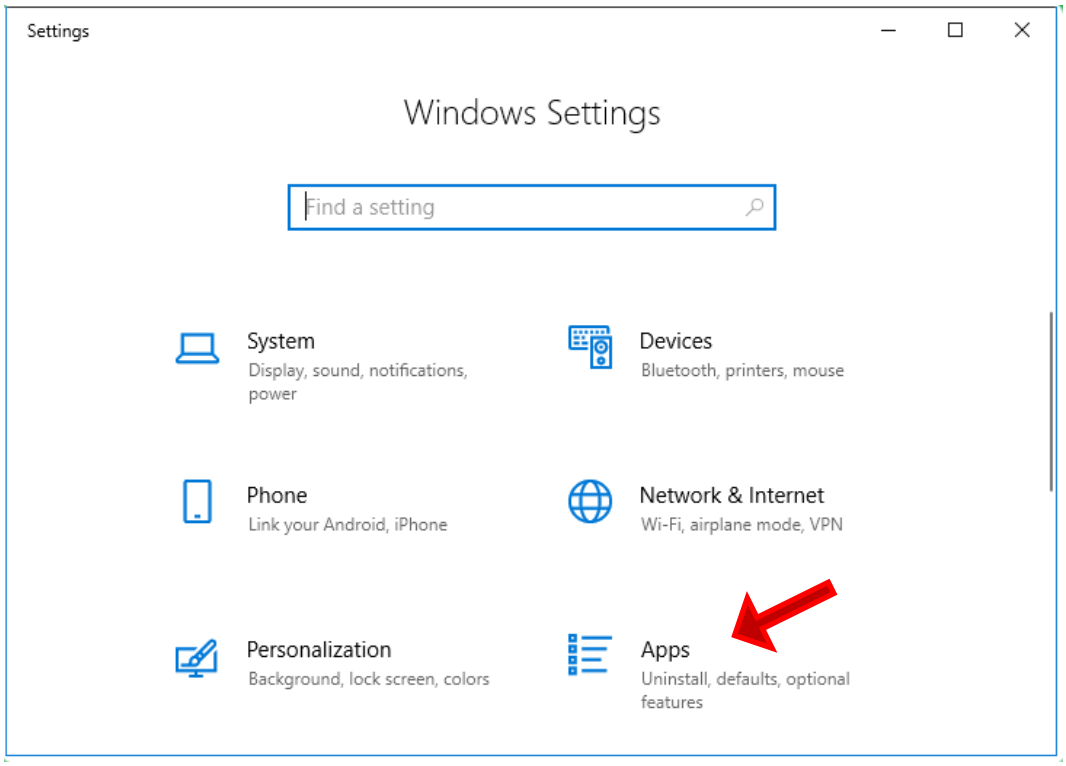
Фигура 1.8. Стартиране на Settings от стартовото меню

По същия начин, контролният панел може да се стартира чрез натискане на старт бутона и избиране на бутона със същото име в стартовото меню (вж. фиг. 1.9.).



Фигура 1.9. Извикване на Settings чрез бутона в стартовото меню

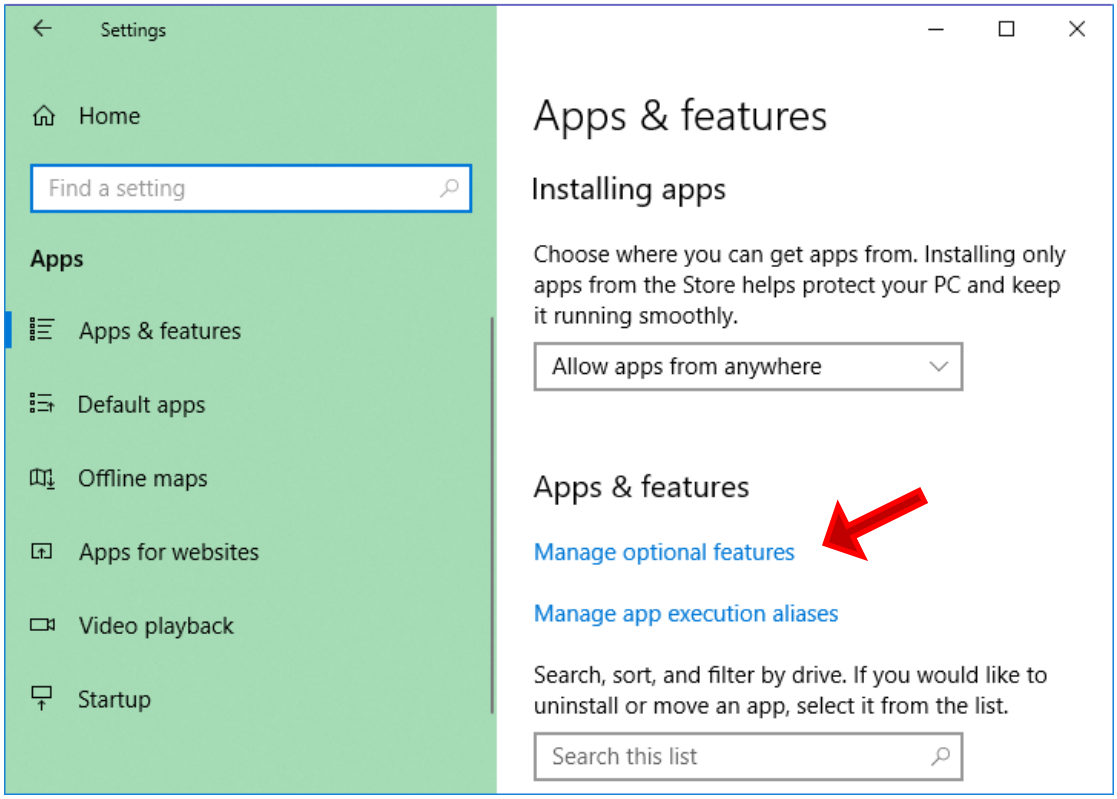
При стартирането на ***Settings***, потребителят може би вижда прозорец, подобен на този, показан на фиг. 1.10.



Фигура 1.10. Основен прозорец на Windows Settings

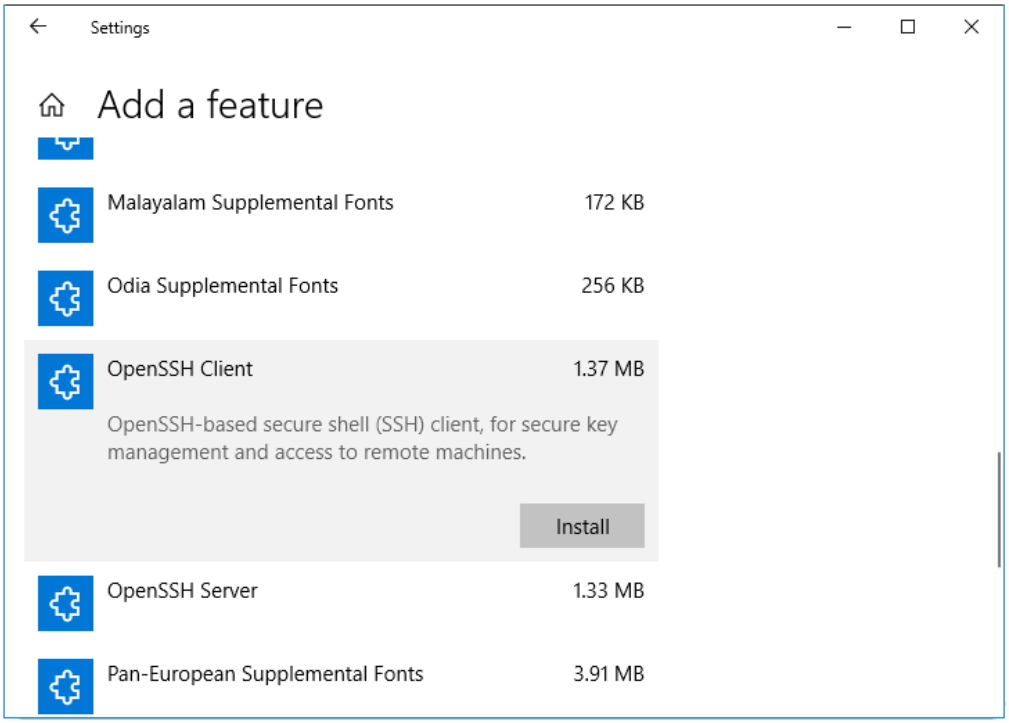
Дори този прозорец да се разминава с показаното като разположение, той със сигурност има елемент ***Apps***. Потребителят трябва да избере именно този елемент, за да може да активира различни модули на операционната система.

Щракването ще препрати потребителя към секция ***Apps & Features***. От там потребителят трябва да щракне върху връзката ***Manage optional features*** (Вж. фиг 1.11).



Фигура 1.11. Секция Apps & Features

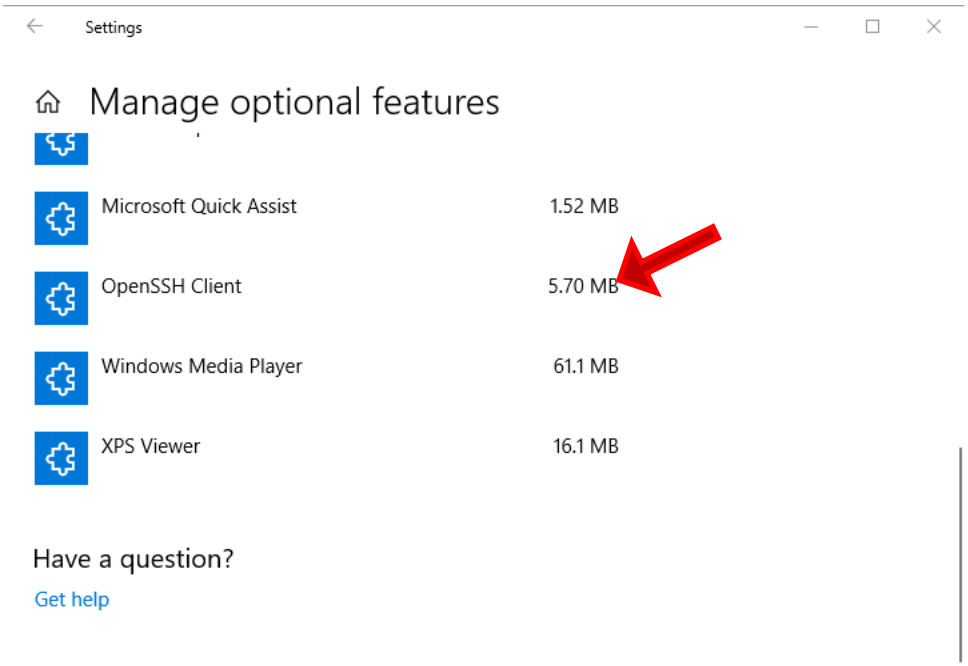
Manage optional features на свой ред ще прехвърли потребителя към подсекция със същото име, в която той ще може да щракне върху бутона ***Add a feature*** и да избере ***OpenSSH Client*** (фиг. 1.12.)



Фигура 1.12. Добавяне на ***OpenSSH*** *от* ***Add a feature***

При щракване върху бутона ***Install***, който ще се появи под описанието на OpenSSH клиента, OpenSSH ще бъде активиран и ще се появи в списъка с вече инсталирани модули. Тази операция може да отнеме няколко минути.

Потребителят може да се върне стъпка назад към ***Manage optional features*** и да провери дали инсталацията е приключила успешно (фиг. 1.13.)



Фиг. 1.13. Подсекция ***Manage optional features***

Вече инсталираният клиент може да се използва от ***Windows Power Shell***. За целта потребителят трябва да стартира ***Power Shell*** и да извика ***ssh*** от командния ред.

Базовият синтаксис на командата е

ssh *потребител@сървър* ***[-p порт]***

Показаният по-долу пример демонстрира първо свързване към сървър с OpenSSH клиент под Windows PowerShell.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Ownjo> ssh owner@195.20.24.18 -p 30022
The authenticity of host '[195.20.24.18]:30022
([195.20.24.18]:30022)' can't be established.
ECDSA key fingerprint is
SHA256:QMwSUTpXh3yILV482a6NyoMcdDdty/IFwyImnTFBXDM.
Are you sure you want to continue connecting (yes/no)? yes

```

```
   Warning: Permanently added '[195.20.24.18]:30022' (ECDSA) to the
 9 list of known hosts.
10 owner@195.20.24.18's password:
11 Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.4.0-45-generic x86_64)
12
13  * Documentation:  https://help.ubuntu.com
14  * Management:     https://landscape.canonical.com
15  * Support:        https://ubuntu.com/advantage
16
17   System information as of Wed May 23 20:33:17 EEST 2018
18
19   System load:  0.0               Processes:            170
20   Usage of /:   39.0% of 3.81GB   Users logged in:      0
     Memory usage: 31%               IP address for ens33:
21 195.20.24.18
     Swap usage:   0%
22
23   => /var is using 91.1% of 13.66GB
24
25  * Meltdown, Spectre and Ubuntu: What are the attack vectors,
26    how the fixes work, and everything else you need to know
27    - https://ubu.one/u2Know
28
29  * Canonical Livepatch is available for installation.
     - Reduce system reboots and improve kernel security. Activate
30 at:
31     https://ubuntu.com/livepatch
32
33 5 packages can be updated.
34 1 update is a security update.
35
36
37 No mail.
38 Last login: Tue May 22 16:44:12 2018 from 77.78.56.10
   owner@bi:~$
```

Както е видно от разпечатката, на ред 4 потребителят се свързва като потребител ***owner*** към сървър с IP адрес 195.20.24.18, чийто SSH сървър работи на нестандартен порт - 30022.

Като се абстрахираме от факта, че потребителското име се дава като параметър още при извикването на ssh клиента, свързването протича по начин, който много се доближава до описания по-горе при PuTTY клиента.

Така например, тъй като това е първо свързване на клиента към сървъра, ред 5 информира, че SSH клиентът няма как автоматично да провери криптографския отпечатък на сървъра. Ред 6 показва получения от мрежата такъв, а ред 7 представлява положителния отговор на потребителя на въпроса следва ли да продължи опита за свързване. Ред 8 информира потребителя, че потвърденият криптографски отпечатък е записан като база за по-нататъшни проверки на самоличността на сървъра.

Струва си да се отбележи също, че новите версии на ***OpenSSH*** показват отпечатъка на криптографския ключ на сървъра в различен формат от ***PuTTY***. Този формат се нарича ***base64*** и е ефективен начин за представяне на дълги числа в сбит вид.

За да се покаже отпечатъка на сървъра по начин, подходящ за сверка с този формат, администратор с локален достъп до системната конзолата би написал

```
sudo ssh-keygen -l -f /etc/ssh/ssh_host_ecdsa_key
```

3. Сигурно прехвърляне на файлове между Windows и Линукс система

В хода на нормалната работа доста често се налага не само да се подават команди към сървърите, но и да се прехвърлят файлове. Така например, след като бъде разработен даден сайт, той трябва да бъде „качен“ на сървъра по някакъв начин.

При прехвърлянето на файлове съществуват същите потенциални проблеми със сигурността като проблемите с отдалечения достъп, които бяха очертани по горе.

Съдържанието на файловете може да бъде подслушано и (евентуално) променено. При това, за да се копира файл на отдалечената система, потребителят отново трябва да предостави валидно потребителско име и парола.

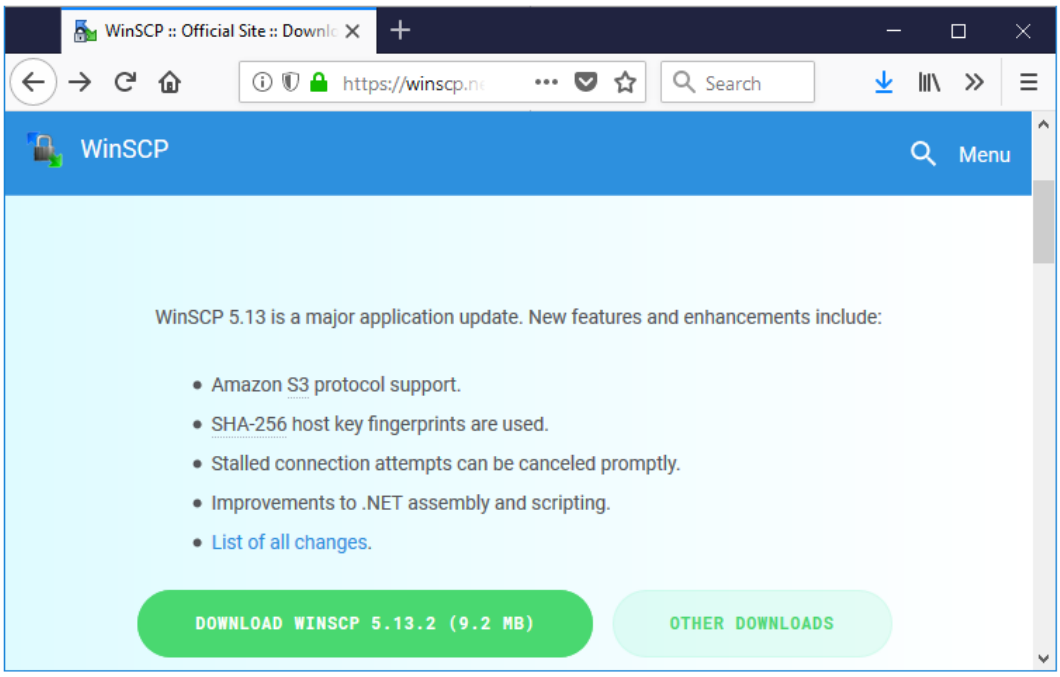
Именно поради тази причина, модерните начини за прехвърляне на файлове между клиенти и сървъри в Интернет са базирани на същите криптографски подходи и същите комуникационни протоколи, като тези, използвани за отдалечено управление. Всъщност, много от разгледаните програми за отдалечено управление на Линукс сървъри или имат вградени средства за прехвърляне на файлове, или са част от по-големи пакети, които имат специален клиент за прехвърляне на файлове.

В настоящото изложение ще разгледаме два клиента за Windows. За копиране на файлове с графичен интерфейс ще демонстрираме безплатния софтуер ***WinSCP***, а за да представим инструментите за ползване от командния ред, ще разгледаме командата ***scp***, която е част от ***OpenSSH*** модула на Windows. Ако потребителят е изпълнил очертаната по-горе последователност за активация на SSH успешно, той има достъп автоматично и до ***scp***.

Настоящето изложение ще пропусне инструментите за копиране на файлове, предлагани от оригиналните автори на ***PuTTY***, тъй като тяхната инсталация е проста, а начинът им на употреба е аналогичен на командния клиент на ***OpenSSH***, който ще бъде разгледан по-подробно. Любопитният читател може да изтегли ***pscp.exe*** от показания по-горе сайт и да провери доколко поведението и функциите на тази програма съвпадат с инструмента на ***OpenSSH***.

3.1. Прехвърляне на файлове с WinSCP

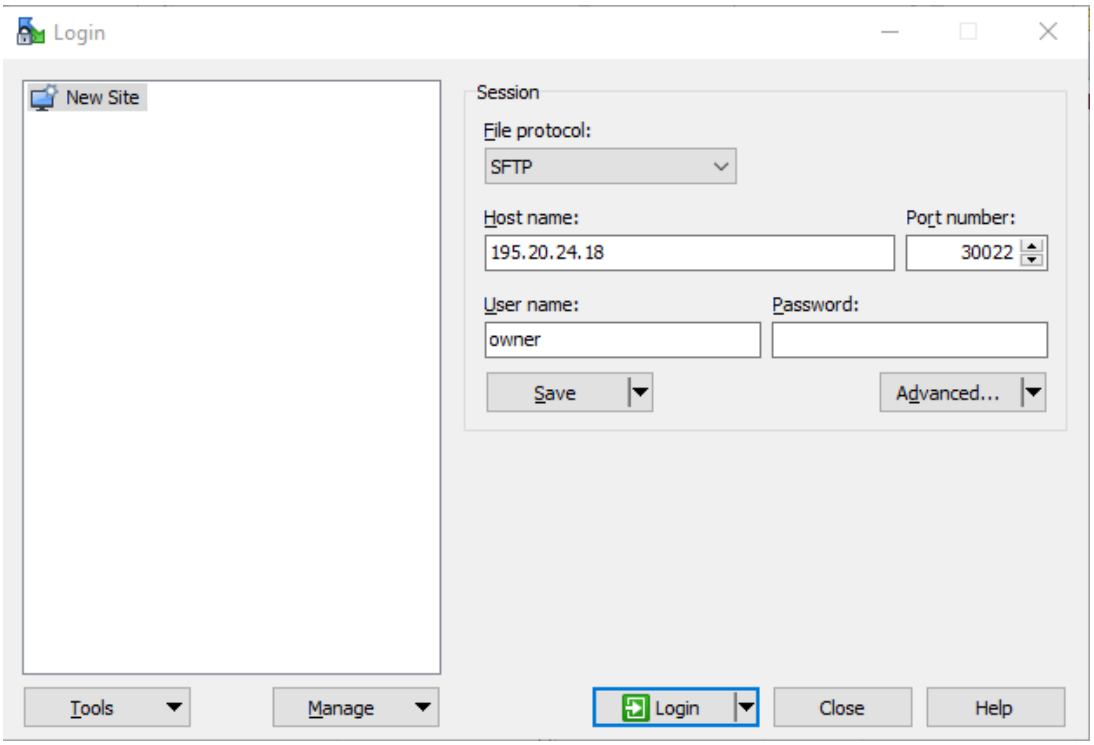
Типичният начин на инсталация на ***WinSCP*** е чрез инсталационна програма, която може да бъде изтеглена от адрес ***https://winscp.net/eng/download.php*** (вж. фиг 1.14.).



Фигура 1.14. Изтегляне на WinSCP от официалния сайт

Самата инсталация на WinSCP е тривиална и по подразбиране създава икона на ***WinSCP*** на работния плот, както и елемент в менюто на ***Windows***.

При стартиране на ***WinSCP*** потребителят вижда диалог за връзка, който е показан на фиг. 1.15.



Фигура 1.15. Начален екран за връзка на ***WinSCP***

В диалога за връзка потребителят указва протокол за трансфер на файлове (***File protocol***), домейн име или IP адрес (***Host name***), порт (***Port number***), както евентуално и потребителско име (***User name***) и парола (***Password***).

WinSCP поддържа няколко протокола за трансфер на файлове, като потребителите, държащи на сигурна връзка към отдалечен ***Linux*** сървър могат да изберат ***SFTP*** (стойността по подразбиране) или ***SCP.*** От гледна точка на крайния потребител посочените протоколи на ***WinSCP*** са напълно аналогични като предлагани функционални възможности.

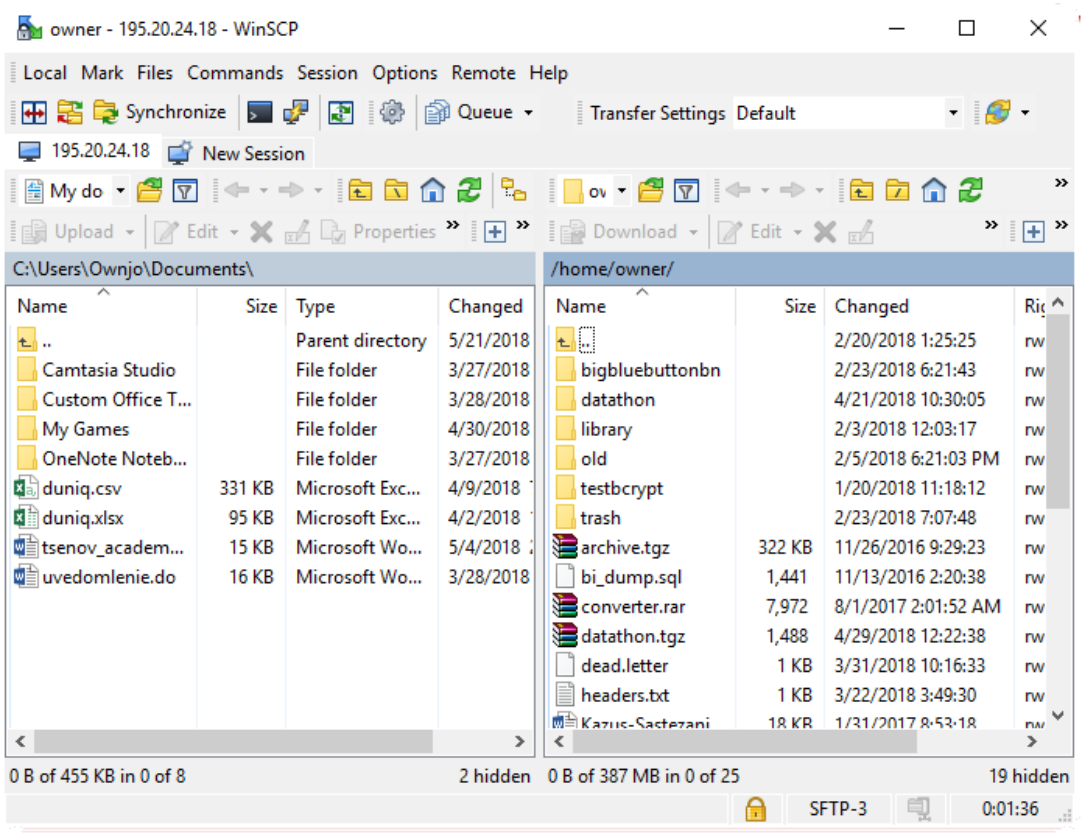
Ако потребителят не въведе потребителско име или парола, такова ще бъде поискано при свързване със системата. Също така, след като бъдат въведени всички настройки (някой от тях се намират под бутона ***Advanced*** на същия прозорец), потребителят ще има възможност да ги запише за по-нататъшно ползване чрез щракане на бутона ***Save***.

Възможността потребителското име или паролата (или и двете) да бъдат изпуснати всъщност позволява на потребителя да избере удобството пред максималната сигурност или обратно.

Ако потребителят запише потребителското име и паролата за достъп, в последващите свързвания със сървъра той няма да бъде питан за тях, което представлява известно нарушение на най-добрите практики при съблюдаване на информационната сигурност. От друга страна загубата не е толкова голяма, тъй като се предполага, че потребителят ще пази собствената си система от всички видове заплахи включително и организационни грешки като предоставяне на достъп до потребителските профили без парола, изхвърлянето на твърди дискове без да бъдат взети мерки механиката и данните да бъдат унищожени и т.н.

След като бъдат въведени всички настройки, потребителят следва да натисне бутона ***Login***. В зависимост от това дали е въвел потребителско име и парола, както и от това дали това е първо свързване с дадения сървър, от потребителя може да се изисква да отговори на допълнителни въпроси. Това включва въвеждане на потребителско име, парола, както и диалог за потвърждаване на криптографския отпечатък на сървъра.

При правилно протичане на свързването, потребителят ще види прозорец, подобен на показания на фиг. 1.16.

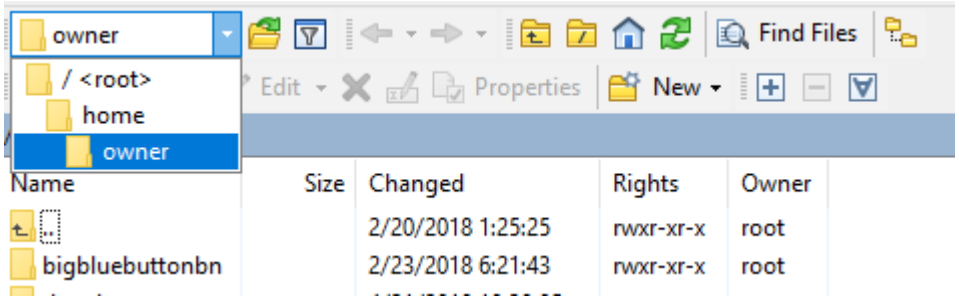


Фигура 1.16. Главен прозорец на WinSCP

В лявата част на прозореца потребителят вижда файловата система на собствения си компютър, докато в дясната той вижда файловата система на сървъра. С помощта на мишката той може да влачи файл или група файлове между двете половини, като това води до трансфер на файловете от едната машина на другата.

Освен прехвърляне на файлове, ***WinSCP*** поддържа още много полезни функции, като създаване на папки, придвижване в дървото на директориите (папките) и т.н. Тези функции са достъпни чрез контекстното меню на десния бутон, както и от лентата с инструменти.

Лентата с инструменти заслужава по-голямо внимание, поради специфичната и организация и концентрацията на функции в нея.



Фигура 1.17. Лента с инструменти на WinSCP

Бутоните в лентата се повтарят два пъти, като тези в лявата част се отнасят за левия панел с файловата система на машината – клиент, докато тези в дясната част се отнасят за файловата система на машината – сървър. На фиг. 1.17 е показана дясната част на лентата.

В най-лявата част на показаната фигура има падащо поле, което показва мястото на текущата директория в йерархията на директориите.

Директориите в ***Unix*** и ***Linux*** са същото нещо като папките в ***Windows***. Разликата се състои само в имената им. Работата с директории ще бъде разгледана подробно в глава 2.

От падащото поле потребителят може да се придвижва една или повече директории „нагоре“ по дървото на директориите докато евентуално достигне до главната директория на файловата система, която в Linux се нарича ***root*** и често се обозначава само с права наклонена черта (“/”).

В средата на показаната фигура се намират допълнителни бутони за придвижване. Бутонът води до придвижване до директорията – родител на текущата директория. Бутонът придвижва потребителя директно до главната директория на файловата система, а бутонът придвижва потребителя до домашната му директория.

Важен бутон е и , който предизвиква презареждане на списъка с файлове и директории от сървъра. Този бутон е полезен, ако трябва да се

покаже файл, който се е появил след свързването със системата, може би като резултат от действията на друг потребител.

Бутонът

, разположен на втория ред на лентата, може да се използва за създаване на нови директории или папки. Бутонът служи за ограничаване показването на наличните файлове до такива, чиито имена отговарят на определени критерии.

3.2. Прехвърляне на файлове със scp

Както бе споменато, ***scp*** е инструмент - част от пакета OpenSSH, който се използва в Linux и (отскоро) в Windows за сигурен отдалечен достъп и копиране на файлове.

За разлика от WinSCP, ***scp*** няма графичен интерфейс и работи на командния ред. За да използва ***scp***, потребителят трябва да е активирал OpenSSH за Windows по описаната по-горе процедура.

Базовият синтаксис на командата за копиране на файлове от текущата към отдалечена система е:

scp [-P *номер_на_порт* ***]*** *път_до_локален_файл потребител@хост:път_на_сървъра*

При копиране на отдалечени файлове от сървъра на локалната система, мястото на двата задължителни аргумента на командата се размества и синтаксисът на командата придобива вида:

scp [-P *номер_на_порт* ***]*** *потребител@хост:път_до_файл_на_сървъра локален_път*

Аргументът ***-P*** позволява на потребителя да укаже нестандартен (т.е. различен от номер 22) порт на SSH сървъра. Квадратните скоби около аргумента в синтаксиса означават, че неговото въвеждане не е задължително.

Другите два аргумента са задължителни и могат да бъдат наречени още „източник“ и „цел“. Файлът (или файловете) се вземат от източника и се копират в целта. За разлика от обикновените команди за копиране на файлове, едно от двете - източникът или целта - могат да се намират на отдалечена система.

При изписване на задължителните аргументи, първият се приема винаги за източник, докато вторият е цел.

Така например:

```
scp lalev@www.google.com:/etc/password “C:\home\archive\”
```

ще предизвика копирането на файл от машина на име www.google.com на локалния компютър. Файлът на отдалечената машина се нарича ***password*** и се намира в директория (папка) ***etc***, която от своя страна е поддиректория (подпапка) на главната директория на сървъра. Файлът, който ще се копира от сървъра на клиента, ще се постави в папка ***archive***, подпапка на папка ***home***, която от своя страна е подпапка на главната директория на устройство “C:”.

Преди да се извърши копирането, разбира се, клиентът ще се опита да установи дали наистина разговаря със сървър ***www.google.com***. Сървърът от своя страна ще провери съществува ли потребител ***lalev***, има ли право този потребител да чете данни от указания файл и дали потребителят наистина е този, за който се представя (последното се проверява чрез въвеждане на паролата за достъп).

Ако изброените проверки завършат с положителен отговор, копирането на файла ще се осъществи. В противен случай ще бъде изведено съобщение за грешка.

Както може би читателите се досещат, информационната сигурност при сървърите от години е много чувствителна тема, тъй като те бяха едни от първите елементи на инфраструктурата на Интернет, които бяха подложени на всевъзможни атаки от хакери. Ето защо не е изненадващо, че съобщенията за грешки при свързване с отдалечен сървър през SSH не са особено специфични, за да не „изтича“ чрез тях полезна информация към подобни хакери.

Ако например горната система ***www.google.com*** наистина предлага SSH /това не е така в реалния свят, не и за случайни потребители, които идват от всички краища на мрежата/ тя няма да отговори със специфично съобщение. Ако потребител ***lalev*** не съществува на сървъра, системата ще поиска три пъти паролата за този потребител преди да прекъсне връзката. Това поведение е същото, което хакерите ще наблюдават, ако потребителят съществува, но не могат да отгатнат паролата му.

По този начин системата няма да разкрие пред хакерите дали съществува или не потребител ***lalev***, което само за себе си е важна информация за всеки, който иска да атакува една система.

По същия начин:

```
scp –P 242 owner@195.20.24.18:/home/owner/test.doc “C:\Users\Owner”
```

Копира файл от система с IP 195.20.24.18. Файлът е в ***/home/owner*** и се нарича ***test.doc***. При копирането, файлът ще бъде поместен в папка ***C:\Users\Owner***.

За разлика от предния пример, този път отдалеченият сървър работи на порт 242, което налага използването на аргумента ***–P 242***. В показания случай можем да наречем аргумента ***owner@195.20.24.18/home/owner/test.doc*** „източник“, понеже описва точно откъде се взема файла, който се копира. ***“C:\Users\Owner”*** от своя страна е „цел“, понеже показва къде ще бъде поместен копираният файл.

Да си представим хипотетично,че потребителят е нанесъл локални корекции на файла ***test.doc*** от предния пример и сега желае да копира същия файл обратно на сървъра. За целта е достатъчно да размести местата на източника и целта:

```
scp –P 242 “C:\Users\Owner\test.doc” owner@195.20.24.18:/home/owner
```

Този път източникът е ***C:\Users\Owner\test.doc”***, а целта е ***owner@195.20.24.18/home/owner***. Целта отново показва местоположението,

където „отива“ копирания файл. Докато източникът показва откъде „идва“ файла.

4. Казус – рестартиране на системата

```
1   Ubuntu 18.04 LTS dragonfly tty1
2
3   dragonfly login: lalev
4   Password:
5   Last login: Sun Feb 18 10:16:57 EET 2018 on tty1
6   Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.4.0-112-generic x86_64)
7
8    * Documentation   https://help.ubuntu.com
9    * Management      https://landscape.canonical.com
10   * Support         https://ubuntu.com/advantage
11
12  0 packages can be updated.
13  0 updates are security updates.
14
15  lalev@dragonfly:~$ ssh owner@195.20.24.18 -p 30022
16  owner@195.20.24.18's password:
17  Welcome to Ubuntu 18.04 LTS (GNU/Linux 4.4.0-63-generic x86_64)
18
19   * Documentation:  https://help.ubuntu.com
20   * Management:     https://landscape.canonical.com
21   * Support:        https://ubuntu.com/advantage
22
23  0 packages can be updated.
24  0 updates are security updates.
25
26
27  Last login: Thu Feb  8 15:13:00 2018 from 195.20.26.30
28  owner@bi:~$ who
29  peter    tty1         Feb 19 19:19
30  dave     pts/1        Feb 20 11:26 (195.20.26.9)
31  owner    pts/2        Feb 20 13:19 (77.78.56.10)
32  owner@bi:~$ write dave
33  Trjabva da restartiram mashinata speshno. Pravish li neshto?
34  owner@bi:~$
35  Message from dave@bi on pts/1 at 13:34 ...
36  Davai
37  owner@bi:~$
38  owner@bi:~$ wall
```

```
39  Restartiram v 13:40
40
41  Broadcast message from owner@bi (pts/2) (Tue Feb 20 13:35:53
42  2018):
43
44  Restartiram v 13:40
45
46  owner@bi:~$
47  owner@bi:~$ sudo at 13:40
48  [sudo] password for owner:
49  warning: commands will be executed using /bin/sh
50  at> reboot
51  at> <EOT>
52  job 1 at Tue Feb 20 13:40:00 2018
53  owner@bi:~$
54  Connection to 195.20.24.18 closed by remote host.
55  Connection to 195.20.24.18 closed.
56  lalev@dragonfly:~$
```

Текущият казус разглежда реална ситуация, свързана с рестартирането на отдалечен сървър.

Както може да се види от разпечатката (редове 1-14), потребителят осъществява успешен вход на виртуална конзола на машина, която се нарича ***dragonfly***.

Промптът на ред 15 (***lalev@dragonfly:~$***) припомня на потребителя точно това, т.е. че е влязъл като потребител ***lalev*** на машина ***dragonfly*** и в момента се намира в домашната[3] си директория (**~**).

Командата на 15 ред, а именно ***ssh owner@195.20.24.18 -p 30022*** осъществява шифрирана връзка с друга машина, която има IP адрес 195.20.24.18. По-рано в главата бе разгледано използването на ***PuTTY*** за шифрирана връзка от Windows към Linux машини. Инструментът ***ssh*** под Linux изпълнява точно същата функция, но подпомага свързването на Linux с други Linux, UNIX и дори Windows машини.

[3] Домашните директории на потребителите са разгледани подробно в глава 2.

owner в случая е потребителското име, под което ще се осъществи достъп до системата с указаното IP. Разбира се, такъв потребител трябва да съществува на отдалечената система и неговата коректна парола за достъп трябва да бъде подадена при свързване.

Аргументът ***–p 30022*** указва, че SSH сървърът, който е отговорен за комуникацията на отдалечения компютър, работи не на стандартния порт 22, а на порт 30022.

Подаването на паролата за потребител ***owner*** на машината ***195.20.24.18*** се извършва на ред 16 от разпечатката и както е видно, паролата не се извежда под никаква форма, за да не се издава никаква информация за нея.

Дори информацията за броя на символите в паролата е огромно улеснение за евентуални злонамерени лица, които биха могли да видят конзолата на системния администратор, като стоят, например зад гърба му.

Въвеждането е успешно и потребителят вижда съобщението на деня на системата с адрес ***195.20.24.18***, както и информация за последния път, когато потребител ***owner*** е осъществил успешно достъп. Това се е случило последно на 8-ми февруари от IP 195.0.26.30 (ред 27).

Промптът на ред 28 припомня на потребителя, че сега вече той действа като потребител ***owner***, а името на системата е ***bi***. Казано по друг начин, промптът напомня, че всички въведени команди оттук нататък ще се изпълняват като потребител ***owner*** на отдалечената система ***bi***.

Самото име на системата (***dragonfly*** или ***bi*** в нашите примери) се задава от системните администратори. Към това име няма други изисквания, освен да е добре разпознаваемо, така че потребителите, които работят на множество системи, да не се объркват къде въвеждат предвидените команди.

Видно е, че администраторът на машината с IP 195.20.24.18 е избрал да я кръсти ***bi***, което в случая е съкращение от „Бизнес информатика“.

На ред 28 потребителят извиква команда ***who***, която показва активните сесии на системата. Изходът на командата, а именно

```
peter    tty1         Feb 19 19:19
dave     pts/1        Feb 20 11:26 (195.20.26.9)
owner    pts/2        Feb 20 13:19 (77.78.56.10)
```

показва, че в момента има три активни потребителски сесии. Потребител ***peter*** се свързва директно от клавиатурата и монитора, т.е. от системната конзола, което е указано от името на конзолата – ***tty1*** – първата виртуална конзола. Датата и часът срещу ***peter*** указват началото на неговата сесия, т.е. 19:19 ч. на 19 февруари.

Потребителите ***dave*** и ***owner*** се свързват отдалечено, което може да се разбере от имената на конзолите (**pts/1** и **pts/2**). Също така потребителите ***dave*** и ***owner*** имат на своите редове IP адрес от който се свързват към ***bi***. ***dave*** се свързва от IP адрес 195.20.26.9, докато нашият потребител – ***owner*** – се свързва от IP адрес 77.78.56.10.

След като се е осведомил за сесиите в системата, потребител ***owner*** изпраща съобщение на потребител ***dave*** с помощта на командата ***write*** (редове 32-33).

Важно е да се отбележи, че програмата ***write*** позволява писането на съобщения на много редове. За да сигнализира, че е приключил въвеждането на съобщението, потребителят ***owner*** е въвел клавишната комбинация ***Ctrl+d*** в края на ред 33. Тази комбинация не се изписва на конзолата и не е видна на разпечатката.

Съобщението е въпрос дали рестартирането на ***bi*** е проблем за **dave**. Скоро нашият потребител получава отговор от ***dave*** (редове 35 и 36) и преминава към рестартирането на системата.

Тъй като рестартирането може да представлява проблем и за другия свързан потребител, ***owner*** предупреждава всички активни потребители с командата ***wall***, която работи по същия начин като ***write***, но не взема като аргумент име на потребителя – получател на съобщението, тъй като

съобщението се изпраща до всички. Съобщението се изпраща до абсолютно всички потребители, което включва и потребителя, който го е изпратил, така че ***owner*** вижда собственото си съобщение на редове 41-44.

В случая съобщението гласи, че системата ще бъде рестартирана в 13:40.

Следващата команда – ***sudo at 13:40*** – всъщност е комбинация от две команди.

Командата ***at 13:40*** приема за изпълнение списък от команди, които да бъдат изпълнени в точно определено време. В нашия случай списъкът се състои от два реда. На първия ред е командата ***reboot***, която е въведена от потребителя. На втория ред се намира текстът <EOT>. Той е резултат от това, че потребителят е въвел ***Ctrl+d*** с което е указал, че е приключил въвеждането на списъка, т.е. списъкът в нашия случай се състои от една команда.

Командата ***sudo***, написана пред командата ***at***, означава, че списъкът с команди, въведени в ***at***, трябва да се изпълни от името и с привилегиите на администратора на системата. Това е направено, защото обикновените потребители нямат право да рестартират системата отдалечено.

Стандартна практика при дистрибуциите от подфамилията на Ubuntu е администраторският акаунт /административният потребител/ да бъде забранен. Вместо това, когато даден потребител трябва да изпълни команди с привилегиите на административния акаунт, той извиква командата ***sudo***, като записва името ѝ пред командата, която се нуждае от привилегиите на администратора, за да извърши функциите си.

Само определени доверени потребители могат да използват командата ***sudo***. Това на практика са администраторите на системата.

Това решение, което отличава Ubuntu от други модерни Linux дистрибуции, е продиктувано от съображения по сигурността. Когато даден потребител е администратор, неговите програми работят с привилегиите и разрешенията на администратора. По този начин грешка на администратора или пробив в сигурността на изпълнените от него програми може да доведе до тотално компрометиране на сигурността на системата.

Ако администраторите изпълняват програмите, които не се нуждаят от разрешенията на администратор, като обикновени потребители, този риск бива избягнат. sudo улеснява именно такъв режим на работа при който администраторът е обикновен потребител до момента, когато трябва да изпълни команда, която се нуждае от привилегиите на администратора, за да извърши функцията си (например ***reboot*** е такава команда).

5. Упражнения

Упражнение 1. Автоматично завършване на команди

Стъпка 1. На промпта на обвивката напишете ***l*** (малка латинска буква "***l***") и натиснете веднъж клавиша ***tab***. Отбележете случва ли се нещо, след което натиснете бързо един след друг два пъти клавиша ***tab***. Показва ли се ред от рода на

Display all 143 possibilities? (y or n)

Натиснете ***y*** (малко латинско "y") и прегледайте списъка, който ще бъде показан.

Стъпка 2. Отново на промпта, напишете "***le***" и натиснете веднъж клавиша ***tab***. Отбележете случва ли се нещо, след което натиснете бързо един след друг два пъти клавиша ***tab***. Този път появява ли се предупредителен надпис? А списък?

Стъпка 3. Отново на промпта, напишете "***les***" и натиснете **tab** веднъж. Отбележете реакцията на системата. Според Вас има ли две команди, които започват с "***les***" ?

Упражнение 2. Извикване на предишно изпълнени команди

Стъпка 1. Въведете на конзолата списъка от следните команди, като след всяка команда натискате клавиша ***Enter***: ***pwd***, ***less***, ***who***, ***free***.

Стъпка 2. Натиснете клавиша „стрелка нагоре" няколко пъти. След това натиснете клавиша „стрелка надолу" няколко пъти. Тъй като Linux конзолите не винаги могат да интерпретират правилно стрелките от клавиатурата, повечето от обвивките по историческа традиция имат алтернативни клавишни комбинации за функциите, изпълнявани от стрелките. Опитайте комбинациите ***Ctrl+p*** и ***Ctrl+n***.

Стъпка 3. Наберете командата

```
vmstat -S K | tail -1 | tr -s ’ ’ | cut  -f 4 -d’ ’
```

но **не** я приключвайте с въвеждането на клавиш ***Enter***. Опитайте комбинациите ***Ctrl+A*** и ***Ctrl+E***. Използвайте клавишите „стрелка наляво“ и „стрелка надясно“ за да придвижите курсора хоризонтално по реда на командата.

Използвайте клавиша ***Del***, за да изтриете символа, върху който се намира курсора. Използвайте клавиша ***Backspace***, за да изтриете символа зад (вляво от) курсора.

Проверете дали ***Ctrl+d*** има същия ефект като ***Del***. Проверете дали ***Ctrl+h*** има същия ефект като ***Backspace***.

Упражнение 3. Клавиши на обвивката *bash*

Ако вашата обвивка по подразбиране е ***bash*** (повечето Linux дистрибуции за десктоп и сървър системи използват именно ***bash*** по подразбиране), изпробвайте следните клавишни комбинации:

Ctrl+a	Премества курсора в началото на реда
Ctrl+e	Премества курсора в края на реда
Ctrl+p	Предишна команда
Ctrl+n	Следваща команда (когато имаме ситуация с извикана преди това предишна команда).
Alt+b	Премества курсора назад (наляво) с една дума
Alt+f	Премества курсова напред (надясно) с една дума
Ctrl+b	Премесва курсора назад (наляво) с един символ
Ctrl+f	Премества курсора напред (надясно) с един символ
Ctrl+xx *(два пъти бързо ‘x’)*	Първо извикване на комбинацията изпраща курсора в началото на реда. Второ извикване връща курсора на позицията, на която е бил преди първото извикване.

Ctrl+d, Del	Изтрива символа, върху който е курсорът.
Ctrl+h, Backspace	Изтрива символа, който е зад (вляво) курсора.
Ctrl+l	Изчиства екрана
Alt+d	Изтрива всички символи надясно (напред) от курсора до първия срещнат интервал. Символът, върху който е курсора, също се изтрива. Ако този символ е интервал, той, заедно със следващите го надясно интервали, се изтриват и действието на командата започва от първия символ, който не е интервал.
Ctrl+k	Изтрива всички символи от курсора до края на реда и ги прехвърля в клипборда.
Ctrl-u	Изтрива всички символи от курсора до началото на реда и ги прехвърля в клипборда.
Ctrl-y	Вмъква съдържанието на клипборда на мястото на курсора

Упражнение 4. Виртуални конзоли

Стъпка 1. Изпробвайте всички клавишни комбинации, които се състоят от ***Ctrl+Alt*** и функционален клавиш (такива са клавишите от ***F1*** до ***F12***). Всеки от тях ли отговаря на виртуална конзола?

Стъпка 2. Когато се намирате на виртуална конзола, опитайте вместо ***Ctrl+Alt+функционален клавиш*** да използвате само ***Alt+функционален клавиш*** за превключване между виртуалните конзоли. Работи ли тази техника когато започвате от конзола с графична среда?

Упражнение 5. Потребителски сесии

Влезте като един и същ потребител на Вашата система два или повече пъти от различна конзола, без да затваряте сесията. Системата позволява ли това?

Напишете командата ***who***. Тази команда показва кой работи в момента на системата. Показва ли отделните Ви сесии на различни редове? Показва ли конкретната конзола (***tty***), от която се извършва достъпа?

Упражнение 6. Получаване на помощ

Командата ***man*** е стандартният начин за получаване на помощ в Линукс. Името и идва от "manual" („ръководство"), тъй като исторически командата и показваната от нея информация са се появили като електронно средство за четене на ръководствата на системата, които до тогава са били само на хартия. По същата причина помощната информация, извеждана от тази програма, се нарича „manpages", което идва от „страници от ръководството".

Линукс и прилежащият софтуер идват с богат набор помощна информация на английски, които са достъпни през програмата ***man***. Те са организирани в „секции" по аналог с хартиените ръководства. Най-важната секция е секция 1, която се отнася за командите на системата. Секции 2 и 3 представляват интерес за програмистите, тъй като те представят съответно програмния интерфейс (API) на ядрото и системните библиотеки на C. Секция 4 представя различните специални файлове, а секция 5 представя важни концепции.

Извикването на помощ за дадена команда се извършва чрез подаването на името на командата като параметър на ***man***. Например ***man who*** ще изведе информация за функциите и параметрите на командата ***who***.

Ако в различните секции съществуват страници с еднакви имена, какъвто например е случаят с ***unlink***, което е име на команда и едновременно име на системна функция, потребителят трябва да посочи и секция, за да получи желаната от него страница. Така например ***man 1 unlink*** ще покаже информация за командата, докато ***man 2 unlink*** ще покаже информация за системната функция на ядрото.

Много често потребителят не е наясно за имената на конкретни програми или функции, а се интересува от дадена концепция. За да го улесни, ***man*** предлага търсене по ключови думи. За целта потребителят трябва да подаде параметъра ***–k*** и да изпише след него ключовата дума. При това положение ***man*** ще изведе имената на релевантните страници и техните секции, които могат да бъдат извикани след това по вече описания начин.

Опитайте да извикате командите ***man umask***, ***man –k acl***, ***man –k permissions***.

ГЛАВА 2. РАБОТА С ФАЙЛОВЕ И ДИРЕКТОРИИ

1. Основни понятия при работата с файлове и директории

1.1. Файлове и директории

Файловата система на Линукс е организирана около файлове и директории по начин, който не е особено различен от модерните популярни потребителски операционни системи. Най-общо казано, файловете съдържат логически свързани данни на някаква тема или данни относно някакъв обект. Файловете например могат да съдържат отделен офис документ, конфигурационна информация на дадена програма, графично изображение и т.н.

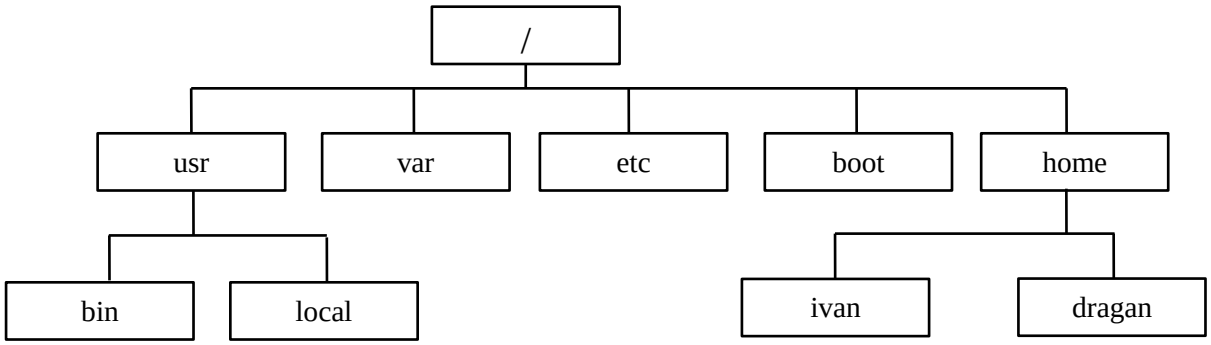
Директориите от своя страна могат да бъдат описани като своеобразни контейнери, които се използват от потребителите за групиране на файловете по даден признак. Всеки файл във файловата система трябва да е разположен в директория, като директорията освен файлове, може да съдържа и други директории. Когато една директория е разположена в друга, казваме че първата директория е *поддиректория* на втората.

Една директория във файловата система по необходимост не се съдържа в никоя друга директория, а служи за своеобразно начало на файловата система. Тази директория може да съдържа поддиректории, които на свой ред да съдържат поддиректории и т.н. (вж. фиг. 2.1)

Разклонената структура, оформена от тази схема, прилича на дърво, поради което началната директория се нарича *„корен“*[4] или - както ние ще я

[4] От англ. „root”.

наричаме - *“главна директория”*. Главната директория в Линукс се обозначава с право наклонена черта (**/**).



Фигура 2.1. Примерно дърво на папки във файлова система под управлението на Линукс.

Читателите са приканени да забележат, че директория ***home*** на фигурата има две поддиректории, които вероятно съдържат файлове, принадлежащи на потребителите ***ivan*** и ***dragan***. Тези директории най-вероятно са били създадени от администратора или неговите програми за автоматизация. По този начин администраторът е използвал механизма на директориите, за да групира файлове по признак. В случая признакът е това кой е собственик на файловете.

Потребителите в Линукс имат пълната свобода да създават директории и поддиректории по начини, които отговарят на възприетата от тях логика и избраните от тях признаци на групиране. Разбира се, даден потребител може да създава поддиректории само в директории, за които има съответните разрешения за достъп.

Файловете и директориите в Линукс имат имена и тези имена са чувствителни към малки и големи букви. Тоест името ***Local*** се счита за различно от името ***local*** и от името ***LoCaL***. Линукс има изискване имената на файловете и поддиректориите в дадена директория да са уникални за тази директория. Тоест в една файлова система могат да съществуват два или повече файла с имена ***info.txt***, стига те да се намират в различни директории. Също така в дадена директория не може да има файл и поддиректория с едно и също име.

Когато трябва да се извърши каквато и да било операция върху даден файл или група файлове, техните имена трябва да бъдат посочени по начин,

който изключва двусмислие. Така например администраторът не може просто да инструктира системата да изтрие файл ***info.txt***, тъй като в различни директории на файловата система може да се намират много файлове с имена ***info.txt***. Той трябва да посочи, явно или неявно, и директорията, в която се намира този файл.

Един от начините това да бъде извършено е името на файла да бъде допълнено с т.нар. *„път“*. Пътят често се поставя преди името на файла и представлява списък с имената на всички директории, през които трябва да се „премине“, за да бъде достигнат файла.

В зависимост от това коя е първата директория в списъка, различаваме два вида пътища – *абсолютни* и *относителни*.

Абсолютните пътища винаги започват от *главната директория* (/) и приключват с директорията, която съдържа файла. В табл. 2.1 са показани примери за абсолютни пътища.

/usr/local/bin/autodump.sh	Файл (или директория) с име ***autodump.sh***, който се намира в директория ***bin***, която е поддиректория на директория ***local***, която на свой ред е поддиректория на директория ***usr***. ***usr*** от своя страна е поддиректория на главната директория. В случая пътят е ***/usr/local/bin*** и е важно да се отбележи, че първата право наклонена черта в пътя е името на главната директория, докато следващите право наклонени черти се използват просто като разделители между имената на директориите.
/bin/bash	Файл (или директория) ***bash***, който се намира в поддиректория ***bin***. За потребителите, свикнали да виждат файлови имена под Windows, е важно е да се отбележи, че в Линукс няма

	ограничения или конвенции, които да изискват файловете да се отличават по имената си от директориите. Така например даден файл може да има име с "разширение" (***info.txt***) или без разширение (***info***). По същия начин дадена директория може да се казва например ***library.git*** или дори ***this.is.ugly.way.of.naming.directory*** Тоест, само по името не можем да гадаем дали даден обект във файловата система е файл или директория.
/lost+found	Файл (или директория), който се намира в главната папка на файловата система.
/	Самата главна папка на файловата система.

Таблица 2.1. Примери за абсолютни пътища

Сборът от абсолютния път и името на файла, както бяха показани в табл. 2.1 , често се наричат *"напълно квалифицирано име на файла"*. Напълно квалифицираното име на файла е уникално. Така например за напълно квалифицираното име на файл ***/usr/local/bin/autodump.sh*** можем да направим следните наблюдения:

1. Не може да има друга поддиректория с името ***usr*** в главната директория.
2. Не може да има друга поддиректория с името ***local*** в директория ***/usr***.
3. Не може да има друга поддиректория с името ***bin*** в директория ***/usr/local***.
4. Не може да има друг файл с името ***autodump.sh*** в директория ***/usr/local/bin***.

Изписването на напълно квалифицирани имена е важно умение, необходимо за задаването на команди към обвивката на Линукс. Така

например, следните три команди копират, преместват и изтриват файл ***httpd.conf***, първоначално разположен в папка ***/usr/local/etc***.

```
cp /usr/local/etc/httpd.conf /etc/apache2/conf.d/apache.conf
mv /etc/apache2/conf.d/apache.conf /etc/apache2/apache.conf
rm /usr/local/etc/httpd.conf
```

Първата команда копира файла под ново име – ***apache.conf*** – в директорията ***/etc/apache2/conf.d***. Втората команда премества този файл в ***/etc/apache2***, а третата команда изтрива оригиналния файл.

1.2. Текуща директория и относителни пътища

Изписването на напълно квалифицирани имена на файловете е трудоемка операция. Поради тази причина Линукс има различни механизми за съкращаване на изписването. Най-мощният от тях е концепцията за текуща директория.

Текущата директория е директорията с която се работи в момента. Тя може да бъде сменяна по усмотрение на потребителя, а ако даден файл или поддиректория се намират в текущата директория, те могат да се указват само с имената им без да се използва път.

Така например командата ***rm info.txt*** премахва файл ***info.txt*** от текущата директория. Ако текущата директория в момента е ***/usr/local/etc/***, напълно квалифицираното име на файла, който ще бъде изтрит, е ***/usr/local/etc/info.txt***.

Самата концепция за текуща директория работи много добре, понеже потребителите доста често изпълняват поредици команди, които действат върху файловете в една директория, след което изпълняват други поредици от команди, които действат в друга директория и т.н. По този начин смяната на

директории позволява повечето файлове да се указват само с кратките си имена.

Показването на текущата директория в повечето обвивки на Линукс се извършва с командата ***pwd***. Както повечето команди на Линукс, ***pwd*** е съкращение от думи, които описват действието на командата. В случая, те са "***present working directory***" т.е. "текуща работна директория". ***pwd*** обикновено не се изпълнява с допълнителни аргументи:

```
1  lalev@dragonfly:~$ pwd
2  /home/lalev
3  lalev@dragonfly:~$
```

В горния пример, след промпта на обвивката (***lalev@dragonfly:~$***), потребителят е написал командата ***pwd***. Следващият ред е изходът от командата, който показва текущата директория (***/home/lalev***). Последният ред е нов промпт на обвивката, който известява потребителя, че командата е приключила и обвивката е готова за въвеждането на нови команди.

Смяната на текущата директория се извършва с командата ***cd***. ***cd*** обикновено има един задължителен параметър – името на директорията, която трябва да стане текуща.

```
1  lalev@dragonfly:~$ pwd
2  /home/lalev
3  lalev@dragonfly:~$ cd marti
4  lalev@dragonfly:~/marti$ pwd
5  /home/lalev/marti
6  lalev@dragonfly:~/marti$ cd /home/lalev
7  lalev@dragonfly:~$ pwd
8  /home/lalev
9  lalev@dragonfly:~$ cd /home/lalev/marti
10 lalev@dragonfly:~/marti$ pwd
11 /home/lalev/marti
12 lalev@dragonfly:~/marti$
```

В горния пример потребителят започва от текуща директория ***/home/lalev*** и сменя текущата директория на ред 3 от разпечатката. Командата

cd на ред 3 изглежда малко по-особена от показаните досега, тъй като директория ***marti*** не е зададена с напълно квалифицирано име. Това означава, че потребителят има предвид директория с име ***marti***, която е поддиректория на текущата директория.

Командата на ред 6 връща ***/home/lalev*** за текуща директория, докато на девети ред потребителят отново променя текущата директория на ***marti***, като този път той задава напълно квалифицираното име.

Освен с напълно квалифицирано име, пътят до даден файл в Линукс може да се указва като за отправна точка се използва не главната, а текущата директория, както и няколко други важни местоположения във файловата система, които са разгледани по-нататък.. Пътищата, формирани по този начин, се наричат *относителни*. Табл. 2.2. показва примери за задаване на път до файл на база текущата директория.

bin/autodump.sh	Файл (или директория) с име ***autodump.sh***, който се намира в директория ***bin***, която е поддиректория на текущата директория. Важно е да се отбележи, че относителните пътища се различават от абсолютните по това, че не започват с името на главната директория – (/).
bash	Файл (или директория) ***bash***, който се намира в текущата директория.
usr/test/info	Файл (или директория) ***info***, който се намира в директория ***test***, която е поддиректория на директория ***usr***. ***usr*** от своя страна е поддиректория на текущата директория.

Таблица 2.2. Примери за относителни пътища

От приложените по-горе примери може да изглежда, че с относителни пътища могат да се адресират само директории и файлове, които се съдържат

в текущата директория и нейните поддиректории. Всъщност, това не е така, тъй като потребителите могат да използват три специални имена на директории. Тези имена са показани в табл. 2.3.

Име	**Значение**
.	Текуща директория
..	Родителска директория
~	Домашна директория на потребителя

Таблица 2.3. Специални символи на обвивката

Името **..** (две точки) е много удобен начин да се укаже *директорията - родител* на дадена директория. *Директорията-родител* на дадена директория е директорията, която съдържа тази директория. Примери за използването на **..** (две точки) са дадени в таблица 2.4.

..	Родителска директория на текущата. Ако текущата директория е например ***/usr/home/lalev***, това специално име указва директория ***/usr/home***. Ако текущата директория е обаче ***/usr***, това специално име указва главната директория.
../..	Родителска директория на родителската директория на текущата. Ако текущата директория е например ***/usr/home/lalev***, показаният път име указва директория ***/usr***. Ако обаче текущата директория е ***/usr/home***, то показания път указва главната директория. Важно е да се отбележи, че главната директория няма родителска директория, поради което използването на **..** когато текуща директория е главната директория няма ефект. В този случай **..** указва пак главната директория.

	Така например, ако текущата директория е ***/usr***, даденият път указва главната директория. В същата ситуация пътят ***../../../../../../../..*** също указва главната директория.
../usr/test	Файл (или директория) ***test***, която се намира в директория ***usr***, която е поддиректория на родителската директория на текущата.

Таблица 2.4. Примери за задаването на родителска директория в относителен път

Понякога е удобно да се укаже по кратък начин текущата директория. За целта се използва специалното име . (точка). Например при извикване на изпълним файл, който се намира в текущата директория, неговото име може да се укаже по следния начин:

```
1  lalev@dragonfly:/bin$ pwd
2  /bin
3  lalev@dragonfly:/bin$ ./date -R
4  Sun, 18 Feb 2018 18:40:35 +0200
5  lalev@dragonfly:/bin$
```

На ред 3 от примера, ***./date*** обозначава файл, който се намира в текущата директория (***/bin***) и се нарича ***date***. Тъй като ***./date*** не е част от друга команда, обвивката приема, че това е име на изпълним файл, който трябва да се изпълни. В случая това е изпълнимата програма ***date***, която се използва за връщане и настройване на текущите дата и време.

Когато стандартните обвивки в Линукс получават име на изпълним файл, те го търсят в специално определен списък от папки. Този списък обикновено не включва текущата директория, тъй като включването поражда проблеми със сигурността.

За да си представим проблема, нека си представим ситуация, в която администраторът да работи в текущата директория ***/home/dragan***. Би могло да се окаже, че потребител ***dragan*** има лоши намерения и е създал изпълнима програма, която изтрива всички файлове и директории. Нормално ***dragan*** няма право да изтрива файлове и директории, които не са негови. Поради тази причина ако ***dragan*** извика тази програма, тя ще спре с грешка, породена от липсата на нужните разрешения.

Но ***dragan*** има подъл план. Той има достъп за писане до ***/home/dragan*** и записва програмата си там с името ***pwd***. Това име съвпада с името на командата ***pwd***, която също е

изпълним файл, разположен в една от специалните папки на системата, предназначени за изпълнимите файлове на системните команди.

Ако обвивката позволяваше изпълнението на файлове, които се намират в текущата директория, извиквайки ***pwd*** в директорията на ***dragan***, нищо неподозиращият администратор всъщност ще извика *злонамерената* програма и тя ще се изпълни с привилегиите на администратора, който има достъп до всички операции върху файловете и директориите, включително изтриването им.

За да предпазят от подобни грешки, повечето обвивки изискват изпълнението на файлове в текущата директория да се укаже като ***./име_на_файла***, което сигнализира на обвивката, че потребителят *наистина възнамерява* да изпълни файл от текущата директория. Ако префиксът ***./*** липсва, командата бива търсена специален списък с директории, който се настройва от администратора и/или от самия потребител. Този списък се нарича "път" (***path***) и обикновено включва всички папки, в които са разположени системните команди.

Когато става въпрос за задаване на пътища до файлове на база текущата директория, трябва да се отбележи, че повечето програми приемат аргументи, започващи с точка или направо с име, като ги третират по идентичен начин.

Така например командите ***cd test/data*** и ***cd ./test/data*** извършват едно и също действие, тъй като ***test/data*** и ***./test/data*** са един и същ относителен път, който започва от текущата директория. Както горният пример демонстрира, в някои редки случаи като изпълнението на файл, ще работи само вариантът, започващ с точка.

Последното специално име ~ (тилда) отбелязва домашната директория на потребителя.

В Линукс всеки потребител с достъп до командния ред има домашна директория и концепцията е подобна на тази в повечето популярни операционни системи. Домашната директория е мястото, предназначено за съхраняване на файловете, които са собственост на дадения потребител.

Обикновено във файловите системи на различните Линукс дистрибуции домашните директории на потребителите са поддиректории на ***/home***. Така например повечето администратори биха очаквали директорията ***/home/dragan*** да е домашната директория на потребител с име ***dragan***. Разбира се,

изключения могат да съществуват, а мястото, което съдържа информация коя е домашната папка на всеки отделен потребител на системата, е файлът ***/etc/passwd***. Този файл е разгледан специално в следващите глави.

Таблица 2.5 дава примери за пътища, зададени спрямо домашната директория на потребителя:

~/bin	Файл (или директория) с име ***bin***, който е поддиректория на домашната директория на потребителя. Ако тя например е ***/home/deshka***, напълно квалифицираното име на файла (или директорията) е ***/home/deshka/bin*** Важно е да се отбележи, че относителните пътища се различават от абсолютните по това, че не започват с името на главната директория – (/).
~/../dragan	Файл (или директория) ***dragan***, който се намира в родителската директория на домашната директория. Така например ако домашната директория е ***/home/deshka***, указаният файл (или директория) е ***/home/dragan***
~/info/data.csv	Файл (или директория) ***data.csv***, който се намира в директория ***info***, която на свой ред е поддиректория на домашната директория на текущия потребител. Така например ако домашната директория е ***/home/deshka***, указаният файл (или директория) е ***/home/deshka/info/data.csv***

Таблица 2.5. Примери за използване на ~ (тилда) за указване на домашната директория в относителни пътища

Използването на тилда (~) за обозначаване на домашната директория на текущия потребител всъщност е най-употребявания вариант на по-обща нотация, която позволява домашните директории на потребителите да се обозначават с тилда, последвана непосредствено от името на потребителя. Така

например ***~ivan*** и ***~dragan*** са по-кратки начини да се обозначат домашните директории на потребителите с имена съответно ***ivan*** и ***dragan***.

1.3. Основни детайли за организацията на файловата система при Линукс

Файловете и директориите са мощни абстракции. Достатъчно е потребителят да постави данните си във файл с помощта на програма като например текстов редактор. След това операционната система поема грижата за това как битовете и байтовете, които съставляват данните на файла, се разполагат в действителност на дисковия носител. Дейностите по разполагане на данните от файловете на диска са изключително сложни и специфични и са резултат от десетилетни усилия файловите системи да бъдат направени по-бързи, по-ефективни и по-устойчиви на откази. Това, че потребителите не трябва да се занимават с такива детайли, е огромно улеснение, което сваля драстично „ценза“, необходим на даден потребител за да работи с компютъра и операционната му система.

Системните администратори също не мислят ежедневно за това как файловете са разположени на диска, но техните отговорности налагат те да знаят повече детайли за работата и устройството на файловата система.

Първият важен детайл касае т.нар. *„дискови дялове“*. Всеки дисков носител в модерните компютърни системи /например твърд диск с класически дизайн на база магнитна плоча, USB памет, SSD диск и т.н./ трябва да бъде разделен на логически дялове, преди да може да се използва за съхраняване на файлове. В зависимост от нуждите, даден диск може да съдържа само един голям дял или може да бъде разделен на множество по-малки дялове. Всеки дял може да се използва независимо и дори можем да имаме ситуация, когато няколко от дяловете се използват от една операционна система, работеща на

даден компютър (например Microsoft Windows), докато останалите се използват от друга операционна система (например Линукс).

Повечето дискови дялове са предназначени за съхраняване на файлове, но съществуват и изключения. Тези дялове, които са предназначени за съхраняване на файлове, поместват данните на файловете, съблюдавайки някаква определена структура. Тази структура се нарича *„файлова система“*. Различните операционни системи работят с различни структури (тоест различни файлови системи). Така например Windows използва файлови системи като FAT32 и NTFS за своите дялове, докато Линукс използва EXT2, EXT3, EXT4, както и много други специализирани файлови системи.

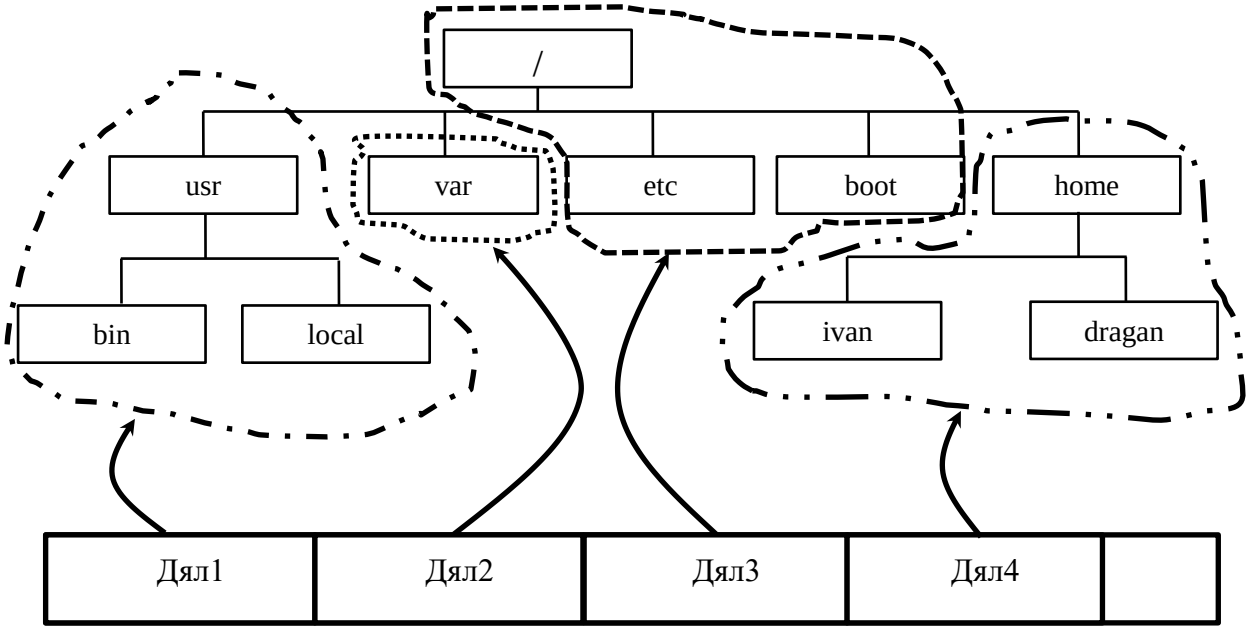
Процесът по изграждането на структурите от данни, които ще съхраняват информация за файловете, се нарича *„форматиране“*. Всеки един новосъздаден дисков дял, който ще съхранява файлове, трябва да се форматира, тоест в него трябва да се създадат структурите от данни на избраната файлова система.

Читателите вероятно са запознати с факта, че в операционните системи от фамилията на Windows, всеки отделен форматиран дисков дял получава буквен идентификатор, като например ***C:***, ***D:*** и т.н. Съответно всеки дял има главна директория, поддиректории и т.н. В Линукс всеки дял също има главна директория и поддиректории, но Линукс предпочита друг подход при който всички устройства и всички дялове се представят на потребителя и на софтуера като една-единствена обединена йерархия от директории. Този подход има огромни предимства, когато става въпрос за преносимост и устойчивост на настройките на потребителските програми.

За да се получи така, че файловете от всички дискови дялове да се намират в една йерархия от директории с една главна директория, отделните дялове в Линукс се „монтират“ в дадена папка. Процесът на *„монтиране“*

поставя главната директория на всеки дял в една от директориите на системата. В някакъв смисъл, това прави йерархията на директориите, която администраторът, потребителите и програмите им виждат, „виртуална“ файлова система, тъй като тя обхваща файловите системи на няколко дискови дяла, които на свой ред могат да бъдат разположени на един диск или на множество дискове. Тази йерархия дори обхваща и устройства като DVD-R и BluRay дискове, които се появяват като поддиректории в йерархията.

На фиг. 2.2. е показана примерната йерархия от директории от фиг. 2.1 по горе. Този път е показано как тя може да бъде съставена от няколко дяла чрез монтиране.



Фигура 2.2. Монтиране на дялове в йерархията на директориите

Така например, според диаграмата, главната директория на дял 1 е монтирана в папка ***/usr***. Когато потребителят влезе в тази папка, той вижда съдържанието на дял 1. Дял 2 е монтиран в папка ***/var***, а дял 4 е монтиран в папка ***/home***. Дял 3 от своя страна е монтиран като главна директория, тоест

неговата главна директория е главна директория за цялата система, а в неговите папки ***/usr***, ***/var*** и ***/home*** са монтирани другите дискови дялове.

Процесът по монтиране на дисковите дялове се извършва при зареждане на операционната система. По всяко време на работа, потребителят може да монтира допълнителни дялове и устройства както и да демонтира такива, при условие обаче че файловете от демонтираните дялове и устройства не са в употреба.

При монтирането на дял в дадена папка, нейното оригинално съдържание обикновено се скрива за сметка на съдържанието на главната папка на монтирания дял или устройство.

Основната „атомарна“ единица, с която боравят файловите системи, е блок от данни. За модерните файлови системи на Линукс, големината на такъв блок е обикновено 4096 байта. При записване на даден файл на диска, неговите данни се поставят в един или повече блокове от данни. Един блок може да съдържа данните само на един файл. Поради тази причина за даден файл винаги се заделят цяло число блокове, независимо от това дали съдържанието на файла запълва докрай последния блок или не.

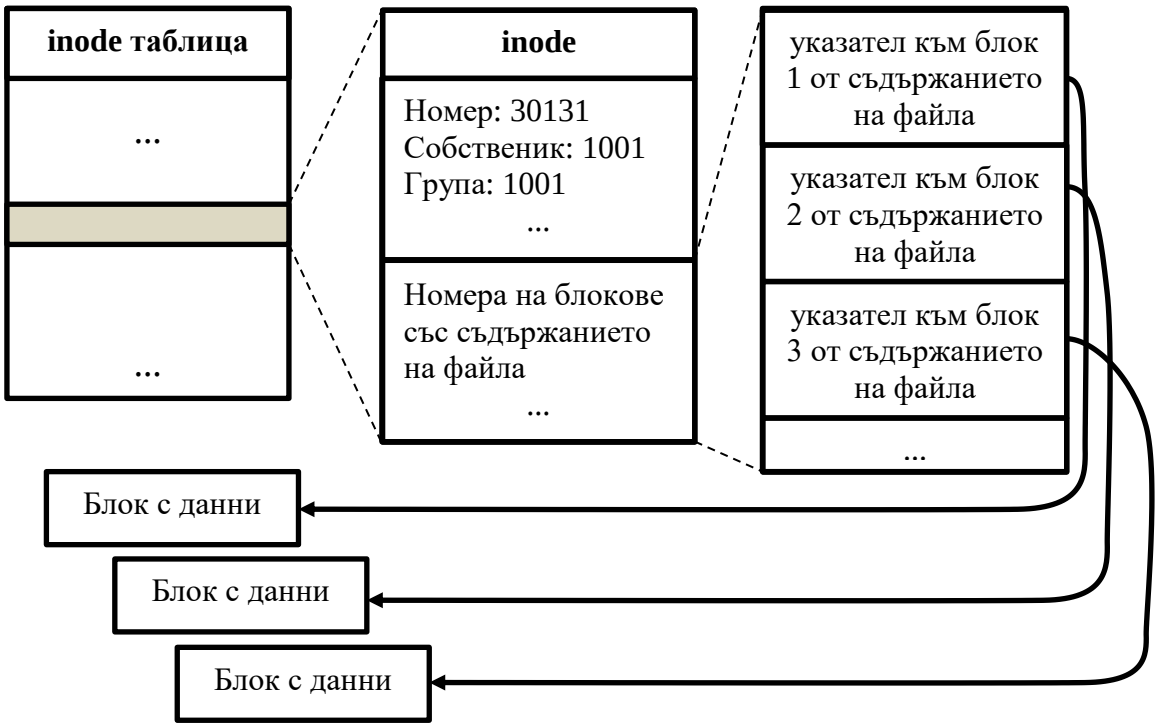
За да съществува яснота кои блокове се отнасят към даден файл, както и за съхраняване на друга служебна информация, за всеки файл в даден дисков дял съществува специална структура - ***inode***[5]. Наред с други данни, ***inode*** структурата съдържа следната важна информация за един файл:

- уникален сериен номер (номер на ***inode***)

[5] Името ***inode*** е наследство от UNIX системите на 80-те години на миналия век, които Линукс емулира и надгражда. Очевидно е, че ***inode*** е съкращение, но както много пъти се случва, историята е забравила от какво словосъчетание е получено то. Основното предположение е, че става въпрос за “***index node***”, което на български може да се преведе като „индексен възел“. За да не въвеждаме обаче излишна терминология, която не кореспондира с оригиналната документация на английски език, която използва винаги съкращението, ние също ще го използваме директно.

- разрешения за достъп до файла
- собственик и група-собственик на файла
- време на последен достъп
- време на последна модификация на съдържанието на файла
- време на последна модификация на служебната информация в ***inode*** структурата
- брой на твърдите връзки към файла
- списък с блоковете на дисковия дял, в които се съхраняват реалните данни на файла

Важно е да се отбележи, че ***inode*** структурата не съдържа информация име на файла или папка, в която се съдържа файла. Поне за файловата система на ниско ниво, файловете имат само уникални номера.



Фигура 2.3. Организация на файловете на дисковия дял

inode структурите се поместват в таблица (вж. фиг.2.3). Всеки форматиран дял има по една такава таблица (заемаща непрекъсната част от дисковия дял или разпределена между групи от блокове с данни).

inode таблицата съдържа данните за всеки един файл на дисковия дял, но тя е „плоска“ структура, която няма йерархия или категории. За да се получи йерархията на директории, поддиректории и файлове, които се съдържат в директорията, Линукс избира да представи директориите на диска като файлове. Тези файлове притежават същите атрибути като останалите (тоест номер на inode, дати на създаване, модификация и достъп, списък с блокове, които съдържат данните на файла и пр.) но тяхното съдържание е оформено по специален начин.

Всяка директория всъщност съдържа проста таблица с две важни колони - име на файл и номер на ***inode*** (вж. фигура 2.4)

име на файл	***inode***
data.csv	13881
info.txt	3199
subdirectory1	9188
subdirectory2	19383

Фигура 2.4. Примерно съдържание на файл-директория

Файловете, указани в таблицата на директорията, се показват от операционната система на потребителите и техните програми, като съдържащи се в тази директория. Номерът на ***inode*** реално дава връзка към ***inode*** таблицата и указва за кой файл става въпрос, докато името на файла е това, което се показва на потребителя.

Имената на файловете „живеят“ в директориите и само там. Ако дадена директория бъде загубена поради например грешка във файловата система,

файловете в нея няма да изчезнат. Въпреки че файловете с техните ***inode*** структури ще са непокътнати на дисковия дял, имената им ще бъдат загубени, а самите файлове ще станат недостъпни за потребителя, тъй като никоя директория не сочи към тях. При поправка на файловата система с инструменти като ***fsck*** подобни загубени и намерени файлове се поставят с нови, автоматично генерирани цифрови имена, в директория ***lost+found***. Всеки дисков дял има такава директория, която обикновено е поддиректория на главната директория на дяла.

2. Работа с директории

2.1. Създаване на директории

Създаването на папка в Линукс се извършва с командата ***mkdir***[6]. Общият вид на командата е

```
mkdir име_на_директория_1 [име_на директория_2 ... име на директория_n]
[ -p ]
```

име_на_директория_1 е името, което ще бъде дадено на новосъздадената директория. По подразбиране директорията с указаното име ще бъде създадена като поддиректория на текущата директория.

Така например ***mkdir test*** изпълнена в текуща директория ***/root*** ще създаде директория с напълно квалифицирано име ***/root/test.*** Ако *име_на_папка* е съдържа абсолютен или относителен път като например ***/home/lalev/new/test*** или ***./log/apache***, директорията ще бъде създадена на указаното от пътя място

[6] Името на командата е съкратено от англ. “Make Directory” - „Създай директория”

във файловата система, като последното (най-дясно) име от пътя ще бъде интерпретирано като име на новата директория.

Така например ***mkdir /home/lalev/new/test*** ще създаде директория с име тест в ***/home/lalev/new,*** а ***mkdir ./log/apache*** ще създаде директория ***apache*** в директория ***log***, която на свой ред е поддиректория на текущата директория. Долният пример показва използването на ***mkdir*** в различни сценарии.

```
 1  lalev@dragonfly:~$ pwd
 2  /home/lalev
 3  lalev@dragonfly:~$ mkdir newapp
 4  lalev@dragonfly:~$ cd newapp
 5  lalev@dragonfly:~/newapp$ mkdir src
 6  lalev@dragonfly:~/newapp$ cd src
 7  lalev@dragonfly:~/newapp/src$ mkdir main
 8  lalev@dragonfly:~/newapp/src$ cd main
 9  lalev@dragonfly:~/newapp/src/main$ cd ~
10  lalev@dragonfly:~$ mkdir newapp/test/java
11  mkdir: cannot create directory ‘newapp/test/java’: No such file or
    directory
12  lalev@dragonfly:~$ mkdir -p newapp/test/java
13  lalev@dragonfly:~$
```

На ред 3, потребителят създава директория с име ***newapp***. Името не съдържа абсолютен или относителен път, което означава, че новата папка ще бъде създадена като подпапка на текущата. Промптът на обвивката показва текущата директория като ~. Това означава, че ***newapp*** ще бъде създадена като поддиректория на домашната директория на потребител ***lalev***. На ред 4, потребителят използва командата ***cd***, за да се придвижи в новосъздадената папка.

На ред 5, както промптът показва, потребителят се намира в новосъздадената директория ***newapp***. Там той издава нова команда, която води до създаването на директория ***~/newapp/src*** (тоест, директория с име ***src***, която е поддиректория на ***newapp***, която на свой ред е поддиректория на домашната папка на потребител ***lalev***.

По-интересната част от примера започва на ред 10. На този ред потребителят се намира отново в домашната си директория. Този път обаче той извиква ***mkdir*** с параметър, който съдържа относителен път. Параметърът се интерпретира от ***mkdir*** като инструкция за създаване на директория ***java*** като поддиректория на ***./newapp/test***. Директория ***newapp*** съществува, но в нея няма поддиректория ***test***, което води до съобщение за грешка.

На ред 12 обаче потребителят въвежда същата команда, този път придружена от аргумент ***-p***. Този аргумент указва модификация на поведението на ***mkdir***. Ако дадена директория от пътя не съществува, ***mkdir*** този път ще я създаде. Както е видно от ред 13 на примера, ***mkdir*** този път завършва успешно и като резултат от неговата дейност са създадени две директории – директорията ***~/newapp/test*** и нейната поддиректория ***~/newapp/test/src***.

Възможност, която се използва относително рядко, е възможността към ***mkdir*** да се подаде не едно име на директория, а множество имена. В базовия синтаксис на командата, посочен по-горе *[име_на_директория_2 ... име_на_директория_n]* представлява именно подобен списък от имена. Така например ***mkdir test1 test2 test3*** ще доведе до създаването на три поддиректории на текущата с указаните имена.

2.2. Премахване на директория

При нужда от изтриване на директории, това може да бъде извършено с командата ***rmdir.*** Базовият синтаксис на командата е

```
rmdir име_на_директория_1  [ име_на директория_2 ... име на
директория_n ]
```

rmdir ще откаже да изтрие директории, които не са празни, т.е. съдържат файлове или поддиректории. Пример за използването на ***rmdir*** е даден по-долу:

```
1 lalev@dragonfly:~$ mkdir prevyear
2 lalev@dragonfly:~$ mkdir prevyear/jan
3 lalev@dragonfly:~$ rmdir prevyear
4 rmdir: failed to remove 'prevyear': Directory not empty
5 lalev@dragonfly:~$ mkdir nextyear
6 lalev@dragonfly:~$ rmdir nextyear
7 lalev@dragonfly:~$ rmdir prevyear/jan prevyear
8 lalev@dragonfly:~$
```

На ред в 3 в примера потребителят прави опит да изтрие директорията ***prevyear***, която е създадена с командата на ред 1. ***prevyear*** обаче има вече поддиректория ***jan***, което води до грешка. Няма такъв проблем с директория ***nextyear*** тъй като към момента на изтриването си (ред 6), тя е празна.

На ред 7, потребителят използва възможността да зададе не един файл, а списък от файлове. Тъй като ***prevyear/jan*** е първа в списъка, тя се изтрива първа, което оставя директорията ***prevyear*** празна. Ето защо изтриването и този път преминава без грешка.

2.3. Показване на съдържанието на директория

Показването на съдържанието на дадена директория е една от най-често използваните от потребителите функции. Командата ***ls*** извършва именно това и може да се извиква с множество параметри, поради което е удобно синтаксисът и да се представя на части.

Извикването на ***ls*** без параметри показва имената на файловете и директориите, които се съдържат в текущата директория.

```
1 lalev@dragonfly:/boot$ ls
2 abi-4.15.0-32-generic         memtest86+.elf
3 abi-4.15.0-34-generic         memtest86+_multiboot.bin
4 abi-4.15.0-36-generic         retpoline-4.15.0-32-generic
5 config-4.15.0-32-generic      retpoline-4.15.0-34-generic
```

```
 6 config-4.15.0-34-generic      retpoline-4.15.0-36-generic
 7 config-4.15.0-36-generic      System.map-4.15.0-32-generic
 8 grub                          System.map-4.15.0-34-generic
 9 initrd.img-4.15.0-32-generic  System.map-4.15.0-36-generic
10 initrd.img-4.15.0-34-generic  vmlinuz-4.15.0-32-generic
11 initrd.img-4.15.0-36-generic  vmlinuz-4.15.0-34-generic
12 lost+found                    vmlinuz-4.15.0-36-generic
13 memtest86+.bin
14 lalev@dragonfly:/boot$
```

В зависимост от това дали ***ls*** извежда съдържанието си директно на конзолата (типичният начин на употреба) или към входа на друга програма, поведението на ***ls*** (в случая, когато се използва без параметри) е леко различно. На показания по горе пример изходът е директно към конзолата, при което ***ls*** извежда имената на файловете като таблица от няколко колони. Броят на колоните е ограничен от дължината на имената и ширината на конзолата в символи. Когато изходът е към друга програма (например филтрите, които ще бъдат разгледани в следващите глави), имената се извеждат по едно на ред, т.е. в една единствена колона.

Когато конзолата поддържа цветове, имената на файловете се изписват в различен цвят, който показва типа на файла и разрешенията за достъп до него. Така например, директориите се изписват в син цвят. Други цветове са показани в таблица 2.6 (повечето от типовете и разрешенията, споменати в таблицата, ще бъдат разгледани по-нататък).

бял на черен фон	нормален файл
син на черен фон	директория
червен на черен фон	архивен файл (файл с разширение .zip, .tar, .gz, .tar.gz и пр.)
лилав на черен фон	файл с графично изображение (например файл с разширение .gif)
циан на черен фон	файл - символична връзка
жълт на черен фон	файл на устройство
бял на червен фон	setuid изпълним файл
бял на кафяв фон	setgid изпълним файл

бял на зелен фон	директории с установен (включен) sticky бит
кафяв на черен фон	FIFO специален файл

*Таблица 2.6. Оцветяване на имената на файловете в изхода на **ls** към конзола, която поддържа цветове*

Когато конзолата не поддържа цветове, няма как от изхода на ***ls*** да се разбере директно дали файлът е например директория. За щастие, ***ls*** поддържа т.нар. „дълъг формат“ на изхода, при който всеки файл или поддиректория се извеждат на един ред, заедно с ключови детайли за размера, разрешенията за достъп, типа и датите на файла. Дългият формат на изхода се включва чрез извикване на ***ls*** с опция ***-l.***

```
1 lalev@dragonfly:/etc/network$ ls -l
2 total 20
3 drwxr-xr-x 2 root root 4096 Sep 11 11:06 if-down.d
4 drwxr-xr-x 2 root root 4096 Sep 11 11:03 if-post-down.d
5 drwxr-xr-x 2 root root 4096 Sep 11 10:45 if-pre-up.d
6 drwxr-xr-x 2 root root 4096 Sep 11 11:07 if-up.d
7 -rw-r--r-- 1 root root  161 Apr  3  2018 interfaces
8 lalev@dragonfly:/etc/network$
```

Дългият формат на изхода на ***ls*** дава 7 колони, последната от които е името на файла или директорията. Първата колона съдържа 10 символа. Първият от тях описва типа на файла, докато останалите 9 (три тройки от по три символа) описват разрешенията за достъп.

На примера, символът ***d*** в началото на първата колона показва, че става въпрос за директория, докато символът **–** (тире) показва, че става въпрос за обикновен файл. Други символи за типа на файла са показани в табл. 2.7.

-	обикновен файл
d	директория
c	специален файл на символно устройство
b	специален файл на блоково устройство
l	файл - символична връзка
s	локален домейн сокет

p	FIFO специален файл[7]

Таблица 2.7. Символ - индикатор за типа на файла в дългата форма на изхода на ***ls***

Втората колона на дългия формат на ***ls*** показва броя на т.нар. *„твърди връзки“*, сочещи към показания файл. Твърдите връзки към файл позволяват на даден файл да фигурира под различни имена в различни директории на файловата система, което дава гъвкавост и позволява логически подреждания на информацията, които иначе биха били невъзможни. Твърдите връзки ще бъдат разгледани по-нататък в настоящата глава.

Колона 3 показва потребителя, който е собственик на файла. В Линукс, всеки файл трябва да е собственост на даден потребител. В нашия пример четирите директории и единия файл, които се намират в ***/etc/network*** са собственост на администратора на системата – потребителят ***root***.

Всеки файл освен потребител-собственик трябва да има и група потребители – собственици. Когато от контекста е ясно, че става дума за разрешения за достъп, вместо „потребител-собственик на файла“ се употребява далеч по-простото „собственик на файла“, и което е по-важното - вместо „група от потребители - собственици на файла“, често се употребява просто „група-собственик на файла“ или дори „група на файла“. Именно групата-собственик на файла е показана в колона 4 на дългия формат на изхода на ***ls***.

Групата-собственик съществува, за да позволи групова работа. Много често възникват ситуации, в които собственикът на файла иска да даде ограничени права върху него и на други потребители, за да улесни работата в

[7] FIFO специалните файлове, също като локалните сокети (наричани още “UNIX домейн сокети”), са част от механизмите за междупроцесна комуникация. На английски FIFO специалните файлове са известни още и като “named pipes” или буквално „именувани тръби“, което често е превеждано неточно на български като „именувани пренасочвания“, тъй като приличат по идея на механизма на пренасочване на изхода на една програма към друга.

екип по дадена задача. Една честа ситуация с правата например е положение при което собственикът на файла може да чете и записва съдържанието на файла, докато членовете на групата-собственик могат само да четат съдържанието на файла без да го променят.

Повечето от файловете рядко се използват групово, което влиза в конфликт с изискването всеки файл да има група-собственик. Ето защо в Линукс проблемът се решава, като за всеки нов потребител се създава и специална група, в която той е единствен член. Обикновено тази група се казва по същия начин като потребителя. Горният пример показва именно такава ситуация, тъй като файловете са собственост на потребител ***root*** и група ***root***. Ако възникне нужда тези файлове да се използват от няколко потребителя, при подобен на гореизложения сценарий, администраторът би създал например група ***projectA***, включваща нужните потребители, след което би сменил групата-собственик на файловете на ***projectA***.

Колона 5 на дългия формат на изхода на ***ls*** показва размера на файловете и директориите в байтове. За обикновен файл числото, показано в тази колона, представлява точния размер на съдържанието на файла. Тоест, ако даден файл съдържа например данни с дължина 312 байта, дължината на файла, показана в колона 5 ще бъде 312 байта, независимо от количеството блокове, които файлът заема на диска.

Бе вече споменато, че файловата система е разделена на блокове с фиксиран размер (обикновено 4096 байта) и един файл на диска заема винаги цял брой такива блокове. Така например дори най-малкият файл с 1-2 символа съдържание на диска би заел един блок от 4096 байта. Файл с реална дължина 5000 байта на диска би заел 2 блока от 4096 или общо 8192 байта и т.н.

Ситуацията при директориите е по-сложна. Числото, показано за дължина на директория, не е нито брой на файловете в директорията нито сумата от размерите на тези файлове. За директориите ***ls*** показва броя на

блоковете, които файлът на директорията заема на диска. Ето защо показваните от ***ls*** големини на директории са кратни на 4096 (или избраният друг размер на блока на файловата система).

Читателят може да експериментира и да проучи информацията за даден файл или директория във файловата система с командата ***stat***. Той трябва да е предупреден обаче, че командата ***stat*** по исторически причини отчита големините в блоковете от 512 байта независимо от реалния размер на блока на файловата система.

Шестата колона на дългия формат на изхода на ***ls*** дава едно от трите времена, които се пазят за всеки файл във файловите системи на Линукс. Тези времена са три: време на последна модификация, време на последен достъп, както и време на последна промяна на служебната информация на файла.

По подразбиране ***ls*** извежда времето на последната модификация на файла, което отразява кога последно съдържанието на файла е било променяно.

Ако ***ls*** се извика с допълнителен параметър ***-u***, тоест ***ls -l -u*** или съкратено ***ls -lu***, вместо времето на последна модификация ще бъде изведено времето на последен достъп до файла. Времето на последен достъп до файла отразява кога последно съдържанието на файла е било прочетено от файловата система.

Ако ***ls*** се извика с допълнителен параметър ***-c***, т.е. ***ls -l -c*** или ***ls -lc***, вместо времето на модификация ще бъде изведено времето на последна промяна на служебната информация на файла. Служебната информация включва името на файла, както и разрешенията за достъп до него, т.е. тя запазва информация за последното преименуване, промяна на разрешенията за достъп, смяна на групата-собственик и пр.

Трите времена всъщност се съхраняват в ***inode*** структурата на файла, очертана по-горе. В документацията и изходния код на модерните Линукс системи те още могат да се видят на места като ATIME (време на последен достъп), MTIME (време на последна модификация) и CTIME (време на последна модификация на служебната информация). ATIME и MTIME са доста интуитивни. В по-старите UNIX системи CTIME всъщност

представлява времето на създаване на файла. С времето се е оказало по-удобно CTIME да се обновява всеки път, когато служебната информация в ***inode*** структурата се променя. Това води до интересни последствия. При смяна на собственика или групата на файла, CTIME се променя, тъй като тези операции засягат информацията в ***inode*** структурата. Тъй като ATIME и MTIME също се съхраняват в ***inode*** структурата, промяната на едно от двете времена би трябвало да води до обновяване и на CTIME. Това обаче е така само за MTIME. Промяната на ATIME не води до промяна на CTIME и това изглежда единственото поле на ***inode*** структурата, чието обновяване не води до обновяване на CTIME. Примерите, които демонстрират това поведение, са оставени като упражнения към настоящата глава.

Нормално ***ls*** сортира файловете по техните имена, но ако към списъка с параметри се добави ***-t***, файловете ще бъдат подредени по колоната за време. Ако вместо това се добави –S, списъкът ще бъде сортиран по големина на файловете, като най-големият файл ще бъде показан първи. Ако по някаква причина е нужно да се обърне сортирането, това става с подаване на параметър ***-r***.

Важен параметър на ***ls*** е ***-a***. Той предизвиква показването на т.нар. скрити файлове. В Линукс име на файл или директория, което започва с точка (например ***.m2***, ***.profile***), се счита за скрит файл. Скритите файлове нормално не се показват в изхода на ***ls***. Подобни файлове и директории най-често съдържат потребителски настройки, които се скриват, за да се избегне инцидентното изтриване (командите за премахване на файлове, също като ***ls*** ще пропуснат скритите файлове, ако не са зададени специални параметри).

Друг важен параметър на ***ls*** е ***-R***, който предизвиква показването на всички поддиректории и тяхното съдържание. На по-професионален език, ***-R*** предизвиква рекурсивно обхождане на поддиректориите и извеждане на тяхното съдържание.

Освен споменатите параметри, ***ls*** приема и списък от един или повече пътища във файловата система, подобно на по-горе разгледаните команди ***mkdir*** и ***rmdir***. Ако такива пътища са зададени, вместо текущата директория ще бъде изведено съдържанието на указаната директория (евентуално –

директории). Пътищата дори могат да указват абсолютно или относително един конкретен файл, при което ***ls*** ще изведе информация само за него.

По долу са дадени примери за използване на ***ls*** с описаните до момента параметри:

```
 1 lalev@dragonfly:/etc/network$ ls -la
 2 total 36
 3 drwxr-xr-x   6 root root  4096 Apr  3  2018 .
 4 drwxr-xr-x 163 root root 12288 Oct 22 11:52 ..
 5 drwxr-xr-x   2 root root  4096 Sep 11 11:06 if-down.d
 6 drwxr-xr-x   2 root root  4096 Sep 11 11:03 if-post-down.d
 7 drwxr-xr-x   2 root root  4096 Sep 11 10:45 if-pre-up.d
 8 drwxr-xr-x   2 root root  4096 Sep 11 11:07 if-up.d
 9 -rw-r--r--   1 root root   161 Apr  3  2018 interfaces
10 lalev@dragonfly:/etc/network$ ls -lar
11 total 36
12 -rw-r--r--   1 root root   161 Apr  3  2018 interfaces
13 drwxr-xr-x   2 root root  4096 Sep 11 11:07 if-up.d
14 drwxr-xr-x   2 root root  4096 Sep 11 10:45 if-pre-up.d
15 drwxr-xr-x   2 root root  4096 Sep 11 11:03 if-post-down.d
16 drwxr-xr-x   2 root root  4096 Sep 11 11:06 if-down.d
17 drwxr-xr-x 163 root root 12288 Oct 22 11:52 ..
18 drwxr-xr-x   6 root root  4096 Apr  3  2018 .
19 lalev@dragonfly:/etc/network$
```

Показаният по-горе пример показва използването на опции ***-l***, ***-a*** и ***-r***. Видимо е, че първото извикване на ***ls*** сортира файловете *възходящо* по имената им, докато второто обръща реда и той става *низходящ*.

Показаният по-горе пример има още интересни детайли, които си заслужава да бъдат дискутирани. Файловете . (точка) и .. (двойна точка) са скрити и се показват само защото е подаден аргумент ***-a***. Тези файлове са начинът, по който се осъществява връзка към текущата директория и директорията-родител. От техните имена произтичат споменатите по-горе „символи“ за задаване на относителни пътища.

Друг важен детайл е, че дългият формат на изхода на командата ***ls*** всъщност започва с реда ***total.*** Това се случва всеки път, когато се показва цялото съдържание на една директория. Този ред съществува по-скоро по съображения за съвместимост с по-старите UNIX системи, които съблюдават стандарта POSIX.

В POSIX-съвместимите системи, числото след ***total*** показва сумарния размер на дисковото пространство, заделено за всички файлове, съдържани в директорията. При това мястото е указано в брой блокове по 512 байта, тъй като това е историческия размер на блока във файловата система на UNIX.

Модерните файлови системи на Линукс използват значително по-големи блокове, поради което информацията, предоставяна от ***total***, не е толкова полезна. В Линукс ***ls*** обикновено нарушава POSIX съвместимостта и показва в реда ***total*** същите размери, но в блокове по 1024 байта. По този начин информацията всъщност показва колко килобайта са резервирани на диска за всички файлове в директорията.

Поведението на ***ls*** се модифицира от променливата на обкръжението POSIXLY_CORRECT. Ако тази променлива е дефинирана и има стойност 1, ***ls*** ще се държи като ***ls*** в UNIX и ще показва размера на файловете в блокове от 512 байта. Ако потребителят желае блоковете, използвани от ***ls***, да имат друг размер, той може да зададе параметъра ***--block-size=xxx*** където ***xxx*** е цяло число. Значително по-модерен начин за извеждане на големините на файловете е използването на аргумента ***-h***. Този аргумент е разгледан по-нататък

Още един интересен детайл касае показаните на предния пример времена. Файлът ***interfaces*** е показан с време на последна модификация ***Apr 3 2018***, докато времето на последна модификация на директорията ***if-up.d*** е показано като ***Sep 11 11:06***. Това се дължи на опита на ***ls*** да икономисва хоризонтално пространство. За файловете, чиито времена са в текущата година, се показват месец, дата и час. За файловете с времена от предходни години се показват месец, ден и година. За да се изведат всички компоненти на времето може да се използва параметърът ***--time-style=long-iso***.

```
 1 lalev@dragonfly:/etc/network$ ls -lau --time-style=long-iso
 2 total 36
 3 drwxr-xr-x   6 root root  4096 2018-10-29 16:35 .
 4 drwxr-xr-x 163 root root 12288 2018-10-29 12:12 ..
 5 drwxr-xr-x   2 root root  4096 2018-10-28 06:07 if-down.d
 6 drwxr-xr-x   2 root root  4096 2018-10-29 12:35 if-post-down.d
 7 drwxr-xr-x   2 root root  4096 2018-10-28 06:07 if-pre-up.d
 8 drwxr-xr-x   2 root root  4096 2018-10-29 11:57 if-up.d
 9 -rw-r--r--   1 root root   161 2018-10-29 16:29 interfaces
10 lalev@dragonfly:/etc/network$ ls if-pre-up.d
11 ethtool  vlan  wireless-tools  wpasupplicant
12 lalev@dragonfly:/etc/network$ ls -lau --time-style=long-iso
13 total 36
14 drwxr-xr-x   6 root root  4096 2018-10-29 16:35 .
15 drwxr-xr-x 163 root root 12288 2018-10-29 12:12 ..
```

```
16 drwxr-xr-x   2 root root  4096 2018-10-28 06:07 if-down.d
17 drwxr-xr-x   2 root root  4096 2018-10-29 12:35 if-post-down.d
18 drwxr-xr-x   2 root root  4096 2018-10-29 18:28 if-pre-up.d
19 drwxr-xr-x   2 root root  4096 2018-10-29 11:57 if-up.d
20 -rw-r--r--   1 root root   161 2018-10-29 16:29 interfaces
21 lalev@dragonfly:/etc/network$
```

Горният пример започва с подобна на предния пример команда, но този път са показани времената на последен достъп до файловете. Директорията ***if-pre-up.d*** има време на достъп ***2018-10-28 06:07***.

На ред 10 ***ls*** е използвана с относителен път за да покаже съдържанието на поддиректорията ***if-pre-up.d***. Тук не е използван дълъг формат на изхода, поради което ***ls*** извежда само списък на имената на файловете. По важното е, че командата ***ls*** на ред 12 извежда отново съдържанието на текущата директория, в което може да се види, че времето на последен достъп до директорията ***if-pre-up.d*** е променено (ред 18). Това се дължи на факта, че за да изведе съдържанието на директорията ***if-pre-up.d***, операционната система изчита файла на директорията от диска. Новото време е именно момента, в който това се е случило - ***2018-10-29 18:28***.

Следващият пример демонстрира някои от вече споменатите параметри на ***ls***. Към тях е добавен и един непредставян до момента параметър ***–h***.

```
 1 lalev@dragonfly:/etc/network$ ls –laRSrh
 2 .:
 3 total 36K
 4 -rw-r--r--   1 root root  161 Apr  3  2018 interfaces
 5 drwxr-xr-x   2 root root 4.0K Sep 11 11:07 if-up.d
 6 drwxr-xr-x   2 root root 4.0K Sep 11 10:45 if-pre-up.d
 7 drwxr-xr-x   2 root root 4.0K Sep 11 11:03 if-post-down.d
 8 drwxr-xr-x   2 root root 4.0K Sep 11 11:06 if-down.d
 9 drwxr-xr-x   6 root root 4.0K Apr  3  2018 .
10 drwxr-xr-x 163 root root  12K Oct 29 12:11 ..
11
12 ./if-up.d:
13 total 32K
14 lrwxrwxrwx 1 root root   32 Aug  9 17:34 wpasupplicant ->
   ../../wpa_supplicant/ifupdown.sh
```

```
15 -rwxr-xr-x 1 root root  227 Apr 20  2018 vlan
16 -rwxr-xr-x 1 root root  484 Jun 27  2016 avahi-daemon
17 -rwxr-xr-x 1 root root  677 Sep 20  2017 ip
18 -rwxr-xr-x 1 root root  923 Sep 19  2017 avahi-autoipd
19 -rwxr-xr-x 1 root root 1.1K Feb 23  2018 postfix
20 -rwxr-xr-x 1 root root 1.7K Jun 30  2016 ethtool
21 drwxr-xr-x 6 root root 4.0K Apr  3  2018 ..
22 drwxr-xr-x 2 root root 4.0K Sep 11 11:07 .
23
24 ./if-pre-up.d:
25 total 24K
26 lrwxrwxrwx 1 root root   32 Aug  9 17:34 wpasupplicant ->
   ../../wpa_supplicant/ifupdown.sh
28 -rwxr-xr-x 1 root root  344 Jun 30  2016 ethtool
29 -rwxr-xr-x 1 root root 2.8K Apr 20  2018 vlan
30 drwxr-xr-x 6 root root 4.0K Apr  3  2018 ..
31 drwxr-xr-x 2 root root 4.0K Sep 11 10:45 .
32 -rwxr-xr-x 1 root root 4.1K Apr 22  2017 wireless-tools
33
34 ./if-post-down.d:
35 total 16K
36 lrwxrwxrwx 1 root root   23 Aug 28 12:37 avahi-daemon -> ../if-
   up.d/avahi-daemon
37 lrwxrwxrwx 1 root root   32 Aug  9 17:34 wpasupplicant ->
   ../../wpa_supplicant/ifupdown.sh
38 -rwxr-xr-x 1 root root 1011 Sep 20  2017 vlan
39 -rwxr-xr-x 1 root root 1.4K Apr 22  2017 wireless-tools
40 drwxr-xr-x 6 root root 4.0K Apr  3  2018 ..
41 drwxr-xr-x 2 root root 4.0K Sep 11 11:03 .
42
43 ./if-down.d:
44 total 16K
45 lrwxrwxrwx 1 root root   32 Aug  9 17:34 wpasupplicant ->
   ../../wpa_supplicant/ifupdown.sh
46 -rwxr-xr-x 1 root root  800 Feb 23  2018 postfix
47 -rwxr-xr-x 1 root root 1015 Sep 19  2017 avahi-autoipd
48 drwxr-xr-x 6 root root 4.0K Apr  3  2018 ..
49 drwxr-xr-x 2 root root 4.0K Sep 11 11:06 .
```

Параметърът ***–R*** предизвиква рекурсивно показване на съдържанието на поддиректориите. Тоест, освен съдържанието на текущата директория (редове 3-10), се показва също съдържанието на всички поддиректории на текущата, заедно с евентуални поддиректории на поддиректориите и т.н. Процесът на

извеждане продължава докато цялото разклонение на дървото на директориите под текущата директория бъде обходено и показано.

Читателят следва да обърне внимание на начина, по който е обозначено и отделено съдържанието на отделните директории. На редове 2, 12, 24, 34 и 43 извеждането на съдържанието на съответната директория започва с извеждането на относителния път до нея. Така например на ред две текущата директория (***/etc/networks***) е обозначена с точка.

Опцията **–S** предизвиква сортирането на информацията за файловете във всяка директория по колоната за размер на файла. Нормално сортирането би се извършило във низходящ ред, но аргументът ***–r*** обръща това поведение и вместо това съдържанието на всяка директория е сортирано възходящо по размера на файловете.

Опцията ***–h*** от своя страна идва от „human” и предизвиква отпечатването на размерите на по-големите файлове в по-удобни за обхващане от човека мерки като килобайти, мегабайти и гигабайти. Ако размерът на даден файл е отпечатан в килобайти, зад числото за неговия размер се поставя буквата “K”. По същия начин буквите M и G обозначават, че размерът е даден в мегабайти или гигабайти.

Като последен пример е показана демонстрация на работата на ***ls*** в комбинация с механизмите на обвивката за масово посочване на файлове.

```
 1 lalev@dragonfly:/$ ls -ld m??
 2 drwxr-xr-x 2 root root 4096 Jan  8  2018 mnt
 3 lalev@dragonfly:/$ ls -ld m?
 4 ls: cannot access 'm?': No such file or directory
 5 lalev@dragonfly:/$ ls -ld m*
 6 drwxr-xr-x 4 root root 4096 Apr 17  2018 media
 7 drwxr-xr-x 2 root root 4096 Jan  8  2018 mnt
 8 lalev@dragonfly:/$ ls -ld m???*
 9 drwxr-xr-x 4 root root 4096 Apr 17  2018 media
10 lalev@dragonfly:/$
```

В примера отново е използван непредставен до момента параметър, а именно ***-d***. Този параметър обикновено се използва когато към ***ls*** се подават относителни или абсолютни пътища. Ако тези пътища указват директория (или директории), нормално ***ls*** ще покаже съдържанието на тази директория. При използване на ***-d*** обаче се показва самият файл на тази директория.

На ред 1 на примера е използван и специалният символ на обвивката - ? (въпросителна). Специалните символи на обвивката могат да се комбинират с нормални символи, за да формират шаблони за търсене на файлови имена.

Така например ***m??*** е инструкция за търсене на имена на файлове, които започват с ***m*** и продължават с два произволни символа. Въпросителната на дадена позиция в шаблона указва, че символът на това място в реалното име на файла може да бъде всеки един възможен символ. Така файл със име ***mnt*** би отговорил на шаблона за търсене ***m??***, тъй като този шаблон изисква файлът да има име от три символа, първият от които ***m***, докато останалите два могат да бъдат всякакви символи. Файл със име ***media*** обаче не би отговорил на същия шаблон, защото е съставен от повече от три символа.

Когато обвивката срещне шаблон в абсолютен или относителен път, тя заменя шаблона със списък на реалните файлове, които му съответстват. Тоест командата ***ls -ld m??*** бива преработена от обвивката в командата ***ls -ld mnt*** още преди командата да бъде изпълнена, тъй като в главната директория има само един файл, който отговаря на шаблона ***m??***. Както е видно от резултата на ред 2, се оказва че този файл всъщност е директория. Ако опцията ***-d*** липсваше, в изхода на командата щяхме да видим съдържанието на самата директория ***mnt***.

На ред 3 е повторена същата команда като тази на ред 1, но този път с шаблон ***m?***. Тъй като в главната директория няма файл с име от два символа, първият от които ***m***, това води до съобщение за грешка.

На ред 5 е използван друг символ за създаване на шаблони – * (звезда), която отговаря на нула или повече произволни символа. Тоест, шаблонът на ред 5 търси файл, който започва с ***m*** и евентуално има други символи, без значение какви. Оказва се, че два файла отговарят на този шаблон. Това са директориите ***media*** и ***mnt*** и обвивката заменя шаблона с двата файла с тях още преди да изпълни командата, което дава ***ls -ld media mnt***.

Важно е да се подчертае, че използването на звезда позволява наличието на произволен брой символи на дадената позиция, но не изисква такива. Това се отличава от въпросителната на дадена позиция, която задължително изисква на символ. Шаблонът на ред 8 търси файлове, чиито имена започват с ***m*** и имат поне 4 символа.

3. Работа с файлове

3.1. Създаване на празни файлове

Едни от най-често изпълняваните операции, свързани с администрирането на системата, са копирането, преместването и изтриването на файлове. Ето защо усвояването им в детайли е една от първите задачи на всеки начинаещ Линукс системен администратор.

Преди да преминем към самото преместване, изтриване и копиране на файловете, е добре да посочим метод за бързо създаване на празни файлове, които да се използват за упражнения.

Такъв начин е командата ***touch***. Базовият синтаксис на тази команда е прост:

```
touch [параметри] файл/файлове
```

touch основно се използва за промяна на времето на модифициране на съдържанието, времето на модифициране на служебната информация и времето на последен достъп до даден файл или файлове. Изпълнението на ***touch*** без параметри върху съществуващ файл променя едновременно и трите времена и ги установява на текущата дата и време, взети към момента на изпълнение на командата.

Ако обаче указаният файл не съществува, ***touch*** ще го създаде, като при това ще остави съдържанието му празно. В следващите примери, ***touch*** се именно по този начин за създаване на примерни файлове, които евентуално ще се копират, преместват или изтриват.

```
 1 lalev@dragonfly:~$ mkdir example1
 2 lalev@dragonfly:~$ cd example1
 3 lalev@dragonfly:~/example1$ touch test1
 4 lalev@dragonfly:~/example1$ touch test2 test3
 5 lalev@dragonfly:~/example1$ ls -la
 6 total 8
 7 drwxrwxr-x  2 lalev lalev 4096 Dec 23 11:22 .
 8 drwxr-xr-x 46 lalev lalev 4096 Dec 23 11:21 ..
 9 -rw-rw-r--  1 lalev lalev    0 Dec 23 11:22 test1
10 -rw-rw-r--  1 lalev lalev    0 Dec 23 11:22 test2
11 -rw-rw-r--  1 lalev lalev    0 Dec 23 11:22 test3
12 lalev@dragonfly:~/example1$
```

В показания по-горе пример, потребителят първо създава празна директория, след което създава три файла в нея с помощта на ***touch***. На ред 3 е показано използването на ***touch*** за създаване на един файл, докато на ред 4 командата се използва за създаване на няколко (в случая два) файла.

3.2. Копиране на файлове

Копирането на файлове в Линукс се извършва от командата ***cp*** (съкр. от англ. „copy” - копиране). Базовият синтаксис на командата е прост, но трябва

да се има предвид, че командата има доста варианти и опции. Най-типичната форма на употреба е дадена по-долу:

```
cp [параметри] източник директория
```

При нея файл или списък файлове (обозначени по-горе като „източник“) се копират в дадена директория.

Долният пример показва конкретна употреба на командата ***cp*** по този начин:

```
 1 lalev@dragonfly:~/example1$ ls -la
 2 total 12
 3 drwxrwxr-x  3 lalev lalev 4096 Dec 23 23:09 .
 4 drwxr-xr-x 46 lalev lalev 4096 Dec 23 16:57 ..
 5 drwxrwxr-x  2 lalev lalev 4096 Dec 23 23:09 info
 6 -rw-rw-r--  1 lalev lalev    0 Dec 23 11:22 test1
 7 -rw-rw-r--  1 lalev lalev    0 Dec 23 11:22 test2
 8 -rw-rw-r--  1 lalev lalev    0 Dec 23 11:22 test3
 9 lalev@dragonfly:~/example1$ cp test1 info
10 lalev@dragonfly:~/example1$ cd info
11 lalev@dragonfly:~/example1/info$ ls -la
12 total 8
13 drwxrwxr-x 2 lalev lalev 4096 Dec 23 23:09 .
14 drwxrwxr-x 3 lalev lalev 4096 Dec 23 23:09 ..
15 -rw-rw-r-- 1 lalev lalev    0 Dec 23 23:09 test1
16 lalev@dragonfly:~/example1/info$
```

На ред 1 потребителят показва съдържанието на директория ***~/example1***. От изхода на командата (по-точно ред 5) е видно, че в директорията има поддиректория на име ***info***.

Същинското копиране се случва на ред 9. Командата ***cp*** на ред 9 взема списък от два аргумента. ***test1*** е името на файла, който ще се копира, а ***info*** е директорията, в която ще бъдат копирани файловете.

Много често при извършване на дейностите по системна администрация се налага копирането не на един, а на множество файлове. Извикването на ***cp*** отделно за всеки файл е непрактично, поради което в ***cp*** също е предвиден

механизъм за копирането на множество файлове. Когато ***cp*** получи повече от два аргумента, какъвто е случаят в долния пример, и последният от тях е директория, то тогава всички аргументи освен последния, се интерпретират като списък с файловете, които ще бъдат копирани.

```
 1 lalev@dragonfly:~/example2$ ls -la
 2 total 12
 3 drwxrwxr-x  3 lalev lalev 4096 Dec 26 10:45 .
 4 drwxr-xr-x 47 lalev lalev 4096 Dec 26 10:44 ..
 5 drwxrwxr-x  2 lalev lalev 4096 Dec 26 10:45 info2
 6 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:44 test1
 7 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test11
 8 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test12
 9 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test13
10 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:44 test2
11 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:44 test3
12 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:44 test4
13 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:44 test5
14 lalev@dragonfly:~/example2$ cp test1 test3 test5 info2
15 lalev@dragonfly:~/example2$ ls -la ./info2
16 total 8
17 drwxrwxr-x 2 lalev lalev 4096 Dec 26 10:45 .
18 drwxrwxr-x 3 lalev lalev 4096 Dec 26 10:45 ..
19 -rw-rw-r-- 1 lalev lalev    0 Dec 26 10:45 test1
20 -rw-rw-r-- 1 lalev lalev    0 Dec 26 10:45 test3
21 -rw-rw-r-- 1 lalev lalev    0 Dec 26 10:45 test5
22 lalev@dragonfly:~/example2$ cp test1? info2
23 lalev@dragonfly:~/example2$ ls -la ./info2
24 total 8
25 drwxrwxr-x 2 lalev lalev 4096 Dec 26 10:46 .
26 drwxrwxr-x 3 lalev lalev 4096 Dec 26 10:45 ..
27 -rw-rw-r-- 1 lalev lalev    0 Dec 26 10:45 test1
28 -rw-rw-r-- 1 lalev lalev    0 Dec 26 10:46 test11
29 -rw-rw-r-- 1 lalev lalev    0 Dec 26 10:46 test12
30 -rw-rw-r-- 1 lalev lalev    0 Dec 26 10:46 test13
31 -rw-rw-r-- 1 lalev lalev    0 Dec 26 10:45 test3
32 -rw-rw-r-- 1 lalev lalev    0 Dec 26 10:45 test5
```

На ред 14 потребителят подава списък от аргументи ***test1 test3 test5 info2***, последният аргумент е директорията, в която ще бъдат копирани файловете, докато ***test1***, ***test2*** и ***test3*** са имената на файловете, които ще бъдат копирани.

Директното задаване на имената на файловете е непрактично, когато става въпрос за копирането на всички файлове в една директория или на група от файлове с подобни имена. За щастие механизмите на обвивката, очертани в точка 2.3, работят и тук.

На ред 22 е демонстрирано копиране, при което потребителят задава файловете, които ще се копират, с помощта на шаблон. Шаблонът ***test1?*** се интерпретира от обвивката като инструкция за търсене на файлове с имена, които започват с „test1“ и задължително съдържат още един произволен символ. В директория ***example2*** три файла отговарят на това условие, а именно ***test11***, ***test12*** и ***test13***. Следователно обвивката ще трансформира командата на ред 22 в командата ***cp test11 test12 test13 example2***.

По подразбиране, командата ***cp*** презаписва файлове без да иска потвърждение. Тоест, ако в *директорията-цел*, т.е. директорията, в която ще бъдат копирани файловете, съществуват файлове със същите имена, като новокопираните файлове, то новокопираните файлове ще презапишат старите файлове без потребителя да бъде питан за потвърждение.

Ако това поведение е нежелателно, то към ***cp*** могат да бъдат подадени опционалните аргументи ***-n*** или ***-i***. Аргументът ***-n*** ще предотврати презаписването на файловете. Тоест, ако в директорията-цел имаме файлове със същите имена като новокопираните файлове, старите файлове ще останат непроменени, а тези от новокопираните файлове, чиито имена съвпадат, просто няма да бъдат копирани.

Ако ***cp*** вместо това е извикана с аргумент ***-i***, ***cp*** ще извежда запитване към потребителя за всяко едно съвпадение на имената. Потребителят ще има възможност да избере дали да копира новия файл върху стария или да запази стария. В следващия пример е демонстрирано именно действието на аргумент ***-i***.

```
 1 lalev@dragonfly:~/example2$ cp -i * info2
 2 cp: -r not specified; omitting directory 'info2'
 3 cp: overwrite 'info2/test1'? y
 4 cp: overwrite 'info2/test11'? y
 5 cp: overwrite 'info2/test12'? n
 6 cp: overwrite 'info2/test13'? t
 7 cp: overwrite 'info2/test2'? n
 8 cp: overwrite 'info2/test3'? n
 9 cp: overwrite 'info2/test4'? n
10 cp: overwrite 'info2/test5'? y
11 lalev@dragonfly:~/example2$
```

На ред 1 потребителят задава шаблон *, който се интерпретира от обвивката като инструкция за указване на всички файлове в текущата директория. Поради тази причина обвивката трансформира знака * в списък от имената на всички файлове в ***example2***. Този списък се състои от файловете ***test1***, ***test11***, ***test12***, ***test13***, ***test2***, ***test3***, ***test4***, ***test5***, както и от директорията ***info2***.

По подразбиране ***cp*** не копира съдържанието на цели директории. Поради това на ред 2 потребителят получава предупредително съобщение, което казва, че директорията ***info2*** и нейното съдържание ще бъдат изпуснати от копирането. Редове 3 до 10 представляват предупредителни съобщения за всеки файл, чието име съвпада с име да файл в директорията-цел ***info2***. Оказва се, че ***info2*** съдържа файлове със същите имена като тези в директория ***example2***, така че на практика потребителят получава съобщение за потвърждаване на презаписването на всеки един файл. Ако потребителят въведе „***y***“ **(**от англ. “yes” - “да”), новокопираният файл ще запише този, който вече съществува в ***info2***. Всички други отговори ще се интерпретират като отказ от презаписване.

Така например на ред 7 потребителят отговаря с „***n***“ и файлът ***test2*** в ***info2*** няма да бъде презаписан от ***test2*** в ***example2***. По-интересен е ред 6, на който потребителят (вероятно по грешка) вместо „***n***“ въвежда „***t***“. Този вход

също се интерпретира като отказ от презаписване и файлът ***test13*** в ***info2*** бива запазен.

Доста често се налага копиране не само на файловете в конкретна директория, но и цялата структура от поддиректории и тяхното съдържание. За тези случаи е предвиден аргументът ***-r***. При извикване на ***cp*** с този аргумент, ***cp*** не прескача директориите, а ги копира заедно с тяхното съдържание и поддиректории. По долния пример демонстрира такова поведение:

```
 1 lalev@dragonfly:~/example2$ cp -r info2 /tmp
 2 lalev@dragonfly:~/example2$ ls -la /tmp/info2/
 3 total 44
 4 drwxrwxr-x  2 lalev lalev  4096 Dec 26 18:46 .
 5 drwxrwxrwt 18 root  root  36864 Dec 26 18:46 ..
 6 -rw-rw-r--  1 lalev lalev     0 Dec 26 18:46 test1
 7 -rw-rw-r--  1 lalev lalev     0 Dec 26 18:46 test11
 8 -rw-rw-r--  1 lalev lalev     0 Dec 26 18:46 test12
 9 -rw-rw-r--  1 lalev lalev     0 Dec 26 18:46 test13
10 -rw-rw-r--  1 lalev lalev     0 Dec 26 18:46 test2
11 -rw-rw-r--  1 lalev lalev     0 Dec 26 18:46 test3
12 -rw-rw-r--  1 lalev lalev     0 Dec 26 18:46 test4
13 -rw-rw-r--  1 lalev lalev     0 Dec 26 18:46 test5
14 lalev@dragonfly:~/example2$ cp -r info2 /etc/
15 cp: cannot create directory '/etc/info2': Permission denied
16 lalev@dragonfly:~/example2$
```

На ред 1 потребителят копира директория ***info2*** в директория ***/tmp***. Ако потребителят не бе избрал опцията ***-r***, копирането нямаше да се осъществи тъй като директорията ***info2*** щеше да бъде прескочена. На ред 14 потребителят се опитва да направи подобна операция, но този път получава съобщение за грешка. Причината за грешката е системата за контрол на достъпа в Линукс, която определя, че потребител ***lalev*** няма права да създава нови файлове и директории в ***/etc***[8].

Системата за контрол на достъпа в Линукс ще бъде разгледана в следващата глава. Дотогава читателите следва да бъдат предупредени, че много от примерите, които те сами

[8] Съобщението “Permission denied” означава буквално „Отказано разрешение“.

биха съставили за да отработят използването на ***cp,*** могат да не сработят поради факта, че нямат достъп за писане до директорията, в която копират файловете.

По принцип потребителите имат пълни права за достъп до домашната си директория и нейните поддиректории, както и до специалната папка ***/tmp***.

Интересна и важна възможност на ***cp*** е преименуването на файл в процеса на копиране. Много често потребителите просто искат да направят копие на даден файл в същата директория преди да го променят. Така например администратор, който редактира даден файл с настройки ***settings.xml*** би могъл да иска да направи копие на файла в същата директория в случай, че неговият опит за промяна на настройките не сработи. Може би администраторът иска да нарече копието на този файл ***settings.xml.old*** или например ***settings.xml.original***.

За да се извърши подобно копиране, ***cp*** трябва да има само два аргумента - име на файл и ново име, под което да бъде копиран файла. Новото име на файла не трябва да съвпада с името на съществуваща директория.

cp *[параметри] файл_за_копиране ново_име_на_файл*

По-долният пример демонстрира такова копиране:

```
 1 lalev@dragonfly:~/example3$ ls -la
 2 total 8
 3 drwxrwxr-x  2 lalev lalev 4096 Dec 27 13:03 .
 4 drwxr-xr-x 47 lalev lalev 4096 Dec 27 13:02 ..
 5 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test11
 6 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test12
 7 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test13
 8 lalev@dragonfly:~/example3$ cp test13 test14
 9 lalev@dragonfly:~/example3$ ls -la
10 total 8
11 drwxrwxr-x  2 lalev lalev 4096 Dec 27 13:03 .
12 drwxr-xr-x 47 lalev lalev 4096 Dec 27 13:02 ..
13 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test11
14 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test12
15 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test13
16 -rw-rw-r--  1 lalev lalev    0 Dec 27 13:03 test14
```

```
17 lalev@dragonfly:~/example3$
```

3.3. Преместване на файлове

Преместването на файлове в Линукс се осъществява с командата ***mv***[9]. ***mv*** целенасочено работи с аналогичен на ***cp*** синтаксис, като основната разлика с гореизложените детайли за работата на ***cp*** се корени в това, че ***mv*** винаги работи рекурсивно.

Долният пример показва преместване на списък от файлове, преместване на списък от файлове, зададен с шаблон на обвивката, както и преименуване на файл.

```
 1 lalev@dragonfly:~/example3$ ls -la
 2 total 12
 3 drwxrwxr-x  3 lalev lalev 4096 Dec 27 13:38 .
 4 drwxr-xr-x 47 lalev lalev 4096 Dec 27 13:02 ..
 5 drwxrwxr-x  2 lalev lalev 4096 Dec 27 13:38 info3
 6 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test11
 7 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test12
 8 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test13
 9 -rw-rw-r--  1 lalev lalev    0 Dec 27 13:03 test14
10 lalev@dragonfly:~/example3$ mv test11 test12 info3
11 lalev@dragonfly:~/example3$ ls -la
12 total 12
13 drwxrwxr-x  3 lalev lalev 4096 Dec 27 13:39 .
14 drwxr-xr-x 47 lalev lalev 4096 Dec 27 13:02 ..
15 drwxrwxr-x  2 lalev lalev 4096 Dec 27 13:38 info3
16 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test13
17 -rw-rw-r--  1 lalev lalev    0 Dec 27 13:03 test14
18 lalev@dragonfly:~/example3$ mv test13 test3
19 lalev@dragonfly:~/example3$ ls -la
20 total 12
21 drwxrwxr-x  3 lalev lalev 4096 Dec 27 13:39 .
22 drwxr-xr-x 47 lalev lalev 4096 Dec 27 13:02 ..
23 drwxrwxr-x  2 lalev lalev 4096 Dec 27 13:38 info3
24 -rw-rw-r--  1 lalev lalev    0 Dec 27 13:03 test14
25 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test3
26 lalev@dragonfly:~/example3$ mv ./info3/test?? .
```

[9] ***mv*** е съкратено от англ. “move” - „премести“.

```
27 lalev@dragonfly:~/example3$ ls -la
28 total 12
29 drwxrwxr-x  3 lalev lalev 4096 Dec 27 13:39 .
30 drwxr-xr-x 47 lalev lalev 4096 Dec 27 13:02 ..
31 drwxrwxr-x  2 lalev lalev 4096 Dec 27 13:39 info3
32 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test11
33 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test12
34 -rw-rw-r--  1 lalev lalev    0 Dec 27 13:03 test14
35 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test3
36 lalev@dragonfly:~/example3$ ls -la info3
37 total 8
38 drwxrwxr-x 2 lalev lalev 4096 Dec 27 13:39 .
39 drwxrwxr-x 3 lalev lalev 4096 Dec 27 13:39 ..
40 lalev@dragonfly:~/example3$
```

Командата на ред 10 премества файловете ***test11*** и ***test12*** в папка ***info3***. Особено интересен обаче е ред 18. Линукс няма специална команда за преименуване на файл, като вместо това за целта се използва ***mv***. Командата на ред 18 премества файла ***test13*** в същата директория, но под ново име - ***test3***. На практика това представлява преименуване на файла и показаната употреба на ***mv*** е стандартният начин, по който се извършва тази операция в Линукс.

На ред 26 потребителят премества файловете, които отговарят на шаблон ***test??*** от ***info3*** в текущата директория. На предните стъпки в ***info3*** бяха преместени файловете ***test11*** и ***test12***, така че в ***info3*** има поне два файла, който отговарят на шаблона. Те, както и евентуално други файлове, чиито имена започват с “test” и завършват с два произволни символа.

Както може да се види от изхода на командата ls на редове 28-35, само файловете ***test11*** и ***test12*** са преместени от ***info3***.

Както бе споменато, ***mv*** има много сходен синтаксис със ***cp***. Това сходство се простира и до поведението по отношение на презаписването на файлове. Ако даден файл в директорията - цел има същото име като някой от копираните файлове, то по подразбиране, той ще бъде презаписан. Ако обаче ***mv*** е извикан с аргумент ***-n***, командата ще „обърне“ поведението си и няма да

премества файлове, чиито имена съвпадат с имена на файлове в директорията - цел.

Аргументът ***-i***, също като при ***cp***, води до това, че ***mv*** ще пита потребителя как да постъпи при всяко едно съвпадение на имената.

3.4. Изтриване на файлове

Изтриването на файлове в Линукс се извършва с командата ***rm***[10]. ***rm*** се използва както за изтриване на файлове, така и за изтриване на директории (тъй като директорията е просто специален файл в Линукс).

```
rm [параметри] списък_с_файлове
```

Списъкът с файлове може да се състои само от един файл или от множество файлове, зададени с поименно или с помощта на шаблони на обвивката.

Долният пример демонстрира подобна употреба:

```
 1 lalev@dragonfly:~/example3$ ls -la
 2 total 12
 3 drwxrwxr-x  3 lalev lalev 4096 Dec 27 18:02 .
 4 drwxr-xr-x 47 lalev lalev 4096 Dec 27 13:02 ..
 5 drwxrwxr-x  2 lalev lalev 4096 Dec 27 17:40 info3
 6 -rw-rw-r--  1 lalev lalev    0 Dec 27 17:40 test11
 7 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test12
 8 -rw-rw-r--  1 lalev lalev    0 Dec 27 13:03 test14
 9 -rw-rw-r--  1 lalev lalev    0 Dec 26 10:45 test3
10 -rw-rw-r--  1 lalev lalev    0 Dec 27 18:02 test4
11 -rw-rw-r--  1 lalev lalev    0 Dec 27 18:02 test511
12 lalev@dragonfly:~/example3$ rm test11
13 lalev@dragonfly:~/example3$ rm test12 test3
14 lalev@dragonfly:~/example3$ ls -la
15 total 12
16 drwxrwxr-x  3 lalev lalev 4096 Dec 27 18:02 .
```

[10] ***rm*** е съкратено от английското “remove” или “премахни”.

```
17 drwxr-xr-x 47 lalev lalev 4096 Dec 27 13:02 ..
18 drwxrwxr-x  2 lalev lalev 4096 Dec 27 17:40 info3
19 -rw-rw-r--  1 lalev lalev    0 Dec 27 13:03 test14
20 -rw-rw-r--  1 lalev lalev    0 Dec 27 18:02 test4
21 -rw-rw-r--  1 lalev lalev    0 Dec 27 18:02 test511
22 lalev@dragonfly:~/example3$ rm info3
23 rm: cannot remove 'info3': Is a directory
24 lalev@dragonfly:~/example3$ rm *
25 rm: cannot remove 'info3': Is a directory
26 lalev@dragonfly:~/example3$ ls -la
27 total 12
28 drwxrwxr-x  3 lalev lalev 4096 Dec 27 18:02 .
29 drwxr-xr-x 47 lalev lalev 4096 Dec 27 13:02 ..
30 drwxrwxr-x  2 lalev lalev 4096 Dec 27 17:40 info3
31 lalev@dragonfly:~/example3$
```

На ред 12 ***rm*** е използван за изтриване на един файл, докато на следващия ред, ***rm*** е използван за изтриване на два файла.

На ред 22 е направен опит за изтриване на директория с ***rm***. По подразбиране обаче, ***rm*** изпуска директориите и извежда съобщение като това на ред 23, което уведомява потребителя, че директория ***info3*** няма да бъде изтрита.

На ред 24, ***rm*** е извикана с шаблон *, който се трансформира от обвивката в списък на всички файлове в директорията. Тоест, това е инструкция за изтриване на всички файлове в ***example3***. ***rm*** извършва изтриването като отново прескача директорията ***info3*** като уведомява потребителя за това. Както можем да се убедим от изхода на последната команда ***ls***, след последователността от операции в директорията остава само един файл, а именно поддиректорията ***info3***.

Доста често в практиката на системната администрация се налага да се изтриват сложни йерархии от директории с техните поддиректории. ***rmdir*** ще откаже да изтрие поддиректории, които съдържат файлове или други поддиректории, а влизането във всяка една поддиректория и изтриването на нейното съдържание е много трудоемко.

За да автоматизира този процес по обхождане на множество поддиректории и изтриване на съдържанието им, в ***rm*** е предвиден аргументът ***-r***. Аргументът работи подобно на ***-r*** при ***cp*** и кара ***rm*** да обходи всяка една поддиректория. За разлика от ***cp***, която копира файлове, при обхождането ***rm*** ще изтрие файловете в поддиректорията преди да изтрие и самата поддиректория. С други думи, ако ***rm -r*** се извика върху поддиректория, това ще предизвика изтриване на цялата поддиректория и всички поддиректории и файлове в нея.

Извикването на ***rm -r*** е изключително опасно, тъй като извикана с погрешни параметри, тази команда може да изтрие голям брой файлове наведнъж. Това важи с още по-голяма сила за случаите когато изтриваните файлове се задават с шаблони на обвивката.

Когато ***rm*** се извиква с аргумент ***-r*** понякога е полезно да се добавят още аргументи, които указват дали ***rm*** трябва да иска потвърждение при изтриване на файлове. Поведението по подразбиране на ***rm*** е такова потвърждение да не се изисква, като това важи дори когато се използва аргумент ***-r***. Ако потребителят обаче зададе едновременно и аргумент ***-i***, ***rm*** ще извежда запитвания дали да бъде изтрит всеки един файл.

Примери за използването на ***rm*** с аргумент ***-r*** са дадени по-долу:

```
 1 lalev@dragonfly:~$ ls -laR example3
 2 example3/:
 3 total 16
 4 drwxrwxr-x  3 lalev lalev 4096 Dec 27 21:31 .
 5 drwxr-xr-x 48 lalev lalev 4096 Dec 27 21:30 ..
 6 drwxrwxr-x  2 lalev lalev 4096 Dec 27 21:30 info3
 7 -rw-rw-r--  1 lalev lalev    0 Dec 27 21:30 test1
 8 -rw-rw-r--  1 lalev lalev    6 Dec 27 21:30 test2
 9
10 example3/info3:
11 total 8
12 drwxrwxr-x 2 lalev lalev 4096 Dec 27 21:30 .
13 drwxrwxr-x 3 lalev lalev 4096 Dec 27 21:31 ..
```

```
14 lalev@dragonfly:~$ rm -ri example3
15 rm: descend into directory 'example3'? y
16 rm: remove regular file 'example3/test2'? n
17 rm: remove directory 'example3/info3'? y
18 rm: remove regular empty file 'example3/test1'? n
19 rm: remove directory 'example3'? y
20 rm: cannot remove 'example3': Directory not empty
21 lalev@dragonfly:~$ ls -laR example1
22 example1/:
23 total 12
24 drwxrwxr-x  3 lalev lalev 4096 Dec 23 23:09 .
25 drwxr-xr-x 48 lalev lalev 4096 Dec 27 21:30 ..
26 drwxrwxr-x  2 lalev lalev 4096 Dec 23 23:09 subdirectory
27 -rw-rw-r--  1 lalev lalev    0 Dec 23 11:22 test1
28 -rw-rw-r--  1 lalev lalev    0 Dec 23 11:22 test2
29 -rw-rw-r--  1 lalev lalev    0 Dec 23 11:22 test3
30
31 example1/subdirectory:
32 total 8
33 drwxrwxr-x 2 lalev lalev 4096 Dec 23 23:09 .
34 drwxrwxr-x 3 lalev lalev 4096 Dec 23 23:09 ..
35 -rw-rw-r-- 1 lalev lalev    0 Dec 23 23:09 test1
36 lalev@dragonfly:~$ rm -r example1
37 lalev@dragonfly:~$
```

Както показва изходът на командата ***ls*** от ред 1, папка ***example3*** съдържа два файла и подпапка. Извикването на ***rm*** с аргументи ***-ri*** (което *е* същото като изписването на двата аргумента поотделно, т.е. ***-r -i***) предизвиква извеждане на запитвания преди изтриването на всеки файл и навлизането във всяка поддиректория.

Съобщението ***rm: descend into directory 'example3'?*** (ред 15) пита потребителя дали ***rm*** да „влезе“ в поддиректория ***example3*** или да я изпусне (заедно със нейното съдържание) от изтриването. Потребителят отговаря утвърдително, поради което ***rm*** влиза в поддиректорията и започва да изтрива файлове. Съобщението на следващия ред - ***rm: remove regular file 'example3/test2'*** пита потребителя дали иска да изтрие първия от файловете в тази поддиректория. Потребителят отговаря отрицателно и файл ***test2*** бива изпуснат от изтриването.

Съобщението на ред 17 - ***rm: remove directory 'example3/info3'*** - пита потребителя дали директория ***info3*** трябва да бъде изтрита. За разлика от ***example3***, която съдържа файлове и други директории, ***info3*** в дадения момент не съдържа нищо, поради това ***rm*** не пита потребителя дали да влезе в директорията, а направо пита дали следва тя да бъде изтрита.

Съобщението на ред 18 пък показва, че ***rm*** различава празните файлове (т.е. тези с дължина 0 байта) от тези, които съдържат нещо. Читателят следва да сравни редове 16 и 18 и да забележи разликата в съобщенията. Тъй като файлът ***test1*** е празен, съобщението за неговото изтриване казва това на потребителя.

На ред 19 потребителят е запитан дали иска да изтрие самата директория ***example3***. Потребителят отговаря утвърдително, но тъй като тя съдържа един файл, който не е изтрит (потребителят е отказал изтриването му на ред 16, ***rm*** отказва да изтрие директорията, за което уведомява потребителя на ред 20.

По-нататъшната част от примера демонстрира и ситуация при която ***rm -r*** е извикана без аргумент ***-i***. Както е видно от изхода на командата ls на ред 21, директория ***example1*** съдържа други директории и множество файлове. Извикването на ***rm -r*** на ред 36 ги премахва наведнъж без каквито и да било запитвания към потребителя.

Изтриването на даден файл под Линукс не унищожава съдържанието му на диска. По-скоро изтриването отбелязва мястото, заемано от файла, като свободно. Евентуално това място бива презаписано от други файлове. Докато (и дори след като се случи това) данните от файла остават съхранени на диска и е възможно, макар и на цената на сериозни усилия, те да бъдат възстановени. Съществуват редица компании, които се специализират в извличането на изтрита или "повредена" (поради отказ на дисковия носител) информация. Понякога подобни дейности по възстановяване на изтрита информация се извършват от злонамерени лица. Друг път подобни дейности се извършват в рамките на разследването на престъпления.

Начинаещите администратори трябва да са наясно, че дори дадена информация да е презаписана, на магнитните дискове остават слаби „следи" от предното състояние на дисковия носител. Тези следи могат да се разчитат от лица, които по някакъв начин са се сдобили със достъп до самите твърди дискове. По същия начин администраторите трябва да

са наясно, че дефектиралите дискове могат да бъдат поправени, поради което не е особено мъдра идея те да се изхвърлят директно в боклука, особено ако са съдържали потенциално чувствителна информация.

Ако даден файл трябва да бъде изтрит по начин, който би предотвратил възстановяването му по-късно, Линукс предлага командата ***shred***. ***shred*** не само изтрива файла, но презаписва неговото съдържание на диска многократно за да направи невъзможно възстановяването му по-късно.

shred също има своите ограничения, които произтичат от тънките детайли на дизайна на файловите системи на Линукс. Ето защо най-чувствителната информация може да се съхранява на диска направо в шифрирано състояние. Линукс поддържа няколко механизъма за дисково шифриране, които обаче едва ли следва да бъдат разглеждани в подобно встъпително изложение.

4. Твърди и символични връзки към файлове

Както бе споменато по-горе, йерархичната структура на директориите има много предимства. Единственият недостатък на директорийната структура е това, че един и същ файл не може да фигурира в различни директории. От време на време обаче това е желателно по най-разнообразни причини. Ето защо еволюцията на повечето операционни системи е довела до възникването на механизми за решаването на този проблем. В Линукс тези механизми са два - така наречените „твърди“ и „символични“ връзки.

Твърдата връзка позволява един и същ файл да фигурира под различни имена в различни директории на файловата система. Символичните връзки по-скоро приличат на хипервръзките в Уеб - те са файлове, чието единствено предназначение е да сочат към други файлове.

4.1. Твърди връзки

Създаването и на двата вида връзки става с командата ***ln***[11]. Синтаксисът на ***ln*** има няколко варианта. Когато се създава една единствена връзка, е подходящ следния синтаксис:

```
ln [-T] цел  източник
```

Цел е файлът, към който сочи връзката. *Източник* от друга страна е името на новосъздадената връзка. В по-долния пример е демонстрирано използването на ***ln*** по този начин за създаването на твърди връзки.

```
 1 lalev@dragonfly:~/example4$ touch test1
 2 lalev@dragonfly:~/example4$ ls -la
 3 total 8
 4 drwxrwxr-x  2 lalev lalev 4096 Dec 28 19:30 .
 5 drwxr-xr-x 48 lalev lalev 4096 Dec 28 17:27 ..
 6 -rw-rw-r--  1 lalev lalev    0 Dec 28 19:30 test1
 7 lalev@dragonfly:~/example4$ ln test1 test2
 8 lalev@dragonfly:~/example4$ ls -la
 9 total 8
10 drwxrwxr-x  2 lalev lalev 4096 Dec 28 19:30 .
11 drwxr-xr-x 48 lalev lalev 4096 Dec 28 17:27 ..
12 -rw-rw-r--  2 lalev lalev    0 Dec 28 19:30 test1
13 -rw-rw-r--  2 lalev lalev    0 Dec 28 19:30 test2
14 lalev@dragonfly:~/example4$ echo "Here comes some content">test1
15 lalev@dragonfly:~/example4$ ls -la
16 total 16
17 drwxrwxr-x  2 lalev lalev 4096 Dec 28 19:30 .
18 drwxr-xr-x 48 lalev lalev 4096 Dec 28 17:27 ..
19 -rw-rw-r--  2 lalev lalev   24 Dec 28 19:31 test1
20 -rw-rw-r--  2 lalev lalev   24 Dec 28 19:31 test2
21 lalev@dragonfly:~/example4$ cat test1
22 Here comes some content
23 lalev@dragonfly:~/example4$ cat test2
24 Here comes some content
25 lalev@dragonfly:~/example4$ ls -lai
26 total 16
```

[11] Съкратено от англ. „link” - “връзка”.

```
27 213030 drwxrwxr-x  2 lalev lalev 4096 Dec 28 19:30 .
28 131073 drwxr-xr-x 48 lalev lalev 4096 Dec 28 17:27 ..
29 134382 -rw-rw-r--  2 lalev lalev   24 Dec 28 19:31 test1
30 134382 -rw-rw-r--  2 lalev lalev   24 Dec 28 19:31 test2
31 lalev@dragonfly:~/example4$ rm test1
32 lalev@dragonfly:~/example4$ ls -lai
33 total 12
34 213030 drwxrwxr-x  2 lalev lalev 4096 Dec 28 19:40 .
35 131073 drwxr-xr-x 48 lalev lalev 4096 Dec 28 17:27 ..
36 134382 -rw-rw-r--  1 lalev lalev   24 Dec 28 19:31 test2
37 lalev@dragonfly:~/example4$ cat test2
38 Here comes some content
39 lalev@dragonfly:~/example4$
```

След като потребителят е създал файл ***test1*** в ***example4***, на ред 7 потребителят извиква командата ***ln test1 test2***. В случая ***test1*** е името на съществуващия файл, към който ще сочи връзката, докато ***test2*** е новото име. Веднъж създадена, твърдата връзка се държи във всяко едно отношение като алтернативно име на файла, което дори може да бъде разположено в друга директория.

Нещо повече, ***test1*** и ***test2*** всъщност са две имена, които сочат към един и същи ***inode***, т.е. един и същи файл. Въпреки че в примера, ***test1*** и ***test2*** са в една директория, това не е задължително. ***test2*** може да бъде създаден в която и да е директория, която обаче се намира на същия дял като файла ***test1***.

Следващите команди демонстрират, че ***test1*** и ***test2*** са един и същ файл. Командата на ред 14 поставя в ***test1*** текста “Here comes some content”, така че той не е повече празен (точното действие на командата ще бъде изяснено в детайли по-нататък). На ред 21 потребителят извежда съдържанието на файла на конзолата. Интересната част от примера се намира на ред 23, където потребителят извежда съдържанието на файла ***test2***. Тъй като ***test1*** и ***test2*** са един и същ файл, потребителят вижда същото съдържание като това, показано за ***test1***.

На ред 25 е показан още един аргумент на командата ***ls***. Аргументът ***-i*** предизвиква извеждането на номера на ***inode*** структурата за всеки файл. Този номер се извежда в първата колона и потребителят може да види от нея (редове 29 и 30) че файловете ***test1*** и ***test2*** имат един и същи ***inode***, тоест са един и същи файл.

Примерът може да се използва за по-ясното демонстриране на още един детайл, свързан с ***ls*** и разгледан по-горе в главата. Втората колона на изхода на ***ls*** (третата, когато се извежда и номерът на ***inode***) съдържа броя на твърдите връзки към файла, тоест броя на имената, под които даден файл фигурира в различните директории. Читателят следва да сравни колона 2 на ред 6 (която съдържа броя на имената, под които фигурира файлът ***test1*** преди да бъде създадена твърдата връзка) с колона 3 на ред 29 (която съдържа същия брой след създаването на връзката).

Изтриването на ред 31 демонстрира интересен детайл, свързан с изтриването на файловете. Командата ***rm*** премахва зададените ѝ имена от директориите, но изтрива даден файл едва когато и последното име, което сочи към ***inode*** структурата на файла, бъде изтрито. В примера изтриването на ***test1*** премахва името от директорията, но тъй като още едно име на файл сочи към същия ***inode*** номер (134382), файлът всъщност не се изтрива. Това е демонстрирано на ред 37, където потребителят извежда съдържанието на ***test2***, което е точно това, което бе зададено за вече „изтрития“ файл ***test1*** на ред 14.

Ако ***ln*** се извика само с един аргумент - файл, който се намира в друга директория - ***ln*** ще създаде твърда връзка към този файл в текущата директория, като името на твърдата връзка ще бъде същото като името на файла - цел.

```
1  lalev@dragonfly:~/example6$ ls -l
```

```
 2 total 0
 3 lalev@dragonfly:~/example6$ ln ../otherdir/somefile1
 4 lalev@dragonfly:~/example6$ ls -l
 5 total 0
 6 -rw-rw-r-- 2 lalev lalev 0 Jan  5 16:02 somefile1
 7 lalev@dragonfly:~/example6$
```

И двата описани до момента начина позволяват създаването на една-единствена връзка с всяко извикване на командата. Това не е удобно когато трябва да се създадат множество връзки. Ето защо ***ln*** поддържа друга форма на подаване на аргументите.

Ако бъде подаден списък с аргументи, последният от които директория, ***ln*** ще създаде в директорията твърди връзки към всеки от аргументите.

```
ln цел_1 цел_2 ... цел_n  директория
```

Възможен е и друг синтаксис, при който директорията идва на първо място:

```
ln -t директория цел_1 цел_2 ... цел_n
```

Следващият пример демонстрира използването на командата ***ln*** по описаните начини за създаване на твърди връзки към множество файлове наведнъж:

```
 1 lalev@dragonfly:~/example6$ ls -l
 2 total 0
 3 -rw-rw-r-- 2 lalev lalev 0 Jan  5 16:02 somefile1
 4 lalev@dragonfly:~/example6$ ls -l ../otherdir/
 5 total 0
 6 -rw-rw-r-- 2 lalev lalev 0 Jan  5 16:02 somefile1
 7 -rw-rw-r-- 2 lalev lalev 0 Jan  5 16:02 somefile2
 8 -rw-rw-r-- 2 lalev lalev 0 Jan  5 16:02 somefile3
 9 lalev@dragonfly:~/example6$ ln -t . ../otherdir/*
10 ln: failed to create hard link './somefile1': File exists
11 rlalev@dragonfly:~/example6$ ls -l
12 total 0
```

```
13 -rw-rw-r-- 2 lalev lalev 0 Jan  5 16:02 somefile1
14 -rw-rw-r-- 2 lalev lalev 0 Jan  5 16:02 somefile2
15 -rw-rw-r-- 2 lalev lalev 0 Jan  5 16:02 somefile3
16 lalev@dragonfly:~/example6$ rm *
17 lalev@dragonfly:~/example6$ ln ../otherdir/* .
18 lalev@dragonfly:~/example6$ ls -l
19 total 0
20 -rw-rw-r-- 2 lalev lalev 0 Jan  5 16:02 somefile1
21 -rw-rw-r-- 2 lalev lalev 0 Jan  5 16:02 somefile2
22 -rw-rw-r-- 2 lalev lalev 0 Jan  5 16:02 somefile3
23 lalev@dragonfly:~/example6$ cd ..
24 lalev@dragonfly:~$
```

Както е видно от първите редове на примера, в директория ***example6*** вече има файл с име ***somefile1***. Командата на ред 9 (***ls -l ../otherdir/****) се интерпретира по следния начин:

Аргументът ***-t***, указва че първият от следващите аргументи е директорията, в която ще се създават твърдите връзки. В случая този първи аргумент е точка, тоест връзките ще бъдат създавани в текущата директория ***example6***. Останалите елементи от списъка, зададени с шаблона на обвивката *, са трите файла в директория ***otherdir***. Те са ***somefile1***, ***somefile2*** и ***somefile3***. Тоест, командата от ред 9 ще създаде нови твърди връзки със същите имена в текущата директория.

В текущата директория обаче вече има файл на име ***somefile1***, поради което създаването на първата твърда връзка е неуспешно. Потребителят е уведомен за това на ред 10. Останалите връзки се създават успешно, както може да се види от изхода на командата ***ls*** на ред 11.

След като на ред 16 всички файлове (включително и твърдите връзки) са премахнати, на ред 17 е демонстриран другият вариант, при който директорията се поставя на последно място.

Синтаксисът, демонстриран на ред 17, прави ***ln*** много подобен на командите за копиране и преместване на файловете, но създава известно

двусмислие. Ако потребителят например издаде командата ***ln test1 test2*** имайки предвид да създаде нова твърда връзка с име ***test2***, може да се случи така, че случайно файл с име ***test2*** вече съществува и той е директория. В такъв случай връзката ще бъде създадена в тази директория. За да предотврати подобни грешки, потребителят може да добави аргумента ***-T***, който указва, че последният аргумент трябва да се интерпретира като стриктно не като директория, а като име на новата връзка.

Долният пример показва разликата между извикването на ***ls*** с аргумент ***-T*** и без него:

```
1 lalev@dragonfly:~$ ln -T otherdir/somefile1 example6
2 ln: failed to create hard link 'example6': File exists
3 lalev@dragonfly:~$ rm example6/*
4 lalev@dragonfly:~$ ln otherdir/somefile1 example6
5 lalev@dragonfly:~$ ls -l example6
6 total 0
7 -rw-rw-r-- 2 lalev lalev 0 Jan  5 16:02 somefile1
8 lalev@dragonfly:~$
```

На ред 1 потребителят иска да създаде твърда връзка към файла ***otherdir/somefile1*** в текущата директория. Името на новата връзка ще е ***example6***. В текущата директория обаче съществува поддиректория със същото име. Както е видно, аргументът ***-T*** показва на ***ls***, че потребителят не е имал предвид да постави нова връзка в тази директория. Поради това командата приключва с грешка. Ако аргументът ***-T*** липсва, същата команда (ред 4) ще създаде връзка към ***otherdir/somefile1*** вътре в папка ***example6*** (редове 3-8 показват точно това).

Описаната току-що грешка изглежда невинна и не прилича на такава, която често се получава при изпълнение на команди в интерактивен режим. Все пак администраторът ще провери дали има такава директория, преди да направи твърда връзка със същото име, а дори и да не провери, лесно може да коригира грешката си по-късно.

Лошото в тази грешка излиза наяве когато става въпрос за изпълнение на същите команди от команден /пакетен/ файл на обвивката или т.нар. „шел скрипт“. Шел скриптовете са разгледани в следващите глави и са основен начин за автоматизация на процесите по системна администрация. Те се пишат предварително от администратора и съдържат списък

от команди за изпълнение в дадена последователност. Много често такива файлове се изпълняват автоматично от системата в уречено време и много често те работят с привилегиите на администратора.

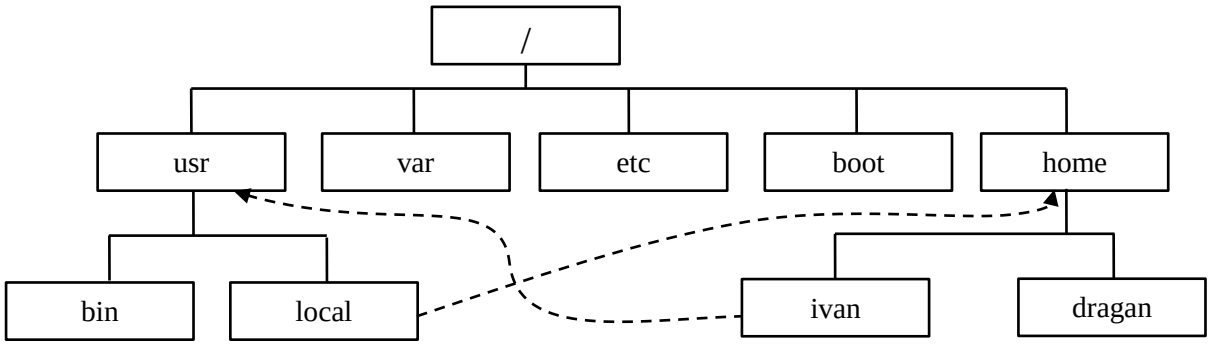
Даден шел скрипт може да провери дали директория със същото име съществува и веднага след като разбере, че такова няма, той може да опита да създаде връзка. Злонамерени лица с дори малък опит като хакери обаче могат да направят програма, която да вмъкне директорията точно след като командния файл провери, че тя не съществува и преди той да извика командата ***ln***.

В такъв случай командният файл ще бъде подведен да направи нещо различно от очакваното, вероятно с привилегиите на администратора. Опцията ***-T*** предотвратява това.

Тук е моментът да се спомене, че опция ***-T*** с абсолютно същото действие и предназначение има и при командите ***cp*** и ***mv***.

Използването на твърди връзки има някои ограничения. Тъй като те сочат към даден ***inode***, а ***inode*** номерата са уникални само в рамките на един дисков дял, твърдите връзки не могат да сочат към файл на друг дисков дял. Това е по-голям проблем, отколкото изглежда на пръв поглед, тъй като Линукс системите по стандарт изискват файловата система да е разделена на множество отделни дялове. Така например изпълнимите програми, които рядко се променят, се намират на отделен дял, който се монтира по стандарт в папката ***/usr***. Променливата информация също се намира на отделен дял, който се монтира в папка ***/var***. При това положение няма как във ***/var*** да се направи твърда връзка към файл, който се намира в поддиректория на ***/usr***.

Твърдите връзки имат и друго ограничение - те не могат да сочат към директории. Директориите в Linux, както многократно бе споменавано до момента, са просто файлове със специална информация, и би следвало към тях да може да се правят твърди връзки. По този начин една цяла поддиректория би могла да се намира на две места във файловата система. Оригиналните UNIX системи са имали подобна възможност. Оказало се обаче, че тази възможност е опасна, тъй като ако не се използва правилно, може да доведе до своеобразно „зацикляне“.



Фигура 2.5. Пример за цикъл във файловата система, който можеше да се получи, ако твърдите връзки към директориите бяха позволени.

На фиг. 2.5 е направен опит за илюстриране на един такъв хипотетичен цикъл във файловата система. Пунктираните стрелки показват, че директория ***/usr/local*** е твърда връзка към ***/home.*** По същия начин ***/home/ivan*** е твърда връзка към ***/usr*** (приемаме, че тази хипотетична система не е изградена по стандартите и всички показани директории се намират на един дял, поради което създаването на твърди връзки е възможно).

Така изградената йерархия от директории има цикъл, който ще доведе до това, че рекурсивното обхождане на някои директории би продължило до безкрайност. Ако потребителят изпълни например командата ***cp -r /home /backup***, това ще предизвика подобно рекурсивно обхождане. От директория ***/home, cp*** ще влезе в поддиректория ***ivan,*** която всъщност е директорията ***/usr***. От ***/usr***, евентуално ***cp*** ще влезе в поддиректория ***local***, която всъщност е директория ***/home***. Обхождането на ***/home*** ще доведе отново до влизане в ***ivan*** и така рекурсията ще продължи до безкрайност.

Горният проблем е причината, поради която повечето системи забраняват създаването на твърди връзки към директории както на обикновени потребители, така и на администратори. Някои по-стари системи забраняват подобно действие само на обикновените потребители с презумпцията, че

администраторите са по-наясно и могат да избегнат подобни нежелани ситуации. При такива системи, аргументът ***-F*** или еквивалентният му ***-d*** позволяват създаването на твърда връзка към директория.

По-долният пример показва подобен опит на Ubuntu 18.04:

```
 1 lalev@dragonfly:~$ sudo bash
 2 [sudo] password for lalev:
 3 root@dragonfly:~# mkdir example7
 4 root@dragonfly:~# cd example7
 5 root@dragonfly:~/example7# pwd
 6 /home/lalev/example7
 7 root@dragonfly:~/example7# mkdir test1
 8 root@dragonfly:~/example7# ln -d test1 test2
 9 ln: failed to create hard link 'test2' => 'test1': Operation not
   permitted
10 root@dragonfly:~/example7# cd ..
11 root@dragonfly:~# rm -rf example7
12 root@dragonfly:~# exit
13 lalev@dragonfly:~$
```

На ред 1 потребителят изпълнява командата ***sudo bash*** за да стартира обвивка с правата на администратор. Командата е успешна и редове 3-13 се изпълняват с привилегиите на администратора, което може да се види от знака **#** (диез). в промпта на обвивката.

При тази последователност, текущата директория не се променя и редове 3-6 показват, че новосъздадената директория ***example7*** е поддиректория на ***/home/lalev***. Тъй като поддиректорията е създадена от администратора, той е неин собственик. По същия начин администраторът е собственик и на поддиректорията ***test1***. На ред 8 е съществената част от експеримента, при която администраторът инструктира системата да направи твърда връзка към директория ***test1***. На ред 9 ***ln*** отказва да направи такава връзка със съобщение за грешка. С останалите редове потребителя премахва директорията, използвана за експеримента и прекратява работата като администратор, излизайки от обвивката.

Опитът демонстрира, че под Ubuntu 18.04 не е възможно създаването на твърда връзка към директория.

Твърдите връзки имат още едно ограничение. Повечето модерни системи не позволяват създаването на твърди връзки към файлове, които не са собственост на текущия потребител освен ако текущият потребител няма права за четене и запис върху тях. Причината за това е, че позволяването на подобна функционалност може да доведе до хакерски атаки. Най-елементарната от тях е предотвратяване на изтриването на даден файл от собственика му. Реалистичен сценарий за подобна атака би било наличието в системата на програма със ***setuid*** флаг (този флаг ще бъде разгледан в следващата глава), която има пробив в сигурността. При това положение администраторът или обновяващата програма може да избере да я изтрие. Това няма да се случи ако даден потребител е направил твърда връзка към нея. Разрешаването на твърди връзки към чужди файлове позволява и по-изтънчени атаки, някои от които са разгледани в упражненията към тази глава.

4.2. Символични връзки

Символичните връзки са сравнително по-модерна възможност на Linux и UNIX системите и са създадени като опит да се премахнат ограниченията на твърдите връзки. Символичните връзки не сочат към конкретен ***inode***, а към път и име на файл във файловата система. Това им дава свободата да сочат към файлове на различни дискови дялове във файловата система. Символичните връзки са различен файл с различен ***inode*** от тяхната цел, поради което те могат да сочат към директории без да създават цикли при копиране или други подобни операции.

Символичните връзки имат само един недостатък и той е че те не са свързани толкова тясно с файла, към който сочат. Възможно е този файл да изчезне при което връзката ще сочи към несъществуващ файл. Това обаче позволява символични връзки да се правят дори към чужди файлове, без да се отварят такива широки възможности за хакерски атаки като в преди демонстрирания случай с твърдите връзки.

Създаването на символични връзки става по абсолютно същия начин като при твърдите връзки с тази разлика, че към командата ***ln*** трябва да бъде добавен допълнителен аргумент ***-s***. Примерът по-долу демонстрира създаването на символични връзки по няколко различни начина:

```
 1 lalev@dragonfly:~/example8$ ls
 2 lalev@dragonfly:~/example8$ ls -la
 3 total 8
 4 drwxrwxr-x  2 lalev lalev 4096 Jan  6 11:59 .
 5 drwxr-xr-x 53 lalev lalev 4096 Jan  6 11:59 ..
 6 lalev@dragonfly:~/example8$ ln -s ../otherdir/somefile? .
 7 lalev@dragonfly:~/example8$ ls -la
 8 total 8
 9 drwxrwxr-x  2 lalev lalev 4096 Jan  6 12:00 .
10 drwxr-xr-x 53 lalev lalev 4096 Jan  6 11:59 ..
11 lrwxrwxrwx  1 lalev lalev   21 Jan  6 12:00 somefile1 ->
   ../otherdir/somefile1
12 lrwxrwxrwx  1 lalev lalev   21 Jan  6 12:00 somefile2 ->
   ../otherdir/somefile2
13 lrwxrwxrwx  1 lalev lalev   21 Jan  6 12:00 somefile3 ->
   ../otherdir/somefile3
14 lalev@dragonfly:~/example8$ rm *
15 lalev@dragonfly:~/example8$ ln -s -t . ../otherdir/somefile1
   /etc/passwd /etc/resolv.conf
16 lalev@dragonfly:~/example8$ ls -la
17 total 8
18 drwxrwxr-x  2 lalev lalev 4096 Jan  6 12:03 .
19 drwxr-xr-x 53 lalev lalev 4096 Jan  6 11:59 ..
20 lrwxrwxrwx  1 lalev lalev   11 Jan  6 12:03 passwd -> /etc/passwd
21 lrwxrwxrwx  1 lalev lalev   16 Jan  6 12:03 resolv.conf ->
   /etc/resolv.conf
22 lrwxrwxrwx  1 lalev lalev   21 Jan  6 12:03 somefile1 ->
   ../otherdir/somefile1
23 lalev@dragonfly:~/example8$ rm *
```

```
24 lalev@dragonfly:~/example8$ mkdir test2
25 lalev@dragonfly:~/example8$ ln -s -T ../otherdir/somefile1 test2
   ln: failed to create symbolic link 'test2': File exists
26 lalev@dragonfly:~/example8$ ln -s ../otherdir/somefile1 test2
27 lalev@dragonfly:~/example8$ ls -la test2/*
28 lrwxrwxrwx 1 lalev lalev 21 Jan 6 13:14 test2/somefile1 ->
   ../otherdir/somefile1
29 lalev@dragonfly:~/example8$
```

Командата на ред 6 създава символични връзки в текущата директория, които сочат към три файла в ***../otherdir***. Изходът на командата ***ls -la*** на ред 7 демонстрира как ***ls*** показва символичните връзки в листинга на директорията. Така например на ред 11 е изписано първо името на символичната връзка, след което след стрелката е изписан пътят и името на файла, към който връзката сочи. Читателят е приканен да забележи също, че типът на файла, е указан в първата колона, с '***l***' за всички символични връзки.

Командите на редове 15 и 25 показват, че също като при твърдите връзки, ***-t*** позволява директорията, в която се създават множество връзки да бъде поставена първа, а ***-T*** означава, че потребителят изрично има предвид второто име от списъка с две имена да бъде име, под което ще се създаде новата връзка, а не директория, в която ще бъде създадена връзка със същото име като оригинала.

Важен факт, на който ще бъде обърнато внимание както тук, така и в главата, посветена на системата за контрол на достъпа до файловете в Линукс, е че разрешенията за достъп на символичните връзки не се използват, поради което те винаги са с разрешения ***rwxrwxrwx.***

Следващият пример демонстрира поведението на символичните връзки, когато те са използвани за указване на директории.

```
 1 lalev@dragonfly:~/example11$ ls -l
 2 total 8
 3 drwxrwxr-x 2 lalev lalev 4096 Jan 24 20:43 subdir1
 4 drwxrwxr-x 2 lalev lalev 4096 Jan 24 20:43 subdir2
 5 lalev@dragonfly:~/example11$ ln -s subdir2 subdir3
```

```
 6 lalev@dragonfly:~/example11$ ln -s subdir3 subdir4
 7 lalev@dragonfly:~/example11$ ln subdir4 subdir5
 8 lalev@dragonfly:~/example11$ ls -l
 9 total 8
10 drwxrwxr-x 2 lalev lalev 4096 Jan 24 20:43 subdir1
11 drwxrwxr-x 2 lalev lalev 4096 Jan 24 20:43 subdir2
12 lrwxrwxrwx 1 lalev lalev    7 Jan 24 20:44 subdir3 -> subdir2
13 lrwxrwxrwx 2 lalev lalev    7 Jan 24 20:44 subdir4 -> subdir3
14 lrwxrwxrwx 2 lalev lalev    7 Jan 24 20:44 subdir5 -> subdir3
15 lalev@dragonfly:~/example11/subdir2$ rm -rf subdir3
16 lalev@dragonfly:~/example11$ ls -l
17 total 8
18 drwxrwxr-x 2 lalev lalev 4096 Jan 24 20:43 subdir1
19 drwxrwxr-x 2 lalev lalev 4096 Jan 24 20:43 subdir2
20 lrwxrwxrwx 2 lalev lalev    7 Jan 24 20:44 subdir4 -> subdir3
21 lrwxrwxrwx 2 lalev lalev    7 Jan 24 20:44 subdir5 -> subdir3
22 lalev@dragonfly:~/example11$ cd subdir4
23 bash: cd: subdir4: No such file or directory
24 lalev@dragonfly:~/example11$
```

Командата ***ln -s subdir2 subdir3*** на ред 5 създава символична връзка към директория. Името на връзката е ***subdir3*** и тя сочи към поддиректорията ***subdir2***. Командата на ред 6 демонстрира, че е напълно възможно една символична връзка да сочи към друга символична връзка. Ако читателят влезе с командата ***cd*** в ***subdir4***, той всъщност ще се озове в ***subdir2***, тъй като ***subdir4*** е символична връзка, която сочи към ***subdir3***, която на свой ред сочи към ***subdir2***.

5. Търсене на файлове с Findutils

GNU Findutils е набор от няколко програми за търсене на файлове във файловата система на Линукс. Най-важните от тях са ***find*** и ***xargs***, които ще бъдат разгледани накратко тук.

find е програма с доста сложен синтаксис, който може да се опише много схематично и непълно по следния начин:

```
find [-H]  [-L]  [-P] списък_с_директории_начални_точки... [логически
израз]
```

Ако ***find*** получи за аргументи една или повече директории, той ще обходи рекурсивно всяка една от тях и ще изведе имената на файловете в нея. В терминологията на ***find*** директориите се наричат „начални точки“[12], а „рекурсивно“ обхождане означава, че ***find*** ще влезе по ред във всяка една поддиректория и нейните поддиректории, „обхождайки“ по този ред всички разклонения на дървото на директориите. Това е показано на долния пример, в който ***find*** е използван за извеждане на имената на всички файлове и поддиректории, които се намират в текущата директория ***/etc/mysql*** и нейните поддиректории:

```
 1 root@dragonfly:/etc/mysql# find .
 2 .
 3 ./debian.cnf
 4 ./mariadb.conf.d
 5 ./mariadb.conf.d/50-mysqld_safe.cnf
 6 ./mariadb.conf.d/tmp_backup_do_not_delete
 7 ./mariadb.conf.d/tmp_backup_do_not_delete/archive.tgz
 8 ./mariadb.conf.d/50-server.cnf
 9 ./mariadb.conf.d/50-client.cnf
10 ./mariadb.conf.d/50-mysql-clients.cnf
11 ./conf.d
12 ./conf.d/mysql.cnf
13 ./conf.d/mysqldump.cnf
14 ./my.cnf
15 ./debian-start
16 ./mariadb.cnf
17 ./my.cnf.fallback
18 root@dragonfly:/etc/mysql# $
```

[12] Горното обяснение, макар и по-ясно, не е съвсем точно. find може да приеме всеки път до всякакъв файл за начална точка. Но ако файлът не е директория, find няма какво да обходи и просто извежда името му. В огромния брой от случаи „началните точки“ са директории, но въпреки това терминът е избран за да отразява факта, че началните точки не са задължително и винаги директории.

По подразбиране, ***find*** извежда имената и пътищата до файловете спрямо началните точки. За начална точка, зададена с относителен път, ***find*** извежда пътищата до обходените файлове като относителни пътища спрямо текущата директория[13]. В горния пример началната точка бе текущата директория, зададена с относителен път, поради което изведените пътищата до файловете са спрямо нея. Когато началните точки са зададени с абсолютен път, ***find*** извежда абсолютен път до всеки един файл.

Най-важната функционалност, предоставяна от ***find***, е възможността списъкът от обходените файлове да бъде ограничен по някакви критерии и програмата да изведе само файловете, които отговарят на тези критерии. С други думи ***find*** може да *търси* файлове по зададени критерии. За целта ***find*** позволява въвеждането като параметър на *логически израз*, който е съставен от набор логически свързани условия. Тези условия описват файловете, които трябва да бъдат търсени.

Логическите условия могат да бъдат групирани в четири категории - *тестове*, *действия, позиционни опции и глобални опции.* Тези условия могат да бъдат свързани помежду си с логически оператори “И“, „ИЛИ“ и „НЕ“.

Тестовете съдържат основната функционалност по търсенето и помагат за указване на желаните свойства на търсените файлове. *Действията* позволяват ***find*** не само да показва, но и да извършва манипулации с намерените файлове, което всъщност е втората най-важна функционална възможност на ***find***. *Позиционните опции* променят поведението на много от тестовете и действията, поставени *след тях* на командния ред, докато *глобалните опции* променят поведението както на тестовете и действията, поставени *както след тях, така и преди тях* на командния ред.

[13] Читателят е приканен да си припомни, че по дефиниция, относителните пътища представляват именно пътищата до файловете, зададени спрямо текущата в момента директория.

Някои по-важни тестове, поддържани от ***find***, са показани в табл. 2.8. На местата, където тестовете изискват подаване на число ***n***, числото може да бъде подадено с префикс + (плюс), префикс - (минус) или без префикс.

Обикновено ***n***, подаден без префикс, означава че се търсят файлове с дадена цифрова характеристика (например големина), която е точно ***n***. Префиксите + или **-** обозначават търсене на файлове с някаква числова характеристика, която е съответно *стриктно* повече от ***n*** или обратното - *стриктно* по-малко от ***n***. Шаблоните на обвивката, регулярните изрази и други части, които съдържат специални символи на обвивката, трябва да бъдат заградени в единични прави кавички (‘) за да предотвратят обвивката да ги изпълни и превърне в списъци на файлове от текущата директория.

-name *шаблон_на_обвивката*
Файловете трябва да имат имена, отговарящи на зададения шаблон на обвивката. Търсенето е само по имената на файловете и не включва пътищата до тях.
-iname *шаблон_на_обвивката*
Идентичен на ***-name*** с тази разлика, че сравненията се извършват без оглед на главни/малки букви. Докато ***-name File1??*** ще покаже файлове с имена ***File111*** и ***File112***, той няма да покаже ***file111*** и ***file112***, тъй като буквената част на имената не е същия регистър като зададения шаблон. За разлика от това ***-iname File???*** ще покаже и четирите файла.
-path *шаблон*
Идентичен с ***-name***, но този път търсенето е върху целия път до файла спрямо началната точка. Отново, допустимите символи са същите, като тези при шаблоните на обвивката.
-ipath *шаблон*
Идентичен с ***-path***, но този път сравненията се извършват без оглед на малки/главни букви.
-regex *регулярен_израз*
Търси файлове, които отговарят на подадения регулярен израз. Търсенето се извършва върху целия път до файла спрямо началната точка, а не само върху неговото име.
-iregex *регулярен_израз*
Идентичен с ***-regex***, но търсенето се извършва без оглед на малки/главни букви.

-size *n[cwbkMG]*
Файлът заема ***n*** единици пространство, като единиците могат да бъдат: c - байтове; w -двубайтови машинни думи (2 байта) b - блокове от 512 /байта k - килобайти M - мегабайти G - гигабайти Единиците пространство, заемани от всеки файл, се закръглят до следващото цяло число преди извършване на теста, т.е. ако даден файл заема 1.5GB, той ще удовлетвори теста ***-size 2G.*** Ако даден файл е 0.5 MB той няма да удовлетвори теста ***-size -1M***, тъй като тестът иска файлове, с размер по-малък от 1MB след закръгляването.
-type *mun*
Типът на файла е от указания тип. Типовете могат да бъдат следните: f - нормален файл l - файл - символична връзка d - директория s - съединение (сокет) p - FIFO специален файл b - специален файл на блоково устройство c - специален файл на символно устройство Търсенето на множество различни файлове под Линукс може да се извърши като типовете се разделят със запетая, тоест ***-type l,s,p,b,c*** ще търси за файлове, които са някой от изброените видове, но не са обикновени файлове или директории.
-amin *n*
Файлът е достъпван последно преди ***n*** минути.
-cmin *n*
inode структурата на файла е променяна посредно преди ***n*** минути.
-mmin *n*
Съдържанието на файла е променяно преди ***n*** минути.
-atime *n*
Съдържанието на файла е достъпвано последно преди ***n*** денонощия.
-ctime *n*
inode структурата на файла е променяна последно преди ***n*** денонощия.
-mtime *n*
Съдържанието на файла е модифицирано последно преди ***n*** денонощия.
-perm *разрешения*
Файлът има режим на достъп точно като описания от разрешенията
-perm *-разрешения*

Файлът има режим на достъп, който съдържа всички разрешения, като евентуално има и разрешения, които са извън описаните. С други думи режимът на достъп до файла е толкова или по-малко рестриктивен от описания.
-perm */разрешения*
Файлът има режим на достъп, който съдържа поне едно от описаните разрешения, като евентуално има и разрешения, които са извън описаните.
-readable
Файлове и директории, които могат да бъдат четени от текущия потребител. За разлика от ***--perm*** тестовете, този тест отчита не само разрешенията за достъп до файловете, но и допълнителни ограничения, като такива, наложени от списъци за контрол на достъпа.
-writable
Файлове и директории, в които текущия потребител може да осъществява запис. За разлика от ***--perm*** тестовете, този тест отчита не само разрешенията за достъп до файловете, но и допълнителни ограничения, като такива, наложени от списъци за контрол на достъпа.
-executable
Файлове и директории, за които текущия потребител има разрешение за изпълнение. За разлика от ***--perm*** тестовете, този тест отчита не само разрешенията за достъп до файловете, но и допълнителни ограничения, като такива, наложени от списъци за контрол на достъпа.
-uid *n*
Файлът е собственост на потребител с ***user id*** (***uid***) *n*.
-user *потребител*
Файлът е собственост на потребител *потребител*
-gid *n*
Файлът е собственост на група с ***group id*** (***gid***) *n*.
-group *група*
Файлът е собственост на група *група*.
-nouser
user id (***uid***) на файла не отговаря на никой от редовете на ***/etc/passwd***
-nogroup
group id (***gid***) на файла не отговаря на никой от редовете на ***/etc/group***
-links *n*
Към файла има *n* на брой твърди връзки
-samefile *файл*
Файлът сочи към същия ***inode*** като този на подадения като параметър на аргумента. С други думи файлът е твърда връзка към *файл*.
-inode *n*
Файлът сочи към ***inode*** с номер *n*.

-empty
Файлът е обикновен файл или директория и е празен, т.е. има големина 0.
-newer *файл*
Файлът е модифициран по-скоро от посочения като параметър *файл*.
-newerXY *файл*
Времето X на файла е по-ново от времето Y на посочения като параметър *файл*. X и Y могат да бъдат всяка една от следните букви: a - последно време на достъп до файла c - последно време на променяне на ***inode*** на файла m - време на модификация на файла
-newerXt *символен_низ_представляващ_дата_и_време*
X може да бъде всяка една от следните букви: a - последно време на достъп до файла c - последно време на променяне на ***inode*** на файла m - време на модификация на файла *символен_низ_представляващ_дата_и_време* може да бъде всеки валиден символен низ, който се поддържа от GNU date.

Таблица 2.8. Тестове на find

Долният пример показва използване на ***find*** с няколко различни теста:

```
 1 lalev@dragonfly:/sbin$ find /usr/bin -size +6M
 2 /usr/bin/lxc
 3 /usr/bin/mysql_embedded
 4 /usr/bin/snap
 5 /usr/bin/gdb
 6 /usr/bin/snapctl
 7 /usr/bin/mariabackup
 8 lalev@dragonfly:/sbin$ ls -la /usr/bin/lxc
 9 -rwxr-xr-x 1 root root 10779792 Nov 23 20:58 /usr/bin/lxc
10 lalev@dragonfly:/sbin$ find /etc/ -name 'resolv*'
11 /etc/dhcp/dhclient-enter-hooks.d/resolved
12 find: ‘/etc/ssl/private’: Permission denied
13 find: ‘/etc/polkit-1/localauthority’: Permission denied
14 find: ‘/etc/lvm/backup’: Permission denied
15 find: ‘/etc/inspircd’: Permission denied
16 /etc/resolvconf
17 find: ‘/etc/amanda’: Permission denied
18 /etc/systemd/resolved.conf
19 /etc/resolv.conf
20 find: ‘/etc/cups/ssl’: Permission denied
21 lalev@dragonfly:/sbin$ find /etc/ -name 'resolv*' 2>/dev/null
```

```
22 /etc/dhcp/dhclient-enter-hooks.d/resolved
23 /etc/resolvconf
24 /etc/systemd/resolved.conf
25 /etc/resolv.conf
26 lalev@dragonfly:/sbin$
```

На ред 1 в примера потребителят търси файлове с размер над 6MB, които се намират в ***/usr/bin*** или нейни поддиректории. В случая знакът + пред 6 означава, че файловете трябва да са стриктно над 6 MB след закръгляване на големината им до цяло число. Тоест най-малкият файл в получения списък има големина 7MB след закръгляването. На ред 8 потребителят проверява големината на един от върнатите от ***find*** файлове, който има размер около 10MB.

На ред 10 потребителят търси файлове в ***/etc*** и поддиректориите, отговарящи на шаблона ***resolv****, тоест файлове, чиито имена започват с ***resolv*** и продължават по произволен начин. Прави впечатление че много от редовете, върнати от ***find***, са грешки от тип „Permission denied“. Тези грешки са доста типични и се получават, когато потребителят, който извиква командата, няма права да влезе и изчете съдържанието на дадена поддиректория. На ред 21 е демонстрирано как пренасочването на изхода може да се използва за избягване извеждането на подобни грешки. Точното значение на текста ***2>/dev/null***, добавен след командата ***find***, ще бъде обяснено в следващите глави.

Трябва също да бъде обърнато внимание, че шаблонът на редове 10 и 21 е предназначен за ***find***, а не за обвивката. Ако този шаблон не беше заграден в единични кавички, обвивката щеше да го превърне в списък от файлове в текущата директория още преди ***find*** да бъде извикана. Това би изменило действието на ***find*** като най-вероятно ще предизвика грешка.

Както е видно от табл. 2.8, ***find*** разполага с многообразни тестове, но още по-мощна възможност е това, че тези тестове могат да бъдат комбинирани, за формулиране на сложно условие за търсене.

Всеки тест, извършен от ***find***, връща стойност TRUE (истина) или FALSE (лъжа). Така тестовете могат да бъдат свързани с помощта на логически оператори AND (логическо и), OR (логическо или) и NOT (логическо не) в едно сложно условие. Ако сложното условие се изчисли до истина, файлът ще бъде изведен (или евентуално - обработен). Ако обратното - изчислението на сложното условие даде резултат FALSE - файлът няма да бъде изведен (или обработен).

Има голяма вероятност читателят да е вече запознат с логическите стойности и логическите оператори, тъй като те се използват във всяка една област на информатиката. Все пак ще направим кратък преглед, който не е заместител на подробно запознаване с материята.

Логическите стойности са TRUE и FALSE - съответно „истина“ и „лъжа“. Последователност от логически стойности, съединени с *логическите оператори* AND, OR и NOT (както и други оператори, които обаче не са нужни на ***find***), формират т.нар. *логически израз*.

Логическите изрази в практиката на информатиката възникват като резултат от опит за формализация на някакво сложно условие, което компютърът трябва да провери. Такъв е и случаят с ***find***. ***find*** изпълнява указаните тестове и за всеки тест връща резултат TRUE или FALSE в зависимост от това дали текущият файл удовлетворява даденото условие. Но да се формулира по сложно условие като например „Файлът е изпълним и е с големина повече от 20MB” отделните тестове трябва да се свържат с логически съюзи. Логическите оператори играят ролята точно на такива съюзи.

Правилата, по които ***find*** проверява верността на сложното условие (или с други думи изчислява стойността на логическия израз) са доста прости:

а/ ако два теста са свързани с AND („логическо И”), резултатът от тяхното изпълнение е TRUE само ако и двата теста върнат TRUE. Всички други варианти за стойности за двата теста дава резултат FALSE за комбинирания израз. С други думи

ТЕСТ 1	AND	ТЕСТ 2	=	Резултат
FALSE		FALSE		FALSE
TRUE		FALSE		FALSE
FALSE		TRUE		FALSE
TRUE		TRUE		TRUE

б/ ако два теста са свързани с OR („логическо ИЛИ”), резултатът от тяхното изпълнение е FALSE само ако и двата теста върнат FALSE. Всички други варианти за стойности за двата теста дава резултат TRUE за комбинирания израз. С други думи

ТЕСТ 1	OR	ТЕСТ 2	=	Резултат
FALSE		FALSE		FALSE
TRUE		FALSE		TRUE
FALSE		TRUE		TRUE
TRUE		TRUE		TRUE

в/ операторът NOT, поставен пред даден тест, обръща неговия резултат. Тоест, ако тестът е върнал TRUE, NOT ще обърне резултата и той ще бъде FALSE и обратното.

Докато горните правила са ясни и интуитивни те не дават отговор на въпроса какво се случва когато два логически оператора се използват за комбиниране на три теста, три логически оператора се използват за комбиниране на 4 теста и т.н. Това е разгледано по-долу:

За да свърже два теста с логическо „и“, потребителят трябва да постави аргумент ***-a*** или ***-and*** между аргументите, които указват двата теста. За да свърже два теста с логическо „или“, потребителят трябва да постави аргумент ***-o*** или ***-or*** между двата аргумента, които указват съответните тестове. Ако между два теста няма ***-a***, ***-and***, ***-o*** или ***-or*** по подразбиране се счита, че двата теста са свързани с логическо „и“. За да обърне резултата от даден тест, потребителят трябва да постави знак ! (удивителна) или ***-not*** пред него.

В долния пример, потребителят извиква ***find*** с два различни теста, комбинирани с ***-and***, ***-or*** или ***-not***.

```
 1 lalev@dragonfly:~$ find /etc -type f -a -amin -60 2>/dev/null
 2 /etc/pam.d/lightdm-greeter
 3 /etc/pam.d/lightdm
 4 /etc/pam.d/systemd-user
 5 /etc/lightdm/lightdm-gtk-greeter.conf
 6 /etc/lightdm/lightdm-gtk-greeter.conf.d/30_lubuntu.conf
 7 /etc/X11/Xwrapper.config
 8 /etc/systemd/user.conf
 9 /etc/fuse.conf
10 lalev@dragonfly:~$ find ~ -name "*.zip" -o -name "*.rar"
11 /home/lalev/Downloads/case.rar
12 /home/lalev/Downloads/H110M-DVS R3.0(7.40)ROM.zip
13 /home/lalev/Downloads/TL-WR841N(US)_V13_180119.zip
14 lalev@dragonfly:~$ find . -size +1024M -size -2048M
15 ./Downloads/drive-download-20180627T213057Z-001.zip
16 lalev@dragonfly:~$
```

На ред 1 потребителят търси в ***/etc*** обикновени файлове (***-type f***), които едновременно са модифицирани преди по-малко от 60 минути (***-amin -60***). Двата теста са свързани с ***-a*** (логическо “и”)***,*** което означава, че и двата трябва да дадат TRUE (истина) за да бъде изведен един файл.

На ред 10 потребителят търси в домашната си директория файлове, които имат разширения ***.zip*** (***-name *.zip***) или (***-name *.rar***). Този път двата теста са свързани с ***-o*** (логическо „или“), което означава че или единият или другият или двата теста трябва да са истина, за да бъде целият логически израз истина и името на файла да бъде изведено. В случая двата теста са взаимно противоречиви и няма как едновременно да са истина, но е достатъчно само един от тях да върне TRUE, за да бъде изведен файла.

Командата на ред 14 търси файлове с размер над 1 и под 2 GB. В случая логическият оператор между двата теста е изпуснат, което означава, че те по подразбиране ще бъдат съединени с логическо „И“.

Читателят следва да обърне внимание на факта, че 1 GB е зададен като 1024 MB и 2 GB са зададени като 2048 MB. Поради закръгляването на големината на файла в понятията на избраните единици за пространство към аргумента ***-size***, евентуална команда ***find . -size +1G -size -2G*** няма да върне никакви файлове, дори да има файлове с размер между един и два гигабайта. В тази евентуална команда, потребителят е избрал мярка „гигабайти“ за измерване на големината на файловете, поради което големината на всеки един файл ще бъде изчислена в гигабайти след което закръглена до цяло число. Едва след това ще бъдат изпълнени тестовете. И тъй като се търсят файлове с размер *стриктно* над 1 и *стриктно* под 2 GB, няма да има такива файлове.

Когато изчислява поредица от повече от два теста, възниква въпросът в какъв ред ***find*** ще изпълни логическите оператори, тъй като изпълнени в различен ред, различните тестове, съединени със съюзи, означават различни условия и съвсем очаквано - връщат различен резултат. Описването на точния начин на изпълнение е много важен при ***find*** особено защото ***find*** изпълнява не само тестове, но понякога изпълнява и действия. Освен че връщат резултат

TRUE и FALSE, действията извършват и някакво друго действие-манипулация върху файловете. В подобни сценарии именно изпълняването или неизпълняването на действието е желания краен резултат, който е по-важен от това дали целият израз ще върне TRUE или FALSE. Подобно на същото явление в програмирането, терминологията на ***find*** нарича резултата от различните действия „*странични ефекти*“.

При изчисляване на израз, който съдържа множество тестове, действия и съюзи, ***find*** съблюдава приоритет на изпълнение на операторите, даден в следната таблица:

	Логически оператор	**Пример**
1	скоби	(израз)
2	-not, !	***-not*** *израз*
3	-and, -a	*израз1* ***-and*** *израз2*
4	-or, -o	*израз1* ***-or*** *израз2*
5	запетайка	израз1,израз2

Таблица. 2.9. Приоритет на изпълнение на логически оператори при find

Няма никаква случайност във факта, че таблицата много прилича на последователността на извършване на действията в „нормалната“ аритметика. Действията, поставени в скоби, имат най-висок приоритет, следващи по приоритет са действията ***-not*** и т.н. Запетайката на ред 5 в табл. 2.9. всъщност не е истински логически оператор и ще бъде разгледан по-долу, когато се разглеждат действията, локалните и глобални опции.

Трябва веднага да бъде наблегнато на факта, че между правилата на алгебрата на числата и логическата алгебра[14] съществуват някои разлики.

[14] Логическата алгебра е известна на професионален език като „Булева алгебра“ по името на първия ѝ изследовател.

Докато винаги трябва да изчислим един числов израз като например ***(3+5)+(2*9)*** до неговия край, за да получим резултата, при логическата алгебра това не винаги е необходимо. Компютърните програми и хардуер (както и в частност командата ***find***) използват този факт за да съкратят изчисленията и (както е случая с ***find***) да постигнат други ефекти. Така например ако два теста на ***find*** са свързани с AND и първият от тях върне резултат FALSE (лъжа), няма никакъв смисъл да се изчислява и вторият, тъй като комбинираният израз ще бъде винаги FALSE. По същия начин, ако два теста са свързани с OR и първият от тях върне TRUE (истина), няма никакъв смисъл да се изчислява и вторият, тъй като резултатът от комбинирания израз ще бъде винаги TRUE. ***find*** отчита тези факти и не изпълнява израза до край веднъж щом са направени достатъчно изчисления за да се определи неговата крайна стойност.

Изчисляването на даден логически израз до край или спирането, когато има налична достатъчно информация, за да се определи неговата крайна стойност, не влияят по никакъв начин на крайния резултат. Но когато са замесени действия, които както бе споменато, не само връщат логическа стойност, но и извършват дадени манипулации, някои от тези действия могат да попаднат в частта от логическия израз, която не се пресмята за даден файл. В такъв случай те няма да бъдат изпълнени за файла. Умелите системни администратори използват този факт за да изпълняват дадени действия само за файлове, които отговарят на определени критерии.

Точният начин по който ***find*** изпълнява логическите изрази е следният:

find винаги разделя израза на две части, които се изчисляват поотделно. Тези части ще наричаме условно „лява“ и „дясна“. Разделянето на израза на две части се случва винаги на логически оператор (изрично написан или вмъкнат по подразбиране между два теста), като избраният оператор винаги е

този с *най-нисък* приоритет. Избирането на логическия оператор с най-нисък приоритет е еквивалентно на поставянето на скоби около операторите с по-висок приоритет и техните операнди, което е точно действието, което човек, изчисляващ с лист и молив даден логически израз би направил. След разделянето, лявата част на израза се пресмята винаги докато в зависимост от особеностите на оператора, дясната част може да не бъде изчислена въобще. За всеки фрагмент от израз, който съдържа в себе си логически оператор, рекурсивно се прилагат същите правила.

В по долния пример са демонстрирани примери за употребата на ***find*** със сложни условия, съставени от множество тестове.

```
 1 lalev@dragonfly:~$ find /etc -type d ! -readable 2>/dev/null
 2 /etc/ssl/private
 3 /etc/polkit-1/localauthority
 4 /etc/lvm/backup
 5 /etc/inspircd
 6 /etc/amanda
 7 /etc/cups/ssl
 8 lalev@dragonfly:~$ find / -size +1G -a \( -name '*.zip' -o -name
   '*.vdi' \) 2>/dev/null
 9 /home/lalev/VirtualBox VMs/OfficeStuff/OfficeStuff.vdi
10 /home/lalev/Downloads/drive-download-20180627T213057Z-001.zip
11 lalev@dragonfly:~$ find /home/lalev -newermt '22:00 Jan 12 2019' -a
   ! -iregex '.*cache.*' -a ! -iregex '.*mozilla.*' 2>/dev/null
12 /home/lalev
13 /home/lalev/.lesshst
14 /home/lalev/VirtualBox VMs/OfficeStuff/OfficeStuff.vdi
15 /home/lalev/VirtualBox VMs/OfficeStuff/Logs/VBox.log
16 /home/lalev/.config/pulse/959df9c3f43d4a5b834b34c4162ddc61-stream-
17 volumes.tdb
18 lalev@dragonfly:~$ find /home/lalev -newermt '23:00 Jan 12 2019' -a
   ! -iregex '.*cache.*' -a ! -iregex '.*mozilla.*' 2>/dev/null
19 /home/lalev
20 /home/lalev/.lesshst
21 /home/lalev/VirtualBox VMs/OfficeStuff/OfficeStuff.vdi
22 lalev@dragonfly:~$ find /home/lalev -newermt '23:50 Jan 12 2019' -a
   ! -iregex '.*cache.*' -a ! -iregex '.*mozilla.*' 2>/dev/null
23 lalev@dragonfly:~$ date
24 Sat Jan 12 23:54:52 EET 2019
25 lalev@dragonfly:~$
```

На ред 1 потребителят търси в ***/etc*** и поддиректориите на ***/etc*** файлове, които са директории, които едновременно с това не са достъпни за четене от текущия потребител. Командата ***find*** на ред 1 има два теста и два логически оператора. От двата логически оператора, операторът ***-a*** има по нисък приоритет (удивителната, както бе споменато по горе, е друг начин да се обозначи логическо „НЕ“). Поради това ***find*** ще раздели израза на две части при ***-a***, които ще бъдат изпълнени поотделно. Това са съответно лявата част ***-type d*** и дясната ! ***-readable***. След това разделяне, ще бъде изпълнена лявата част, която се състои само от тест. Ако лявата част даде FALSE, тоест файлът не е директория, изчислението ще приключи, тъй като един от двата операнда на AND е FALSE, което означава че целият израз е FALSE. Като резултат този файл, който не е директория няма да бъде изведен в изхода на командата ***find***.

Ако обаче ***-type d*** върне TRUE, дясната част ще бъде изпълнена. Тъй като операторът ***-not*** няма ляв операнд, десният му операнд ***-readable*** ще бъде изпълнен винаги. Ако ***-readable*** върне FALSE, ***! -readable*** ще бъде TRUE. Стигне ли се до този момент от изчислението, при което двете части на оригиналния израз са известни и първият от тях е TRUE, операторът AND помежду ще бъде изпълнен. За файлове, които не са достъпни за четене, крайният резултат от логическия израз ще бъде TRUE, което ще доведе до извеждането на имената и пътищата до тях в изхода на командата ***find***.

Казано с други думи, командата на ред 1 на горния пример е еквивалентна на командата ***find /etc (-type -d) -a (! (-executable))***.

Когато потребителят иска да измени реда на изпълнение на логическите оператори, той може да постави сам скоби около тези оператори, които трябва да бъдат изпълнени първо. Той трябва обаче да се съобрази с факта, че скобите са специален символ на обвивката. Ако те не бъдат префиксирани със знака \

(“обратна наклонена черта”), обвивката ще опита да ги изпълни вместо да ги подаде на ***find***, което най-вероятно ще доведе до грешка.

Използването на скоби с ***find*** е илюстрирано на ред 8 в примера. Читателят трябва да забележи, че между обратно наклонената черта и знакът, който я следва, не трябва да има интервали или други символи.

В примера на ред 8 логическият израз ще бъде изпълнен в следната последователност:

1. Тъй като скобите са с по-висок приоритет от ***-a***, изразът ще бъде разделен на лява част ***-size +1G*** и дясна част ***-name '*.zip' -o -name '*.vdi'***.
2. Лявата част ще бъде изпълнена, давайки TRUE или FALSE. Ако файлът е над 1 GB, изчислението ще продължи. В противен случай, тъй като между лявата и дясната част стои логическо „И“, изчислението ще спре с резултат FALSE.
3. Дясната част се състои от два теста, свързани с логически оператор. Дясната част на свой ред ще бъде разделена на две части на единствения логически оператор, който в случая е ***-o***.
4. Новата лява част (***-name “*.zip”***) ще бъде изпълнена. Ако тя даде TRUE, резултатът от операцията логическо “ИЛИ“ ще е ясен, тъй като първият операнд е TRUE. Ако обаче новата лява част е FALSE, тогава ще се пресметне новата дясна част (***-name “*.rar”***). Ако тя е TRUE, резултатът от логическото „ИЛИ“ ще бъде TRUE. Ако тя е FALSE, резултатът от логическото „ИЛИ“ ще бъде FALSE.
5. На този етап ***find*** ще знае резултата от оригиналната лява и оригиналната дясна част на израза. Първият резултат е бил TRUE, тъй като иначе изчислението ще е приключило по-рано. Ако вторият също

е TRUE, крайният резултат ще е TRUE и пътя и името на текущия файл ще бъдат изведени.

Редове 11, 18 и 22 демонстрират една и съща конструкция, която използва теста ***-newerXt*** за да търси файлове, които имат дата на модификация след фиксирана дата. Първият тест търси файлове, модифицирани преди 22:00 на 12 януари 2019 година, докато следващите два теста придвижват тази дата напред към текущия момент, показвайки все по-малък списък от последно модифицирани файлове. Всички тези търсения използват тестовете ***-iregex*** за да изключат директории и файлове с имена, съдържащи думите ***cache*** и ***mozilla***.[15]

Ако ***find*** можеше само да търси файлове с определени свойства, той пак щеше да бъде напълно функционална и полезна програма. ***find*** обаче има още по богата и разнообразна функционалност, който произтича от възможността му да извършва различни манипулации върху намерените файлове благодарение на споменатите по-горе *действия*.

Действията приличат на тестовете по това, че те връщат логическа стойност. За разлика от тестовете, действията модифицират състоянието на файловата система или предизвикват извеждане на текст в изхода на ***find***. Вече бе споменато, че такива „допълнителни“ функции всъщност се наричат „*странични ефекти*“.

Най-простото действие, поддържано от ***find*** е отпечатването на името на файла в изхода на ***find***. Това се извършва от действието ***-print***, което винаги връща стойност TRUE. Всъщност, всяка последователност от тестове, която *няма* действие в себе си, по подразбиране получава ***-print*** в края на командата. Тоест извикването например на ***find / -type d -o -executable -o -perm g=w***

[15] Регулярните изрази ще бъдат разгледани подробно в главата за филтри по нататък.

(в командата има няколко теста, но няма експлицитно посочено действие) е еквивалентно на ***find / \(-type d -o -executable -o -perm g=w \) -print.***

-delete е друго действие, поддържано от ***find***. То води до изтриването на файлове и връща TRUE ако файла е изтрит успешно.

Може би най-полезната възможност на ***find*** касае изпълнението на зададени от потребителя команди върху намерените файлове. Това става с действията ***-exec***, ***-execdir***, ***-ok*** и ***-okdir***. Действията ***-ok*** и ***-okdir*** са аналогични съответно на ***-exec*** и ***-execdir***, с тази разлика, че първата двойка изисква потвърждение от потребителя преди изпълняването на всяка команда, докато втората двойка не прави това.

Вариантите с наставка "dir" изпълняват действието на командата в директорията, в която се намира файла. Вариантите, които нямат наставка "dir" извикват командата от директорията, в която се изпълнява ***find***.

Вариантите с наставка ***-execdir*** и ***-okdir*** съществуват, тъй като класическите, специфицирани в POSIX стандарта аргументи ***-exec*** и ***-ok*** водят до потенциални проблеми със сигурността.

Ако администраторът стартира ***find***, ***find*** и командите, стартирани с ***-exec*** и ***-ok*** ще работят с неговите привилегии. Но те ще оперират върху файлове и директории, които евентуално са собственост на други потребители. Това означава, че тези файлове и техните имена могат да бъдат манипулирани, за да подведат ***find*** да изпълни действия, които администраторът не възнамерява да извърши.

Единият начин за извършване на атаки срещу сигурността чрез ***find*** е възможен, защото съществува (макар и миниатюрна) разлика във времето, когато ***find*** изпълнява тестовете преди ***-exec*** или ***-ok*** и времето, когато ***find*** стартира командата, аргумент на ***-exec***. Да речем, че администраторът изпълни ***find /home -path "*bakup*" -type f -exec /bin/rm \;*** В същото време в ***/home/lalev/trap.bakup/,*** собственост на потребител ***lalev***, има файл с име ***passwd***, който също е собственост на ***lalev***. Този файл ще отговаря на критериите на търсенето, поради което евентуално ***find*** ще опита да изпълни ***rm /home/lalev/trap.backup/passwd***.

Тъкмо преди това да се случи, но след като ***find*** е определил, че файлът отговаря на посочените критерии, програма, написана точно за целта от злонамерения потребител ***lalev***, ще изтрие цялата директория ***trap.backup***. Това е позволено за потребителя ***lalev***, тъй като директорията е негова собственост. Веднага щом изтрие директорията, той ще постави символична връзка със същото име, която сочи към ***/etc***. По този начин при изпълнение на ***rm /home/lalev/trap.backup/passwd***, изтритият файл ще бъде ***/etc/passwd***. Това вече е действие, което не може да бъде извършено директно от

потребителя ***lalev***, и е опасно за системата,тъй като ***/etc/passwd*** е важен конфигурационен файл..

За разлика от ***-exec*** и ***-ok***, които изпълняват командите от директорията, от която е извикан ***find***, аргументите ***-execdir*** и ***-okdir*** първо влизат в директорията на файла, след което изпълняват командата от тази директория. Този метод предоставя защита срещу горната атака по две причини:

Първо, след като влезе в директорията, ***find*** извършва още една проверка, която се състои в това да се провери дали текущата директория е същата като оригиналната директория в която ***find*** е влязъл. Тази двойна проверка ще „улови“ подмяната на директорията със символична връзка и ***find*** ще откаже да изпълни командата върху дадения файл.

Второ, макар че между двойната проверка и изпълнението на командата - параметър на ***-execdir*** и ***-okdir*** - отново съществува промеждутък от време, този път злонамереният потребител ***lalev*** не може да изтрие директорията, тъй като директории в Линукс не могат да бъдат изтривани, когато даден процес ги е отбелязал като текущи.

За нещастие, подходът на ***-execdir*** и ***-okdir*** отваря пространство за провеждането на други атаки. Те стават възможни само ако се съчетаят две обстоятелства:

Първо, шаблонът за текуща директория (.) да е в променливата PATH на обвивката, и второ, администраторът да е подал командата - параметър на ***-execdir*** или ***-okdir*** без да указва абсолютен път до нея.

Променливата PATH, както бе дискутирано в допълнителния материал към Глава 1, съдържа списък от директории с изпълними файлове, които да бъдат претърсени за изпълним файл със същото име, когато потребителят извика команда без да указва път до нея. Възможно е администраторът по непредпазливост да е посочил път, който съдържа знака . (точка) обозначаващ текущата в момента директория.Това при всички случаи е лоша практика, но в конкретния сценарий ще доведе до това, че злонамереният потребител може да постави „фалшиви“ изпълними файлове в своята директория, които носят имената на най-често използваните команди. При комбинация с *втория* фактор - задаване на командата без абсолютен път до нея, тези фалшиви файлове ще бъдат извикани от ***find*** с привилегиите на администратора.

За да се избегне всяка възможност за провеждане на горните атаки е добра идея извикването на ***find*** да се извършва по следния начин:

1. ***-execdir*** и ***-okdir*** да се извикват вместо ***-exec*** и ***-ok***

2. Командите, изпълнявани от ***find,*** винаги да се задават с абсолютния път до изпълнимия им файл

3. Никога да не се поставя шаблона на обвивката . (точка) в PATH

На този етап любознателният читател може би се пита какво ако горната атака с поставяне на символична връзка вместо директорията ***trap.passwd*** се замени с поставянето на твърда връзка към файла ***/etc/passwd***. За да работи такава атака, трябва да са изпълнени няколко предпоставки:

Първо, вместо ***rm*** администраторът трябва да изпълнява някаква друга команда, тъй като ***rm*** просто ще премахне допълнителната твърдата връзка. Това не е невъзможно, тъй като съществуват множество смислени варианти за подобна команда, която може да е например ***echo*** с пренасочване, ***chmod***, ***chown*** и пр.

Второ, връзката трябва да се намира на същия дисков дял като папката ***/etc***. Това също не е невъзможно, тъй като обикновено всички потребители могат да създават файлове и директории в ***/tmp***, която обикновено е на същия дял като главната директория и ***/etc.***

Трето, трябва да е възможно злонамерените потребители да създават твърди връзки към файлове, които са собственост на други потребители. Тази последна предпоставка е направена невъзможна на модерните Линукс системи с помощта на конфигурационна настройка на ядрото, която по подразбиране не разрешава създаването на твърди връзки към чужди файлове (***fs.protected_hardlinks***). Незапознат системен администратор обаче, би могъл да отмени тази забрана и да направи системата податлива и на тази атака.

Поставянето на твърди и символични връзки в правилния момент вместо други файлове, представлява универсална категория от атаки, които действат върху широк кръг софтуерни програми, работещи с файловата система. Те засягат дори други операционни системи като Windows, който има същите механизми, някои от тях въведени поради опитите да се постигне POSIX съвместимост.

Модерните операционни системи и софтуер се опитват да намалят ефекта от тях по множество начини, което прави тяхното изпълнение все по-трудно за евентуални атакуващи. Някои от тези начини, използвани от Linux, са разгледани в упражненията. Така например новите версии на ***find*** ще откажат да изпълнят ***-execdir*** или ***-okdir***, ако шаблонът за текуща работна директория е в променливата PATH.

Познаването на възможните атаки, включително и тези, срещу които модерният софтуер прилага защити, е изключително полезно за системния администратор, когато той влезе в ролята на програмист и започне да съставя шел скриптове за обвивката. В тази роля той е в позиция да повтори много от същите грешки, които са отстранени в масовия софтуер като уеб сървъри, сървъри за ел. поща и пр. Ето защо е полезно да се споменат няколко съвета за това как системата и програмите, работещи върху нея могат да се „заздравят“ срещу атаки с поставяне на връзки.

1. На дадена система конзолен достъп следва да имат само администраторите.

Това изискване не е трудно за постигане на много системи, които съществуват, за да предлагат услуги от „по-високо“ ниво като уеб сървъри, сървъри за ел. поща и пр., които работят чудесно и без потребителите да имат достъп до конзолата. Може да се каже, че системите, в които редовите потребители имат конзолен достъп, са голяма рядкост в наши дни. Но именно тези системи крият най-големи рискове за сигурността.

2. Много системи позволяват потребителя да качи файлове с определени имена на сървъра дори когато той няма конзолен достъп. Така например услугите FTP или SAMBA или дори всяко уеб приложение, което по някаква причина се нуждае от възможността потребителите да качват файлове, са в подобно положение. Когато това е възможно, системата не трябва да оставя потребителя да определя вътрешното име на файла, тъй като това води до друг клас атаки, споменати по-долу.

Златният стандарт в тази област, съблюдаван от повечето уеб приложения, предвижда името, което е дадено от потребителя, да се промени и файлът да получи случайно име. Двойката „реално име“ - „случайно име“ може бъде съхранена отделно, например в база от данни. Случайното име се състои от дълга последователност от случайно избрани букви (и/или цифри). Използването на такива имена обикновено се съчетава с поставянето на всички файлове, чиито имена започват с ***a*** в поддиректория, която се казва ***a***, всички файлове, чиито имена започват с ***b*** в поддиректория, която се казва ***b*** и т.н. Това намалява размера на файловете-директории, които трябва да бъдат изчетени, за да се намери ***inode*** номера на файла и ускорява намирането на файла във файловата система.

3. Програмите и в частност шел скриптовете, написани от администратора, трябва да избягват създаването на временни файлове в ***/tmp*** или друга директория, върху която имат права и непривилегировани потребители Добра идея е имената на такива файлове да бъдат колкото се може по-непредсказуеми, както и да се използва добре т.нар. "***sticky***" бит.

Синтаксисът на аргументите ***-exec***, ***-execdir***, ***-ok*** и ***-okdir*** следва един и същи формат:

```
find [-H]  [-L]  [-P] списък_с_начални_точки... [логически израз]-exec
команда [аргументи на командата] ;
```

По-долу са демонстрирани примери на използването на ***find*** с ***-exec*** и сродни аргументи.

```
 1 lalev@dragonfly:~/example11$ ls -l
 2 total 12
 3 -rw-rw-r-- 1 lalev lalev  512 Jan 17 16:19 test1
 4 -rw-rw-r-- 1 lalev lalev 1024 Jan 17 16:19 test2
 5 -rw-rw-r-- 1 lalev lalev 2048 Jan 17 16:48 test3
 6 lalev@dragonfly:~/example11$ find . -size 1b -print
 7 ./test1
 8 lalev@dragonfly:~/example11$ find . -size 1b -exec /bin/rm {} \;
 9 lalev@dragonfly:~/example11$ ls -l
10 total 8
11 -rw-rw-r-- 1 lalev lalev 1024 Jan 17 16:19 test2
12 -rw-rw-r-- 1 lalev lalev 2048 Jan 17 16:48 test3
13 lalev@dragonfly:~/example11$ find . -size +1b -print -exec /bin/rm
   {} +
14 .
15 ./test3
16 ./test2
17 /bin/rm: cannot remove '.': Is a directory
18 lalev@dragonfly:~/example11$ ls
19 lalev@dragonfly:~/example11$
```

На ред 1 потребителят ***lalev*** проверява съдържанието на текущата директория. Както е видно от изхода на командата ***ls***, тя съдържа три файла с големина съответно 512, 1024 и 2048 байта.

На ред 6 потребителят търси файлове с големина след закръгляването точно 1 блок с големина 512 байта, докато на ред 8 той изтрива тези файлове.

Винаги е добра практика преди да се изпълни дадено действие върху файловете, намерени от ***find***, да се провери дали критериите за търсене са зададени правилно. За целта на мястото на ***-delete*** или ***-exec*** може да се постави ***-print***. Командата на ред 6 прави точно това, като извежда имената на файловете, които предстои да бъдат изтрити на ред 8. Както е видно от изхода на ***find*** на ред 7, точно един файл с име ***test1*** отговаря на зададените критерии.

Командата на ред 8 използва същия тест като предната (***-size 1b***). Този път обаче след теста е добавено действието ***-exec /bin/rm {} \;*** Преди изпълнението на ***/bin/rm,*** двойните фигурни скоби ще бъдат заменени от пътя до файла, обработван в момента. В контекста на текущия пример това означава, че командата на ред 8 ще извърши едно извикване на ***/bin/rm*** за файла ***test1***, а реално извиканата команда ще бъде ***/bin/rm ./test1***.

Точката и запетаята в края на ред 8 обозначават края на списъка с параметри на ***-exec***. След този знак може да бъде поставен следващ логически оператор и аргумент.

Читателят може да забележи, че примерът от ред 8 не използва ***-delete*** а ***-exec /bin/rm*** за изтриване на външни файлове. Двете действия постигат еквивалентен ефект, но всъщност извикването на външна команда за изтриване има по-голям смисъл, когато става въпрос за използване на ***-ok*** и ***-okdir***.

Тъй като ***find*** работи с файлове, е напълно логично, че на практика всяка от извикваните програми чрез ***-exec***, ***-execdir***, ***-ok*** и ***-okdir*** приема файл или списък от файлове за аргументи. Ето защо ***find*** има редица механизми за предаване на списъка от намерени файлове на изпълняваните външни програми.

Най-често използваният механизъм за подаване на файлови имена към външни команди вече бе показан в примерите по-горе. Ако потребителят

изпише {} в аргументите на командата, подадена като параметър на ***-exec***, ***find*** ще замени {} с името на файла, който се обработва в момента. С използването на {}, командата след ***-exec*** ще бъде изпълнена толкова пъти, колкото пъти обработката на логическо условие за даден файл достигне до ***-exec*** частта. С други думи ако 1000 файла отговарят на условията, зададени на ***find*** преди ***-exec*** частта, командата след exec ще се изпълни 1000 пъти за всеки един отделен файл.

Понякога синтаксисът на извикваната команда изисква файловите имена да бъдат подадени като списък от файлове, а командата да бъде изпълнена само веднъж. ***find*** позволява подаването на параметри към такива команди с помощта на конструкцията {} +. Това е демонстрирано на ред 13 в горния пример.

Конструкцията {} +, поставена като аргумент на извикваната от ***find*** команда, кара ***find*** да изчака и събере имената и пътищата до всички файлове, които отговарят на предхождащите условия. Едва когато ***find*** има целия списък с всички файлове, отговарящи на условията, той извиква командата. Този път вместо по един на ред, пътищата до файловете са на един ред, разделени с интервали.

Командата на ред 13 в горния пример изпълнява три логически условия свързани по подразбиране с оператори AND. Две от тези условия са действия. Първо се изпълнява тестът ***-size +1b***, който проверява дали файлът е с големина, стриктно по-голяма от един блок. Ако тази проверка даде TRUE за даден файл, обработката преминава към следващото условие, което е действието ***-print.*** Това води до отпечатването на името и пътя до файла на конзолата.

От изхода на командата можем да видим, че няколко файла отговарят на условието за големина. Тези файлове са съответно . (файлът на текущата

директория), ***test2*** и ***test3***. ***-print*** връща винаги TRUE, което означава, че обработката продължава към следващото условие, което е ***-exec***. Тъй като този ***-exec*** има в параметрите си {} +, ***find*** ще „складира“ пътищата до файловете, достигнали тази фаза на обработката, в списък. Едва когато ***find*** обходи и тества всички файлове, той ще подаде списъка на командата, указана в ***-exec*** опцията. Като резултат, командата ***rm*** ще бъде извикана само веднъж и в случая има вида ***rm . ./test3 ./test2***.

Ако вместо командата на ред 13, потребителят бе написал ***find . -size +1b -print -exec /bin/rm {} \;*** (без + след {}), това щеше да доведе до извикването на ***rm*** три пъти:

```
rm .
rm ./test3
rm ./test2
```

Поради по-особения начин на изпълнение, ***-exec*** с ***{} +*** винаги трябва да бъде последно условие. Поради това не се налага списъкът с извикваните от него програми и техните аргументи да бъде завършван с точка и запетая.

Друга алтернатива на ***{} +*** е използването на ***find*** в комбинация с командата ***xargs***. ***xargs*** взема на стандартния си вход списък от файлове, всеки на отделен ред, и го превръща в списък на един ред, който бива подаден като аргумент на избрана команда. Синтаксисът на използване на ***xargs*** заедно с ***find*** може да бъде представен по следния начин:

```
find .......... | xargs команда [аргументи на командата]
```

Долният пример илюстрира конкретно как се използва ***xargs***.

```
1 lalev@dragonfly:~/test34$ find . -type f
2 ./alpha
3 ./beta
4 lalev@dragonfly:~/test34$ find . -type f | xargs
5 ./alpha ./beta
```

```
 6 lalev@dragonfly:~/test34$ find . -type f | xargs rm -f
 7 lalev@dragonfly:~/test34$ find . -type f
 8 lalev@dragonfly:~/test34$
```

На ред 1 потребителят извиква ***find*** за да намери всички обикновени файлове в текущата директория. Оказва се че в директорията има само два такива файла. ***find***, разбира се, извежда пътя до всеки един от тях на отделен ред (това са редове 2 и 3 на разпечатката).

На ред 4 потребителят чрез знака **|** постановява пренасочване при което изходът на командата ***find*** този път не се извежда на конзолата, а става вход на ***xargs***. Както е видно от ред 5, ***xargs*** трансформира списъка с файлове от формат по един на ред към формат „много на ред“, разделени с интервали.

В типичния случай на използването на ***xargs***, потребителят ще иска не просто да превърне списъка от един формат в друг, но и да извика команда, на която да предаде този списък. Това е демонстрирано на ред 6, където ***xargs*** е използван да изтрие двата намерени файла. Читателят следва да забележи, че ***rm -f*** няма аргументи - имена на файлове именно защото ***xargs*** ще ги подаде. ***xargs*** винаги ще добави списъка с аргументи на последно място в извикваната команда. Тоест в конкретния пример, реално изпълнената от ***xargs*** команда ще бъде ***rm -f ./alpha ./beta***.

Използването на командата ***xargs*** по илюстрирания горе начин крие и някои рискове за сигурността особено когато става въпрос за екзотични и стари файлови системи. На теория имената на файловете в Линукс могат да съдържат всеки символ освен специалния символ ***null*** (ASCII код 0). Въпреки че модерните ядра и инструменти ще се опитат по всякакъв начин да предотвратят вредните ефекти от поставянето на други специални символи във файловите имена, все още съществува теоретичната възможност знак за нов ред да се окаже в дадено файлово име без да бъде обезопасен с подходящата *ескейп последователност*. Името на този файл ще се изведе на два реда (заради знака

за нов ред) и на практика ще се преобразува в две други. ***xargs*** няма да може да определи дали това са имена на два отделни файла или става въпрос за един файл. Този дефект може да бъде експлоатиран от зложелатели, за да се накара да изпълни неоторизирани операции със системни файлове.

За да се избегне тази възможност, ***find*** може да бъде накаран да разделя имената на файловете не със знак за нов ред а със знак ***null***. Съответно ***xargs*** има аргумент, който указва на програмата да очаква ***null*** в ролята на разделител.

По-долният пример илюстрира този начин за употреба на ***xargs***.

```
 1 lalev@dragonfly:~/test34$ find . -type f
 2 ./alpha
 3 ./beta
 4 lalev@dragonfly:~/test34$ find . -type f -print0
 5 ./alpha./betalalev@dragonfly:~/test34$
 6 lalev@dragonfly:~/test34$ find . -type f -print0 | xargs --null
 7 ./alpha ./beta
 8 lalev@dragonfly:~/test34$ find . -type f -print0 | xargs -0 rm -f
 9 lalev@dragonfly:~/test34$
```

Примерът с ***find*** на ред 1 е аналогичен на предния и демонстрира, че отново в директорията имаме файлове с имена ***alpha*** и ***beta***. На ред 4 потребителят добавя действието ***-print0*** към края на командата. То отменя подразбиращото се действие ***-print*** и води до това че след името на всеки файл не се отпечатва знака за нов ред в Линукс, а се отпечатва знакът ***null***. Този знак не се извежда на конзолата, поради което изходът на ред 5 изглежда малко хаотичен. Но предназначението на ***-print0*** не е да изведе красиво изглеждащ списък към конзолата, а да изведе списък, който не може да бъде манипулиран. Типичният ползвател на този списък е програмата ***xargs.***

На ред 6 е демонстрирано как полученият изход от ***find*** може да се препрати като вход на ***xargs.*** За да знае ***xargs***, че списъкът от ***find*** е разделен с ***null*** символи, потребителят е задал аргумент ***--null.*** Както е видно, това

дава очакваните резултати - двата намерени файла са изведени от ***xargs*** като списък на един ред, разделен с интервали.

На последния ред потребителят най-накрая използва ***xargs*** за да стартира команда ***rm***, която изтрива двата файла. Тук вместо ***--null*** потребителят е задал ***-0***, което има същия ефект (***--null*** и ***-0*** са две алтернативни имена на един и същ аргумент).

До момента бяха демонстрирани някои основни тестове и действия, поддържани от ***find***. В сравнение с тях глобалните и позиционни опции се използват значително по-рядко. Въпреки това ние ще представим няколко от тях.

Опциите ***-mindepth n*** и ***-maxdepth n*** позволяват на потребителя да укаже на колко нива надолу в йерархията на директориите да започне (или съответно - да приключи) търсенето. Така например ***-maxdepth 1*** ще ограничи обхождането до началните точки и техните преки поддиректории. ***-xdev*** не позволява търсенето да преминава към други дискови дялове, било защото те са монтирани в поддиректории на началните точки или затова, че ***find*** е инструктиран да следва символични връзки.

Пример за позиционна опция е аргументът ***-regextype mun*** който важи само за ***-regex*** или ***-iregex*** опциите, които го следват. Различните основни типове регулярни изрази ще бъдат разгледани в следващите глави.

Освен всички други настройки ***find*** поддържа и три аргумента, които трябва да бъдат поставени задължително на първо място, непосредствено след името на командата.

Ако потребителят зададе аргумента ***-L***, ***find*** ще следва символичните връзки. Тоест тестовете и действията ще се отнасят за файла, към който сочи дадената символична връзка. Ако връзката е счупена, т.е. файлът, към която връзката сочи не съществува, тестовете ще се прилагат към самата връзка.

Ако потребителят зададе аргумента ***-H***, връзките няма да бъдат следвани, освен ако не са зададени като начални точки.

Ако потребителят не зададе ***-H*** или ***-L***, поведението на ***find*** по подразбиране е да не се следват символичните връзки, независимо дали те са зададени като начална точка или са открити в процеса на обхождане на поддиректориите. Това експлицитно може да се зададе и с аргумента ***-P***. При това положение тестовете ще се прилагат върху самата символична връзка, а не към файла, който сочи, и ***find*** ще извежда името на самата връзка в списъка с файлове.

6. Казус - стартиране на програма в chroot среда

Стартирането на програма в ***chroot*** среда е един от най-старите методи за повишаване сигурността на системата чрез изолиране на достъпа на дадено критично приложение до файловата система. ***chroot*** работи като сменя главната директория с някоя от поддиректориите на системата. Тоест, за стартираното чрез ***chroot*** приложение, поддиректорията става главна директория, като по този начин, директориите извън избраната поддиректория стават недостъпни за приложението.

В този казус, системният администратор е избрал да стартира сървър за услугата IRC[16]. От различните софтуерни програми - сървъри за IRC - администраторът е избрал да инсталира InspIRCd[17]. Дистрибуцията на Линукс, която е инсталирана на сървърния хардуер е Ubuntu, версия 18.04 LTS.

[16] Услугата IRC позволява разговори в реално време в Интернет. Изключително популярна преди появата на социалните мрежи като Twitter и Facebook, днес тя има сравнително малък кръг, но за сметка на това изключително лоялни ползватели. Вж. **https://bg.wikipedia.org/wiki/Internet_Relay_Chat**

[17] Вж. **http://www.inspircd.org/**

В първата част на казуса, администраторът инсталира ***inspircd*** през пакетната система на Ubuntu.

```
lalev@dragonfly:~$ sudo bash
root@dragonfly:~# apt-get install inspircd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libtre5
Suggested packages:
  sqlite3 default-mysql-server ldap-server postgresql gnutls-bin
tre-agrep
The following NEW packages will be installed:
  inspircd libtre5
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 0 B/1,448 kB of archives.
After this operation, 9,360 kB of additional disk space will be
used.
Do you want to continue? [Y/n] y
Selecting previously unselected package libtre5:amd64.
(Reading database ... 265719 files and directories currently
installed.)
Preparing to unpack .../libtre5_0.8.0-6_amd64.deb ...
Unpacking libtre5:amd64 (0.8.0-6) ...
Selecting previously unselected package inspircd.
Preparing to unpack .../inspircd_2.0.24-1ubuntu1_amd64.deb ...
Unpacking inspircd (2.0.24-1ubuntu1) ...
Processing triggers for ureadahead (0.100.0-20) ...
Setting up libtre5:amd64 (0.8.0-6) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for systemd (237-3ubuntu10.9) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Setting up inspircd (2.0.24-1ubuntu1) ...
Created symlink /etc/systemd/system/multi-
user.target.wants/inspircd.service →
/lib/systemd/system/inspircd.service.
Processing triggers for systemd (237-3ubuntu10.9) ...
Processing triggers for ureadahead (0.100.0-20) ...
root@dragonfly:~# systemctl stop inspircd
root@dragonfly:~# systemctl disable inspircd
Synchronizing state of inspircd.service with SysV service script
with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install disable inspircd
root@dragonfly:~#
```

На ред 1 администраторът, който до момента използва системата като обикновен потребител, придобива административни привилегии като извиква ***sudo***. Изпълняваната от ***sudo*** команда е стандартната обвивка ***bash***, което означава, че оттук нататък всяка въведена команда ще бъде изпълнена като администратора на системата - ***root***.

На ред 2 администраторът инсталира InspIRCd, извиквайки системата за управление на пакети в Ubuntu чрез програмата ***apt-get***.

Системата за управление на пакети в Ubuntu отговаря за инсталирането и обновяването на допълнителния софтуер, инсталиран на компютъра. Въпреки това кратко описание на нейните функции, нейната роля в автоматизиране на работите по системна администрация е огромна.

В по старите Unix варианти от 80-те и 90-те години на миналия век, от които води началото си и Линукс, подобни системи за управление на пакети не съществуваха. От администраторите много често се очакваше да компилират софтуера от изходния му код, за да го инсталират на системата. Това работеше задоволително, тъй като софтуерът бе значително по-прост и монолитен по своя дизайн. „Прост и монолитен“ например означаваше, че дадена софтуерна програма зависеше от много по-малко на брой външни библиотеки с програмен код. Това също означаваше, че когато такива библиотеки се използват, те се свързваха с програмата по времето на компилирането и ставаха част от изпълнимия файл. Веднъж разполагайки с изпълнимия файл на инсталираната програма, администраторът не трябваше да извършва други дейности по поддръжка на тези външни библиотеки.

С времето софтуерът значително се усложни и специализира. Като резултат от тази еволюция, програмите започнаха да зависят от все по-голям брой външни библиотеки с функции и други програми, за да извършват правилно дейността си. Своеобразен вододел в това отношение бе преминаването към т.нар. „динамично свързване“. Някои от външните библиотеки, които се използваха от различните програми, бяха толкова популярни, че на практика се използваха от всяка една програма. Тогава при зареждането на например две програми, които използват една такава популярна библиотека, две копия на един и същи програмен код трябваше да бъдат заредени и да заемат място в оперативната памет на компютъра. Това се оказа натоварващо за хардуера, поради което бързо бе въдена друга практика. Вместо да се свързват във всеки изпълним файл още при компилирането, външните библиотеки се инсталират отделно и се зареждат в паметта на компютъра, когато даден изпълним файл ги изиска. Това позволява на операционната система да следи за дубликати и вместо да зарежда повторно код в паметта, да „насочи“ втората, третата и пр. стартирани програми към вече заредената външна библиотека в паметта. Този механизъм именно получи името „динамично свързване“, а библиотеките заредени по този начин се наричат „*споделени библиотеки*“.

Докато динамичното свързване свърши страхотна работа по разтоварване оперативната памет на компютъра, това че библиотеките заживяха собствен живот като отделни файлове на системата, които трябва да са налице, за да се изпълнят правилно

изпълнимите програми, създаде много грижи на администраторите. Много често новата версия на някоя изпълнима програма изискваше и нова версия на споделена библиотека. В същото време други инсталирани програми не можеха да работят с новата версия, поради което на системата трябваше да се пазят няколко версии на една и съща библиотека. Понякога библиотеките имаха свои конфигурационни файлове и директории с допълнителни файлове и различните версии на библиотеките влизаха в конфликт една с друга за тях. Всичко това създаде сериозни предизвикателства пред процеса на инсталиране и обновяване на софтуера. Пакетните системи се появиха, за да отговорят на тези предизвикателства. Те управляват зависимостите между инсталираните програми и споделените библиотеки и позволяват времето за инсталиране на софтуер да се сведе до минимум. Те също свеждат вероятността за пропуски при обновяване на стар софтуер до минимум, издигайки сигурността на системата на ново ниво.

Различните дистрибуции на Линукс вярват в различни философии по отношение управлението на пакетите, което води до това, че различните дистрибуции имат различни системи за управление на пакетите. Двете най-популярни системи са Advanced Package Tool (***apt***), която се използва от Debian Linux и неговите деривати (Ubuntu, Linux Mint и т.н.) и Yellowdog Updater, Modifier (***yum***), който се използва в дистрибуциите, които водят началото си от RedHat (Fedora, CentOS, RedHat Enterprise Linux и пр.).

Редове 3-14 са информация, изведена от ***apt-get*** и предназначена за администратора. На редове 6 и 7 ***apt-get*** информира администратора, че за да работи правилно ***InspIRCd***, трябва да бъде инсталирана една допълнителна външна библиотека с функции, която към момента липсва на системата. Това е ***libtre5***[18] . Ако ***libtre5*** зависи на свой ред от други външни библиотеки, те ще бъдат добавени към окончателния списък за инсталация, който е показан на редове 10-12. Оказва се, че ***libtre5*** не зависи от такива библиотеки и окончателният списък за инсталация се състои от ***inspircd*** (самият сървърен софтуер) и ***libtre5***.

Редове 13 и 14 информират администратора, че при инсталирането на посочените пакети ще бъдат изтеглени 1.4 MB архиви, които след разархивиране на диска ще заемат около 9.3 MB на диска.

[18] Повечето имена на пакети са съкращения, които произтичат от името на библиотеката/програмата и нейната версия. В случая става въпрос за пета версия на TRE https://laurikari.net/tre/.

На ред 15 ***apt-get*** пита администратора дали е съгласен да се извърши описаната инсталация. Администраторът отговаря утвърдително с "***y***".

Инсталацията на сървърен софтуер в много случаи води до неговото стартиране с параметри по подразбиране. Ако администраторът нямаше други цели, като например по-добрата изолация на този сървър от останалите процеси и файлове в системата, казусът би приключил тук. Целта на администратора обаче е да създаде ***chroot*** среда и да стартира ***inspircd*** от нея.

Затова на ред 34 администраторът спира с командата ***systemctl stop inspircd*** току-що стартирания от пакетната система сървър. С командата на ред 35 (***systemctl disable inspircd***) той забранява стартирането на ***inspircd*** при стартиране на операционната система. По този начин той подсигурява, че ***inspircd*** ще е инсталирана на системата, но няма да се стартира извън средата, която тепърва предстои да бъде подготвена.

След тези дейности, администраторът се насочва към втората подзадача - издирването на изпълнимия файл на ***inspircd*** сървъра и определянето на всички споделени библиотеки, от които зависи неговата работа. Тук е моментът да се отбележи, че освен ***libtre5***, може да има и други такива библиотеки за които ***apt-get*** не предупредил, тъй като те са били вече инсталирани на системата.

Администраторът има намерение да издири всички такива библиотеки и да ги пренесе, заедно с изпълнимата програма, конфигурационните и допълнителните файлове, в поддиректория на системата. След това смята да ограничи ***inspircd*** до тази поддиректория като използва ***chroot***.

Ако след смяната на главната директория с ***chroot***, допълнителните библиотеки, конфигурационните и допълнителните файлове не са на същите места спрямо новата главната директория, ***inspircd*** просто няма да стартира изпълнението си.

```
39 root@dragonfly:~# find / -name inspircd 2>/dev/null
40 /etc/default/inspircd
41 /etc/logrotate.d/inspircd
42 /etc/inspircd
43 /etc/init.d/inspircd
44 /usr/share/lintian/overrides/inspircd
46 /usr/share/doc/inspircd
47 /usr/sbin/inspircd
48 /usr/lib/inspircd
49 /usr/lib/inspircd/inspircd
50 root@dragonfly:~# ldd /usr/sbin/inspircd
51         linux-vdso.so.1 (0x00007ffeb33d1000)
52         libstdc++.so.6 => /usr/lib/x86_64-linux-gnu/libstdc++.so.6
   (0x00007fe5e8d72000)
53         libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2
   (0x00007fe5e8b6e000)
54         librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1
   (0x00007fe5e8966000)
55         libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1
   (0x00007fe5e874e000)
56         libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0
   (0x00007fe5e852f000)
57         libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
   (0x00007fe5e813e000)
58         libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6
   (0x00007fe5e7da0000)
59         /lib64/ld-linux-x86-64.so.2 (0x00007fe5e93de000)
60 root@dragonfly:~#
```

На ред 39 администраторът инициира търсене на всички файлове с име ***inspircd*** в цялата система. Тъй като инсталираният пакет се нарича по същия начин, администраторът вероятно действа по интуиция, правейки догадката че инсталирането на пакет с име ***inspircd*** ще инсталира изпълними файлове и/или директории със същото име.

Частта ***2>/dev/null*** представлява пренасочване на изхода на програмата ***find***, което предизвиква това, че всички съобщения за грешка (или такива с допълнителна информация) няма да бъдат показани. Това означава че в показания изход ще има само имената на намерените файлове.

Пренасочванията на вход и изход ще бъдат най-подробно разгледани в следващите глави.

Както се оказва догадката е била коректна и ***find*** извежда списък с абсолютните пътища до файлове и директории, които се казват ***inspircd***. Ако това не беше така, администраторът щеше вероятно да прибегне до по-сигурни, макар и по-бавни начини за намирането на разположението на изпълнимия файл. Такъв начин например е прочитането на стартиращите скриптове на различните услуги в ***/etc/init.d***.

Най-полезната информация за администратора идва от ред 47, който показва, че в директория ***/usr/sbin*** има файл с името ***inspircd***. Тъй като тази директория по конвенция съдържа изпълними файлове, това вероятно означава, че този файл е изпълним и извикването му е начина за стартиране на сървъра.

Другите редове също подсказват, че освен изпълнимия файл, сървърът зависи и от други файлове. Такива например са файловете в ***/etc/inspircd.*** По конвенция, файловете в директория ***/etc*** и нейните поддиректории съдържат конфигурационна информация и е много вероятно сървърът да не може да стартира правилно без тях.

На ред 50 администраторът използва командата ***ldd*** за да разбере кои са външните библиотеки, необходими за работата на ***/usr/bin/inspircd***, и къде са файловете, които ги съдържат. Изходът на ***ldd*** представлява таблица с редове. Всеки ред представлява една библиотека, като форматът на реда е следния:

- име на библиотеката
- разделител =>

- място във файловата система, където зареждащата програма е открила споделената библиотека
- местоположение в адресното пространство, където споделената библиотека е заредена (представено като адрес в шестнадесетична форма, поставен в скоби)[19].

Два реда се отклоняват от този шаблон. Първият ред (ред 59 на разпечатката) представлява т.нар. „виртуален динамично споделен обект“, който не е реална библиотека, а начин на ядрото на Линукс да изложи достъп до често използвани функции. Щом не е реален файл, този обект не трябва да се копира в ***chroot*** средата. Последният ред всъщност е зареждащата програма, която отговаря за зареждането на останалите споделени библиотеки. Тя също не е типичен споделен обект, но трябва да бъде копирана на правилното място в ***chroot*** средата. Останалите редове показват местоположението на всички нужни споделени библиотеки.

След като е определил местоположенията на важните файлове, третата подзадача, която трябва да бъде изпълнена от администратора всъщност е създаването на средата и копирането на файловете.

```
61 root@dragonfly:~# mkdir -p /jails/inspircd
62 root@dragonfly:~# ldd /usr/sbin/inspircd | cut -f 2 | sed 's/.*=>
   \(.*\)/\1/g'|tail +2|cut -f 1 -d\
63 /usr/lib/x86_64-linux-gnu/libstdc++.so.6
64 /lib/x86_64-linux-gnu/libdl.so.2
65 /lib/x86_64-linux-gnu/librt.so.1
66 /lib/x86_64-linux-gnu/libgcc_s.so.1
67 /lib/x86_64-linux-gnu/libpthread.so.0
```

[19] Линукс системите сравнително отскоро използват рандомизация на разположението на адресното пространство (ASLR) като средство за борба с хакерските атаки. Това означава, че показаните адреси ще се менят всеки път. Информацията от тези адреси не е предназначена за системните администратори, а по-скоро за програмистите, които се опитват да проследят и отстранят евентуални грешки. Важно е да се отбележи обаче, че ***ldd*** може да бъде манипулирана да стартира разглежданата програма, поради което трябва да се използва с внимание върху непознати изпълними файлове.

```
68 /lib/x86_64-linux-gnu/libc.so.6
69 /lib/x86_64-linux-gnu/libm.so.6
70 /lib64/ld-linux-x86-64.so.2
71 root@dragonfly:~# cp -L --preserve --parents -v -t
   /jails/inspircd/  `ldd /usr/sbin/inspircd | cut -f 2 | sed 's/.*=>
   \(.*\)/\1/g'|tail +2|cut -f 1 -d\ |xargs`
72 /usr -> /jails/inspircd/usr
73 /usr/lib -> /jails/inspircd/usr/lib
74 /usr/lib/x86_64-linux-gnu -> /jails/inspircd/usr/lib/x86_64-linux-
   gnu
75 '/usr/lib/x86_64-linux-gnu/libstdc++.so.6' ->
   '/jails/inspircd/usr/lib/x86_64-linux-gnu/libstdc++.so.6'
76 /lib -> /jails/inspircd/lib
77 /lib/x86_64-linux-gnu -> /jails/inspircd/lib/x86_64-linux-gnu
78 '/lib/x86_64-linux-gnu/libdl.so.2' -> '/jails/inspircd/lib/x86_64-
   linux-gnu/libdl.so.2'
79 '/lib/x86_64-linux-gnu/librt.so.1' -> '/jails/inspircd/lib/x86_64-
   linux-gnu/librt.so.1'
80 '/lib/x86_64-linux-gnu/libgcc_s.so.1' ->
   '/jails/inspircd/lib/x86_64-linux-gnu/libgcc_s.so.1'
81 '/lib/x86_64-linux-gnu/libpthread.so.0' ->
   '/jails/inspircd/lib/x86_64-linux-gnu/libpthread.so.0'
81 '/lib/x86_64-linux-gnu/libc.so.6' -> '/jails/inspircd/lib/x86_64-
   linux-gnu/libc.so.6'
82 '/lib/x86_64-linux-gnu/libm.so.6' -> '/jails/inspircd/lib/x86_64-
   linux-gnu/libm.so.6'
83 /lib64 -> /jails/inspircd/lib64
84 '/lib64/ld-linux-x86-64.so.2' -> '/jails/inspircd/lib64/ld-linux-
   x86-64.so.2'
85 root@dragonfly:~# cp -r -L --preserve --parents /etc/inspircd
   /jails/inspircd/
86 root@dragonfly:~# cp --preserve --parents /usr/sbin/inspircd
   /jails/inspircd/
87 root@dragonfly:~#
```

На ред 61 администраторът създава поддиректорията, в която ще бъде поместена ***chroot*** средата. Тя е ***/jails/inspircd***.

На ред 62 администраторът продължава с копиране на споделените библиотеки. Въпреки че споделените библиотеки могат да бъдат копирани една по една с ***cp***, а директориите в ***chroot*** средата могат да бъдат създадени една по една с ***mkdir***, администраторът избира малко по-технически интензивен подход. На ред 62 администраторът използва филтри, за да извлече

от изхода на ***ldd*** само колоната с абсолютните пътища до споделените библиотеки. Използването на филтри ще бъде разгледано в детайли в следващите глави, но читателят, който се опитва да пресъздаде казуса, трябва да бъде веднага посъветван, че последният символ след наклонената черта на ред 62 е интервал и това не може да се покаже на разпечатката.

Командата на ред 62 е нещо като „генерална репетиция“ за същинското копиране, което става на ред 71. Вероятно командата на ред 62 е издадена от администратора за да провери дали списъкът с файлове за копиране (редове 63-70) е извлечен правилно.

Списъкът е коректен но още не може да бъде подаден като аргумент на ***cp***, тъй като всеки файл е показан на нов ред. Това, от което администраторът се нуждае, е списък със същите файлове, но на един ред, разделени с интервал помежду си. Той ще постигне това на ред 71 с въвеждането на леки модификации.

Командата на ред 71 е ***cp***, извикана със специални аргументи. Аргументът ***-t /jails/inspircd/*** указва че подадения списък с файлове ще бъде копиран в папката ***/jails/inspircd.*** Аргументът ***-t*** позволява местата на директорията и файловете за копиране да бъдат обърнати и директорията, в която ще се копират файловете, да бъде указана преди списъка със самите файлове.

Аргументът ***--parent*** инструктира ***cp*** да пресъздаде пътищата спрямо директорията-цел, така че ако директорията-цел стане главна директория, абсолютните пътища до копията на файловете спрямо новата главна директория да същите като пътищата от старата главна директория до оригиналите на файловете. Ако ***cp*** се извика без аргумента ***--parent***, за копиране на файла ***/lib/x86_64-linux-gnu/libc.so.6*** в ***/jails/inspircd***, абсолютният път до новосъздаденото копие ще бъде

/jails/inspircd/libc.so.6. Ако обаче аргументът ***--parent*** бъде подаден, абсолютният път до новосъздаденото копие ще бъде ***/jails/inspircd/lib/x86_64-linux-gnu/libc.so.6,*** а ако поддиректорите по този път отвъд ***/jails/inspircd*** липсват, cp ще ги създаде.

Аргументът ***--preserve*** указва на ***cp*** да запази собствениците и разрешенията за достъп на копираните файлове. Ако този аргумент не бъде зададен, новосъздадените копия ще са собственост на администратора. Това е проблем, тъй като вероятно ***inspircd*** няма да работи с привилегиите на администратора, поради което може да му бъде отказан достъп до някои от файловете, ако те станат собственост на администратора. Системата за контрол на достъпа до файловете е разгледана подробно в следващата глава.

Аргументът ***-L*** инструктира ***cp*** да не копира самите символични връзки, а файловете, към които те сочат. Тази команда е необходима, тъй като в списъка със споделени библиотеки за копиране има множество символични връзки. Те съществуват, за да се улесни обновяването на библиотеките от пакетната система, но в случая администраторът се нуждае от самите файлове, към които връзките сочат.

Така например споделената библиотека ***/usr/lib/x86_64-linux-gnu/libstdc++.so.6*** в списъка на ***ldd*** е символична връзка, която в действителност сочи към ***/usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.25***. Аргументът ***-L*** ще доведе до това, че ***cp*** ще копира ***/usr/lib/x86_64-linux-gnu/libstdc++.so.6.0.25*** вместо връзката.

Аргументът ***-v*** води до това, че ***cp*** ще изведе ред с информация за всеки копиран файл и всяка създадена директория. По този начин администраторът може да проследи (редове 72-84 от разпечатката) всяко действие командата.

Най-интересното нещо в командата на ред 71 обаче е това, което следва аргументите. Там, където трябва да се намира списъка с файлове за копиране,

стои по същество командата от ред 62, заградена с обратно кавички (`). Това е възможност на обвивката, която позволява изхода на една команда да се вмъкне в съдържанието на друга команда. Когато обвивката срещне част от команда, заградена с обратни кавички, обвивката изпълнява командите в кавичките и вмъква техния изход на мястото на кавичките. Тоест

```
`ldd /usr/sbin/inspircd | cut -f 2 | sed 's/.*=> \(.*\)/\1/g'|tail +2|cut
-f 1 -d\ |xargs`
```

ще бъде заменена от

```
/usr/lib/x86_64-linux-gnu/libstdc++.so.6 /lib/x86_64-linux-
gnu/libdl.so.2 /lib/x86_64-linux-gnu/librt.so.1 /lib/x86_64-linux-
gnu/libgcc_s.so.1 /lib/x86_64-linux-gnu/libpthread.so.0 /lib/x86_64-
linux-gnu/libc.so.6 /lib/x86_64-linux-gnu/libm.so.6 /lib64/ld-linux-
x86-64.so.2
```

Последното е точно списъкът с файлове, които командата cp от ред 71 ще копира. Наблюдателните читатели сигурно са забелязали, че към последователността от команди на ред 62 е добавена командата ***xargs***, която превръща списъка с файлове от формат с по един файл на ред във формат „всичко на един ред“, при който отделните файлове са отделени с интервал - точно трансформацията, която бе нужна, за да се превърне списъка с файлове във форма, удобна за подаване на командата ***cp***.

На ред 85 администраторът също копира съдържанието на цялата папка ***/etc/inspircd*** в ***/jails/inspircd/etc/inspircd***, използвайки рекурсивното обхождане, показано по-горе в главата.

Последното действие на администратора е да копира самия изпълним файл, което се случва на ред 86.

Следващата задача, която трябва да бъде изпълнена от администратора, е пробен тест на средата, за да се определи дали има пропуски при пренасянето на файловете в нея.

```
 88 root@dragonfly:~# ls -lR /jails/inspircd/
 89 /jails/inspircd/:
 90 total 16
 91 drwxr-xr-x 3 root root 4096 Jan  7 19:32 etc
 92 drwxr-xr-x 3 root root 4096 Jul 30 19:56 lib
 93 drwxr-xr-x 2 root root 4096 May  3  2018 lib64
 94 drwxr-xr-x 4 root root 4096 Apr  3  2018 usr
 95
 96 /jails/inspircd/etc:
 97 total 4
 98 drwxrwx--- 2 irc irc 4096 Jan  7 12:38 inspircd
 99
100 /jails/inspircd/etc/inspircd:
101 total 12
102 -rw-r--r-- 1 irc irc 2838 Aug 11  2017 inspircd.conf
103 -rw-r--r-- 1 irc irc   40 Aug 11  2017 inspircd.motd
104 -rw-r--r-- 1 irc irc   31 Aug 11  2017 inspircd.rules
105
106 /jails/inspircd/lib:
107 total 4
108 drwxr-xr-x 2 root root 4096 Jan  7 19:03 x86_64-linux-gnu
109
110 /jails/inspircd/lib/x86_64-linux-gnu:
111 total 3936
112 -rwxr-xr-x 1 root root 2030544 Apr 16  2018 libc.so.6
113 -rw-r--r-- 1 root root   14560 Apr 16  2018 libdl.so.2
114 -rw-r--r-- 1 root root   96616 Jul 28 09:13 libgcc_s.so.1
115 -rw-r--r-- 1 root root 1700792 Apr 16  2018 libm.so.6
116 -rwxr-xr-x 1 root root  144976 Apr 16  2018 libpthread.so.0
117 -rw-r--r-- 1 root root   31680 Apr 16  2018 librt.so.1
118
119 /jails/inspircd/lib64:
120 total 168
121 -rwxr-xr-x 1 root root 170960 Apr 16  2018 ld-linux-x86-64.so.2
122
123 /jails/inspircd/usr:
124 total 8
125 drwxr-xr-x 3 root root 4096 Jan  6 17:35 lib
126 drwxr-xr-x 2 root root 4096 Jan  7 12:37 sbin
127
128 /jails/inspircd/usr/lib:
```

```
129 total 4
130 drwxr-xr-x 2 root root 4096 Jan  7 12:37 x86_64-linux-gnu
131
132 /jails/inspircd/usr/lib/x86_64-linux-gnu:
133 total 1560
134 -rw-r--r-- 1 root root 1594832 Jul 28 09:13 libstdc++.so.6
135
136 /jails/inspircd/usr/sbin:
137 total 840
138 -rwxr-xr-x 1 root root 858448 Aug 12  2017 inspircd
139 root@dragonfly:~# chroot --userspec irc:irc /jails/inspircd/
140 /usr/sbin/inspircd
141 ERROR: Could not open initial logfile
142 /usr/lib/inspircd/logs/startup.log: No such file or directory
143 root@dragonfly:~# cp -r -L --preserve --parents /usr/lib/inspircd/
    /jails/inspircd/
144 root@dragonfly:~# chroot --userspec irc:irc /jails/inspircd/
    /usr/sbin/inspircd
145 Inspire Internet Relay Chat Server
146 (C) InspIRCd Development Team.
147
148 Developers:
149      Brain, FrostyCoolSlug, w00t, Om, Special, peavey
150      aquanight, psychon, dz, danieldg, jackmcbarn
151      Attila
152
153 Others:               See /INFO Output
154 InspIRCd Process ID: 4316
155 Failed to write PID-file '/var/run/inspircd.pid', exiting.
156 root@dragonfly:~# mkdir -p /jails/inspircd/var/run
157 root@dragonfly:~# touch /jails/inspircd/var/run/inspircd.pid
158 root@dragonfly:~# chown irc /jails/inspircd/var/run/inspircd.pid
159 root@dragonfly:~# chroot --userspec irc:irc /jails/inspircd/
    /usr/sbin/inspircd
160 Inspire Internet Relay Chat Server
161 (C) InspIRCd Development Team.
162
163 Developers:
164      Brain, FrostyCoolSlug, w00t, Om, Special, peavey
165      aquanight, psychon, dz, danieldg, jackmcbarn
166      Attila
167
168 Others:               See /INFO Output
169 InspIRCd Process ID: 4386
170
171
```

```
172 Loading core
173 commands.....................................................
174 InspIRCd is now running as 'irc.local'[811] with 1024 max open
    sockets
```

На ред 88 администраторът издава команда, с която преглежда всички файлове в средата. Освен потвърждение на досега проведените операции, изходът на командата дава и друга важна информация. Както е видно от редове 98-104, конфигурационните файлове са собственост на потребител ***irc***, което подсказва, че ***inspircd*** е конфигуриран да работи с привилегиите на този потребител. Администраторът ще използва тази информация, когато стартира IRC сървъра по-долу.

На ред 139 с командата ***chroot --userspec irc:irc /jails/inspircd/ /usr/sbin/inspircd*** администраторът най-сетне използва ***chroot*** за да стартира ***inspircd***. Синтаксисът на командата ***chroot*** е следният:

chroot [--userspec потребител:група] нова_главна_директория [изпълним_файл]

Важно е да се отбележи, пътят до изпълнимия файл при ***chroot*** се задава спрямо новата директория. Тоест, командата на ред 139 стартира ***/jails/inspircd/usr/sbin/inspircd*** с нова главна директория - ***/jails/inspircd/***. При това програмата се стартира с привилегиите на потребител ***irc*** и група ***irc***.

Тук е моментът да бъдат вмъкнати няколко забележки. Въпреки че ***chroot*** не го изисква, ***chroot*** следва винаги да стартира процесите с привилегии на потребители, различни от администратора. В противен случай ***chroot*** е безполезен като механизъм за сигурност, тъй като администраторът или процес, работещ с привилегиите на администратора, може лесно да се измъкне

от средата, като извика отново ***chroot*** направо от програмния код и върне старата главна директория.

Друга важна особеност на ***chroot*** е това, че ако не се посочи изпълним файл, ***chroot*** ще опита да изпълни по подразбиране файла ***/bin/sh*** взет спрямо новата главна директория. Ако този файл съществува, това ще стартира нова обвивка с новата главна директория.

Командата на ред 139 не е напълно успешна, програмата стартира, но прекратява изпълнението си със съобщение за грешка (редове 141 и 142), което казва, че програмата не може да установи достъп до ***/usr/lib/inspircd/logs/startup.log*** спрямо новата главна директория.

Това се дължи на това, че ***inspircd*** разчита на наличието на папка ***/usr/lib/inspircd*** и файловете в нея, а администраторът не я е копирал в ***chroot*** средата - грешка, която администраторът коригира на ред 143.

Повторният опит за стартиране на ред 144 също не е съвсем успешен, тъй като ***inspircd*** се „оплаква“ от това, че не може да създаде файла ***/var/run/inspircd.pid*** спрямо новата главна.

Администраторът взема мерки да създаде такъв празен файл и директорията, която ще го съдържа в ***chroot*** средата. Командата на ред 158 прави новия файл собственост на потребител ***irc***, тъй като в противен случай ***inspircd***, който работи с привилегиите на същия потребител, няма да може да го достъпи за писане.

Това се оказва достатъчно и на ред 159 администраторът най-сетне успява да стартира ***inspircd*** в ***chroot*** средата.

Въпреки че целта на администратора, поставена в казуса, е постигната, тук е моментът да направим малко по-задълбочен анализ и да отговорим на въпроса „Какво сме спечелили и какво сме загубили чрез използването на ***chroot*** по този начин?”.

Отговорът на този въпрос е различен според това по какъв начин и доколко даден хакер ще успее да манипулира ***inspircd*** използвайки потенциални, може би все още неизвестни нито на администратора, нито на разработчиците, грешки в кода. Значение за

отговора на този въпрос има и това колко е сложен софтуера и от колко много външни библиотеки и инструменти зависи той, за да работи нормално.

Ако евентуален хакер успее да манипулира ***inspircd*** само да чете файлове на системата и да ги показва на хакера, хакерът има сериозен проблем. Без ***chroot*** той би имал широк достъп до информация за системата, нейните конфигурационни файлове и въобще всички файлове, които са достъпни за четене от потребителя ***irc***. С ***chroot*** хакерът би имал само достъп до файловете в ***/jails/inspircd***.

Ако даден хакер успее да манипулира ***inspircd*** да записва файлове, той би могъл да записва файлове на всички места, където потребител ***irc*** има права да записва файлове. При такъв пробив, хакерът ще може да качи допълнителни файлове в ***chroot*** средата и да използва ресурсите на системата за свои цели (например „изкопаване“ на Bitcoin). ***chroot*** средата ще ограничи такъв пробив до ***/jails/inspircd*** и ще предпази например директории като ***/tmp*** от по-нататъшни атаки, които хакерът би извършил за да получи по-големи привилегии на системата.

Освен това, имайки така ограничена среда с малък брой файлове, администраторът би открил подобни промени в средата, ако от време на време извършва проверка. Още по-добре, той може да настрои автоматична проверка чрез шел скрипт, който да проверява за новопоявили се или променени файлове в средата.

Ако хакерът успее да манипулира ***inspircd*** да изпълни файлове, тогава той би могъл да потърси ***setuid*** изпълними файлове с проблеми, които биха му позволили да разшири пробива и да стане администратор. С новопридобитите администраторски привилегии, той би се „измъкнал“ от ***chroot*** средата и би получил пълен контрол над системата. ***chroot*** средата обаче представлява препятствие пред него, тъй като прави невъзможен достъпа до такива файлове, разположени в ***/usr/sbin***, ***/sbin*** и други места на файловата система на Linux.

Ако цел на хакера обаче са проблеми в ядрото на операционната система и той намери как да ги експлоатира през ***inspircd***, ***chroot*** не може да направи нищо съществено, тъй като той ограничава само достъпа до файловете, а не до приложните интерфейси (API) на ядрото.Освен това ***chroot*** средата не винаги може да се организира по показания начин. Понякога изпълнимият файл, който трябва да бъде ограничен, изисква твърде много други изпълними файлове и външни библиотеки. При такава ситуация администраторите са изкушени да използват команди като ***mount -o bind /usr/bin /jails/inspircd/usr/bin***. Горната команда би направила всеки файл от оригиналната директория ***/usr/bin*** достъпен и в ***chroot*** средата. Такъв подход дава на хакера повече ресурси, с които да работи и намалява значително ползата от ***chroot*** като инструмент за сигурност. В много случаи обаче това е неизбежно, тъй като гореописаният подход изисква експоненциално повече работа, когато стартираната програма стане по-сложна и зависима от външни библиотеки.

Какво се губи с използването на ***chroot***?

Без съмнение най-страшната загуба е невъзможността да се използва пакетната система, за автоматично обновяване и управление на зависимости. Това е сериозна загуба, тъй като сега администраторът трябва да следи „ръчно“ дали ***inspircd*** (или използваните от нея библиотеки) имат нови пробиви в сигурността. За щастие, това може да се коригира чрез автоматично извикван скрипт, който просто да копира файловете в средата, когато оригиналите им са променени. Такъв скрипт в голяма степен би повторил командите от казуса, но все още е доста „крехък“ в ситуации, когато новата версия на софтуера започва да зависи от нова, неизползвана досега външна библиотека.

Още по-лошо, възможно е изпълнението на ограничаваната програма да започне нормално и в някакъв бъдещ момент от време да спре, изненадващо за администратора, тъй като за пръв път е възникнала ситуация, която изисква липсващ изпълним файл. Подобно развитие на ситуацията може да се предотврати само ако администраторът има повече опит с дадената програма.

В крайна сметка ***chroot*** е толкова по-добра бариера срещу хакери, колкото администраторът успее да се възползва от нейните предимства и да неутрализира недостатъците ѝ. Но ***chroot*** винаги ще има недостатъци, които произтичат най-вече от това, че той ограничава достъпа само до файловата система, а не до другите важни ресурси на компютъра.

Съществуват нови методи за изолация на критични процеси, които имат множество предимства спрямо ***chroot***. Такива са виртуализацията чрез хипервайзор (***Xen***) или леките контейнери (***docker***). Друг път за изолация е използването на SELinux или AppArmor. Демонстрацията на тези средства обаче трудно кореспондира с характера на настоящия текст, който има за цел да демонстрира основните възможности на Линукс.

7. Упражнения

Упражнение 1. Създаване на файлове и директории

Създайте във Вашата домашна директория папка ***excercise1*** и пресъздайте в нея, с помощта на командите за работа с директории, йерархията, показана на фиг. 2.1. (Тоест, ***excercise1*** ще играе ролята на главната директория и всички поддиректории, показани на фигурата, ще бъдат поддиректории на ***excercise1***.

Упражнение 2. Скрити файлове и шаблоните на обвивката

Проверете дали на Вашата система следната последователност от команди дава същите резултати:

```
 1 lalev@dragonfly:~$ mkdir test15
 2 lalev@dragonfly:~$ cd test15
 3 lalev@dragonfly:~/test15$
 4 lalev@dragonfly:~/test15$ touch .test1 .test2 test3 test4 test5.doc
 5 lalev@dragonfly:~/test15$ ls -la
 6 total 8
 7 drwxrwxr-x  2 lalev lalev 4096 Feb 13 16:12 .
 8 drwxr-xr-x 78 lalev lalev 4096 Feb 13 15:31 ..
 9 -rw-rw-r--  1 lalev lalev    0 Feb 13 16:12 .test1
10 -rw-rw-r--  1 lalev lalev    0 Feb 13 16:12 .test2
```

```
11 -rw-rw-r--  1 lalev lalev    0 Feb 13 16:12 test3
12 -rw-rw-r--  1 lalev lalev    0 Feb 13 16:12 test4
13 -rw-rw-r--  1 lalev lalev    0 Feb 13 16:12 test5.doc
14 lalev@dragonfly:~/test15$ ls -la *
15 -rw-rw-r-- 1 lalev lalev 0 Feb 13 16:12 test3
16 -rw-rw-r-- 1 lalev lalev 0 Feb 13 16:12 test4
17 -rw-rw-r-- 1 lalev lalev 0 Feb 13 16:12 test5.doc
18 lalev@dragonfly:~/test15$ ls -la ??????
19 ls: cannot access '??????': No such file or directory
20 lalev@dragonfly:~/test15$ ls -la .?????
21 -rw-rw-r-- 1 lalev lalev 0 Feb 13 16:12 .test1
22 -rw-rw-r-- 1 lalev lalev 0 Feb 13 16:12 .test2
23 lalev@dragonfly:~/test15$ ls -d .?
24 ..
25 lalev@dragonfly:~/test15$ ls -d ??
26 ls: cannot access '??': No such file or directory
27 lalev@dragonfly:~/test15$ ls -d *
28 test3  test4  test5.doc
29 lalev@dragonfly:~/test15$ ls -la test5*
30 -rw-rw-r-- 1 lalev lalev 0 Feb 13 16:12 test5.doc
31 lalev@dragonfly:~/test15$ ls -la test5?doc
32 -rw-rw-r-- 1 lalev lalev 0 Feb 13 16:12 test5.doc
33 lalev@dragonfly:~/test15$
```

Илюстрираният пример показва, че скритите файлове (тоест тези, чиито имена започват с точка), както и специалните имена . (точка) и .. (две точки) се обработват по различен начин при съпоставянето им със символите - шаблони на обвивката.

Звездата в командата ***ls -la **** на ред 14 би трябвало да се транслира в списък с всички файлове в директорията, включително . (точка) и .. (две точки). Това обаче не е така. Специалните имена и скритите файлове явно се изпускат при формиране на списъка и всъщност командата на ред 14 става ***ls -la test3 test4 test5.doc***.

Ситуацията е подобна при използването на шаблон ? (въпросителна). На ред 18, потребителят търси файлови имена с общо 6 символа дължина, като всеки от символите може да бъде произволен. Имената ***.test1*** и ***.test2*** са с дължина точно толкова символи, но не се транслират в списък и резултатът е

грешка. По подобен начин на ред 25 ***ls -la ??*** не дава никакви резултати, въпреки че специалното име .. (две точки) отговаря на шаблона.

Ситуацията се променя коренно, когато точката се постави експлицитно в началото на шаблона, както е направено на редове 20 и 23. Този път скритите файлове стават част от списъка, подаван към командата.

Командите на редове 29 и 31 показват, че точката на първа позиция в името на файла се третира по различен начин от точката в началото. Така например на ред 29 името ***test5.doc*** отговаря на шаблон ***test5****. По същия начин въпросителната в шаблона ***test5?doc*** води до включването на файла ***test5.doc*** в формирания списък.

Описаното поведение е типично за обвивката ***bash***, която е стандартна за Linux системите, което означава, че повечето читатели трябва да наблюдават точно показаното поведение. Важно е да се помни, когато се копират файлове (и особено когато за целта се използва шаблона *), че системата ще се държи точно по този начин и може да изпусне скрити файлове и директории. Понякога (например при ***git*** хранилища) това дори е желан ефект.

Ако обвивката поддържа вградената команда ***shopt***, описаното по-горе поведение на шаблоните може да бъде променено до някаква степен. Това рядко е нужно, но любознателните читатели могат да опитат да въведат команда ***shopt -s dotglob*** и да повторят командите от горния пример. Те би следвало да забележат, че скритите файлове и по-специално точката в началото вече се третират като останалите символи. Специалните имена . (точка) и .. (две точки) обаче продължават да се третират от обвивката по стария начин.

Други интересни опции на shopt са ***failglob*** и ***nullglob***. Те касаят ситуации, в която няма файлове, които да отговарят на шаблоните * и ?. Поведението по подразбиране в такъв случай е самите символи - шаблони да бъдат подадени без преобразуване като параметри на извикваните команди.

Тоест командата ***ls -la **** в директория, в която няма файлове, не се модифицира от обвивката.

```
 1 lalev@dragonfly:~$ mkdir test16
 2 lalev@dragonfly:~$ cd test16
 3 lalev@dragonfly:~/test16$ ls -la *
 4 ls: cannot access '*': No such file or directory
 5 lalev@dragonfly:~/test16$
```

Съобщението на ред 4 от горния пример идва от самата команда ***ls***, която съобщава, че файл с име '*' не съществува.

Ако опцията ***failglob*** е включена, при същата ситуация обвивката няма да стартира командата въобще.

```
 1 lalev@dragonfly:~/test16$ ls
 2 lalev@dragonfly:~/test16$ shopt -s failglob
 3 lalev@dragonfly:~/test16$ shopt | grep failglob
 4 failglob        on
 5 lalev@dragonfly:~/test16$ ls -la *
 6 bash: no match: *
 7 lalev@dragonfly:~/test16$
```

В торния пример потребителят активира ***failglob*** на ред 2, което води до ситуация при която изпълнението на ***ls -la **** в празната директория ***test16*** води до това, че обвивката прекратява командата. Съобщението на ред 6 идва именно от обвивката и гласи, че няма файлове, които да отговарят на шаблона *.

При активирана опция ***nullglob*** липсата на файлове, които да отговарят на шаблона води до това, че на извикваната команда се подава празен низ.

```
 1 lalev@dragonfly:~/test16$ shopt -u failglob
 2 lalev@dragonfly:~/test16$ shopt -s nullglob
 3 lalev@dragonfly:~/test16$ ls -la *
 4 total 8
 5 drwxrwxr-x  2 lalev lalev 4096 Feb 14 09:25 .
 6 drwxr-xr-x 79 lalev lalev 4096 Feb 15 22:32 ..
 7 lalev@dragonfly:~/test16$ ls -la ???
 8 total 8
 9 drwxrwxr-x  2 lalev lalev 4096 Feb 14 09:25 .
10 drwxr-xr-x 79 lalev lalev 4096 Feb 15 22:32 ..
```

```
11 lalev@dragonfly:~/test16$
```

На показания пример при включена опция ***nullglob*** командата на ред 3 ***ls -la **** се транслира от обвивката до ***ls -la***. Тъй като не съществуват файлове, които да отговорят на шаблона * и опцията ***nullglob*** е включена, звездата се транслира в празен низ. Същото се случва с шаблона ??? на ред 7, поради което и двете команди имат напълно еквивалентен изход.

ГЛАВА 3. КОНТРОЛ НА ДОСТЪПА ДО ФАЙЛОВЕТЕ

1. Базова система за контрол на достъпа до файловете в Линукс

Когато на дадена система работят множество потребители съществува реалната възможност един потребител да достъпи неправомерно файловете на друг потребител. Този неправомерен достъп може да се осъществи случайно или преднамерено и може да се изразява не само в прочитане на файл с поверително съдържание, но също и в изтриване или неоторизирано модифициране на чужди файлове. За да се избегне тази неприятна ситуация, всички многопотребителски операционни системи имат някаква система за контрол на достъпа до файловете.

В Линукс могат да бъдат конфигурирани няколко подобни системи за контрол на достъпа до файловете като ние ще наричаме основната от тях „базова“.

Базовата система за контрол на достъпа е достатъчно гъвкава и в същото време достатъчно сигурна и елегантна, за да може сама да осигури адекватно ниво на сигурност при много от реално инсталираните системи. Останалите механизми за контрол на достъпа до файлове в Линукс се инсталират и активират по желание и работят като надстройки на базовата система.

1.1. Собственост върху файловете и разрешения за достъп до тях

В Линукс всеки файл има собственик и група (група – собственик). Това позволява задаването на отделни права на достъп до файла за три отделни „категории“ потребители. Те са:

1. Собственик на файла

2. Членове на групата - собственик
3. Всички останали потребители

За всяка категория потребители могат да се зададат права за три вида действия с файла. (Понякога, в подходящия контекст, ние ще наричаме тези права „*разрешения за достъп*“). Те са:

1. Четене на файла
2. Запис на файла
3. Изпълнение на файла

Собственикът на файла може да променя правата за достъп на останалите групи потребители. Собственикът също може да променя коя е групата-собственик на файла при условие че новата група отговаря на определени условия.

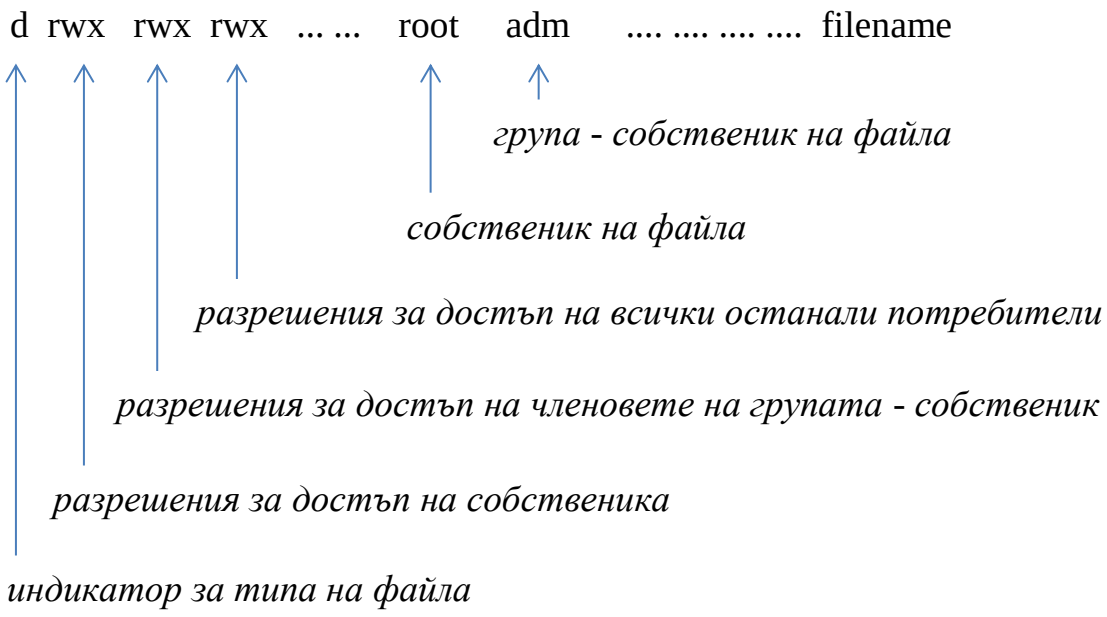
Членовете на групата собственик, както и всички останали потребители, могат да ползват файла при указания за тях режим на достъп но никога не могат да променят разрешенията за достъп до файла или неговата група.

Ограниченията, наложени от базовата система, важат за всички потребители с *едно важно изключение*. Това е администраторът на системата (потребителят ***root***), който може да осъществява достъп безусловно до всеки файл, независимо от зададените права за достъп. Администраторът също може безусловно да сменя режима на достъп, както и собственика и групата-собственик на файла.

Както бе дискутирано накратко в предната глава, собственикът, групата-собственик и правата за достъп до файла са важна информация, която се извежда в дългия формат на командата ***ls***.

```
1 root@dragonfly:~/test31# ls -l
2 total 4
3 drwxrwxrwx 2 root adm 4096 Mar 10 11:52 filename
4 root@dragonfly:~/test31#
```

В показания по-горе пример, правата за достъп до файла ***filename*** са показани в първата колона, докато собственикът и групата собственик се намират съответно в колони 3 и 4 (вж. фиг. 3.1).



Фигура 3.1. Права за достъп до файл

Първата колона съдържа 10 символа. Първият от тях е индикатор за типа на файла. Той няма отношение към системата за контрол на достъпа и бе разгледан подробно в предната глава. Останалите 9 символа всъщност са три групи от по три символа.

Първите три символа представляват правата на собственика да чете, записва и изпълнява файла. Следващата тройка символи представляват правата, които имат членовете на групата – собственик, а последните три символа се отнасят за всички останали потребители, тоест тези, които не са собственици на файла и не принадлежат към групата – собственик

Правата на собственика са зададени от първата тройка по следния начин:

Ако първият от трите символа е ***r***, собственикът има право да чете съдържанието файла. Ако вместо това на същото място има - (тире), това означава, че собственикът няма такова разрешение. Правото за четене дава възможност на потребителя например да визуализира съдържанието на документа (ако файлът съдържа документ), както и да копира файла на ново място.

Ако на втората от трите позиции има ***w***, собственикът може да осъществява запис във файла. По същия начин ако на тази позиция има тире, потребителят няма това разрешение. Правото за запис във файла позволява на потребителя да модифицира неговото съдържание.

На трета позиция има или ***x*** или тире. Наличието на ***x*** обозначава правото на собственика да изпълнява файла, докато тирето означава липсата на такова право.

Правото за изпълнение е по-особено, тъй като далеч не всички файлове във файловата система са изпълними програми. Пример за неизпълними файлове например са документите на Word. Даването на право за изпълнение на подобен върху файл няма особен смисъл и няма да предизвика неговото изпълнение, тъй като действието по изпълнение ще доведе до съобщение за грешка от обвивката или операционната система още преди да бъде изпълнен файла.

Следващите две тройки са напълно аналогични, но както бе вече споменато, съответните права за четене, запис и изпълнение се отнасят за членовете на групата-собственик и за всички останали потребители.

В конкретния пример по-горе, собственикът на файла (***root***), членовете на група ***adm*** (групата-собственик на файла), както и всички останали потребители имат всички права върху файла - четене, запис и изпълнение. В реалния свят това рядко е така, като обикновено собственикът на файла има

най-много права, членовете на групата-собственик имат по-малко права и всички останали потребители имат най-малко права. Долният пример илюстрира подобно разпределение.

```
1 root@dragonfly:~/test32# ls -la
2 total 8
3 drwxr-xr-x  2 root  root  4096 Mar 10 13:39 .
4 drwxr-xr-x 89 lalev lalev 4096 Mar 10 13:40 ..
5 -rw-r-----  1 lalev users    0 Mar 10 13:39 assessment.xlsx
6 -rw-r--r--  1 lalev lalev    0 Mar 10 13:39 notes
7 -rw-rw-r--  1 lalev users    0 Mar 10 13:39 report.docx
8 root@dragonfly:~/test32#
```

Изходът от командата ***ls -la*** показва, че файлът ***assesment.xlsx*** (ред 5) е собственост на потребител ***lalev*** и група ***users***. Потребителят ***lalev*** има права да чете и записва във файла, но не и да го изпълнява. Членовете на група ***users*** могат само да четат съдържанието на файла, докато всички останали потребители не могат нито да четат, нито да пишат, нито да изпълняват файла.

Файлът ***report.docx*** (ред 7) е собственост на същите потребител и група, но този път правата за достъп на потребителя и членовете на групата собственик са едни и същи - и двете категории потребители могат да четат и записват файла. Правата за достъп на всички останали потребители позволяват само четене на съдържанието на файла.

Файлът ***notes*** (ред 6) може би прави по-странно впечатление на начинаещите потребители, тъй като неговият собственик и група-собственик изглеждат еднакви. Те се казват ***lalev*** и всъщност са потребител и група с едно и също име. Групите с имена на потребители са специална конвенция в Линукс и се използват когато даден файл не се споделя със специална група от други потребители и следователно не са нужни специални права на групата-собственик.

1.2. Потребители и групи в Линукс

Организацията на потребителите и групите от потребители в Линукс е важна и неразделна част от системата за контрол на достъпа, поради което тук тя ще бъде разгледана по-подробно.

Читателите, запознати с глава 1, вече знаят, че всеки потребител в Линукс има свое уникално потребителско име. Докато използването на потребителското име е удобно за самите потребители и администратора на системата, важно е да се знае, че всеки потребител вътрешно получава уникален номер, известен като ***uid***[20]. Именно номерът на потребителя се записва в полето за собственик на файла в неговия ***inode*** възел.

Всеки потребител в Линукс членува в една или повече групи. Определен брой групи със стандартизирани имена се създават за нуждите на системата още при инсталиране на всяка Linux дистрибуция. Членството в тях носи определени права, като например правото за четене и запис от външни CD-ROM, DVD-ROM и BR-ROM устройства. Други групи се формират от администратора според нуждата на потребителите от достъп до файлове и ресурси. Така например ако много потребители работят по даден проект, администраторът на системата ще ги включи в група, която примерно може да се казва ***project1***. Когато даден потребител-член на групата иска да сподели файл с групата (но не и с всички останали потребители, които не са в групата), той ще зададе група-собственик на файла ***project1*** и ще даде нужните разрешения за достъп на групата-собственик. Това може да изглежда така:

```
1 lalev@dragonfly:~/test33$ ls -l
2 total 0
3 -rw-r----- 1 lalev project1 0 Mar 10 16:07 analysis.doc
4 lalev@dragonfly:~/test33$
```

[20] ***uid*** идва от “User Identifier” - “потребителски идентификатор”.

Също като при потребителите, освен уникални имена, които се използват за удобство от потребителите и администратора, групите също така имат и уникални номера. Именно уникалният номер се поставя в ***inode*** структурата на файла, за да обозначи, че групата с този номер е група-собственик на файла.

Всеки потребител има една главна група, която става автоматично група-собственик на всички нови файлове, създавани от потребителя. Командата ***id*** може да бъде използвана за показване на ***uid*** на потребителя, неговата главна група, както и всички други, „допълнителни“ групи, в които потребителят членува.

```
1 lalev@dragonfly:~$ id
2 uid=1000(lalev) gid=1000(lalev)
3 groups=1000(lalev),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd),114(l
  lalev@dragonfly:~$
4
```

Първият ред от изхода на командата (ред 2 на разпечатката) показва, че потребител ***lalev*** има ***uid*** 1000, а неговата главна група се казва ***lalev*** и има уникален номер също 1000. Уникалният номер на главната група на потребителя се нарича ***gid***[21].

Вторият ред от изхода на командата представлява списък от номерата и имената на всички групи, в които членува потребителя. За потребителя ***lalev*** някои от тези групи са ***adm***, ***cdrom*** и т.н.

Съответствията между потребителското име и неговия номер се съхраняват във файла ***/etc/password***. Този файл също съхранява номера на главната група на всеки потребител, както и друга важна информация за потребителите на системата. По-долния пример показва част от съдържанието на този файл:

```
1 lalev@dragonfly:~$ (cat /etc/passwd | head -10) && (cat /etc/passwd
2 | grep lalev)
```

[21] ***gid*** идва от „Group Identifier” - “идентификатор на група”.

```
 3 root:x:0:0:root:/root:/bin/bash
 4 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
 5 bin:x:2:2:bin:/bin:/usr/sbin/nologin
 6 sys:x:3:3:sys:/dev:/usr/sbin/nologin
 7 sync:x:4:65534:sync:/bin:/bin/sync
 8 games:x:5:60:games:/usr/games:/usr/sbin/nologin
 9 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
10 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
11 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
12 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
13 lalev:x:1000:1000:Angelin Lalev,,,:/home/lalev:/bin/bash
14 lalev@dragonfly:~$
```

Командите в примера използват филтри, които са разгледани в следващите глави, но по същество показват части от файла ***/etc/passwd***, тъй като неговата дължина е доста голяма и би увеличила излишно редовете в разпечатката.

Редовете в ***/etc/passwd*** са разделени на полета, отграничени едно от друго с двоеточие. Първото поле е името на потребителя, докато третото и четвъртото поле са съответно неговият номер и номера на основната група, в която потребителят членува.

Съответствията между номерата и имената на групите се съхраняват във файла ***/etc/group***. Долният пример показва малък брой редове от този файл.

```
 1 lalev@dragonfly:~$ (cat /etc/group | head -12) && (cat /etc/group |
 2 grep ^lalev)
 3 root:x:0:
 4 daemon:x:1:
 5 bin:x:2:
 6 sys:x:3:
 7 adm:x:4:syslog,lalev
 8 tty:x:5:
 9 disk:x:6:backup
10 lp:x:7:
11 mail:x:8:
12 news:x:9:
13 uucp:x:10:
14 man:x:12:
15 lalev:x:1000:
16 lalev@dragonfly:~$
```

Форматът на редовете в ***/etc/group*** е подобен на този на ***/etc/passwd***. Двоеточието отново е разделител между отделните полета, като първото поле е името на групата, докато третото поле е нейния уникален номер.

От ***/etc/passwd***, показан по-горе, можем да видим, че потребител ***lalev*** има ***uid*** 1000 и ***gid*** 1000. С други думи уникалният номер на потребителя е 1000, а номерът на основната му група също е 1000. От ***/etc/group*** можем да видим, че група с номер 1000 всъщност се казва ***lalev***.

Четвъртото поле на ***/etc/group*** служи за съхранение на информацията за допълнителните групи на всеки потребител. Ако даден потребител членува в дадена група и тази група не е основната му група, неговото име се появява в това поле в реда на групата. Така от горната разпечатка можем да заключим, че потребителите ***lalev*** и ***syslog*** членуват в група ***adm***, като ***adm*** не е основна група за никой от двамата потребители.

Ако дадена група е основна за даден потребител, тази информация се съхранява в ***/etc/passwd*** и името на потребителя не присъства в четвъртото поле от реда на групата в ***/etc/group***. Така например от горните разпечатки можем да видим, че потребителят ***lalev*** има за главна група групата ***lalev***.
Тъй като групата ***lalev*** е основна за потребителя ***lalev***, неговото име липсва в четвъртото поле на ред 15 от горната разпечатка на ***/etc/group***.

Линукс по конвенция се създава по една нова група за всеки нов потребител. Тази нова група носи същото име като това на потребителя и има само един член - самият потребител. Освен това, тази група най-често е основната група на новосъздадения потребител. При това положение, всеки файл, който потребителят създава, получава по подразбиране за група-собственик групата на потребителя. Долният пример илюстрира това поведение:

```
1 lalev@dragonfly:~/test34$ id
```

```
2 uid=1000(lalev) gid=1000(lalev)
  groups=1000(lalev),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lxd),114(l
  lalev@dragonfly:~/test34$ touch newfile
3 lalev@dragonfly:~/test34$ ls -l newfile
4 -rw-rw-r-- 1 lalev lalev 0 Apr  8 11:59 newfile
5
```

На примера, потребител ***lalev*** има за главна група групата ***lalev***, която съдържа само него. Тъй като ***lalev*** е главна група за потребителя, по подразбиране всеки файл, който потребителя създаде, ще получи за група-собственик именно нея, което е видно от ред 5 на разпечатката.

Илюстрираното поведение на практика означава, че новите файлове по подразбиране не се споделят с никоя специална група потребители, която да има по-малко права от собственика, но повече права от всички останали. Това е така, тъй като за потребителя ***lalev*** ще се приложат правата на собственика, а не правата на групата ***lalev***, независимо че той членува в нея.

Използването на подобни групи от един потребител за новите файлове е добър избор по подразбиране, тъй като повечето файлове на потребителите не се използват в групов режим. Ако даден потребител желае да даде повече права върху даден файл на дадена специална група потребители, то той ще смени групата-собственик на файла по подходящ начин след като веднъж файлът е създаден.

1.3. Команди за смяна на собственика и групата на даден файл

За смяна на собственика и групата на даден файл в Линукс се използват съответно командите ***chown***[22] и ***chgrp***[23]. Командата ***chown*** може да сменя

[22] Съкратено от англ. „change owner” - „смени собственика“.
[23] Съкратено от англ. „change group” - „смени група“

потребителя и групата на даден файл, докато ***chgrp*** се използва специфично за промяна на групата.

Синтаксисът на ***chown*** е:

chown *[параметри] [потребител]:[група] файл/файлове*

По-долу са посочени няколко примера за използването на ***chown***:

```
 1 root@dragonfly:/opt/example3# ls -lad alpha
 2 drwxrwxr-x 2 root root 4096 Nov  4 13:49 alpha
 3 root@dragonfly:/opt/example3# chown lalev alpha
 4 root@dragonfly:/opt/example3# ls -lad alpha
 5 drwxrwxr-x 2 lalev root 4096 Nov  4 13:49 alpha
 6 root@dragonfly:/opt/example3# chown :lalev alpha
 7 root@dragonfly:/opt/example3# ls -lad alpha
 8 drwxrwxr-x 2 lalev lalev 4096 Nov  4 13:49 alpha
 9 root@dragonfly:/opt/example3# chown root:root alpha
10 root@dragonfly:/opt/example3# ls -lad alpha
11 drwxrwxr-x 2 root root 4096 Nov  4 13:49 alpha
12 root@dragonfly:/opt/example3#
```

Редове 1 и 2 на горната разпечатка показват, че в ***/opt/example*** съществува поддиректория ***alpha***, собственост на потребител ***root*** и група ***root***. От промпта също е видно, че потребителят в момента работи като администратор.

На ред 3 администраторът изпълнява командата ***chown lalev alpha***, която води до това, че директорията ***alpha*** става собственост на потребител ***lalev***. В този вариант на командата не е указана група, което означава, че групата-собственик на файла остава непроменена. Редове 4 и 5 показват именно това.

На ред 6 е показан вариант, който сменя само групата и не променя потребителя - собственик на файла. ***chown*** очаква потребителя и групата да бъдат предоставени във формат *потребител:група*. Когато липсва група,

разделящото двоеточие може да бъде изпуснато, но когато липсва потребител, то трябва да бъде поставено пред името на групата.

На ред 9 пък е показан пример за най-типичната употреба на командата, при която се задават едновременно потребител и група. Редът е интересен и като илюстрация за това как администраторът не се влияе от ограниченията на системата за достъп до файловете. Нормално един потребител не може да сменя собствеността на файловете, които са собственост на друг потребител. В противен случай нищо нямаше да спира потребителите да "крадат" файлове един от друг и да заобикалят по този начин системата за контрол на достъпа. На ред 9 се случва обаче точно това - потребителят ***root*** променя собствеността на файловете, които принадлежат на друг потребител. Това е възможно именно защото администраторът не подлежи на ограниченията, наложени от системата за контрол на достъпа.

Добавянето на параметър ***-R*** или ***--recursive*** указва, че промяната на собствеността ще засегне не само указаната директория, но и съдържанието на поддиректориите, техните поддиректории и т.н. до обхождане на цялото разклонение от дървото на директориите, което се намира „под" указаната директория.

Особено интересен е параметърът ***--from*** който позволява задаването на текущ собственик и текуща група. Ако е използван този параметър, смяната на собственика и/или групата на файла ще се осъществи само за файловете, които имат указания текущ собственик и/или група. Това е бърз и автоматизиран начин за администратора да прехвърли например всички файлове на ***dragan*** на потребител ***ivan***. Поради различни причини от реалния свят, подобна операция се налага доволно често, поради което тук тя заслужено е илюстрирана с пример:

```
1  root@dragonfly:/opt/example3# ls -la
```

```
 2 total 20
 3 drwxrwxr-x 5 root  root  4096 Nov  5 09:24 .
 4 drwxr-xr-x 4 root  root  4096 Nov  1 14:12 ..
 5 drwxrwxr-x 2 lalev lalev 4096 Nov  4 13:49 alpha
 6 drwxrwxr-x 2 root  root  4096 Nov  1 14:09 beta
 7 drwxrwxr-x 2 root  root  4096 Nov  1 14:09 gamma
 8 root@dragonfly:/opt/example3# chown --from lalev:lalev www-
   data:www-data *
 9 root@dragonfly:/opt/example3# ls -la
10 total 20
11 drwxrwxr-x 5 root     root     4096 Nov  5 09:24 .
12 drwxr-xr-x 4 root     root     4096 Nov  1 14:12 ..
13 drwxrwxr-x 2 www-data www-data 4096 Nov  4 13:49 alpha
14 drwxrwxr-x 2 root     root     4096 Nov  1 14:09 beta
15 drwxrwxr-x 2 root     root     4096 Nov  1 14:09 gamma
16 root@dragonfly:/opt/example3#
```

Още един типичен сценарий, при който може да се наложи използването на параметъра ***--from*** касае ситуация, при която собственикът или групата-собственик на файла са били изтрити от файловете ***/etc/passwd*** и/или ***/etc/group***. Долният пример показва изхода от командата ***ls*** в този случай, както и командата за смяна на собственика:

```
 1 root@dragonfly:~/test34# ls -l
 2 total 0
 3 -rw-rw-r-- 1 1001  1001 0 Apr  8 18:37 alpha
 4 -rw-rw-r-- 1 1001 staff 0 Apr  8 18:37 beta
 5 root@dragonfly:~/test34# chown --from 1001 lalev *
 6 root@dragonfly:~/test34# ls -l
 7 total 0
 8 -rw-rw-r-- 1 lalev  1001 0 Apr  8 18:37 alpha
 9 -rw-rw-r-- 1 lalev staff 0 Apr  8 18:37 beta
10 root@dragonfly:~/test34# chown --from :1001 :lalev *
11 root@dragonfly:~/test34# ls -l
12 total 0
13 -rw-rw-r-- 1 lalev lalev 0 Apr  8 18:37 alpha
14 -rw-rw-r-- 1 lalev staff 0 Apr  8 18:37 beta
15 root@dragonfly:~/test34#
```

Тъй като в ***/etc/passwd*** и ***/etc/group*** няма редове за потребител и група с номера 1001, на съответните позиции в разпечатката на ***ls*** са показани самите номера. Това не е проблем, тъй като ***chown*** може да приема за параметри както

имена на потребители и групи, така и съответстващите им номера. При това те са напълно взаимозаменяеми. Навсякъде където трябва да се подаде име на потребител може да се подаде номер и обратното.

Използването на ***chown*** с номер вместо с име на потребител е демонстрирано на редове 5 и 10.

Като последен пример за употребата на ***chown*** е даден случай на използването му със списък файлове.

```
 1 root@dragonfly:/opt/example3# ls -la
 2 total 20
 3 drwxrwxr-x 5 root     root     4096 Nov  5 09:24 .
 4 drwxr-xr-x 4 root     root     4096 Nov  1 14:12 ..
 5 drwxrwxr-x 2 www-data www-data 4096 Nov  4 13:49 alpha
 6 drwxrwxr-x 2 www-data www-data 4096 Nov  1 14:09 beta
 7 drwxrwxr-x 2 root     root     4096 Nov  1 14:09 gamma
 8 root@dragonfly:/opt/example3# chown root:root alpha beta
 9 root@dragonfly:/opt/example3# ls -la
10 total 20
11 drwxrwxr-x 5 root root 4096 Nov  5 09:24 .
12 drwxr-xr-x 4 root root 4096 Nov  1 14:12 ..
13 drwxrwxr-x 2 root root 4096 Nov  4 13:49 alpha
14 drwxrwxr-x 2 root root 4096 Nov  1 14:09 beta
15 drwxrwxr-x 2 root root 4096 Nov  1 14:09 gamma
16 root@dragonfly:/opt/example3#
```

Поддържането на подаването на списък с файлове е важно, тъй като това означава, че ***chown*** може да работи с шаблони на обвивката като . (точка) и * (звезда). Подобна употреба, впрочем, бе илюстрирана в по-предните примери.

Синтаксисът на ***chgrp*** е подобен на този на ***chown***:

chgrp *[параметри] група файл/файлове*

По долу са дадени примери за употребата на ***chgrp***:

```
 1 root@dragonfly:/opt/example3# ls -la
 2 total 20
 3 drwxrwxr-x 5 root root 4096 Nov  5 09:24 .
 4 drwxr-xr-x 4 root root 4096 Nov  1 14:12 ..
 5 drwxrwxr-x 2 root root 4096 Nov  4 13:49 alpha
```

```
 6 drwxrwxr-x 2 root root 4096 Nov  1 14:09 beta
 7 drwxrwxr-x 2 root root 4096 Nov  1 14:09 gamma
 8 root@dragonfly:/opt/example3# chgrp -R www-data alpha
 9 root@dragonfly:/opt/example3# ls -la
10 total 20
11 drwxrwxr-x 5 root root     4096 Nov  5 09:24 .
12 drwxr-xr-x 4 root root     4096 Nov  1 14:12 ..
13 drwxrwxr-x 2 root www-data 4096 Nov  4 13:49 alpha
14 drwxrwxr-x 2 root root     4096 Nov  1 14:09 beta
15 drwxrwxr-x 2 root root     4096 Nov  1 14:09 gamma
16 root@dragonfly:/opt/example3# chgrp -v -R root alpha
17 changed group of 'alpha/test2' from www-data to root
18 changed group of 'alpha/test3' from www-data to root
19 changed group of 'alpha/test' from www-data to root
20 changed group of 'alpha/test4' from www-data to root
21 changed group of 'alpha' from www-data to root
22 root@dragonfly:/opt/example3# chgrp www-data alpha/test3
23 root@dragonfly:/opt/example3# chgrp -c -R root alpha
24 changed group of 'alpha/test3' from www-data to root
25 root@dragonfly:/opt/example3#
```

На ред 8 е демонстриран сравнително стандартен начин на употреба на ***chgrp*** върху директория. Параметърът ***-R***, също като при ***chown***, изпълнява операцията рекурсивно, върху цялото съдържание на посочените директории, включително на поддиректориите и тяхното съдържание.

Понякога е полезно да бъдат изведени имената на файловете, които се обработват. В такъв случай на помощ идват два параметъра, които също се поддържат и от ***chown***:

Параметърът ***-v*** (идващ от думата “verbose” - подробно) прави така, че за всеки файл, който бива обработван от ***chgrp***, се извежда по един ред. Такива редове на примера са 17 до 21. Ред 20 например казва, че за файл ***alpha/test4*** е сменена групата-собственик от ***www-data*** на ***root***.

Параметърът ***-c***, илюстриран на редове 23-24 е подобен на ***-v*** с тази разлика, че при него се извежда информация само за файловете, чиято група реално се променя като резултат от извикването на командата.

Много важен въпрос, който не може да бъде подминат, е това доколко потребител, различен от ***root***, може да манипулира собствеността на файлове. Както беше вече изтъкнато, такъв потребител не може да променя собствеността на файлове, които не са негови, по никакъв начин. Да приемем обаче, че става дума за файлове, които са собственост на потребителя. Доколко той може да промени тази собственост, като промени собственика и/или групата собственик на файла?

По отношение на собственика на файла, нещата стоят по същия начин. При създаване на нов файл, неговият собственик става автоматично потребителят, който го е създал. Различен от ***root*** потребител не може да промени собственика на файла, така че новият собственик да е друг потребител. Това е направено, за да се избегне ситуация, в която потребителят създава и „подхвърля“ файл със зловредно съдържание на друг потребител.

Потребител, различен от ***root***, обаче може да смени групата на файл, който е негова собственост. Този процес има важни ограничения, които предотвратяват подобно на горното „подхвърляне“ на файлове. Потребителят може да смени групата-собственик на файл само с група-собственик, в която потребителят членува. Долният пример илюстрира тази ситуация:

```
 1 lalev@dragonfly:/opt/example3$ id
 2 uid=1000(lalev) gid=1000(lalev)
   groups=1000(lalev),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),
   110(lxd),114(lpadmin),115(sambashare)
 3 lalev@dragonfly:/opt/example3$ ls -la
 4 total 20
 5 drwxrwxr-x 5 root  root   4096 Nov  5 09:24 .
 6 drwxr-xr-x 4 root  root   4096 Nov  1 14:12 ..
 7 drwxrwxr-x 2 lalev daemon 4096 Nov  9 15:03 alpha
 8 drwxrwxr-x 2 root  root   4096 Nov  1 14:09 beta
 9 drwxrwxr-x 2 root  root   4096 Nov  1 14:09 gamma
10 lalev@dragonfly:/opt/example3$ chown root alpha
11 chown: changing ownership of 'alpha': Operation not permitted
12 lalev@dragonfly:/opt/example3$ chgrp adm alpha
13 lalev@dragonfly:/opt/example3$ chgrp daemon alpha
```

```
14 chgrp: changing group of 'alpha': Operation not permitted
15 lalev@dragonfly:/opt/example3$ chgrp lalev alpha
16 lalev@dragonfly:/opt/example3$
```

Директория ***alpha*** този път има потребител-собственик ***lalev*** и група-собственик ***daemon*** (ред 7). Това е доста странна ситуация, тъй като потребителят ***lalev*** не е член на ***daemon***. До нея може да се стигне само ако администраторът на системата е задал групата-собственик или ако група ***daemon*** има парола за достъп, която потребителят знае и използва заедно с командата ***newgrp***.

На ред 10 е демонстрирано, че потребител, различен от ***root***, не може просто да смени собственика на файла, дори да има пълни права върху този файл. На реда потребител ***lalev*** се опитва да смени собственика на ***alpha***. Самият файл е собственост на същия потребител, а новият собственик е потребителят ***root***. Поради споменатото ограничение, разбира се, промяната не се осъществява и води до извеждане на съобщение за грешка.

На ред 12 потребителят ***lalev*** променя групата на ***alpha***. Тъй като ***lalev*** е собственик на файла и в същото време ***lalev*** се числи към новата група-собственик (а именно ***adm***), този път операцията е успешна.

На ред 13 потребителят ***lalev*** се опитва да върне групата-собственик на същия файл на началната ѝ стойност - ***daemon***. Това води до грешка, тъй като ***lalev*** не е член на групата ***daemon***. При тази ситуация само администраторът на системата би могъл да върне групата-собственик на файла на ***daemon***.

Също като при ***chown***, при ***chgrp*** името на групата може да бъде заместено съответно от нейния номер. По долу е показан подобен пример:

```
 1 lalev@dragonfly:~/example4$ ls -l
 2 drwxrwxr-x  2 lalev lalev 4096 May  5 07:08 alpha
 3 drwxrwxr-x  2 lalev lalev 4096 May  5 07:08 beta
 4 drwxrwxr-x  2 lalev lalev 4096 May  5 07:08 gamma
 5 lalev@dragonfly:~/example4$ ls -ln
 6 drwxrwxr-x  2 1000 1000 4096 May  5 07:08 alpha
```

```
 7 drwxrwxr-x  2 1000 1000 4096 May  5 07:08 beta
 8 drwxrwxr-x  2 1000 1000 4096 May  5 07:08 gamma
 9 lalev@dragonfly:~/example4$ id
10 uid=1000(lalev) gid=1000(lalev)
11 groups=1000(lalev),4(adm),24(cdrom),27(sudo),30(dip),
12 46(plugdev),110(lxd),114(lpadmin),115(sambashare),135(project1)
13 lalev@dragonfly:~/example4$ chgrp 4 alpha
14 lalev@dragonfly:~/example4$ ls -ln
15 drwxrwxr-x  2 1000    4 4096 May  5 07:08 alpha
   drwxrwxr-x  2 1000 1000 4096 May  5 07:08 beta
   drwxrwxr-x  2 1000 1000 4096 May  5 07:08 gamma
16 lalev@dragonfly:~/example4$
```

На ред 1 потребителят извежда съдържанието на директорията в дълъг формат, използвайки повече или по-малко стандартния параметър ***-l***. На ред 5 обаче, потребителят задава и параметъра ***-n***, който модифицира изхода на ***ls***, така че вместо имената, в него са показани номерата на потребителя и групата-собственик.

Командата на ред 10 показва ***uid*** и ***gid*** на потребителя, както и допълнителните групи, в които потребителят членува, заедно с техните номера. Това прави по-лесно да се разбере смисъла на командата на ред 13, която използва номера на група ***adm*** за указване на новата група-собственик на директория ***alpha***.

Както трябва да бъде видно от командата ***ls*** на ред 14, това работи без проблеми и новата група собственик на ***alpha*** е наистина ***adm***.

Използването на номерата вместо имената на потребителите и групите е необичайна практика, която се налага в някои редки случаи, един от които бе дискутиран по-горе. В над 99% от случаите обаче, администраторите работят с имената, тъй като те са значително по-удобни.

1.4. Команди за задаване права за достъп до файловете

Изложението до момента най-подробно обясни какви права за достъп могат да се зададат чрез базовата система, как се задават собственика и групата-собственик на файловете, как са организирани самите групи на една Линукс система и дори - как вътрешно файловата система записва данните за собствеността върху файловете. Последният основен елемент, който трябва да бъде представен, е как се задават самите права за достъп. Това се случва с помощта на командата ***chmod***, която има следния синтаксис:

chmod *[параметри] права файл/файлове*

Правата за достъп при ***chmod*** могат да се зададат в две форми. Първата форма е разчита на символи за да представи правата, докато втората представя правата като числа в осмична бройна система. Първата форма е далеч по-често използваната, но представянето на правата като осмични числа има специфично приложение, така че добре подготвеният администратор трябва да познава и двете.

„Символната“ форма се състои от три части:

1. Символи - индикатори за това чии права се задават;

Символите индикатори могат да бъдат ***u***, ***g*** и ***o*** и съответно обозначават правата на потребителя[24], правата на групата[25] и правата на останалите потребители[26].

2. Знак за равенство, плюс или минус;

Знак плюс означава, че към съществуващите права се добавят нови. Знак минус означава, че от съществуващите права се премахват указани права. Знак за

[24] “***u***” идва от „***user***”, т.е. „потребител“.
[25] “***g***” идва от „***group***”, т.е. „група”.
[26] “***o***” идва от „***other***” т.е. „други”.

равенство означава, че съществуващите права се отменят и се задават нови на тяхно място.

3. Символи - индикатори за правата, които се задават;

Символите - индикатори за права биват основно три. ***r*** означава достъп за четене, ***w*** означава достъп за писане, ***x*** означава достъп за изпълнение.

По долу са дадени примери за използването на ***chown*** с права, зададени по различен начин:

```
 1 lalev@dragonfly:~/modes$ ls -la
 2 total 8
 3 drwxrwxr-x  2 lalev lalev 4096 Nov 10 09:10 .
 4 drwxr-xr-x 37 lalev lalev 4096 Nov 10 09:09 ..
 5 -rw-rw-r--  1 lalev lalev    0 Nov 10 09:10 file1
 6 -rw-rw-r--  1 lalev lalev    0 Nov 10 09:10 file2
 7 -rw-rw-r--  1 lalev lalev    0 Nov 10 09:10 file3
 8 -rw-rw-r--  1 lalev lalev    0 Nov 10 09:10 file4
 9 -rw-rw-r--  1 lalev lalev    0 Nov 10 09:10 file5
10 -rw-rw-r--  1 lalev lalev    0 Nov 10 09:10 file6
11 lalev@dragonfly:~/modes$ chmod ugo=r file1
12 lalev@dragonfly:~/modes$ ls -la file1
13 -r--r--r-- 1 lalev lalev 0 Nov 10 09:10 file1
14 lalev@dragonfly:~/modes$ chmod ug+w file1
15 lalev@dragonfly:~/modes$ ls -la file1
16 -rw-rw-r-- 1 lalev lalev 0 Nov 10 09:10 file1
17 lalev@dragonfly:~/modes$ chmod o-r file1
18 lalev@dragonfly:~/modes$ ls -la file1
19 -rw-rw---- 1 lalev lalev 0 Nov 10 09:10 file1
20 lalev@dragonfly:~/modes$
```

Изходът от командата ***ls*** на върха на разпечатката показва, че файлът ***file1*** има права за достъп ***rw-rw-r--***. Това означава, че собственикът (***lalev***) може да чете и записва файла. Членовете на групата ***lalev*** имат същите права, а всички останали потребители имат права само за четене.

Командата на ред 11 задава напълно нови права за достъп до файла ***file1,*** тъй като в символния низ с новите права се използва знак за равенство. ***r*** означава, че новозададените права позволяват единствено четене, а ***ugo*** означава, че се коригират едновременно правата и на трите категории

потребители - потребител, група и всички останали. Резултатът, както показват редове 12 и 13, са права ***r--r--r--***, тоест, всички категории потребители сега могат само да четат файла.

Командата на ред 14 добавя права за писане към съществуващите права на потребителя и групата, което води до права ***rw-rw-r--.*** А на ред 17 потребителят премахва правата за четене на всички останали потребители. Получените по този начин права ***rw-rw----*** означават, че освен членовете на групата ***lalev*** и потребителя ***lalev***, никой няма да има достъп до файла.

chmod дори приема списък от символни форми, които указват различни права за различните групи. Това е показано на следващия пример:

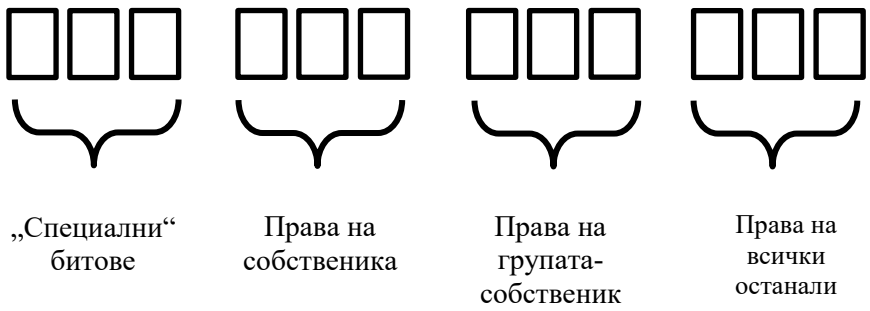
```
 1 lalev@dragonfly:~/modes$ ls -la
 2 total 8
 3 drwxrwxr-x  2 lalev lalev 4096 Nov 10 09:10 .
 4 drwxr-xr-x 38 lalev lalev 4096 Nov 20 16:43 ..
 5 -rwxr-xr-x  1 lalev lalev    0 Nov 10 09:10 file1
 6 -rwxr-xr-x  1 lalev lalev    0 Nov 10 09:10 file2
 7 -rwxr-xr-x  1 lalev lalev    0 Nov 10 09:10 file3
 8 -rwxr-xr-x  1 lalev lalev    0 Nov 10 09:10 file4
 9 -rwxr-xr-x  1 lalev lalev    0 Nov 10 09:10 file5
10 -rwxr-xr-x  1 lalev lalev    0 Nov 10 09:10 file6
11 lalev@dragonfly:~/modes$ chmod u-w,g-x,g+w,o=r *
12 lalev@dragonfly:~/modes$ ls -la
13 total 8
14 drwxrwxr-x  2 lalev lalev 4096 Nov 10 09:10 .
15 drwxr-xr-x 38 lalev lalev 4096 Nov 20 16:43 ..
16 -r-xrw-r--  1 lalev lalev    0 Nov 10 09:10 file1
17 -r-xrw-r--  1 lalev lalev    0 Nov 10 09:10 file2
18 -r-xrw-r--  1 lalev lalev    0 Nov 10 09:10 file3
19 -r-xrw-r--  1 lalev lalev    0 Nov 10 09:10 file4
20 -r-xrw-r--  1 lalev lalev    0 Nov 10 09:10 file5
   -r-xrw-r--  1 lalev lalev    0 Nov 10 09:10 file6
   lalev@dragonfly:~/modes$
```

В примера, на ред 11, на командата ***chmod*** са зададени четири инструкции във символна форма, които се изпълняват по реда на задаването си.

1.5. Задаване на права за достъп в осмична бройна система

Както бе вече споменато, правата за достъп могат да се представят и като числа в осмична бройна система. Това представяне всъщност е по-близко до това как тези права се съхраняват във файловата система.

В ***inode*** структурата на всеки файл има поле с дължина 12 бита (т.е. 12 двоични цифри), което съхранява правата за достъп до файла. Вж. фиг. 3.2.



Фигура 3.2. Битове за правата за достъп до файла

В тази структура за всяка от категориите потребители на файла са заделени три бита, представляващи съответно правата за четене, запис и изпълнение на съответната категория. Битът е единица, ако съответното право е гарантирано на категорията потребители, и нула - ако не е. Така права ***rwx*** се представят с последователност от битове 111, докато права ***r-x*** и ***rw-*** са съответно 101 и 110. В таблица 3.1 са дадени всички възможни комбинации от три бита и тяхното представяне като осмични числа.

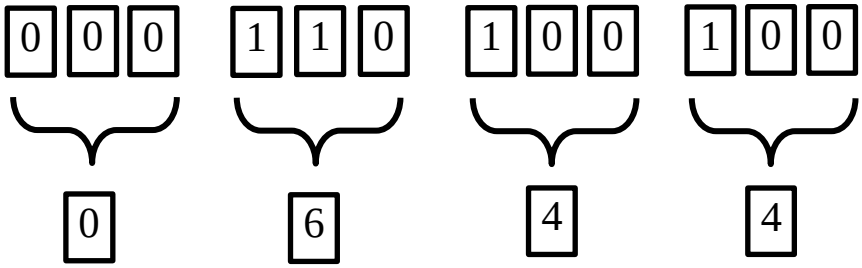
Права в „символна“ форма	**Права, представени като битове**	**Права, представени като осмично число**
---	000	0
--x	001	1
-w-	010	2
-wx	011	3

r--	100	4
r-x	101	5
rw-	110	6
rwx	111	7

Таблица 3.1. Битове и права

Прави впечатление, тъй като една осмична цифра се представя в компютъра от точно три двоични, че всяка възможна осмична цифра отговаря на дадена комбинация от права. Нещо повече, тези осмични цифри, които имат само една единица в двоичното си представяне (тоест осмичните цифри 1,2 и 4) отговарят точно на едно право (1 - само изпълнение, 2 - само запис, 4 - само четене), а всички останали цифри могат да се представят като сума от тях. Така например 6 (права ***rw-***) е равно на 4 (права ***r--***) плюс 2 (права ***-w-***). Това не е някакво специално свойство, а е просто следствие от това, че използваме отделните битове за представяне на всяко едно от правата.

Тъй като една осмична цифра се представя от точно три двоични, дванадесетте бита могат да се представят с четири осмични цифри, като по този начин се формира число, представено в осмична бройна система, което отговаря на правата за достъп до файла. Превръщането е директно, като вместо всяка от тройките битове се записва съответстващата осмична цифра.



Фигура 3.3. Права в осмична бройна система

В долната таблица са дадени примери за подобни последователности.

Права за достъп	Права за достъп в двоична бройна система	Права за достъп в осмична бройна система
rwxrwxrwx	000 111 111 111	0777
rwxr-xr-x	000 111 101 101	0755
rw-------	000 110 000 000	0600
---rw---x	000 000 110 001	0061

Таблица 3.2. Права за достъп в „символна“ форма, в двоична форма и права, представени като числа в осмична бройна система.

Трябва да се отбележи, че докато Линукс приема задаването на права в „символна“ и „осмична“ форма, няма начин правата да се задават директно в двоична бройна система.

По долу е даден пример за използването на права в „осмична“ форма с ***chmod***.

```
 1 lalev@dragonfly:~/modes$ ls -la
 2 total 8
 3 drwxrwxr-x  2 lalev lalev 4096 Nov 10 09:10 .
 4 drwxr-xr-x 38 lalev lalev 4096 Nov 20 16:43 ..
 5 -rw-rw----  1 lalev lalev    0 Nov 10 09:10 file1
 6 -rw-rw-r--  1 lalev lalev    0 Nov 10 09:10 file2
 7 -rw-rw-r--  1 lalev lalev    0 Nov 10 09:10 file3
 8 -rw-rw-r--  1 lalev lalev    0 Nov 10 09:10 file4
 9 -rw-rw-r--  1 lalev lalev    0 Nov 10 09:10 file5
10 -rw-rw-r--  1 lalev lalev    0 Nov 10 09:10 file6
11 lalev@dragonfly:~/modes$ chmod 777 file1
12 lalev@dragonfly:~/modes$ chmod 0700 file2
13 lalev@dragonfly:~/modes$ chmod 300 file3
14 lalev@dragonfly:~/modes$ ls -la
15 total 8
16 drwxrwxr-x  2 lalev lalev 4096 Nov 10 09:10 .
17 drwxr-xr-x 38 lalev lalev 4096 Nov 20 16:43 ..
18 -rwxrwxrwx  1 lalev lalev    0 Nov 10 09:10 file1
19 -rwx------  1 lalev lalev    0 Nov 10 09:10 file2
```

```
20 --wx------  1 lalev lalev    0 Nov 10 09:10 file3
21 -rw-rw-r--  1 lalev lalev    0 Nov 10 09:10 file4
22 -rw-rw-r--  1 lalev lalev    0 Nov 10 09:10 file5
23 -rw-rw-r--  1 lalev lalev    0 Nov 10 09:10 file6
24 lalev@dragonfly:~/modes$ chmod -222 *
25 lalev@dragonfly:~/modes$ ls -la
26 total 8
27 drwxrwxr-x  2 lalev lalev 4096 Nov 10 09:10 .
28 drwxr-xr-x 38 lalev lalev 4096 Nov 20 16:43 ..
29 -r-xr-xr-x  1 lalev lalev    0 Nov 10 09:10 file1
30 -r-xr--r--  1 lalev lalev    0 Nov 10 09:10 file2
31 -r-xr--r--  1 lalev lalev    0 Nov 10 09:10 file3
32 -r--r--r--  1 lalev lalev    0 Nov 10 09:10 file4
33 -r--r--r--  1 lalev lalev    0 Nov 10 09:10 file5
34 -r--r--r--  1 lalev lalev    0 Nov 10 09:10 file6
35 lalev@dragonfly:~/modes$
```

На ред 11 потребителят задава права за четене, запис и изпълнение на всички категории за файл ***file1***. Важно е да се отбележи, че вместо 0777 потребителят използва 777, тъй като изпуснатата водеща цифра 0 не променя осмичното число точно както водещата нула няма значение за смисъла на едно десетично число. Водещата нула все още може да бъде изписана, както е показано на следващия ред с файл ***file2***. Този файл получава разрешения за четене, писане и запис, но само от страна на собственика, тъй като цифрите за групата и потребителя са нули.

Ред 13 показва малко странно разпределение на правата, тъй като този път собственикът може само да записва и изпълнява файла, но не и да чете съдържанието му.

На ред 24 е показана още по „екзотична“ форма на задаване на права, при която от числата за правата на всички файлове, се „премахва“ осмичното число 222. 222 представлява комбинация от права за запис съответно на потребителя, групата и всички останали потребители. С други думи, тази команда е еквивалентна на ***chmod ugo-w ****.

Задаването на права директно в осмична бройна система рядко е удобно, но правата в осмична бройна система се използват достатъчно често по други поводи и са важна част от знанията на системните администратори на Линукс. Един от случаите, който налага да се работи с права, зададени в осмична бройна система, е задаването на т.нар. ***umask*** стойности.

umask стойностите са механизъм, който идва от UNIX и служи за определяне на правата за достъп, които се дават по подразбиране на новосъздадените файлове. ***umask*** стойностите се интерпретират като осмични числа също като разрешенията за достъп до файловете, но всъщност представляват разрешенията, които новосъздаденият файл *няма* да получи. За целта, когато се създава нов файл, системата *премахва* ***umask*** стойността началната стойност 0666 (***rw-rw-rw***). Началната стойност 0666 означава, че всички категории потребители без изключение могат да записват и четат файла. Премахването на разрешенията, описвани от ***umask*** стойността, задава по-рестриктивен режим на достъп до файла по усмотрение на потребителя.
Читателят може да погледне отново ред 13 на горния пример, за да види демонстрация как действа премахването на осмично число от разрешенията за достъп до даден файл.

Всеки процес (т.е. всяка стартирана програма) в Линукс има собствена ***umask*** стойност. Когато една програма стартира друга, новостартираната програма получава ***umask*** стойността като част от служебната информация, „наследена“ от стартиращата програма. Новостартираната програма може да използва тази ***umask*** стойност или на свой ред да я промени. С други думи, ***umask*** стойността е по-скоро указание, което добре написаните програми следват във всички случаи, освен когато по-изключение не се налага по-рестриктивен или по-либерален режим на достъп до новосъздаваните файлове.

Чрез ***umask*** стойностите потребителят и стартираните от него процеси могат да задават разрешенията за достъп до всеки новосъздаден файл веднага при създаването без да се налага на втора стъпка да издават команди за уточняването на тези разрешения. Това опростява работата и подобрява сигурността на системата.

В ***bash*** – стандартната обвивка, използвана от дистрибуциите на Линукс – ***umask*** стойностите могат да се манипулират с команда, която носи същото име. Нейната употреба е демонстрирана в следващия пример:

```
 1 lalev@lalev-desktop:~/example5$ umask
 2 0002
 3 lalev@lalev-desktop:~/example5$ touch file1
 4 lalev@lalev-desktop:~/example5$ ls -la
 5 total 8
 6 drwxrwxr-x  2 lalev lalev 4096 May  6 14:12 .
 7 drwxr-xr-x 21 lalev lalev 4096 May  6 14:11 ..
 8 -rw-rw-r--  1 lalev lalev    0 May  6 14:12 file1
 9 lalev@lalev-desktop:~/example5$ umask 022
10 lalev@lalev-desktop:~/example5$ touch file2
11 lalev@lalev-desktop:~/example5$ ls -la
12 total 8
13 drwxrwxr-x  2 lalev lalev 4096 May  6 14:13 .
14 drwxr-xr-x 21 lalev lalev 4096 May  6 14:11 ..
15 -rw-rw-r--  1 lalev lalev    0 May  6 14:12 file1
16 -rw-r--r--  1 lalev lalev    0 May  6 14:13 file2
17 lalev@lalev-desktop:~/example5$ umask 066
18 lalev@lalev-desktop:~/example5$ touch file3
19 lalev@lalev-desktop:~/example5$ ls -la
20 total 8
21 drwxrwxr-x  2 lalev lalev 4096 May  6 14:13 .
22 drwxr-xr-x 21 lalev lalev 4096 May  6 14:11 ..
23 -rw-rw-r--  1 lalev lalev    0 May  6 14:12 file1
24 -rw-r--r--  1 lalev lalev    0 May  6 14:13 file2
25 -rw-------  1 lalev lalev    0 May  6 14:13 file3
26 lalev@lalev-desktop:~/example5$
```

На ред 1 от примера, потребителят издава командата ***umask*** без параметри. Извикана по този начин, командата дава текущата ***umask*** стойност на обвивката. Тази стойност е 0002, което означава, че новосъздадените

файлове няма да могат да бъдат записвани от членове на категорията „всички останали потребители“.

На ред 3 потребителят стартира програмата ***touch***. Тъй като, технически, обвивката стартира програмата, ***touch*** получава ***umask*** стойността 0002 от обвивката и действа според нея. Така новосъздаденият файл ***file1*** има разрешения за достъп 0664 (***rw-rw-r--)***. Тази стойност се получава, като от 0666 (***rw-rw-rw-***), т.е. “четене и запис за всички“, се премахне 0002 (***-------w-***).

На ред 9 потребителят променя ***umask*** стойността на обвивката на 0022, което означава, че потребителят не желае членовете на групата-собственик и всички останали потребители да могат да записват данни в новосъздадените файлове. При това положение достъп за запис в новосъздадените файлове ще има само собственикът им.

Командата ***touch*** получава новата ***umask*** стойност от обвивката и създава ***file2*** според указанията. Прави впечатление, че при задаването на ***umask*** стойността на ред 9, потребителят изтърва една водеща нула и вместо 0022 записва 022. Изпускането на водещите нули е напълно допустимо, тъй като не променя числото. В този смисъл, потребителят можеше да зададе същата ***umask*** стойност и като 22.

На ред 17 потребителят задава дори по-рестриктивната ***umask*** стойност 0066, която означава, че новите файлове няма да са достъпни нито за четене, нито за запис както за членовете на групата-собственик, така и за всички останали потребители.

Читателите, които владеят аритметиката в двоична и осмична бройна система, може би ще се запитат дали „премахването“, за което говорихме по-горе, всъщност е изваждане. Въпросът е резонен, тъй като 0666-0002 в осмична бройна система дава точно 0664.

За правилно зададените ***umask*** стойности, реалният процес по изчисляване на разрешенията на новосъздадените файлове наистина не се различава от изваждането. Това, което се случва в действителност обаче, е че първо системата прилага побитов NOT

оператор върху ***umask*** стойността и комбинира този междинен резултат със стойността 0666 чрез побитов AND. В долния пример цифрите в скобите обозначават каква система е записано числото:

```
0002(8) = 000 000 000 010(2)
NOT 000 000 000 010(2) = 111 111 111 101(2) = 7775(8)


0666(8) = 000 110 110 110(2)
AND
7775(8) = 111 111 111 101(2)
---------------------------------------
0664(8) = 000 110 110 101(2)
```

Идеята на побитовия NOT е да създаде число, което има двоични нули само в онези позиции, в които не желаем да имаме права. За горния пример това е разрешението на всички други потребители да записват във файла. Операцията AND пък има за резултат единица само когато и двата бита – вход са единици. Това означава, че там, където междинният резултат има единица, стойността на бита в другия операнд ще се ***пренесе*** непроменена в резултата. Там, където в междинния резултат има нула, в крайният резултат също ще има нула.

umask 0666 дава разрешения за новосъздадения файл 0000. Същият резултат имаме и при ***umask*** 0667, 0677 и 0777. По същия начин ***umask*** 0000 е еквивалентна на ***umask*** 0001. Общото между всички посочени маски, е че те имат, на едно или повече места, бит за изпълнение, което идва да укаже, че потребителят не желае да дава права за изпълнение на новосъздадените файлове. Но това е излишно, тъй като системата по-подразбиране не дава такива права. Дори в този случай, гореописаният механизъм работи правилно, въпреки че ***umask*** стойността не е зададена съвсем коректно.

Последният важен детайл, който трябва да бъде споменат, е това, че начинаещите потребители не трябва да се притесняват за ***umask*** стойностите твърде много, тъй като системата задава доста смислени стойности по подразбиране на обвивката още когато потребителят влезе в системата. За новите Линукс системи, които създават група със същото име за всеки потребител, стойността на ***umask*** при влизане в системата и стартиране на обвивката е 0002 за обикновени потребители и 0022 за администратора на системата – потребителят ***root***. За по-старите системи, които не създават такива групи, стойността на ***umask*** е 0022 за всички потребители. Разбира се, тази

начална стойност за ***umask*** може да се модифицира от конфигурационните файлове на системата.

1.6. Разрешения за достъп до директории

Докато интерпретацията на трите вида права за достъп по отношение на обикновен файл е директна и интуитивна, същите права, приложени към директориите, се нуждаят от малко повече обяснения. Една директория, също както всеки обикновен файл, може да има разрешения за четене, запис и изпълнение.

Както бе споменато на няколко пъти до момента в изложението, директорията е реализирана като специален файл. За данните, поставяни в този файл, може да се мисли като таблица, съставена от две колони (вж. фиг. 3.4).

Име на файла	**Номер на *inode* на файла**
file1	1939013
file2	1390319
directory1	1394908

Фигура 3.4. Структура на директория

За да може да изведе съдържанието на една директория, потребителят трябва да има правото да прочете съдържанието на нейния файл. Това право се гарантира от разрешението за четене.

За да може да създаде, изтрие или преименува един файл от директорията, потребителят трябва да има права за запис в тази директория. В противен случай потребителят не би могъл да изтрие/промени реда на изтривания/преименувания файл в таблицата на директорията.

Разрешението за изпълнение от друга страна позволява на потребителя да „влиза“ в директорията и, което е по-важно, да подава като аргументи на различни команди абсолютни и относителни пътища, които съдържат

директорията и нейни поддиректории. Образно казано, разрешението за изпълнение на директориите се интерпретира като разрешение за “влизане“ или „преминаване“ през директорията.

Долният пример тези и други особености, касаещи интерпретацията на разрешенията за достъп по отношение на директориите:

```
 1 lalev@lalev-desktop:~/example6$ ls -l
 2 total 12
 3 drwx------ 2 lalev lalev 4096 May  8 21:02 dir1
 4 drwx------ 3 lalev lalev 4096 May  8 21:02 dir2
 5 drwx------ 2 lalev lalev 4096 May  8 21:00 dir3
 6 lalev@lalev-desktop:~/example6$ ls -l dir1
 7 total 0
 8 -rw-rw-r-- 1 lalev lalev 0 May  8 21:02 file1
 9 -rw-rw-r-- 1 lalev lalev 0 May  8 21:02 file2
10 -rw-rw-r-- 1 lalev lalev 0 May  8 21:02 file3
11 lalev@lalev-desktop:~/example6$ cat dir1/file1
12 lalev@lalev-desktop:~/example6$ chmod u-wx dir1
13 lalev@lalev-desktop:~/example6$ ls -l dir1
14 ls: cannot access 'dir1/file2': Permission denied
15 ls: cannot access 'dir1/file3': Permission denied
16 ls: cannot access 'dir1/file1': Permission denied
17 total 0
18 -????????? ? ? ? ?            ? file1
19 -????????? ? ? ? ?            ? file2
20 -????????? ? ? ? ?            ? file3
21 lalev@lalev-desktop:~/example6$ ls -l -i dir1
22 ls: cannot access 'dir1/file2': Permission denied
23 ls: cannot access 'dir1/file3': Permission denied
24 ls: cannot access 'dir1/file1': Permission denied
25 total 0
26 ? -????????? ? ? ? ?            ? file1
27 ? -????????? ? ? ? ?            ? file2
28 ? -????????? ? ? ? ?            ? file3
29 lalev@lalev-desktop:~/example6$ cat dir1/file1
30 cat: dir1/file1: Permission denied
31 lalev@lalev-desktop:~/example6$ ls -l dir2
32 total 4
33 -rw-rw-r-- 1 lalev lalev    0 May  8 21:02 file1
34 -rw-rw-r-- 1 lalev lalev    0 May  8 21:02 file2
35 -rw-rw-r-- 1 lalev lalev    0 May  8 21:02 file3
36 drwxrwxr-x 2 lalev lalev 4096 May  8 21:02 subdir2_1
37 lalev@lalev-desktop:~/example6$ chmod u-rw dir2
```

```
38 lalev@lalev-desktop:~/example6$ ls -l dir2
39 ls: cannot open directory 'dir2': Permission denied
40 lalev@lalev-desktop:~/example6$ cd dir2/subdir2_1
41 lalev@lalev-desktop:~/example6/dir2/subdir2_1$ cd ..
42 lalev@lalev-desktop:~/example6/dir2$ ls
43 ls: cannot open directory '.': Permission denied
44 lalev@lalev-desktop:~/example6/dir2$ cd ..
45 lalev@lalev-desktop:~/example6$ cat dir2/file1
46 lalev@lalev-desktop:~/example6$
```

На ред 1 от примера, потребителят извежда съдържанието на директория ***example6***. Както е видно от редове 2 до 5, директорията съдържа три поддиректории - ***dir1***, ***dir2*** и ***dir3***. Разрешенията на тези поддиректории са еднакви и разрешават на собственика им, който всъщност е текущия потребител, четене, запис и изпълнение.

Командата на ред 6 показва съдържанието на ***dir1***, като по този начин демонстрира, че тези права позволяват потребителя да показва съдържанието на поддиректориите без никакви проблеми.

Ред 11 показва, че разрешенията на ***dir1*** и файловете в нея също позволяват на потребителя да изведе съдържанието на всеки един файл в директорията. Конкретно изведеният в случая файл е ***dir1/file1***. Той е празен, поради което командата ***cat*** завършва без съобщение за грешка, но и без изведено съдържание на екрана.

Командата на ред 12 премахва правата на потребителя за запис и изпълнение на директорията. По този начин потребителят остава само с права за четене. Това води до интересна ситуация, когато следващата команда (ред 13) прави опит да покаже отново съдържанието на ***dir1***. Опитът за извеждане на това съдържание в дълъг формат произвежда няколко съобщения за грешки (редове 14-16), както и съдържанието на директорията (редове 18-20). Този път обаче всички колони в изхода на командата ***ls*** без самите имена на файловете са запълнени с въпросителни.

Изясняването на причините за това поведение е най-добрият начин за пълно разкриване на разликата между правата за четене и изпълнение на директории в Линукс. Това изясняване трябва да започне по-отдалеч със споменаването на факта, че в Линукс, както и във всички модерни многопотребителски операционни системи, програмите не достъпват директно файловете на диска, а разчитат на специални функции на ядрото на операционната система, които могат да се извикват по специален начин от потребителските програми в хода на изпълнението им. Този набор от функции се нарича най-общо API, което идва от „интерфейс за програмиране на приложни програми”. Именно тези функции на ядрото следят за прилагането на разрешенията върху файловете и отказват достъп на процеси, които са стартирани от потребители без нужните права за четене, запис или изпълнение.

За да покаже съдържанието на директория ***dir1***, командата ***ls*** трябва да извика такава функция, която дава като резултат съдържанието на директорията. За да се случи това, ***ls*** дава на операционната система името на директорията, чието съдържание е нужно. Операционната система преценява дали текущия потребител, стартирал ***ls***, има право за четене на директорията и в зависимост от това връща или грешка или данни за съдържанието на директорията под формата на таблица, не особено различна от показаната по-горе.

В случая потребителят ***lalev*** има права за четене върху ***dir1***, така че тази първа операция минава успешно, като ***ls*** получава имената и ***inode*** номерата на файловете.

За да изведе собственика, групата-собственик, размера и всички други детайли, командата ***ls*** се нуждае от информацията в ***inode*** възела на всеки файл в директорията. За на получи тази информация ***ls*** трябва да извика друга функция на операционната система, като подаде пътя до всеки един файл

поотделно. Най-важното нещо, което трябва да се подчертае тук е, че никоя функция на цялото API на Линукс не позволява каквато и да е работа с файл (отваряне, четене, справка за собственика и разрешенията) по номера неговия ***inode*** възел. Вместо това програми като ***ls***, които използват функциите на операционната система, трябва да подават имена и пътища.

В контекста на примера, това означава, че ***ls*** трябва да подаде относителния път до ***file1 (./dir1/file1***) за да получи данните от inode възела. Операционната система ще провери този път за директории, върху които потребителят няма право за изпълнение, т.е. не може да влиза в тях. Ако такива има, независимо от това че такъв файл съществува и че разрешенията на самия файл позволяват достъп, достъпът на потребителя ще бъде отказан.

Случаят с ***dir1*** е точно такъв. Потребителят няма разрешение за изпълнение върху директорията, затова всяка справка за файл, който се съдържа в ***dir1*** или нейни поддиректории, ще бъде отказана от операционната система независимо какво казват разрешенията на индивидуалния файл. Съобщенията за грешки на редове 14-17 са именно резултат от отказа на операционната система да предостави на ***ls*** данни от ***inode*** възлите на файловете ***./dir1/file1***, ***./dir1/file2*** и ***./dir1/file3***. При това положение ***ls*** няма друг избор освен да изведе имената на файловете без други данни. Това ще се случи дори когато ***ls*** извежда данни в кратка форма на цветна конзола, тъй като оцветяването на някои файлове също изисква достъп до данни от ***inode*** възлите им (вж. упр. 1)

Командата ***ls -l -i dir1*** на ред 21 има дори по-странни и неочаквани резултати. Параметърът ***-i*** трябва да изведе номерата на ***inode*** възлите на всеки файл като първа колона, предхождаща останалите стандартни колони на дългия формат на командата ***ls***. Потребителят има права за четене на директорията ***dir1*** което означава, че ***ls*** би трябвало да има достъп до тези

номера. Въпреки това в новата първа колона, изведените резултати са отново въпросителни. Оказва се, че ***ls*** наистина има достъп (вж. упр.2), но поради избора на разработчиците на ***ls,*** програмата се опитва да вземе тези номера от запитванията за съдържанието на ***inode*** структурите, което е неуспешно поради вече описаните причини.

Илюстрация на същия ефект – липса на бит за изпълнение на директория, в която се намира даден файл – е и командата на ред 29. Тя е повторение на по-раншната команда, с която потребителят изведе съдържанието на ***dir1/file1***. Единственото различно нещо този път е фактът, че потребителят вече няма право за изпълнение върху ***dir1***, поради което този опит завършва със съобщение за грешка.

Примерът продължава с илюстрирането на поведението на системата, когато имаме обратната ситуация – наличието на разрешение за изпълнение, но липсата на разрешение за четене. На ред 31 потребителят изпълнява стандартна команда ***ls -l*** върху ***dir2***, която има разрешения ***rwx------***. Тъй като потребителят има всички нужни разрешения, това действие преминава без проблеми.

На следваща стъпка, разрешенията за четене и запис са премахнати от ***dir2***, при което същата команда на ред 38 извежда съобщение за грешка, тъй като потребителят вече няма разрешение да чете съдържанието на файла на директорията.

Въпреки това, ако потребителят знае имената на файлове или поддиректории в ***dir2***, той може да подава валидни пътища към тях на всякакви команди за работа с файлове. При условие, че разрешенията на тези поддиректории също позволяват достъп, потребителят ще може да извършва успешни операции с тях.

На ред 40 потребителят изпълнява командата ***cd dir2/subdir2_1*** и демонстрира точно този факт. Потребителят има всички права върху ***subdir2_1***, но няма права за четене върху ***dir2***. Правата за четене не са нужни за успешна операция по смяна на директорията. Това, което е важно, е потребителят да има права за изпълнение върху ***dir2*** и да даде валиден път до поддиректория. Двете условия са изпълнени, поради което операцията приключва успешно. Ред 45 демонстрира същата ситуация с извеждането на файл в директория ***dir2***.

Следващият пример демонстрира ефектите от наличието или липсата на разрешение за запис върху директория.

```
 1 lalev@lalev-desktop:~/example6$ ls -la dir3
 2 total 8
 3 drwx------ 2 lalev lalev 4096 May 10 07:58 .
 4 drwxrwxr-x 5 lalev lalev 4096 May  8 21:00 ..
 5 -rw-rw-r-- 1 lalev lalev    0 May 10 07:58 file1
 6 -rw-rw-r-- 1 lalev lalev    0 May 10 07:58 file2
 7 lalev@lalev-desktop:~/example6$ chmod ugo-rwx dir3/file2
 8 lalev@lalev-desktop:~/example6$ ls -la dir3
 9 total 8
10 drwx------ 2 lalev lalev 4096 May 10 07:58 .
11 drwxrwxr-x 5 lalev lalev 4096 May  8 21:00 ..
12 -rw-rw-r-- 1 lalev lalev    0 May 10 07:58 file1
13 ---------- 1 lalev lalev    0 May 10 07:58 file2
14 lalev@lalev-desktop:~/example6$ rm dir3/file2
15 rm: remove write-protected regular empty file 'dir3/file2'? y
16 lalev@lalev-desktop:~/example6$ ls -l dir3
17 total 0
18 -rw-rw-r-- 1 lalev lalev 0 May 10 07:58 file1
19 lalev@lalev-desktop:~/example6$ chmod u-x dir3
20 lalev@lalev-desktop:~/example6$ rm dir3/file1
21 rm: cannot remove 'dir3/file1': Permission denied
22 lalev@lalev-desktop:~/example6$ chmod u+x dir3
23 lalev@lalev-desktop:~/example6$ chmod u-w dir3
24 lalev@lalev-desktop:~/example6$ rm dir3/file1
25 rm: cannot remove 'dir3/file1': Permission denied
26 lalev@lalev-desktop:~/example6$ mv dir3/file1 dir3/file3
27 mv: cannot move 'dir3/file1' to 'dir3/file3': Permission denied
28 lalev@lalev-desktop:~/example6$ touch dir3/file4
29 touch: cannot touch 'dir3/file4': Permission denied
```

```
30 lalev@lalev-desktop:~/example6$ chmod u+w dir3
31 lalev@lalev-desktop:~/example6$ touch dir3/file4
32 lalev@lalev-desktop:~/example6$ mv dir3/file1 dir3/file3
33 lalev@lalev-desktop:~/example6$ ls -l dir3
34 total 0
35 -rw-rw-r-- 1 lalev lalev 0 May 10 07:58 file3
36 -rw-rw-r-- 1 lalev lalev 0 May 10 08:00 file4
37 lalev@lalev-desktop:~/example6$
```

На ред 1 потребителят показва съдържанието на поддиректория **dir3**. Както е видно от ред 3 на разпечатката, потребителят има всички права за достъп до директорията. На ред 7 потребителят премахва правата за четене, запис и изпълнение от файла ***file2*** в директорията, а командата на ред 8 потвърждава, че операцията е преминала успешно.

При това положение потребителят няма да има достъп до съдържанието на файла, но както демонстрира командата ***rm dir3/file2*** на ред 14, това не пречи файлът да бъде изтрит. Това е така тъй като изтриването на файла всъщност е процес на редакция на съдържанието на самата директория, а потребителят има права за запис в нея.

Читателят трябва да забележи, че ***rm*** ще забележи тази необичайна ситуация и по подразбиране (т.е. ако не е зададен параметър ***-f***), ще попита за потвърждение.

Редове 19 до 21 демонстрират, че независимо дали потребителят има достъп за запис до директорията, ако той няма и разрешение за изпълнение, системата ще откаже да изтрие файла, тъй като, както при горния пример, пътят до файла преминава през директория, „влизането“ в която е забранено.

На редове 22 до 25 потребителят възстановява разрешението за изпълнение, но премахва разрешението за запис, което има същия ефект – изтриването не е възможно, този път защото потребителят няма право да редактира съдържанието на директорията.

Редове 26-37 са демонстрация на това, че в директория, която не е разрешена за запис, освен изтриване са забранени и операциите по създаване и преименуване на файлове.

Обикновено правата за четене и изпълнение върху директории се дават заедно, тъй като встрани от няколкото изкуствени примера по-горе, на практика няма начин потребител, който не може да чете съдържанието на директорията да знае имената на файлове и поддиректории. Ако е нужно потребителят да създава, изтрива или преименува файлове, то тогава тези две права се допълват и с разрешение за запис

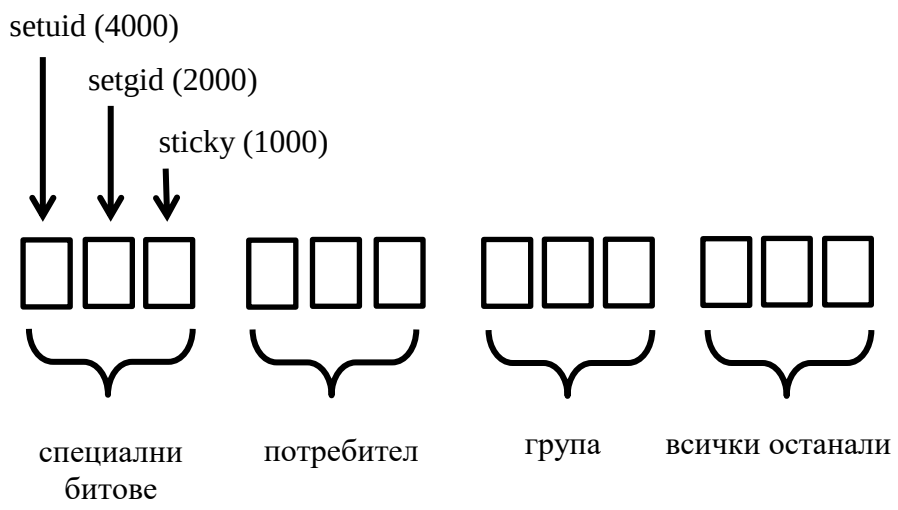
Използването на бита за изпълнение като средство за контрол на достъпа до цели директории е гениална идея, която подобрява сигурността на системата в практически ситуации. И без нея системата за сигурност на Линукс би била консистентна, но би изисквала администраторите и потребителите зорко да следят за разрешенията на всеки един файл поотделно.

За да работи битът за изпълнение в ролята на ограничение, функциите на ядрото не трябва да позволяват каквато и да е манипулация на файлове само по даден техен ***inode*** номер. Това наистина е така, но поражда въпрос, който може би притеснява много от читателите. Защо тогава ***inode*** номерата въобще се показват на потребителите? Защо не останат скрити и капсулирани в ядрото?

Един отговор на този въпрос е това, че когато не е проблем за сигурността, философията на Линукс е да не скрива информация от потребителите. Информацията за ***inode*** номерата може да се окаже от полза поне за един потребител – системния администратор при ситуации като диагностика на дисковата система или възстановяване на повредени и изтрити данни.

1.7. Setuid, setgid и „sticky” битове

Както вече бе споменато, освен деветте бита, представляващи разрешенията за четене, запис и изпълнение на трите категории потребители, файловете и директориите в Linux имат и три специални бита. Те са известни като ***setuid***, ***setgid*** и ***sticky*** бит.



Фигура 3.5. setuid, setgid и sticky битове.

Всяка една програма, стартирана от потребител в Линукс, нормално работи с неговите привилегии. Така например ако потребителят няма право да чете даден файл, то и стартираните от него програми нямат това право. От време на време обаче се налага програмите, стартирани от потребителя, да работят с по-високи привилегии от тези на самия потребител, което налага задаването на ***setuid*** и ***setgid*** битове.

До момента споменахме и частично разгледахме функциите на поне една подобна програма. Това беше програмата ***sudo,***.която позволява на потребителя да изпълнява команди с привилегиите на потребителя ***root***. Ако трябва да обясним по-подробно действията, извършени от ***sudo***, те могат да се разделят на следните стъпки:

1. ***sudo*** проверява дали потребителят е член на групите ***sudoers*** и/или ***adm,*** което на практика прави потребителя един от администраторите на системата.

2. ***sudo*** изисква от потребителя да въведе наново собствената си парола, която ще бъде сверена с тази в ***/etc/shadow*** за да е сигурно, че някой не се е възползвал от забравена отворена конзола.
3. ***sudo*** стартира указаната от потребителя команда с привилегиите на потребителя ***root***.

Ако всяка една програма в Линукс се изпълняваше стриктно с привилегиите на потребителя без изключения, програмата ***sudo*** нямаше да може да изпълни третата стъпка и да стартира програма с правата на администратора на системата.

Читателят навярно забелязва известна непрецизност в използваната терминология. До момента когато говорихме за „администратор на системата“ ние веднъж имахме предвид самият потребител ***root***, докато други пъти говорихме за потребители, които имат право да изпълняват ***sudo***. За повечето практически цели двете са еквивалентни, тъй като всеки, който може да изпълнява командата ***sudo***, може да стартира всяка команда в Линукс с привилегиите на потребителя ***root***.

Друга програма, която трябва да се изпълнява с привилегии, по-високи от тези на потребителите, които я извикват, е програмата ***passwd***. ***passwd*** позволява на даден потребител да смени паролата си. За целта програмата редактира файла ***/etc/shadow***, който съдържа кодираните пароли на всички потребители. Тази информация е чувствителна, поради което файлът е собственост на ***root*** и има разрешения ***rwx------***. С други думи, потребители, различни от ***root***, не могат да осъществяват какъвто и да било достъп до този файл.

Друг, по-малко рестриктивен режим за достъп до ***/etc/shadow***, би бил невъзможен, тъй като в противен случай даден потребител ще може да копира собствената си кодирана парола на мястото, където стои паролата на администратор, и да влезе от името на този администратор със собствената си парола! Това, че само ***root*** ще може да чете и пише този файл обаче представлява неприемливо ограничение, тъй като това би означавало че потребителите не биха могли сами да сменят паролите си при нужда.

За щастие, ***passwd*** извършва редакциите от името на потребителя с привилегиите на администратора, като спазва стриктно ограничението, че

потребителят може да използва програмата само за смяна на собствената си парола.

Начинът, по който двете споменати програми, както и много други програми в Линукс, биват стартирани от един потребител, но работят от името на друг, са именно ***setuid*** и ***setgid*** битовете.

Ако даден файл има установен в единица ***setuid*** бит, когато този файл бъде изпълнен, той ще бъде стартиран *не с привилегиите на потребителя, който го е стартирал, а с привилегиите на собственика на файла*. Ако файлът има установен в единица ***setgid*** бит, *то той ще се изпълни с привилегиите на групата на собственика, а не на текущия потребител*.

Това поведение е илюстрирано с по-долния „синтетичен“ пример. Този пример временно дава администраторски привилегии на програма, която нормално няма нужда от тях, за да демонстрира начина на работа на двата бита.

```
 1 lalev@lalev-desktop:~$ mkdir example7
 2 lalev@lalev-desktop:~$ cd example7
 3 lalev@lalev-desktop:~/example7$ sudo bash
 4 [sudo] password for lalev:
 5 root@lalev-desktop:~/example7# cp /usr/bin/id .
 6 root@lalev-desktop:~/example7# ls -l id
 7 -rwxr-xr-x 1 root root 43224 May 11 16:27 id
 8 root@lalev-desktop:~/example7# chmod u+s id
 9 root@lalev-desktop:~/example7# ls -l id
10 -rwsr-xr-x 1 root root 43224 May 11 16:27 id
11 root@lalev-desktop:~/example7# exit
12 exit
13 lalev@lalev-desktop:~/example7$ ./id
14 uid=1000(lalev) gid=1000(lalev) euid=0(root)
   groups=1000(lalev),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),116
   (lpadmin),126(sambashare)
15 lalev@lalev-desktop:~/example7$ id
16 uid=1000(lalev) gid=1000(lalev)
   groups=1000(lalev),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),116
   (lpadmin),126(sambashare)
17 lalev@lalev-desktop:~/example7$ sudo bash
18 root@lalev-desktop:~/example7# chmod u-s,g+s id
19 root@lalev-desktop:~/example7# ls -l id
```

```
20 -rwxr-sr-x 1 root root 43224 May 11 16:31 id
21 lalev@lalev-desktop:~/example7$ ./id
22 uid=1000(lalev) gid=1000(lalev) egid=0(root)
   groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),116(lpa
   dmin),126(sambashare),1000(lalev)
23 lalev@lalev-desktop:~/example7$ id
24 uid=1000(lalev) gid=1000(lalev)
   groups=1000(lalev),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),116
   (lpadmin),126(sambashare)
25 lalev@lalev-desktop:~/example7$ rm id
26 rm: remove write-protected regular file 'id'? y
27 lalev@lalev-desktop:~/example7$
```

На редове 1-5 потребителят създава нова празна директория и създава в нея копие на програмата ***id***, която както вече бе демонстрирано, извежда на конзолата ***uid*** и ***gid*** номерата на текущия потребител.

На ред 3 потребителят използва командата ***sudo bash*** за да стартира обвивка с привилегиите на администратор, а на ред 8 като администратор задава нови разрешения за достъп на копието на командата ***id***.

Новият момент тук е на ред 8, където командата ***chmod u+s id*** установява ***setuid*** бита в 1. Синтаксисът на ***chmod*** в случая е подобен на този при другите разрешения, но трябва да се помни че ***setuid*** битът е един и команди като ***g+s*** и ***o+s*** няма да дадат очакваните резултати.

Командата на ред 9 показва състоянието на разрешенията след извършването на тези операции. Те са показани от ***ls*** като ***-rwsr-xr-x***. Тук малко ***s*** обозначава, че файлът има едновременно разрешение за изпълнение от страна на собственика и ***setuid*** бит. Освен ***x*** и малко ***s***, на същата позиция ***ls*** може да изведе голямо ***S***. Това се случва когато файлът има ***setuid*** бит 1, но няма разрешение за изпълнение от страна на собственика.

setuid битът указва, че който и да изпълни копието на командата ***ls***, тя ще работи с привилегията на собственика на това копие, показан на ред 10. Това, разбира се, е потребителят ***root***, който няколко реда по-нагоре създаде

файла чрез копиране. Тук е моментът да се отбележи също, че на практика всеки потребител на системата може да стартира тази програма, тъй като тя има разрешение за изпълнение от всички останали потребители.

Ред 13 е най-важната част от настоящия пример. На този ред потребителят стартира копието на ***ls***, което има ***setuid*** бит. Тъй като програмата ***id*** извежда данни за самоличността на потребителя, нейният изход трябва да отрази разликата в привилегиите.

Това наистина е така. На ред 14 заедно с обичайните параметри ***uid*** и ***gid***, id извежда и друга стойност, отразена като ***euid=0(root)***. ***euid*** идва от "effective uid"[27] и заедно с другите параметри отразява факта, че потребител ***lalev*** е стартирал програмата и е собственик на нейния процес, но тя работи с привилегиите на потребителя ***root.***

На ред 15 за съпоставка е изпълнена оригиналната програма, която няма ***setuid*** бит. Нейният изход е аналогичен, с тази разлика, че няма показан параметър ***euid***, което отразява факта, че програмата работи с привилегиите на потребителя, който е я стартирал, т.е. ***lalev***.

На ред 18 потребителят извиква ***chmod*** с два параметъра. Първият от тях ***- u-s*** – премахва ***setuid*** бита от копието на ***id***, докато вторият, ***g+s*** задава ***setgid*** бит на същия файл..

Както е видно от ред 20, визуализацията на ***setgid*** бита е аналогична на ***setuid*** бита, с тази разлика, че буквата малко ***s*** се показва на мястото на разрешението за изпълнение от членовете на групата собственик. По същия начин тази буква обозначава, че файлът има едновременно разрешение за изпълнение от страна на членовете на групата-собственик и ***setgid*** бит и по същия начин голяма буква ***S*** тук би означавала само наличие на ***setgid*** бит.

[27] От. англ. "действащ" uid.

Този път извикването на копието на ***id*** показва друг параметър – ***egid=0(root)***. Той е аналогичен на ***euid*** по горе и идва от "effective gid"[28] По същия начин този параметър обозначава, че групата на процеса (т.е. на стартираната програма) не е същата като главната група на потребителя.

Когато даден процес има ***egid***, различен от ***gid*** на потребителя, който го е стартирал, създадените от него файлове имат за група-собственик тази група, която е указана от ***egid***.

Илюстрираният механизъм на превключване на привилегии чрез ***setuid*** и ***setgid*** битове може да се използва за превключване към всеки един потребител и група, макар че най-честата употреба на тези битове е за превключване към привилегиите на потребителя ***root***.

Както горната дискусия загатна, използването на програми със зададен ***setuid*** или ***setgid*** бит в някои случаи е неизбежно. Използването на някой от двата бита обаче крие рискове за сигурността. Ако например ***passwd*** не успее да спази ограничението всеки потребител, различен от ***root***, да може да сменя само собствената си парола, тази програма тутакси ще се превърне в начин за заобикаляне на системата за сигурност. По същия начин ако ***sudo*** не контролираше кой потребител може да извиква програмата, тогава всеки потребител щеше да може да изпълнява команди от името на потребителя ***root*** (и впрочем, от името на който пожелае потребител).

Използването на ***setuid*** и ***setgid*** битове „отмества" проверките за сигурност от самата базова система за контрол на достъпа към програмата, която има установени тези битове. Образно казано, ***setuid*** и ***setgid*** битовете са широк „портал" към функциите на администратора, а самата програма трябва да играе ролята на стриктна „охрана" на този портал. Ето защо

[28] От. англ. "действащ" gid.

програмите със ***setuid*** и ***setgid*** битове трябва да са написани много внимателно, като се съблюдават всички добри практики по писане на сигурен програмен код. Обикновено тези програми са част от системата и получават ***setuid*** и ***setgid*** битовете си още при началната инсталация. Много рядко системният администратор ще променя тези битове. Това например може да се случи със скриптове, написани от самия него.

В тази връзка даването на ***setuid*** и ***setgid*** битовете на случайно избрани изпълними програми, както прави горния пример, е много, много лоша идея независимо дали го прави администратор или случаен потребител.

Завършваме дискусията с кратък преглед на няколко истински ***setuid*** програми и дискусия на тяхната сигурност.

```
 1 lalev@lalev-desktop:/$ find /usr/bin -perm 4755 2>&1
 2 /usr/bin/pkexec
 3 /usr/bin/chfn
 4 /usr/bin/arping
 5 /usr/bin/passwd
 6 /usr/bin/chsh
 7 /usr/bin/gpasswd
 8 /usr/bin/traceroute6.iputils
 9 /usr/bin/sudo
10 /usr/bin/newgrp
11 lalev@lalev-desktop:/$ ls -la /usr/bin/passwd
12 -rwsr-xr-x 1 root root 59640 Mar 22 21:05 /usr/bin/passwd
13 lalev@lalev-desktop:/$ find /usr/bin -perm 2755 2>&1
14 /usr/bin/crontab
15 /usr/bin/chage
16 /usr/bin/expiry
17 /usr/bin/ssh-agent
18 /usr/bin/wall
19 /usr/bin/bsd-write
20 /usr/bin/mlocate
21 lalev@lalev-desktop:/$ ls -la /usr/bin/wall
22 -rwxr-sr-x 1 root tty 30800 Oct 15  2018 /usr/bin/wall
23 lalev@lalev-desktop:/$ ls -la /dev/tty?
24 crw--w---- 1 root tty 4, 0 May 12 09:09 /dev/tty0
25 crw--w---- 1 root tty 4, 1 May 12 09:09 /dev/tty1
26 crw--w---- 1 root tty 4, 2 May 12 09:09 /dev/tty2
27 crw--w---- 1 root tty 4, 3 May 12 09:09 /dev/tty3
```

```
28 crw--w---- 1 root tty 4, 4 May 12 09:09 /dev/tty4
29 crw--w---- 1 root tty 4, 5 May 12 09:09 /dev/tty5
30 crw--w---- 1 root tty 4, 6 May 12 09:09 /dev/tty6
31 crw--w---- 1 root tty 4, 7 May 12 09:09 /dev/tty7
32 crw--w---- 1 root tty 4, 8 May 12 09:09 /dev/tty8
33 crw--w---- 1 root tty 4, 9 May 12 09:09 /dev/tty9
34 lalev@lalev-desktop:/$ cat /etc/group | grep tty
35 tty:x:5:
36 lalev@lalev-desktop:/$
```

На ред 1 на примера, потребителят извършва търсене в директория ***/usr/bin*** за файлове, които имат ***setuid*** бит. Търсенето на файлове е още едно от местата, където използването на разрешения в осмична форма е по-удобно, а важното нещо, което трябва да бъде забелязано, е това че ***setuid*** е най-старшият („най-левият“) от трите специални бита. Поради тази причина в осмична бройна система ***setuid*** битът добавя 4000 към осмичното число.

В списъка от няколко програми забелязваме дискутираните до момента ***sudo*** и ***passwd***. На ред 11 потребителят извежда с помощта на ***ls*** разрешенията за достъп до ***passwd***. От тях е видно, че всеки може да изпълни ***passwd***, но веднъж стартирана, програмата се изпълнява с привилегиите на нейния-собственик ***root***.

На ред 13 потребителят прави подобно на предното търсене, но този път търси програми със ***setgid*** бит. ***setgid*** битът е средният от трите бита, поради което той добавя 2000 към осмичното число.

Една интересна програма в получения списък е програмата ***wall***, която бе представена в първа глава, когато бяха обсъждани възможностите за комуникация между отделните потребители. Оказва се, че програмата е собственост на потребител ***root*** и има група-собственик ***tty***. Програмата може да бъде изпълнявана от всеки потребител, но веднъж изпълнена, нейната група става ***tty***. Причините за това стават ясни когато се разгледа в детайли как ***wall***

извършва функциите си по извеждане на съобщения на виртуалните конзоли на всички потребители.

За всяка виртуална конзола съществува специален файл, който е разположен в директория ***/dev***. Някои от тези файлове са показани на редове 23-33. Те са начинът, по който потребителските програми могат да взаимодействат с реалните устройства като например клавиатура, мишка и екран, които „съставляват“ конзолата и се управляват от ядрото. Когато в този специален файл се запише определен символен низ, той се извежда на конзолата, което е точно начинът по който ***wall*** успява да реализира функциите си.

Разрешенията на ***tty*** файловете са доста рестриктивни, за да се избегне ситуация, при която даден потребител безконтролно манипулира конзолата на друг потребител, като например изведе фалшиво подканяне за вход в системата. Разрешенията позволяват пълен достъп на потребителя ***root***, но позволяват на членовете на групата ***tty*** само запис, което е точно достатъчно за ***wall***, за да може да изпише съобщения на всички активни конзоли. ***wall***, разбира се, е собственост на потребителя ***root***, което значи че други потребители не могат да променят програмата, а самият програмен код на ***wall*** винаги се грижи съобщенията да са правилно форматирани, за да не подведат потребителя, който ги получава.

Редове 34 и 35 демонстрират факта, че никой потребител в системата няма група ***tty***, и единственият начин да се използват правата на групата за запис във виртуалните конзоли е като се използват ***setgid*** програми, каквато е ***wall***..

Последният от трите специални бита е т.нар. ***sticky*** бит. Той е най-младшият от трите бита поради което той добавя 1000 към разрешенията на файла в осмична форма. Причината за съществуването на ***sticky*** бита е

необходимостта в Линукс системата да има директории, в които да могат да записват всички потребители. Най-важните от тези директории ***/tmp*** и ***/dev/shm***. Тези директории позволяват на отделните потребителски програми да съхраняват временни файлове с данни по начин, който не зависи от привилегиите на потребителя, който ги стартира.

Необходимостта програмите да използват временни файлове е факт от живота, който произтича от начина, по който са организирани самите компютри (например те имат ограничена оперативна памет, която винаги ще е с по-малък обем от твърдите дискове на системата). Къде ще се съхраняват тези файлове, е предмет на инженерно решение, което има своите предимства и недостатъци.

Така например една стратегия за създаването на временни файлове би била програмите да съхраняват временните файлове в домашната директория на потребителите, които ги стартират. По този начин програмистите ще са сигурни, че програмите ще имат правата да създадат временен файл, тъй като (както бе споменато многократно) повечето програми работят с привилегиите на потребителя, който ги стартира. Но някои потребители в Линукс нямат домашна директория! Такъв потребител например е системният потребител ***nobody***.

Подходът на Линукс към проблема следва установените традиции от UNIX и предвижда използването на една стандартна директория за всички процеси на системата. Това е директорията ***/tmp***. Използването на една директория за временните файлове на всички процеси предотвратява разпръскването на временните файлове на различни места във файловата система и пречи на „задръстването“ на дисковете със „забравени“ от програмите неизтрити временни файлове. Тъй като директорията съдържа само временни файлове, това дава свобода на системата да извършва автоматично почистване. Директорията ***/tmp*** обикновено се изчиства при всяко стартиране на системата. Администраторът пък получава, чрез механизма за монтиране, свободата да реши на кой и какъв диск да разположи тази директория. На системи с излишно много памет, тази директория може да бъде директно виртуален диск, разположен в оперативната памет на компютъра.

/dev/shm е точно такъв виртуален диск, който се монтира на ***Ubuntu*** и други системи. Файловете, разположени на този диск се помещават директно в оперативната памет на компютъра. Този диск се разраства според това какво се поставя на него, както максималното му разрастване се определя от администратора. На много системи, ограничението по подразбиране не надвишава половината от наличната оперативна памет.

/dev/shm не е директория за временни файлове в смисъл, че програмите винаги ще записват по подразбиране временните си файлове в ***/tmp***. Администраторът или други потребители могат да поставят ръчно в ***/dev/shm*** файлове, които имат временен характер, и да укажат на програмите да ги използват оттам. Това подобрява многократно скоростта на работа с тях. Недостатъкът е, че към момента на спиране на системата, евентуални промени по тези файлове трябва да се пренесат на „нормален“ диск или те ще бъдат загубени.

Изборът на „централизиран“ подход като този с ***/tmp*** и ***/dev/shm*** има и недостатъци. Тези недостатъци са най-вече по отношение на сигурността и са разгледани по-долу.

Директориите ***/tmp*** и ***/dev/shm*** позволяват четене, запис и изпълнение за всички категории потребители. Това означава, че всеки потребител може да създаде файл в тях. За съжаление това означава също, че всеки потребител може да изтрие файлове в тях, дори тези файлове да не са негова собственост. Това е много опасно особено за шел скриптовете, които не могат да държат временните файлове постоянно отворени, което би предотвратило изтриването им.

Предназначението на ***sticky*** бита е да предотврати тази заплаха. Директории, които имат ***sticky*** бит позволяват изтриването на файл в тях само ако потребителят, който го изтрива, е негов собственик. С други думи, за да изтрие файл в директория със ***sticky*** бит, потребителят не само трябва да има разрешение за запис в директорията, но трябва и да бъде собственик на изтривания файл.

Долният пример демонстрира как се задава и какъв е ефектът от използването на ***sticky*** бит.

```
 1 lalev@lalev-desktop:~$ pwd
 2 /home/lalev
 3 lalev@lalev-desktop:~$ sudo mkdir example8
 4 [sudo] password for lalev:
 5 lalev@lalev-desktop:~$ ls -lad example8
 6 drwxr-xr-x 2 root root 4096 May 16 11:13 example8
 7 lalev@lalev-desktop:~$ sudo chmod ugo+rw example8
 8 lalev@lalev-desktop:~$ ls -lad example8
 9 drwxrwxrwx 2 root root 4096 May 16 11:13 example8
10 lalev@lalev-desktop:~$ sudo touch example8/root1
11 lalev@lalev-desktop:~$ sudo touch example8/root2
12 lalev@lalev-desktop:~$ ls -la example8
13 total 8
14 drwxrwxrwx  2 root  root  4096 May 16 11:14 .
15 drwxr-xr-x 30 lalev lalev 4096 May 16 11:13 ..
16 -rw-r--r--  1 root  root     0 May 16 11:14 root1
17 -rw-r--r--  1 root  root     0 May 16 11:14 root2
18 lalev@lalev-desktop:~$ rm example8/root1
19 rm: remove write-protected regular empty file 'example8/root1'? y
20 lalev@lalev-desktop:~$ sudo chmod o+t example8
```

```
21 lalev@lalev-desktop:~$ ls -lad example8
22 drwxrwxrwt 2 root root 4096 May 16 11:14 example8
23 lalev@lalev-desktop:~$ rm example8/root2
24 rm: remove write-protected regular empty file 'example8/root2'? y
25 rm: cannot remove 'example8/root2': Operation not permitted
26 lalev@lalev-desktop:~$
```

На ред три в примера, потребителят създава директория ***example8***, като използва командата ***sudo*** за да направи това от името на потребителя ***root***. По този начин ***example8*** става по подразбиране собственост на ***root***, което е демонстрирано от изхода на командата ***ls*** на ред 5. Потребителят още веднъж изпълнява команда от името на ***root*** на ред 7 за да даде пълни разрешения за достъп на всички категории потребители. Както е видно от ред 9, след изпълнението на тези операции, всеки потребител може да влиза и да създава файлове в директория ***example8***.

На редове 10 и 11, потребителят отново използва ***sudo*** а да създаде два файла, собственост на ***root*** в директорията. Независимо, че файловете са собственост на ***root***, всеки потребител може да ги изтрие, тъй като директорията, която ги съдържа позволява запис от страна на абсолютно всички потребители. Това е демонстрирано на ред 18, когато потребителят изтрива от собствено име файла ***example8/root1***.

Следващата команда на ред 20 е особено важна, тъй като тя показва как се задава ***sticky*** бит във символна форма. Правилният начин е чрез подаване на параметъра ***o+t*** на ***chmod*** (***u+t*** и ***g+***t няма да дадат желания ефект, тъй като ***sticky*** битът е само един и се отнася за всички категории потребители)

Както е видно от ред 22, ***ls*** уведомява потребителя за това, че дадена директория има ***sticky*** бит като показва буквата ***t*** на мястото на разрешението за изпълнение на всички останали потребители. Ако директорията има и разрешение за изпълнение, показаната буква е малка, докато ако няма, показаната буква е голямо ***T***.

2. Списъци за контрол на достъпа

Освен базовата система за контрол на достъпа, последните версии на повечето важни Линукс дистрибуции поддържат по подразбиране и още една система, която допълва първата. Тази система разчита на т.нар. „списъци за контрол на достъпа“.

За да могат се използват списъците за контрол на достъпа, трябва да са изпълнени три условия:

1. Поддръжката за тях да е компилирана в ядрото;
2. Архитектурата на файловата система, която се използва на дадения дисков дял, да позволява тяхното използване по принцип;
3. Конкретният дисков дял да е монтиран от администратора с параметри, които карат ядрото да вземе под внимание ограниченията, зададени от тези списъци;

Повечето модерни дистрибуции на Линукс използват ядра с компилирана поддръжка на списъци за контрол на достъпа и монтират дисковите дялове с нужните опции. Що се отнася до второто изискване, списъци за контрол на достъпа понастоящем се поддържат от архитектурите на ext2, ext3, ext4, XFS, JFS, ReiserFS и Btrfs, което гарантира, че освен ако потребителят не използва някаква екзотична файлова система, той ще може, при желание да използва този механизъм за контрол да достъпа до файловете.

За съжаление, определянето на това дали дадена файлова система е монтирана с поддръжката на списъци не е толкова лесна задача. Що се отнася до файловата система ***ext4*** (най-популярната сред инсталираните Линукс системи), по-старите версии на Линукс ядрата не активират поддръжката на

списъци, ако тя не е изрично зададена от администратора. По-новите версии подхождат по обратен начин - ако администраторът изрично не е забранил активацията им, те се активират по подразбиране. Други файлови системи задължително използват списъци и не могат да бъдат монтирани без тяхната поддръжка. Самото активиране или дезактивиране на поддръжката за списъци може да бъде направено поне на две места – в ***/etc/fstab*** файла и в опциите по подразбиране на файловата система, достъпни чрез ***tune2fs***.

Ето защо най-лесният и универсално приложим начин за проверка на това дали даден дисков дял поддържа списъци за контрол на достъпа е да се проведе експеримент, като за се зададе такъв списък на тестов файл. Пример за това е даден по-долу, като резултатът в дадения случай е положителен:

```
 1 lalev@dragonfly:~$ touch testfile
 2 lalev@dragonfly:~$ setfacl -m testfile u:root:rwx testfile
 3 lalev@dragonfly:~$ getfacl testfile
 4 # file: testfile
 5 # owner: lalev
 6 # group: lalev
 7 user::rw-
 8 user:root:rwx
 9 group::rw-
10 mask::rwx
11 other::r--
12 lalev@dragonfly:~$ ls -la testfile
13 -rw-rwxr--+ 1 lalev lalev 0 May 27 14:18 testfile
14 lalev@dragonfly:~$
```

Н ред 2, потребителят добавя правило към списъка за контрол на достъпа до ***testfile***, който указва, че потребител ***root*** получава права за четене, запис и изпълнение върху файла. Командата приключва без грешки, което е индикация, че всички споменати по-горе условия са изпълнени. Командата на ред 3 пък показва съдържанието на целия списък за контрол на достъпа до файла ***testfile***, като на ред 8 е видно, че към този списък е добавено правилото, зададено по-рано на ред 2.

На този етап е важно да се забележи, че командата ***ls*** показва наличието на „специални“ правила в списъка за контрол на достъпа (вж. ред 13 на разпечатката) като *след* традиционните разрешения за достъп на файла, тя извежда знак + (плюс).

Следващият пример показва как би протекло изпълнението на същите команди на система и/или дисков дял, които не поддържат списъци за контрол на достъпа.

```
1  lalev@dragonfly:~$ mkdir example22
2  lalev@dragonfly:~$ cd example22/
3  lalev@dragonfly:~/example22$ dd if=/dev/zero of=vdisk.img bs=1M
4  count=40
5  40+0 records in
6  40+0 records out
7  41943040 bytes (42 MB, 40 MiB) copied, 0.175393 s, 239 MB/s
8  lalev@dragonfly:~/example22$ mkfs.ext4 vdisk.img
9  mke2fs 1.44.1 (24-Mar-2018)
10 Discarding device blocks: done
11 Creating filesystem with 40960 1k blocks and 10240 inodes
12 Filesystem UUID: d39190af-af08-4588-9d6c-e551136be3e8
13 Superblock backups stored on blocks:
14         8193, 24577
15
16 Allocating group tables: done
17 Writing inode tables: done
18 Creating journal (4096 blocks): done
19 Writing superblocks and filesystem accounting information: done
20
21 lalev@dragonfly:~/example22$ mkdir mountpoint
22 lalev@dragonfly:~/example22$ sudo mount -o loop,noacl vdisk.img
   ./mountpoint/
23 lalev@dragonfly:~/example22$ cd mountpoint/
24 lalev@dragonfly:~/example22/mountpoint$ sudo chown lalev:lalev .
25 lalev@dragonfly:~/example22/mountpoint$ touch testfile
26 lalev@dragonfly:~/example22/mountpoint$ setfacl -m u:root:rwx
   testfile
27 setfacl: testfile: Operation not supported
28 lalev@dragonfly:~/example22/mountpoint$
```

Демонстрираният по-горе пример извършва доста работа преди същинския експеримент на ред 26, тъй като на модерни Линукс дистрибуции е

доста трудно да се симулира ситуация, в която списъците за контрол на достъпа не работят. Въпреки това подобни ситуации възникват доволно често в практиката, за да им обърнем внимание в настоящото изложение. Така например, читателите, които използват Ubuntu подсистемата за Windows 10, ще забележат, че не се нуждаят от никакви допълнителни стъпки за да демонстрират, че списъците за контрол на достъпа не работят под нейно управление.

На ред 3 от примера потребителят използва командата за блоково копиране ***dd*** за да изчете 40 пъти по 1 мегабайт от специалното устройство **/dev/zero** и да ги постави във файл с име ***vdisk.img***. Тъй като при четене на битове от ***/dev/zero*** операционната система връща винаги двоични нули, резултатът от изпълнението на командата е създаването на 40 мегабайтов файл, пълен с нули.

На ред 8, потребителят използва командата ***mkfs.ext4*** за да създаде файлова система вътре във файла ***vdisk.img***. Програмата ***mkfs.ext4*** представлява стандартната програма за „форматиране“ на дискови дялове в Линукс. Ядрото на Линукс представя дисковете и другите запаметяващи устройства като файлове пред всички програми, включително и тези за форматиране. Поради тази причина ***mkfs.ext4*** не прави разлика дали работи върху реален диск или, както е в нашия случай, върху обикновен файл. И в двата случая извършените действия са еднакви - празният файл се попълва с нужните на файловата система структури от данни като суперблок и таблица на inode възлите и като резултат от тези операции, той става готов за съхраняване на файлове. Можем да мислим за ***vdisk.img*** като за „виртуален диск“, който ще бъде монтиран на дадена позиция във файловата система.

На ред 21 потребителят създава директорията, в която ще се монтира създадения виртуален диск. В случая това е поддиректорията ***mountpoint***. На

ред 22 се извършва същинското монтиране. Тъй като потребителят задава допълнителни опции за монтирането, той трябва да изпълни командата като администратор. Подадените към ядрото опции и по-специално опцията ***noacl*** инструктират системата да не взема под внимание и да не позволява задаването на списъци за контрол на достъпа.

Командата е успешна и потребителят поема контрол над виртуалния диск, чието съдържание оттук нататък ще се появява в директорията ***/home/lalev/example22/mountpoint*** на файловата система.

Както бе споменато в началото на обяснението, на ред 26 потребителят издава същата команда като тази от предния пример. Този път обаче тя не сработва, при което се извежда съобщението ***Operation not supported.***

На този етап е нужно да отделим внимание на това какво представляват списъците за контрол на достъпа. Тези списъци са съставени от набор от правила. Всяко правило определя дали даден потребител или група потребители ще могат да четат, записват или изпълняват даден файл. За разлика от базовата система, която поддържа разрешения за само три категории потребители - собственик, група-собственик и „всички останали“ - списъците за контрол на достъпа нямат ограничения за броя на правилата. Това означава, че ако е нужно, разрешения за достъп могат да бъдат задавани индивидуално за всеки потребител и/или група на системата. По този начин списъците за контрол на достъпа позволяват по-детайлно уточняване на правата за достъп до всеки един файл.

Точно както разрешенията при базовата система за контрол на достъпа, всеки файл или директория в Линукс имат свой индивидуален списък за контрол на достъпа. В него могат да се поставят пет вида правила, които са представени в долната таблица (заградените в квадратни скоби елементи са опционални).

Вид	**Синтаксис**	**Примери**
1. Правило за потребителя - собственик на файла	**u::[r|-][w|-][x|-]** или **user::[r|-][w|-][x|-]**	а/ Потребителят-собственик може само да чете съдържанието на файла: **u::r** или **u::r--** или **user::r** или **user::r--** б/ Потребителят-собственик може да чете и записва съдържанието на файла: **u::rw** или **u::rw-** или **user::rw** или **user::rw-** в/ Потребителят-собственик може да чете и изпълнява файла: **u::rx** или **u:r-x** или **user::rx** или **user::r-x** г/ Потребителят-собственик може да чете, записва и изпълнява файла: **u::rwx** или **user::rwx** д/ Потребителят-собственик няма права за четене, писане или изпълнение на файла **u::---**
2. Правило за конкретен потребител, вероятно различен от потребителя - собственик на файла	**u:потребителско_име: [r|-][w|-][x|-]** или **user:потребителско_име: [r|-][w|-][x|-]**	а/ потребител lalev може да чете, файла: **u:lalev:r** или **u:lalev:r--** или **user:lalev:r** или **user:lalev:r--** б/ потребител lalev може да чете и записва файла: **u:lalev:rw** или **u:lalev:rw-** или **user:lalev:rw** или **user:lalev:rw-**

		в/ потребител lalev може да чете и изпълнява файла: **u:lalev:rx** или **u:lalev:r-x** или **user:lalev:rx** или **user:lalev:r-x**
3. Правило за групата-собственик на файла	**g::[r|-][w|-][x|-]** или **group::[r|-][w|-][x|-]**	а/ членовете на групата - собственик могат да четат файла **g::r** или **g::r--** или **group::r** или **group::r--** б/ членовете на групата - собственик могат да четат, записват и изпълняват файла **g::rwx** или **group::rwx**
4. Правило за конкретна група, вероятно различна от групата - собственик	**g:име_на_група:[r|-][w|-][x|-]** или **group:име_на_група:[r|-][w|-][x|-]**	а/ членовете на група **adm** могат да четат и записват файла **g:adm:rw** или **g:adm:rw-** или **group:adm:rw** или **group:adm:rw-** б/ членовете на група adm могат да записват и изпълняват файла **g:adm:wx** или **g:adm:-wx** или **group:adm:wx** или **group:adm:-wx**
5. Правило за всички останали потребители, които не са собственик, група-собственик или не са споменати от друго	**o::[r|-][w|-][x|-]** или **other::[r|-][w|-][x|-]**	а/ Всички други потребители могат да записват файла **o::w** или **o::-w-** или **other::w** или **other::-w-**

конкретно правило		

Таблица 3.3. Видове правила в списъците за контрол на достъпа

Наблюдателните читатели сигурно са забелязали, че правилата за потребителя собственик и групата-собственик се задават по същия начин като правилата за конкретен потребител и конкретна група потребители, само че при първите името на потребителя или групата са изпуснати. Освен това ***user***, ***group*** и ***other*** могат да се изпишат пълно или да се съкратят съответно като ***u***, ***g*** или ***o***.

Също не е трудно да се забележи, че правилата от вид 1, 3 и 5, тоест правилата за достъп на потребителя-собственик, групата-собственик и правилото за всички останали потребители са идентични по смисъл с разрешенията, които могат да бъдат задавани от базовата система за контрол на достъпа.

Списъците за контрол на достъпа, които съдържат само споменатите видове правила, се наричат минимални. Всеки файл във файловата система получава такъв минимален списък по подразбиране и правилата в този списък отговарят дословно на разрешенията, дадени чрез базовата система. Ако потребителят промени някое от правилата в такъв минимален списък, тази промяна се отразява веднага и в разрешенията на базовата система и обратното - ако някой промени разрешенията чрез базовата система, тези промени се отразяват и в правилата на минималния списък на момента.

Долният пример демонстрира процеса на модификация, като използва момента да представи и командите за манипулиране на списъци.

```
1  lalev@dragonfly:~/example23$ touch testfile
2  lalev@dragonfly:~/example23$ ls -l testfile
3  -rw-rw-r-- 1 lalev lalev 0 May 31 12:45 testfile
4  lalev@dragonfly:~/example23$ getfacl testfile
5  # file: testfile
```

```
6   # owner: lalev
7   # group: lalev
8   user::rw-
9   group::rw-
10  other::r--
11
12  lalev@dragonfly:~/example23$ setfacl -m u::rwx testfile
13  lalev@dragonfly:~/example23$ getfacl testfile
14  # file: testfile
15  # owner: lalev
16  # group: lalev
17  user::rwx
18  group::rw-
19  other::r--
20
21  lalev@dragonfly:~/example23$ ls -la testfile
22  -rwxrw-r-- 1 lalev lalev 0 May 31 12:45 testfile
23  lalev@dragonfly:~/example23$
```

На ред 1, потребителят създава примерен файл, който получава стандартни разрешения за достъп от базовата система. Както е видно от командата ***ls*** на ред 2, тези разрешения са ***rw-rw-r--***.

На ред 4, потребителят използва командата ***getfacl*** за да изведе на стандартния изход списъка за контрол на достъпа до файла. Тъй като файлът ***testfile*** не е получавал специален списък за контрол на достъпа, системата генерира минимален списък за контрол на достъпа, базиран на разрешенията на базовата система. На първите няколко реда, които започват със знака **#** (диез), системата извежда информация за името на файла, както и за собственика, групата-собственик и наличието на евентуални специални битове. Всеки от следващите редове представлява едно правило за достъп. Тъй като в конкретния случай става дума за минимален списък, той е съставен точно от три правила, отговарящи на трите категории в базовата система - права на потребителя (ред 8), права на групата собственик (ред 9) и всички останали (ред 10). Видно е, че тези права отговарят точно на показаните в изхода на предхождащата команда ***ls***.

На ред 12 потребителят използва командата ***setfacl*** за да модифицира списък за контрол на достъпа. Параметърът ***u::rwx*** показва, че се променя правилото, което указва правата на собственика и че новите права на собственика върху файла позволяват четене, писане и изпълнение.

Командата на ред 13 показва, че промените са влезли в сила, а особено интересна е командата ***ls*** на ред 21, която показва, че промените, нанесени чрез ***setfacl*** са отразени и в разрешенията за достъп на базовата система.

Както бе вече илюстрирано, модифицирането на даден списък за контрол на достъпа става с командата ***setfacl***. Нейният базов синтаксис за тази цел е:

setfacl -m *правило_1 [,правило_2][...][, правило_n] файл/списък от файлове*

Командата приема едно или повече спецификации на правила, разделени със запетаи, които ще бъдат използвани за модификация на правилата в списъците за контрол на достъпа на един или повече файлове.

Под „модифициране“ тук се разбира това, че новото правило се добавя към съществуващия списък. Ако при този процес се окаже, че потребителят или групата, за които се отнася правилото, вече имат правило в списъка, това старо правило се заменя с новото.Следващият пример демонстрира това по-подробно:

```
1   lalev@dragonfly:~/example23$ getfacl testfile
2   # file: testfile
3   # owner: lalev
4   # group: lalev
5   user::r--
6   group::r--
7   other::---
8
9   lalev@dragonfly:~/example23$ setfacl -m u::rw,user:www-data:r
10  testfile
11  lalev@dragonfly:~/example23$ getfacl testfile
12  # file: testfile
13  # owner: lalev
```

```
14 # group: lalev
15 user::rw-
16 user:www-data:r--
17 group::r--
18 mask::r--
19 other::---
20
21 lalev@dragonfly:~/example23$
```

Както е видно от командата на ред 1, файлът ***testfile*** този път има минимален списък за контрол на достъпа, който позволява само четене от страна на потребителя-собственик и групата-собственик. На ред 9 потребителят издава команда ***setfacl*** с две правила, разделени от запетая.

Първото от тези правила дава на потребителя-собственик права за четене и запис върху файла. Но това ново правило влиза в конфликт със съществуващото правило в списъка за контрол на достъпа, което казва, че потребителят собственик може само да чете файла. При това положение, новото правило ще замени старото и както е видно от ред 15 на разпечатката, оттук нататък потребителят-собственик ще има права за четене и запис.

Второто правило касае конкретния потребител ***www-data***, който получава права за четене. Тъй като в списъка до момента няма правило за този потребител, новото правило ще бъде добавено към него без да подменя съществуващо такова.

В крайна сметка, за всеки конкретен потребител и група, както и за категориите потребител-собственик, група-собственик и всички останали, в даден списък за контрол на достъпа може да има най-много едно правило. Даването на ново такова заменя старото, ако такова е съществувало на първо място.

Списъци, които съдържат правила от видове 2 и 4 в таблицата, т.е. такива, които посочват конкретно име на потребител или група, за която се отнасят, се наричат разширени списъци за контрол на достъпа. В горния

пример вече създадохме такъв, тъй като добавянето на правилото ***user:www-data:r*** превърна минималният списък от предния пример в разширен такъв. Най-важното нещо, което трябва да се знае за разширените списъци, е това, че тяхното „оценяване“, т.е. процесът на вземане на решение на тяхна база за това дали даден потребител ще получи достъп или не, се извършва по малко по-различен начин.

Като вече бе споменато, разширените списъци позволяват задаването на прецизни правила за достъп, които не могат да бъдат изразени с помощта на разрешенията на базовата система за контрол на достъпа. Именно затова програмата ***ls*** извежда знак + (плюс) след разрешенията на файл, който има и такъв разширен списък (това бе демонстрирано в пример по-рано в главата).
Това е достатъчно да ориентира потребителите и администратора, но за съжаление това не може да се каже за програмите, които са написани преди появата на списъците за контрол на достъпа. Подобни програми, които разбират само разрешенията за контрол на достъпа на базовата система, могат да бъдат заблудени от наличието на такива списъци. Като резултат от това те могат да дадат неволно право за достъп на потребители, които според разрешенията на базовата система не биха имали подобно право.

За да се минимизира тази възможност дизайнерите на списъците за контрол на достъпа са се опитали да сближат максимално интерпретациите на двата вида разрешения. При базовата система, единственият начин потребител, който не е собственик на файла, да получи по-специални права от всички останали, е той да се включи в групата-собственици на файла. При тази специална интерпретация, както бе загатнато и в началото на главата, групата-собственици би могла да бъде наречена „потребители със по-специални права“. Обикновено тези права са по-малки от тези на собственика и по-големи от тези на всички останали.

Докато при базовата система всички членове на групата-собственици имат еднакви права, за разширените списъци може да се мисли като за механизъм, който дава възможност за по-нататъшно, по-прецизно специфициране на правата на всеки потребител, който ще има по-специални права от всички останали.

Именно тази интерпретация е избрана от създателите и дизайнерите на списъците за контрол на достъпа. За целта, когато потребителят модифицира списък за контрол на достъпа като добави правило за конкретен потребител, конкретна група или пък модифицира разрешенията за групата собственици, ***setfacl*** ще промени разрешенията на групата-собственик на базовата система. Новите стойности за базовата система ще бъдат комбинация от всички разрешения, дадени на потребителите с по-специален достъп. Така например, ако според даден разширен списък за контрол на достъпа един конкретен потребител може да чете файла, друг конкретен потребител може да го записва, а групата-собственик на файла може да го изпълнява, в разрешенията на базовата система ще бъдат записани и трите права. В този смисъл, разрешенията на групата-собственик на файл, който има разширен списък за контрол на достъпа, ще представляват „максималните“ права, които имат отделните потребители с по-специален достъп.

За читателите, запознати с двоичната бройна система и побитовите операции, които бяха разгледани накратко в специалните разяснения към предната глава, ще е полезно да знаят, че когато става дума за разширен списък за контрол на достъпа, ***setfacl*** ще изпълни побитово ИЛИ върху всички разрешения на конкретните потребители, конкретните групи и групата-собственик. Получената стойност ще бъде записана на мястото на разрешенията на групата собственик в базовата система.

Долният пример илюстрира описаното поведение:

```
1  lalev@dragonfly:~/example23$ ls -la testfile
2  -rw----r-- 1 lalev lalev 0 Jun  2 22:54 testfile
3  lalev@dragonfly:~/example23$ getfacl testfile
4  # file: testfile
```

```
5  # owner: lalev
6  # group: lalev
7  user::rw-
8  group::---
9  other::r--
10
11 lalev@dragonfly:~/example23$ setfacl -m user:www-data:r testfile
12 lalev@dragonfly:~/example23$ getfacl testfile
13 # file: testfile
14 # owner: lalev
15 # group: lalev
16 user::rw-
17 user:www-data:r--
18 group::---
19 mask::r--
20 other::r--
21
22 lalev@dragonfly:~/example23$ ls -la testfile
23 -rw-r--r--+ 1 lalev lalev 0 Jun  2 22:54 testfile
24 lalev@dragonfly:~/example23$ setfacl -m group:adm:w testfile
25 lalev@dragonfly:~/example23$ getfacl testfile
26 # file: testfile
27 # owner: lalev
28 # group: lalev
29 user::rw-
30 user:www-data:r--
31 group::---
32 group:adm:-w-
33 mask::rw-
34 other::r--
35
36 lalev@dragonfly:~/example23$ ls -la testfile
37 -rw-rw-r--+ 1 lalev lalev 0 Jun  2 22:54 testfile
38 lalev@dragonfly:~/example23$ setfacl -m group::rx testfile
39 lalev@dragonfly:~/example23$ getfacl testfile
40 # file: testfile
41 # owner: lalev
42 # group: lalev
43 user::rw-
44 user:www-data:r--
45 group::r-x
46 group:adm:-w-
47 mask::rwx
48 other::r--
49
50 lalev@dragonfly:~/example23$ ls -la testfile
```

```
51 -rw-rwxr--+ 1 lalev lalev 0 Jun  2 22:54 testfile
52 lalev@dragonfly:~/example23$
```

Командата на ред 1 в примера демонстрира, че този път ***testfile*** може да бъде четен и записван от собственика си - потребителя ***lalev***. Групата собственик няма никакви права, докато всички останали потребители имат права само за четене. Командата ***setfacl*** на ред 3 показва минималния списък за контрол на достъпа, който отговаря на разрешенията на базовата система.

Командата ***setfacl -m user:www-data:r testfile*** на ред 11 дава права на конкретен потребител, който не е собственик на файла. Това превръща списъка за контрол на достъпа в разширен. Командата ***getfacl*** на ред 12 показва как изглежда този списък. Наред с разрешенията на потребителя-собственик (ред 16), разрешенията на потребителя www-data (ред 17), разрешенията на групата-собственик (ред 18), и разрешенията на всички останали потребители (ред 20), на ред 19 има нов, елемент - ***mask***.

Елементът ***mask*** е именно стойността, която ще се впише на мястото на разрешенията на групата-собственик на базовата система. Както бе споменато по-горе, тя се получава като комбинация от правата на всички потребители, които не са собственика на файла и не попадат в категория всички останали. За нашия пример, тези потребители са потребителят ***www-data*** и членовете на групата-собственик. Тъй като групата-собственик няма разрешения, а потребителят ***www-data*** може само да чете файла, стойността на елемента ***mask*** ще бъде ***r--***. Така разрешенията на файла в базовата система ще показват, че групата-собственик има права за четене, което е показано от командата на ред 22. В действителност, обаче тази стойност трябва да се интерпретира само като индикатор, че в правилото за контрол на достъпа има поне един потребител, който не е собственик на файла и който има право за четене.

Примерът демонстрира същата особеност още два пъти. На ред 24 потребителят дава права за писане на специално посочената група ***adm***. Това води до промени в параметъра ***mask***, който след тази операция приема стойност ***rw-***. Тази стойност се записва в разрешенията на базовата система за групата-собственик и се интерпретира като индикатор, че в списъка за контрол на достъпа има поне един потребител (или група потребители), различен от собственика, който може да чете файла. Тази стойност също показва, че съществува и поне един потребител (или група потребители), различен от собственика (но не задължително различен от първия потребител), който може да записва във файла.

На ред 38 потребителят, чрез списък за контрол на достъпа, дава права за четене и изпълнение на групата собственик. Нейните разрешения се добавят към параметъра ***mask*** и той придобива вида ***rwx***.

Тъй като списъците за контрол на достъпа се прилагат след като са приложени разрешенията на базовата система (точният начин как се случва това ще бъде разгледан след малко), разрешение ***rwx***, записано в правата на групата-собственик в базовата система не означава, че членовете на тази група ще имат пълни права върху файла. Допълнителното ограничение в списъка за контрол на достъпа, показано на ред 45 от разпечатката, ще ограничи записа във файла от страна на член на тази група.

Елегантният замисъл на идеята за интерпретацията на правата на групата като права на всички потребители с по-специален режим на достъп става ясен, когато се разгледа какво се случва, когато потребител или програма използват инструментите за работа с разрешенията на базовата система върху файл, който има разширен списък за контрол на достъпа. Тъй като тези инструменти могат да се извикват от потребители и най-вече програми, които имат или нямат представа за съществуването на разширените списъци за контрол на

достъпа, разрешенията на групата в базовата система вземат приоритет пред разрешенията на всички потребители и групи с по-специални права, споменати в списъка за контрол на достъпа. По този начин, ако новите разрешения на групата-собственик в базовата система не позволяват определен вид достъп до файла (например изпълнение), това право ще бъде отказано на всеки един потребител с по-специален режим, споменат в списъка за контрол на достъпа, независимо какво казват разрешенията в неговото правило.

Долният пример продължава от мястото, където приключи предния, и илюстрира това поведение:

```
1  lalev@dragonfly:~/example23$ ls -l testfile
2  -rw-rwxr--+ 1 lalev lalev 0 Jun  2 22:54 testfile
3  lalev@dragonfly:~/example23$ getfacl testfile
4  # file: testfile
5  # owner: lalev
6  # group: lalev
7  user::rw-
8  user:www-data:r--
9  group::r-x
10 group:adm:-w-
11 mask::rwx
12 other::r--
13
14 lalev@dragonfly:~/example23$ chmod g-wx testfile
15 lalev@dragonfly:~/example23$ ls -l testfile
16 -rw-r--r--+ 1 lalev lalev 0 Jun  2 22:54 testfile
17 lalev@dragonfly:~/example23$ getfacl testfile
18 # file: testfile
19 # owner: lalev
20 # group: lalev
21 user::rw-
22 user:www-data:r--
23 group::r-x                      #effective:r--
24 group:adm:-w-                   #effective:---
25 mask::r--
26 other::r--
27
28 lalev@dragonfly:~/example23$
```

На ред 14 потребителят използва инструмента за манипулиране на разрешенията на базовата система ***chmod***. В конкретния пример, той премахва разрешенията за запис и изпълнение на групата-собственик.

Тъй като файлът има разширен списък за контрол на достъпа, тези разрешения се интерпретираха като „максимално“ възможните за даден потребител, група или комбинация от потребители и групи със специални права. Сега потребител, или може би програма, която може би не разбира разширените списъци, иска да намали тези права. Единственият начин да се запази съвместимостта е тези права да се премахнат от всички потребители с по-специален достъп. Списъците за контрол на достъпа правят именно това.

Командата на ред 17 показва отново списъка за контрол на достъпа след налагането на ограничението. Сега ***getfacl*** показва параметър ***mask***, който отговаря на правата на групата-собственик в базовата система. От тези права бяха премахнати разрешенията за запис и изпълнение, поради което ***mask*** съдържа само разрешение за четене (***r--***). Новите елементи са частите, обозначени с ***effective*** (от англ. „действителни”), които показват реалните права, които има всеки потребител в списъка.

Така например правилото, показано на ред 23 в разпечатката, казва, че групата собственик може да чете и изпълнява файла. Но това влиза в конфликт с разрешенията на базовата система и параметъра ***mask***, което означава, че групата-собственик ще може да осъществява само такъв достъп, който е позволен и от двете системи. В случая това е само операцията четене. Именно този факт е обозначен с последователността ***#effective:r--.***

По същия начин, правилото за групата ***adm*** казва, че тя може само да записва във файла, но тъй като това влиза в противоречие с параметъра ***mask***, на практика групата ***adm*** няма никакви права за достъп до файла (правилото в списъка забранява четене, а разрешенията на групата-собственик в базовата

система позволяват само четене). По този начин действителните права на групата са нулеви, което е обозначено с ***#effective:---***.

Параметърът mask (или „маска“ на български) обозначава, че правата на групата при разширените списъци се интерпретират като битова маска. Когато ***setfacl*** добавя ново правило, той нормално използва побитово ИЛИ (OR), за да преизчисли тази маска. Това на практика има точно описания ефект - маската обозначава „максималната“ комбинация от права на потребителите със по-специален достъп.

Когато списъкът за контрол на достъпа се оценява, т.е. се взема решение на негова база дали даден потребител ще осъществи определен вид достъп до файла, стойността на групата-собственик се комбинира със стойността на всяко конкретно правило за потребители или групи със по-специални права чрез побитово И (AND). Тази операция се нарича „маскиране“ и дава името на параметъра. Операцията оставя единици само там, където и двата входни бита са единици, и дава точно описания по-горе ефект.

Списъците за контрол на достъпа могат да бъдат задавани както за файлове, така и за директории, като смисълът на гарантираните права за четене, запис и изпълнение на директория е същият, като при базовата система. Директориите обаче могат да имат и специален, допълнителен списък, който уточнява правата на новосъздадените файлове в директорията. Ако такъв липсва, новите файлове получават минимален списък за контрол на достъпа, който бе разгледан вече по-горе, и който отговаря на разрешенията на базовата система (за нов файл те се определят от ***umask*** параметъра). Ако такъв списък обаче съществува, той се задава на файла. В хода на създаване на файла програмата, която го създава, може да предпише ограничения в разрешенията, давани чрез базовата система. Ако такива ограничения бъдат зададени за групата-собственик на новия файл, те се интерпретират като маска, която ограничава разрешенията, дадени чрез правила на всички потребители и групи с по-специален достъп, точно както бе дискутирано по-горе.

Задаването на списък по подразбиране на директория се извършва като към параметрите на командата ***setfacl*** се добави ***-d***. Премахването на такъв списък се извършва, като ***setfacl*** се извика за една директория или списък от

директории с параметър ***-k***. Употребата на ***setfacl*** с тези параметри е демонстрирано в долния пример.

```
1  lalev@dragonfly:~$ mkdir example24
2  lalev@dragonfly:~$ ls -ld example24/
3  drwxrwxr-x 2 lalev lalev 4096 Jun  4 12:41 example24/
4  lalev@dragonfly:~$ getfacl example24
5  # file: example24
6  # owner: lalev
7  # group: lalev
8  user::rwx
9  group::rwx
10 other::r-x
11
12 lalev@dragonfly:~$ setfacl -d -m user:www-data:r example24
13 lalev@dragonfly:~$ getfacl example24
14 # file: example24
15 # owner: lalev
16 # group: lalev
17 user::rwx
18 group::rwx
19 other::r-x
20 default:user::rwx
21 default:user:www-data:r--
22 default:group::rwx
23 default:mask::rwx
24 default:other::r-x
25
26 lalev@dragonfly:~$ touch example24/testfile
27 lalev@dragonfly:~$ ls -l example24/testfile
28 -rw-rw-r--+ 1 lalev lalev 0 Jun  4 12:42 example24/testfile
29 lalev@dragonfly:~$ getfacl example24/testfile
30 # file: example24/testfile
31 # owner: lalev
32 # group: lalev
33 user::rw-
34 user:www-data:r--
35 group::rwx                       #effective:rw-
36 mask::rw-
37 other::r--
38
39 lalev@dragonfly:~$ mkdir example24/testdir
40 lalev@dragonfly:~$ getfacl example24/testdir
41 # file: example24/testdir
42 # owner: lalev
```

```
43 # group: lalev
44 user::rwx
45 user:www-data:r--
46 group::rwx
47 mask::rwx
48 other::r-x
49 default:user::rwx
50 default:user:www-data:r--
51 default:group::rwx
52 default:mask::rwx
53 default:other::r-x
54
55 lalev@dragonfly:~$ setfacl -k example24/
56 lalev@dragonfly:~$ getfacl example24
57 # file: example24
58 # owner: lalev
59 # group: lalev
60 user::rwx
61 group::rwx
62 other::r-x
63
64 lalev@dragonfly:~$ getfacl example24/testdir/
65 # file: example24/testdir/
66 # owner: lalev
67 # group: lalev
68 user::rwx
69 user:www-data:r--
70 group::rwx
71 mask::rwx
72 other::r-x
73 default:user::rwx
74 default:user:www-data:r--
75 default:group::rwx
76 default:mask::rwx
77 default:other::r-x
78
79 lalev@dragonfly:~$
```

На ред 1 потребителят създава нова директория - ***example24***. Новосъздадената директория има разрешения ***rwxrwxr-x***, както е видно от ред 3 на разпечатката и първоначално има само минимален списък за контрол на достъпа, който е показан от командата на ред 4.

На ред 12 потребителят използва параметъра ***-d*** на ***setfacl*** за да направи модификации по списъка по подразбиране на директория ***example24***. Както е видно от примера, освен добавянето на параметъра ***-d*** синтаксисът на командата не търпи промени. Тъй като към момента на извикване на директорията няма списък по подразбиране, системата ще генерира един на базата на съществуващите разрешения на базовата система по отношение на директорията. Тъй като потребителят-собственик на ***example24*** има права за четене, писане и изпълнение, тези права ще влязат в него. И тъй като командата на ред 12 не дава никакви указания за промяна на правата на потребителя-собственик, тези права са показани на ред 20, който гласи ***default:user::rwx***. Както е видно, ***setfacl*** показва списъка за директорията и списъка по подразбиране за новите файлове в директорията заедно, като префиксира правилата на втория с думата ***default***.

По подобен начин стоят нещата с правата на групата-собственик и всички останали (редове 22 и 24). Те идват от разрешенията на базовата система за ***example24*** и влизат в списъка по подразбиране, тъй като директорията няма такъв към момента, когато потребителят поиска да модифицира този списък на ред 12.

На ред 22, потребителят създава нов файл в ***example24*** и демонстрира, че този нов файл е получил разрешенията, дадени в списъка по подразбиране за тази директория. Така например правото за четене на потребител ***www-data*** в списъка по подразбиране на директорията се е прехвърлило в списъка на новия файл. Правилото за потребителя-собственик и правилото за всички останали са се прехвърлили, но с промени. От двете правила е премахнато правото за изпълнение. Това е така, тъй като ***touch*** създава файл, който не е изпълним. ***touch*** използва само базовата система за сигурност, когато създава файла. Така когато ***touch*** зададе разрешения ***rw-*** за потребителя-собственик, те се

прехвърлят в списъка за контрол на достъпа в правилото за потребителя-собственик. Същото се отнася и за всички останали.

Тук е моментът да се наблегне, че разрешенията на потребителя-собственик и разрешенията на категория „всички останали“ се прехвърлят директно между двете системи и те никога не могат да бъдат различни. Когато с инструментите на едната се манипулират разрешенията в една от тези две категории, тези промени се отразяват веднага в другата.

Различия може да има само между правата на групата-собственик, показани в базовата система и правата на групата-собственик в списъка за контрол на достъпа. Тези различия могат да съществуват само когато става въпрос за разширен списък за контрол на достъпа.

Правилото за групата-собственик в списъка по подразбиране (даващо права ***rwx***), е прехвърлено в списъка на новия файл без промяна (ред 35), но тъй като ***touch*** е задала чрез базовата система по-рестриктивни разрешения за групата-собственик на файла (***rw-***), маската ограничава реалните права на групата собственик до четене и писане. Същото се отнася и за правата на потребителя ***www-data***, което се прехвърля от списъка по подразбиране без промяна, но бива ограничено от маската. В случая това няма ефект, тъй като маската дава по-големи права от тези, които правилото дава на ***www-data***.

На редове 39-40 потребителят създава поддиректория на ***example24*** и демонстрира, че списъкът за контрол на достъпа по подразбиране се наследява и от новата поддиректория. На редове 55 и 56 използва параметъра ***-k*** за да премахне списъка по подразбиране от ***example24*** и демонстрира, че след тази операция ***example24*** наистина няма такъв списък.

На ред 64 потребителят демонстрира, че изтриването на списъка по подразбиране на директорията-родител не премахва автоматично списъците по подразбиране на поддиректориите. За да се обхванат поддиректориите при всякакъв вид модификации на списъците, ***setfacl*** и ***getfcl*** поддържат параметъра ***-R***.

Много важен въпрос, който касае използването на списъци за контрол на достъпа е как се оценяват списъци с наглед противоречащи си правила, като този, който е показан следващия пример:

```
1   lalev@dragonfly:~/example25$ touch testfile
2   lalev@dragonfly:~/example25$ ls -l testfile
3   -rw-rw-r--+ 1 lalev lalev 0 Jun  4 17:34 testfile
4   lalev@dragonfly:~/example25$ setfacl -b testfile
5   lalev@dragonfly:~/example25$ ls -l
6   total 0
7   -rw-rw-r-- 1 lalev lalev 0 Jun  4 17:34 testfile
8   lalev@dragonfly:~/example25$ getfacl testfile
9   # file: testfile
10  # owner: lalev
11  # group: lalev
12  user::rw-
13  group::rw-
14  other::r--
15
16  lalev@dragonfly:~/example25$ setfacl -m user:lalev:--- testfile
17  lalev@dragonfly:~/example25$ setfacl -m group:adm:r testfile
18  lalev@dragonfly:~/example25$ id
19  uid=1000(lalev) gid=1000(lalev)
    groups=1000(lalev),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),1
    10(lxd),114(lpadmin),115(sambashare),135(project1)
20  lalev@dragonfly:~/example25$ getfacl testfile
21  # file: testfile
22  # owner: lalev
23  # group: lalev
24  user::rw-
25  user:lalev:---
26  group::rw-
27  group:adm:r--
28  mask::rw-
29  other::r--
30  lalev@dragonfly:~/example25$
```

На ред 1 от примера, потребителят създава нов файл с ***touch***. Командата ***ls*** на ред 2 от примера показва, че директорията ***examlpe25*** има списък за контрол на достъпа по подразбиране, чиито правила са прехвърлени към новия файл. Потребителят в случая явно не желае тези правила да се отнасят към новия файл, поради което изтрива всички правила от разширения списък, които

описват по-специален достъп на потребители и по този начин превръща разширения списък в минимален, който отразява разрешенията на базовата система. Изтриването се извършва, като се подаде параметър ***-b*** към ***setfacl.***

На редове 16 и 17 потребителят добавя правила към минималния списък, като отново го превръща в разширен, а на ред 18 извиква командата ***id***, от чиито изход е видно, че потребител ***lalev*** е член на група ***adm***.

На ред 20, потребителят извежда списъка за контрол на достъпа на файла ***testfile.*** Според правилото за потребителя-собственик (ред 24), потребителят ***lalev***, който е и собственик на файла, има права за четене и писане. Следващо правило в списъка обаче, забранява всякакъв достъп на потребител ***lalev***, който този път е посочен като конкретен потребител (ред 25). В същото време потребител ***lalev*** е единствен член на групата-собственик, която има права за четене и писане (ред 26) и е член на група ***adm***, която има разрешения само за писане. Кое от тези права получава приоритет?

Полезно е да се знае, че правилата в списъците за контрол на достъпа се прилагат във определен ред. Той може да бъде описан по следния начин:

1. Ако потребителят е собственик на файла, към него се прилага правилото за собственика на файла. Ако това правило гарантира искания достъп, потребителят го получава. Ако обратното - правилото забранява достъпа, то системата отказва на потребителя достъп без да обработва други правила.
2. Ако потребителят не е собственик на файла, се проверяват всички правила за конкретен потребител (един потребител може да има максимум едно конкретно правило в даден списък). Ако се окаже, че има конкретно правило за потребителя, неговите разрешения определят дали потребителя получава достъп. Ако потребителят

получи отказ, този отказ е окончателен и не се проверяват други правила.

3. Системата проверява правилата за групата-собственик и правилата за конкретни групи. Тъй като потребителят може да членува в много групи, може да се окаже, че в списъка съществуват няколко правила, които се отнасят за групи, в които потребителят членува. Ако дори само едно от тях позволява желания вид достъп, той ще бъде даден на потребителя от системата, при условие че маската го позволява. Ако се окаже, че никое от тях не позволява желания достъп или маската не го позволява, системата ще откаже достъп на потребителя без да обработва по-нататъшни правила.
4. Може да се укаже, че никое от правилата не в списъка за контрол на достъпа не се отнася за групи, в които потребителят членува. При това положение, най-накрая системата ще се позове на правилото за категория „всички останали“. Ако то позволява достъп, той ще бъде даден на потребителя. Ако обратното - това правило не позволява желания вид достъп, той ще бъде отказан.

В контекста на изложения алгоритъм, системата ще провери дали потребител ***lalev*** е собственик на файла. Тъй като това е така, в сила ще влезе правилото на ред 24, което дава права за писане и четене. Ако поисканият достъп касае едно от двете права, той ще бъде даден на потребителя. Ако потребител ***lalev*** се опита да изпълни файла, тъй като правилото на ред 24 забранява това, той ще получи отказ.

Последен важен детайл, който трябва да бъде разгледан, за да е пълно изложението, касае факта, че списъците за контрол на достъпа могат да бъдат записвани и зареждани от файл. Това позволява копирането на дадени разрешения от един файл на друг или дори от един файл на множество други.

Тъй като формата на този файл е обикновен текст, разрешенията дори може да се редактират с текстов редактор, преди да бъдат дадени на друг файл или файлове.

За да се запишат разрешенията във файл е достатъчно да се пренасочи изхода на командата ***getfacl*** към файл. За да се вземат данните от този файл и да се поставят в списък за контрол на достъпа на друг файл, следва да се използва опцията ***--set-file*** на ***setfacl***.

Долният пример илюстрира как могат да се използват тези опции.

```
1   lalev@dragonfly:~/example27$ touch testfile
2   lalev@dragonfly:~/example27$ touch testfile1
3   lalev@dragonfly:~/example27$ setfacl --set u::rw,g::rw,o::r,u:www-
    data:r testfile
4   lalev@dragonfly:~/example27$ getfacl testfile
5   # file: testfile
6   # owner: lalev
7   # group: lalev
8   user::rw-
9   user:www-data:r--
10  group::rw-
11  mask::rw-
12  other::r--
13
14  lalev@dragonfly:~/example27$ getfacl testfile > perm.backup
15  lalev@dragonfly:~/example27$ setfacl --set-file=perm.backup
16  testfile1
17  lalev@dragonfly:~/example27$ getfacl testfile1
18  # file: testfile1
19  # owner: lalev
20  # group: lalev
21  user::rw-
22  user:www-data:r--
23  group::rw-
24  mask::rw-
25  other::r--
26
27  lalev@dragonfly:~/example27$ setfacl -x u:www-data *
28  lalev@dragonfly:~/example27$
```

На редове 1 и 2 потребителят създава два нови файла - ***testfile*** и ***testfile1***. Командата на ред 3 е малко по-различен и неразглеждан досега

вариант на ***setfacl***, който не просто променя, а заменя целият списък за контрол на достъпа до файла с нов. Новият списък, който е посочен на реда след опцията ***--set*** в този случай трябва да съдържа задължително правила за потребителя-собственик, групата-собственик и категория „всички останали“. Правилата на ред 3 покриват това изискване, като добавят още едно правило, което касае специално потребителя ***www-data***.

Резултатният списък за контрол на достъпа е показан на редове 5-13. На ред 14 потребителят използва знак > (по-голямо) за да пренасочи изхода от командата от конзолата към файла ***perms.backup***. Пренасочванията на изхода ще бъдат разгледани подробно в следващите глави, но на този етап е достатъчно да се каже, че в резултат на операцията файлът ***perms.backup*** получава идентично на редове 5-13 съдържание, а именно изходът на командата ***getfacl testfile***.

На ред 15 потребителят използва ***setfacl*** в комбинация с опция ***--set-file*** за да подмени напълно списъкът за контрол на достъпа на ***testfile1*** с този, който е записан в ***perms.backup.*** Командата на ред 16 демонстрира, че тази операция приключва с успех.

Важно е да се отбележи, че ***setfacl*** третира всички символи, които следват знака # като коментари и напълно игнорира тяхното значение. Тоест, редът

```
# owner: lalev
```

няма да предизвика промяна на собственика на файла, дори потребител ***lalev*** да не беше собственик. ***setfacl*** не може да се използва за задаване на собственика или групата на файла, както и ***setuid***, ***setgid*** и ***sticky*** битове.

Запознатите с механизмите на пренасочване на изхода на обвивката ще оценят по-кратък начин за прехвърляне на списъка от един на друг файл, а именно комбинацията от команди

```
getfacl testfile | setfacl --set-file=- testfile1
```

За финал на настоящия пример и настоящата точка, ред 27 демонстрира как се изтриват конкретни правила от списъците за контрол на достъпа на един или повече файлове. В случая се премахва правилото за потребител ***www-data*** и от двата файла в директорията, което превръща в минимални списъците за контрол на достъпа и на двата файла.

3. Казус – потребителски директории на Apache сървър

Потребителските директории са възможност на уеб сървъра Apache, която позволява всеки потребител да има собствена уеб страница, чиито файлове се намират в домашната папка на този потребител и следователно са под негов контрол. Apache сървър с активирани потребителски директории и адрес ***http://www.example.com*** ще реагира по специален начин, когато получи адрес от рода на ***http://www.example.com/~bogomil/***. Ако потребителят ***bogomil*** съществува, сървърът ще се опита да сервира уеб-съдържание от указаната в конфигурацията поддиректория на домашната папка на потребителя.

Настоящият казус демонстрира инсталацията на Apache и PHP под CentOS 7.6 и активирането на потребителските директории. В хода на този процес казусът илюстрира ползата от използване на списъци за контрол на достъпа, както и наличието на още една, доста по-сложна за конфигурация система за контрол на достъпа, която излиза извън обхвата на настоящата книга.

Казусът започва с инсталацията на Apache и PHP.

```
1   [root@localhost ~]# yum install httpd php
2   Loaded plugins: fastestmirror
3   Determining fastest mirrors
     * base: mirrors.neterra.net
```

```
 * extras: mirrors.neterra.net
 * updates: mirrors.neterra.net
base
| 3.6 kB  00:00:00
extras
| 3.4 kB  00:00:00
updates
| 3.4 kB  00:00:00
(1/4): base/7/x86_64/group_gz
| 166 kB  00:00:01
(2/4): extras/7/x86_64/primary_db
| 200 kB  00:00:01
(3/4): updates/7/x86_64/primary_db
| 5.0 MB  00:00:27
(4/4): base/7/x86_64/primary_db
| 6.0 MB  00:00:34
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.4.6-89.el7.centos will be installed
--> Processing Dependency: httpd-tools = 2.4.6-89.el7.centos for package: httpd-2.4.6-89.el7.centos.x86_64
--> Processing Dependency: /etc/mime.types for package: httpd-2.4.6-89.el7.centos.x86_64
--> Processing Dependency: libaprutil-1.so.0()(64bit) for package: httpd-2.4.6-89.el7.centos.x86_64
--> Processing Dependency: libapr-1.so.0()(64bit) for package: httpd-2.4.6-89.el7.centos.x86_64
---> Package php.x86_64 0:5.4.16-46.el7 will be installed
--> Processing Dependency: php-common(x86-64) = 5.4.16-46.el7 for package: php-5.4.16-46.el7.x86_64
--> Processing Dependency: php-cli(x86-64) = 5.4.16-46.el7 for package: php-5.4.16-46.el7.x86_64
--> Running transaction check
---> Package apr.x86_64 0:1.4.8-3.el7_4.1 will be installed
---> Package apr-util.x86_64 0:1.5.2-6.el7 will be installed
---> Package httpd-tools.x86_64 0:2.4.6-89.el7.centos will be installed
---> Package mailcap.noarch 0:2.1.41-2.el7 will be installed
---> Package php-cli.x86_64 0:5.4.16-46.el7 will be installed
---> Package php-common.x86_64 0:5.4.16-46.el7 will be installed
--> Processing Dependency: libzip.so.2()(64bit) for package: php-common-5.4.16-46.el7.x86_64
--> Running transaction check
---> Package libzip.x86_64 0:0.10.1-8.el7 will be installed
--> Finished Dependency Resolution

```

```
35  Dependencies Resolved
36
37  ================================================================
    ========================
38   Package              Arch            Version
    Repository        Size
39  ================================================================
    ========================
40  Installing:
41   httpd                x86_64          2.4.6-89.el7.centos
    updates          2.7 M
42   php                  x86_64          5.4.16-46.el7
    base             1.4 M
43  Installing for dependencies:
44   apr                  x86_64          1.4.8-3.el7_4.1
    base             103 k
45   apr-util             x86_64          1.5.2-6.el7
    base              92 k
46   httpd-tools          x86_64          2.4.6-89.el7.centos
    updates           90 k
47   libzip               x86_64          0.10.1-8.el7
    base              48 k
48   mailcap              noarch          2.1.41-2.el7
    base              31 k
49   php-cli              x86_64          5.4.16-46.el7
    base             2.7 M
50   php-common           x86_64          5.4.16-46.el7
    base             565 k
51
52  Transaction Summary
53  ================================================================
    ========================
54  Install  2 Packages (+7 Dependent packages)
55
56  Total download size: 7.7 M
57  Installed size: 27 M
58  Is this ok [y/d/N]: y
59  Downloading packages:
60  warning: /var/cache/yum/x86_64/7/base/packages/libzip-0.10.1-
    8.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID f4a80eb5:
    NOKEY
61  Public key for libzip-0.10.1-8.el7.x86_64.rpm is not installed
62  (1/9): libzip-0.10.1-8.el7.x86_64.rpm
    |  48 kB  00:00:00
    Public key for httpd-tools-2.4.6-89.el7.centos.x86_64.rpm is not
    installed
```

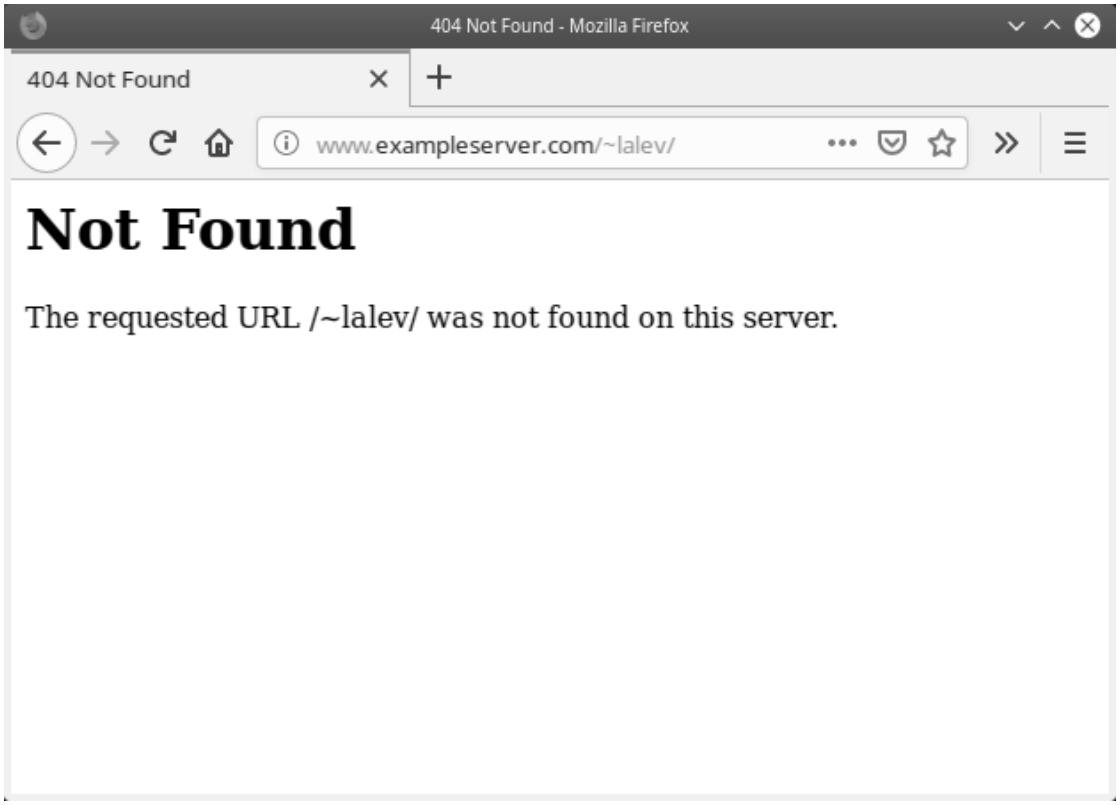
```
63  (2/9): httpd-tools-2.4.6-89.el7.centos.x86_64.rpm
    |  90 kB  00:00:01
64  (3/9): apr-1.4.8-3.el7_4.1.x86_64.rpm
    | 103 kB  00:00:01
65  (4/9): mailcap-2.1.41-2.el7.noarch.rpm
    |  31 kB  00:00:00
66  (5/9): apr-util-1.5.2-6.el7.x86_64.rpm
    |  92 kB  00:00:01
67  (6/9): php-common-5.4.16-46.el7.x86_64.rpm
    | 565 kB  00:00:08
68  (7/9): php-5.4.16-46.el7.x86_64.rpm
    | 1.4 MB  00:00:12
69  (8/9): httpd-2.4.6-89.el7.centos.x86_64.rpm
    | 2.7 MB  00:00:16
70  (9/9): php-cli-5.4.16-46.el7.x86_64.rpm
    | 2.7 MB  00:00:17
71  ----------------------------------------------------------------
    -------------------------
72  Total                                                   420
    kB/s | 7.7 MB  00:00:18
73  Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
74  Importing GPG key 0xF4A80EB5:
75   Userid     : "CentOS-7 Key (CentOS 7 Official Signing Key)
    <security@centos.org>"
76   Fingerprint: 6341 ab27 53d7 8a78 a7c2 7bb1 24c6 a8a7 f4a8 0eb5
77   Package    : centos-release-7-6.1810.2.el7.centos.x86_64
    (@anaconda)
77   From       : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
78  Is this ok [y/N]: y
79  Running transaction check
80  Running transaction test
81  Transaction test succeeded
82  Running transaction
83    Installing : apr-1.4.8-3.el7_4.1.x86_64
    1/9
84    Installing : apr-util-1.5.2-6.el7.x86_64
    2/9
85    Installing : httpd-tools-2.4.6-89.el7.centos.x86_64
    3/9
86    Installing : libzip-0.10.1-8.el7.x86_64
    4/9
87    Installing : php-common-5.4.16-46.el7.x86_64
    5/9
88    Installing : php-cli-5.4.16-46.el7.x86_64
    6/9
89
```

```
  Installing : mailcap-2.1.41-2.el7.noarch
90 7/9
  Installing : httpd-2.4.6-89.el7.centos.x86_64
91 8/9
  Installing : php-5.4.16-46.el7.x86_64
92 9/9
  Verifying  : httpd-tools-2.4.6-89.el7.centos.x86_64
93 1/9
  Verifying  : mailcap-2.1.41-2.el7.noarch
94 2/9
  Verifying  : apr-util-1.5.2-6.el7.x86_64
95 3/9
  Verifying  : php-cli-5.4.16-46.el7.x86_64
96 4/9
  Verifying  : libzip-0.10.1-8.el7.x86_64
97 5/9
  Verifying  : httpd-2.4.6-89.el7.centos.x86_64
98 6/9
  Verifying  : php-5.4.16-46.el7.x86_64
99 7/9
  Verifying  : php-common-5.4.16-46.el7.x86_64
100 8/9
  Verifying  : apr-1.4.8-3.el7_4.1.x86_64
101 9/9
102
Installed:
  httpd.x86_64 0:2.4.6-89.el7.centos                php.x86_64
0:5.4.16-46.el7
103
104 Dependency Installed:
  apr.x86_64 0:1.4.8-3.el7_4.1                       apr-
105 util.x86_64 0:1.5.2-6.el7
  httpd-tools.x86_64 0:2.4.6-89.el7.centos           libzip.x86_64
106 0:0.10.1-8.el7
  mailcap.noarch 0:2.1.41-2.el7                      php-cli.x86_64
107 0:5.4.16-46.el7
  php-common.x86_64 0:5.4.16-46.el7
108
109 Complete!
[root@localhost ~]#
```

На ред 1 потребителят издава командата ***yum install httpd php***, която инструктира пакетната система на CentOS да пристъпи към инсталация на уеб сървъра Apache и модулите за програмиране на PHP. Повечето от

съобщенията, които следват са свързани с диагностика и информиране на потребителя за прогреса на операцията. На редове 37-53 ***yum*** информира потребителя какви пакети ще бъдат инсталирани и колко място ще заемат те на диска, а на ред 58 задава въпроса дали потребителя иска да изпълни зададените транзакции. Отговорът е утвърдителен и инсталацията протича както е указано.

Ако към този момент потребител отвори адреса на даден валиден потребител (например ***http://www.exampleserver.com/~lalev/***) в браузър, той ще види следното съобщение:



Фигура 3.6. Съобщение за грешка 404

Причината за това е, че модулът, който отговаря за пренасочването на горния URL адрес към правилната папка във файловата система не е активиран. Следващите стъпки на администратора са свързани именно с активирането на този специален модул.

```
1   [root@localhost ~]# cd /etc/httpd/conf.modules.d/
2   [root@localhost conf.modules.d]# cat 00-base.conf |grep userdir
3   LoadModule userdir_module modules/mod_userdir.so
4   [root@localhost conf.modules.d]# cd ../conf.d/
5   [root@localhost conf.d]# ls userdir.conf
6   userdir.conf
7   [root@localhost conf.d]#
```

На ред 1 потребителят преминава в директорията с конфигурацията на модулите на сървъра Apache. Там той проверява дали в конфигурационните файлове фигурира директивата

```
LoadModule userdir_module modules/mod_userdir.so
```

Ако тази директива фигурира, това означава, че модулът за потребителски директории се зарежда при стартиране на Apache. Както е видно от ред 3, това е така.

Следващата стъпка е да се редактира конфигурацията на този модул и да се определят потребителите, които ще имат право да притежават уеб папки. Конфигурацията на модула се намира във файла ***/etc/httpd/conf.d/userdir.conf.*** На ред 5 администраторът проверява дали такъв файл съществува на системата. Резултатът е положителен.

Следващата стъпка е редактирането на конфигурацията. Администраторът може да извърши това с любимия си текстов редактор. Крайният резултат следва да изглежда като показания на долната разпечаткаq която за краткост изключва евентуални коментари (редовете, започващи с #).

```
1   <IfModule mod_userdir.c>
2       UserDir enabled lalev
3
4       UserDir public_html
5   </IfModule>
6
7   <Directory "/home/*/public_html">
8       AllowOverride FileInfo AuthConfig Limit Indexes
9       Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
10      Require method GET POST OPTIONS
11  </Directory>
```

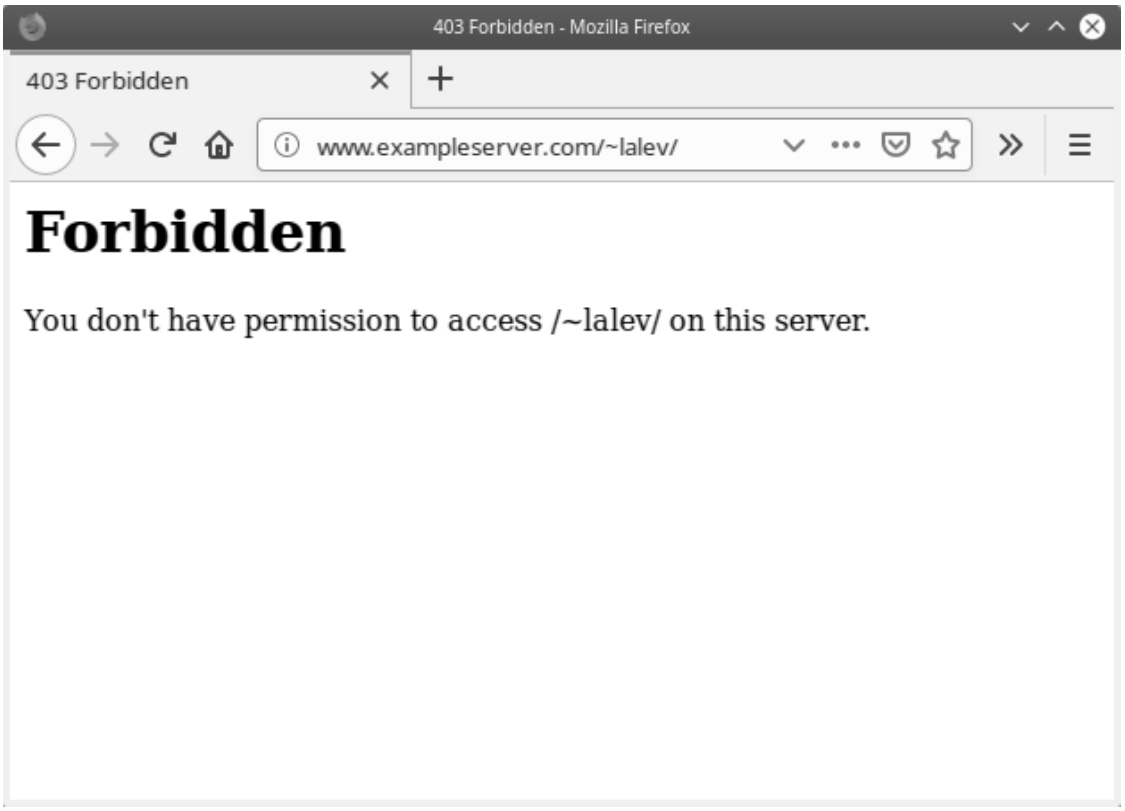
Показаната конфигурация е консервативна. Ред 2 инструктира модула да разреши уеб директория само на потребител ***lalev***. Ред 4 указва, че при извикване на ***http://www.exampleserver.com/~lalev/*** или друг адрес на валиден потребител на системата, който освен това е в списъка на ред 2, системата ще потърси поддиректория ***public_html*** в домашната директория на този потребител и ще сервира уеб-съдържанието от там. Редовете 7-11 са стандартни правила за достъп до подобни директории и не следва да се променят, освен в посока на по-нататъшно ограничаване на функции, които не се използват.

След промяната на този файл, администраторът трябва да рестартира сървъра, за да влязат промените в сила. Това се извършва по следния начин:

```
1  [root@localhost conf.d]# systemctl restart httpd
2  [root@localhost conf.d]#
```

Командата на ред 1 преминава без съобщения за грешка, което указва успешна операция.

Ако потребител се опита да зареди посочения по-горе URL адрес в този момент, той ще види различно съобщение. Вместо код за грешка 404 (Ресурсът не е намерен), Apache вече връща код за грешка 403 (Достъпът забранен).



Фигура 3.7. Съобщение за грешка 403

Причините за това са видни от долната разпечатка:

```
1  [root@localhost conf.d]# cd /home/
2  [root@localhost home]# ls -la
3  total 0
4  drwxr-xr-x.  3 root  root   19 Jun  4 16:49 .
5  dr-xr-xr-x. 17 root  root  224 Jun  4 13:36 ..
6  drwx------.  2 lalev lalev  62 Jun  4 16:49 lalev
7  [root@localhost home]#
```

Домашната директория на потребител ***lalev*** (вж. ред 6) не позволява достъп за други потребители или групи, а при CentOS сървърът Apache работи с привилегиите на специален потребител, а именно потребителят ***apache***!

Администраторът е изправен пред проблем. Ако не разчита на списъци за контрол на достъпа, той трябва да разреши преминаването през домашната директория на потребител ***lalev*** за категория „всички останали потребители“, като в същото време гарантира на същата тази категория права за четене и

изпълнение в поддиректорията ***public_html***. Но това ще даде достъп на всеки потребител на системата до тази директория и ще му позволи да прочете чувствителна информация като пароли за сървър за бази от данни, които могат да се намират в PHP скриптовете на потребителя. Дори по-лошо, може да се окаже, че уеб сървърът се нуждае от права за запис на файлове в ***public_html***. Тогава администраторът трябва да разреши на всички потребители на системата да записват в ***/~lalev/public_html***!

Това очевидно е нежелателно и администраторът трябва да преглътне огромен проблем в сигурността или да избере друг подход, базиран на списъци за контрол на достъпа:

```
1  [root@localhost home]# setfacl -m u:apache:rx lalev
2  [root@localhost home]# cd lalev
3  [root@localhost lalev]# setfacl -m u:apache:rwx public_html
4  [root@localhost lalev]# setfacl -d -m u:apache:rwx public_html
```

На ред 1, администраторът дава разрешения на потребителя ***apache*** да преминава през домашната директория на потребител ***lalev***. След това на ред 3, администраторът дава права за четене, писане и изпълнение на поддиректорията ***public_html*** на потребителя ***apache*** (ред 3) и настройва списък по подразбиране за ***public_html*** (ред 4), което подсигурява, че файловете и директориите, създадени там от потребител ***lalev*** автоматично ще бъдат достъпни и за потребителя ***apache***.

Ако даден потребител на този етап се опита да отвори потребителската папка на потребителя ***lalev***, той ще получи отново съобщение 403 – Достъпът забранен. Причините този път се коренят в още една система за контрол на достъпа, която е от различен клас спрямо представените до момента две системи.

SELinux е третата система за сигурност, която ще бъде спомената в настоящата глава и спада към категорията на системите, които позволяват

осъществяването на т.нар. “Mandatory Access Control”. Представените до момента две системи разчитат на потребителите и техните програми сами да определят правилата на достъп до файловете. Поради тази причина те често се обозначават с понятието „Discretionary Access Control”, което на български в случая може да се преведе като „контрол на достъпа по желание“. При Discretionary Access Control системите, грешки от страна на потребителите или бъгове в техните програми могат да оставят критични файлове неправилно защитени.

SELinux позволява администраторът да укаже правила за ползване на файлове, директории и други ресурси в системата, които не могат да бъдат отменяни от потребителите, които са собственици на ресурсите. В този смисъл, правилата са задължителни, което дава и името „Mandatory Access Control”. SELinux се използва и е включен по подразбиране на Fedora, CentOS и RHEL, като той идва с широк комплект правила, които акомодират най-различни режими на използване на критичния за системата софтуер.

Оказва се, че за да може да се използват потребителски уеб директории, правилата на SELinux трябва да допуснат потребителя ***apache*** да използва файлове и директории, разположени под ***/home*** – нещо, което той обикновено не прави, което е и причината то да е забранено от стандартните правила на SELinux. Диагностиката на проблема и корекциите, предприети от администратора са показани на следващата разпечатка:

```
1   [root@localhost active]# wget http://www.exampleserver.com/~lalev/
2   --2019-06-04 17:33:56--  http://www.exampleserver.com/~lalev/
3   Resolving www.exampleserver.com (www.exampleserver.com)...
4   127.0.0.1
5   Connecting to www.exampleserver.com
6   (www.exampleserver.com)|127.0.0.1|:80... connected.
7   HTTP request sent, awaiting response... 403 Forbidden
8   2019-06-04 17:33:56 ERROR 403: Forbidden.
9
10  [root@localhost active]# setenforce 0
```

```
11 [root@localhost active]# wget http://www.exampleserver.com/~lalev/
12 --2019-06-04 17:34:04--  http://www.exampleserver.com/~lalev/
13 Resolving www.exampleserver.com (www.exampleserver.com)...
14 127.0.0.1
15 Connecting to www.exampleserver.com
16 (www.exampleserver.com)|127.0.0.1|:80... connected.
17 HTTP request sent, awaiting response... 200 OK
18 Length: 675 [text/html]
19 index.html: No such file or directory
20
21 Cannot write to ‘index.html’ (No such file or directory).
22 [root@localhost active]# setenforce 1
23 [root@localhost active]# setsebool -P httpd_enable_homedirs true
24 [root@localhost active]# wget http://www.exampleserver.com/~lalev/
25 --2019-06-04 17:34:19--  http://www.exampleserver.com/~lalev/
26 Resolving www.exampleserver.com (www.exampleserver.com)...
27 127.0.0.1
28 Connecting to www.exampleserver.com
29 (www.exampleserver.com)|127.0.0.1|:80... connected.
30 HTTP request sent, awaiting response... 200 OK
31 Length: 675 [text/html]
32 index.html: No such file or directory
33
34 Cannot write to ‘index.html’ (No such file or directory).
35 [root@localhost active]#
```

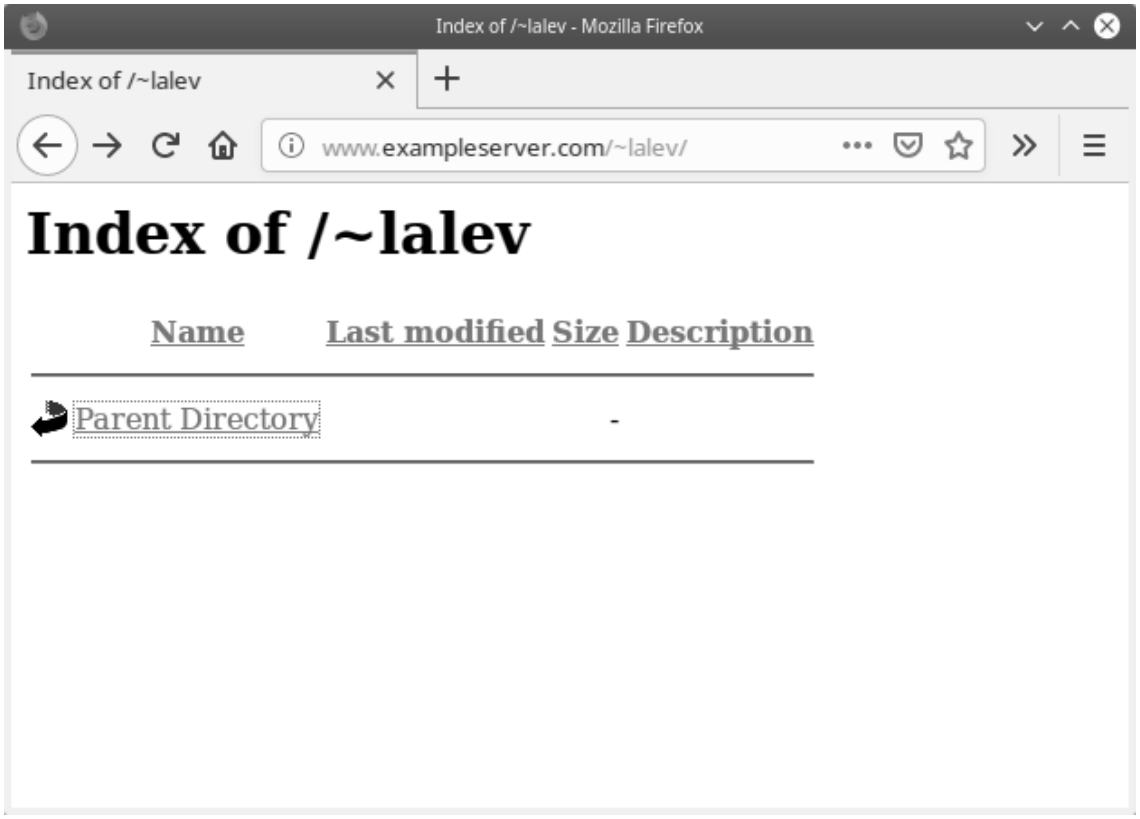
На ред 1 администраторът използва програмата ***wget*** за да изтегли страницата на потребителя ***lalev***. Уеб сървърът, тъй като е ограничен от SELinux, не може да достъпи папката /***home/lalev/public_html*** поради което продължава да връща грешка 403 - факт, който ***wget*** съобщава на ред 7.

На ред 10 администраторът временно изключва прилагането на правилата на SELinux със командата ***setenforce 0***. Повторното извикване на ***wget*** при тези условия връща успех (код 200) въпреки че ***wget*** се оплаква, че не може да намери файл ***index.html***. Това е така, тъй като потребителят ***lalev*** все още не е поставил такъв във своята директория. За директории, които нямат ***index.html*** уеб сървърите връщат списък на файловете в тях, при условие, че това не е забранено в конфигурацията на сървъра. Както се оказва, това не е забранено в конкретния случай и потребител, който използва графичен браузър

в този момент ще може да види съдържанието на директория ***public_html***, което показва че потребителската директория най-сетне е достъпна.

За съжаление изключването на SELinux подкопава сигурността на системата, поради което може да се използва само за диагностика. На ред 22 администраторът отново включва SELinux и използва специалната команда ***setsebool -P httpd_enable_homedirs true*** за да активира готовите правила, които позволяват използването на потребителски уеб директории. След изпълнението на тази команда SELinux ще позволява на потребителя ***apache*** да чете и използва файловете в домашните директории на потребителите, при условие че списъците за контрол на достъпа да позволяват това.

Както е видно от командата на ред 24, този път изтеглянето на страницата на потребител ***lalev*** е успешно, което подсказва на администратора, че достъпът до домашните уеб директории работи.



Фигура 3.8. Уеб папка на потребител lalev. В папката няма файлове, поради което се показва празен списък.

4. Упражнения

Упражнение 1. Разрешения за достъп до директории

Създайте примерна директория с произволно име. В директорията, с помощта на ***touch*** създайте няколко файла. Използвайте ***chmod*** за да зададете разрешения за достъп ***r------***. От родителската директория изпълнение командата ***ls -color=none***, за да покажете съдържанието на директорията в кратка форма и без оцветяване. След това изпълнете просто ***ls*** за да получите кратка форма с оцветяване.

Как интерпретирате получените резултати?

Упражнение 2. Алтернативна програма за извеждане на съдържанието на директориите.

Създайте нова директория и с любимият си текстов редактор за Линукс създайте файл ***listdir.c*** със следното съдържание:

```
 1 #define _GNU_SOURCE
 2 #include <stdio.h>
 3 #include <sys/types.h>
 4 #include <dirent.h>
 5
 6 #include <sys/stat.h>
 7 #include <unistd.h>
 8 #include <string.h>
 9
10 int
11 main (int argc, char* argv[])
12 {
13   DIR *dp;
14   struct dirent *ep;
15   struct stat sb;
16   char *filename;
```

```


  dp = opendir (argv[1]);
  if (dp != NULL)
    {
      while (ep = readdir (dp)) {
        printf("%lu %s\n", ep->d_fileno, ep->d_name);
        asprintf(&filename, "./%s/%s", argv[1], ep->d_name);
      }
      (void) closedir (dp);
    }
  else
    perror ("Couldn't open the directory");

  return 0;
}
```

Когато сте уверени, че сте въвели всичко от съдържанието на тази програма на езика C, запишете файла и го компилирайте с командата ***gcc listdir.c -o listdir***

Ако на Вашата система не сте компилирали нищо на C до момента ще се наложи да инсталирате основните инструменти за разработка на C. Във фамилиите от родословното дърво на Debian, това става най-лесно с командата ***sudo apt-get install gcc build-essential*** а във Fedora, CentOS и RHEL това може да бъде направено с командата ***yum group install "Development Tools"***

Компилацията на програмата по гореуказания начин, при условие че всичко е въведено точно и не възникнат грешки при компилация, ще създаде изпълним файл с името ***listdir***. Този файл ще взема задължително като параметър името на директория. Резултатът от изпълнението на файла е извеждане на съдържанието на директорията в две колони - ***inode*** и име на файл. Изпълнете ***listdir*** върху директория, в която имате само разрешение за четене (точно като тази в упражнение 1). Можете ли да видите номерата на ***inode*** възлите? Изпълнете ***ls -lai*** върху същата директория, което би

трябвало да покаже номерата на ***inode*** възлите в първата колона. Виждате ли ги?

Какви заключения може да направите за ефекта от даването на разрешение само за четене на директория. Съвпадат ли с описаното поведение в изложението по-горе?

Упражнение 3. Използване на find за търсене на файлове със специфични разрешения.

Използвайте ***find***, за да намерите всички ***setuid*** и ***setgid*** файлове на системата. Има ли директория или директории, в които изглежда се съхраняват преобладаващо такива файлове?

Използвайте командата ***find*** за да намерите всички директории на системата, които имат разрешение ***r*** за всички останали но нямат разрешение ***x*** за тази категория. Има ли такива директории?

Упражнение 4. Sticky бит на директории, които не са достъпни за категория „всички останали потребители“

Използвайте ls, за да разгледате разрешенията за достъп на папката ***/var/spool/cron/crontabs***. Кой е собственик на директорията? Коя е групата-собственик на директорията? Имат ли членовете на категория „всички останали потребители“ права за четене, „изпълнение“ или запис в директорията?

Може ли потребител, член на групата ***crontab***, да чете съдържанието на директорията? А може ли такъв потребител да създава файлове в нея? Ако може да създава файлове, може ли такъв потребител да трие файловете на друг потребител в същата група?

Упражнение 5. Използване на mktemp

Тъй като шел скриптовете са едни от най-уязвимите програми по отношение на използването на временни файлове в ***/tmp***, отчасти и поради факта, че те не могат да държат такива файлове постоянно отворени, е много важно те да използват непредсказуеми имена за тези файлове. Един от инструментите, които имат за цел да помогнат на програмистите на такива шел скриптове е програмата ***mktemp***. Извикайте я без параметри и вижте изхода и. Проверете дали ***mktemp*** всъщност създава указания файл, както и това какви разрешения има файла.

ПРИЛОЖЕНИЕ А – ИНСТАЛИРАНЕ НА UBUNTU SERVER 18.04 LTS

Дистрибуцията Ubuntu Linux се предлага в множество разновидности. Основните от тях се определят от предназначението на системата, на която ще се инсталира Linux. Ubuntu Server е оптимизирана инсталация на сървъри, а Ubuntu Desktop е предназначена за използване от крайни потребители[29]. Разделението на Ubuntu Server и Ubuntu Desktop касае основно вида на инсталационната програма и инсталираните по подразбиране пакети. Вече инсталирана система Ubuntu Server може да се превърне в Ubuntu Desktop и обратно чрез пакетната система без да се налага преинсталация.

По правило Ubuntu Desktop предлага графичен инсталатор с по-малко на брой опции, докато Ubuntu Server предлага инсталация в текстов режим, която набляга на различни опции и особено на инсталацията на дисковете на системата – въпрос, който е от огромно значение за сървърите, но не е от интерес за обикновените потребители, които обикновено използват само един диск в стандартна конфигурация.

Ubuntu Desktop версията по подразбиране се предлага с Gnome 3 десктоп среда. Съществуват обаче и официални подварианти, които заменят Gnome 3 с други десктоп среди като KDE (Kubuntu), Xfce(Xubuntu), LXQt(Lubuntu) и дори Gnome 2 (Ubuntu MATE). Веднъж инсталиран, даден подвариант на Ubuntu може да се превърне в друг чрез използване на пакетната система.

[29] Към тези две разновидности, Ubuntu наскоро добави и Ubuntu for IOT, която е силно специализирана дистрибуция за вградени “System-on-Chip”-базирани компютри като Raspberry PI. Тази дистрибуция използва специално оптимизирани ядра и система за управление на пакети, която се различава драстично от този, предлагана по подразбиране с Ubuntu Server и Ubuntu Desktop.

Освен да реши кой от горните два варианта ще бъде инсталиран, потребителят трябва да вземе решение и коя версия на Ubuntu да бъде инсталирана.

Ubuntu има за цел да подготвя нова версия на всеки 6 месеца. Издаването на нова версия се случва във фиксирани месеци на годината – април и октомври. Тези месеци, заедно с годината формират номера на версията. Така например Ubuntu Desktop 18.04 е версията на Ubuntu, издадена през април 2018 година. Ubuntu Server 18.10 е версията на Ubuntu Server, която е издадена през октомври 2018 година.

С едно важно изключение, споменато по-долу, Ubuntu поддържа всяка от версиите си по 9 месеца. След края на този период не се издават обновявания по сигурността, което означава че преди изтичането му администраторът трябва да мигрира инсталираната система към новата версия.

Миграцията към нова версия на операционната система на всеки 6 до 9 месеца не е особено привлекателно за организации, които имат множество сървъри или имат критичен софтуер, чиято миграция към нова версия на операционната система крие рискове за работата на самата организация. За такива потребители, Ubuntu предлага версии, които се поддържат по 5 години. Тези версии са известни като LTS или “Long Time Support” („дългосрочна поддръжка”). Новите LTS версии не са специално съставени а се избират от останалите на всеки 2 години. Така например, версия 14.04 бе LTS докато следващите три версии 14.10, 15.04, 15.10 бяха „нормални“. Версия 16.04 също бе LTS. Последната, най-актуална към момента на излизане на тази книга, LTS версия е 18.04, като следващата LTS версия евентуално ще бъде 20.04.

Така очертаният цикъл на издаване на нови версии означава, че към даден момент може да съществува една най-актуална LTS версия и една най-актуална „нормална“ версия.

Настоящото изложение ще се концентрира върху LTS версията на Ubuntu Server. Ubuntu Server е избрана, тъй като инсталацията на Ubuntu Desktop е доста проста и директна, а изборът на LTS гарантира, че описаният инсталационен процес ще остане непроменен в следващите няколко години.

1. Проверка на хардуерните изисквания

Ubuntu има някои минимални изисквания спрямо хардуера, които трябва да бъдат посрещнати. Като правило минималните изисквания за Ubuntu Desktop са по-високи от тези за Ubuntu Server, но трябва да се има предвид, че сървър, работещ на система която едва покрива минималните изисквания, реално не може да обслужва голям брой клиенти. Това не пречи подобен сървър да се използва за обучителни цели.

Минималните изисквания на Ubuntu Server 18.04 LTS по спецификация са 32 битов процесор с 256 MB памет и 1.5 GB дисково пространство. В действителност успешната инсталация на Ubuntu Server върху подобна система зависи от множество специфични стъпки, които биха представлявали проблем за начинаещи потребители.

Тъй като голяма част от процесорите, произведени след 2010 година са 64 битови, стандартният инсталационен носител на Ubuntu Server вече не се предлага в 32-битов вариант. Инсталацията на Ubuntu върху 32-битова машина все още е възможно чрез алтернативен инсталационен носител с отделна инсталационна програма – т.нар. “Netboot CD”. Тази инсталационна програма обаче налага доста по-сложен процес на инсталация. По същия начин, въпреки че инсталираната система би се заредила успешно на система с по-малко от 256 MB памет, инсталационната програма на стандартния инсталационен носител

няма да сработи при по-малко от 512MB памет на системата, което отново налага използването на алтернативния инсталатор.

Това означава, че препоръчителните минимални характеристики на система с Ubuntu Server са 64 битов процесор с около 1 GHz тактова честота, 1 GB оперативна памет и поне 2GB дисково пространство.

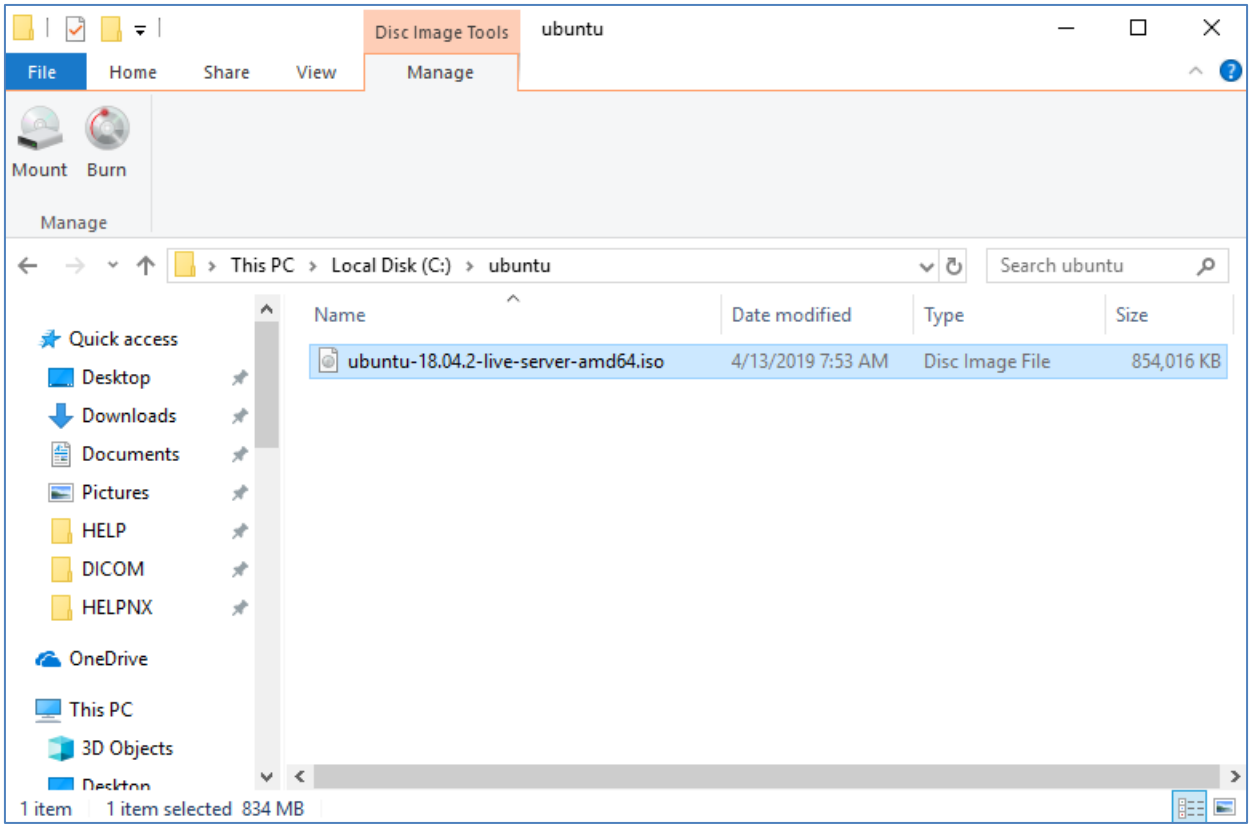
2. Изтегляне на ISO изображенията

Инсталацията на Ubuntu, независимо дали става въпрос за инсталация чрез DVD диск или USB флаш памет, започва с изтеглянето на ISO изображения. Намирането на правилното изображение за изтегляне може да стане през главния сайт. Полезно е да се знае че на адрес ***https://www.ubuntu.com/download/server*** винаги ще се намира страница с връзка към последната LTS версия.

2a. Създаване на DVD-ROM

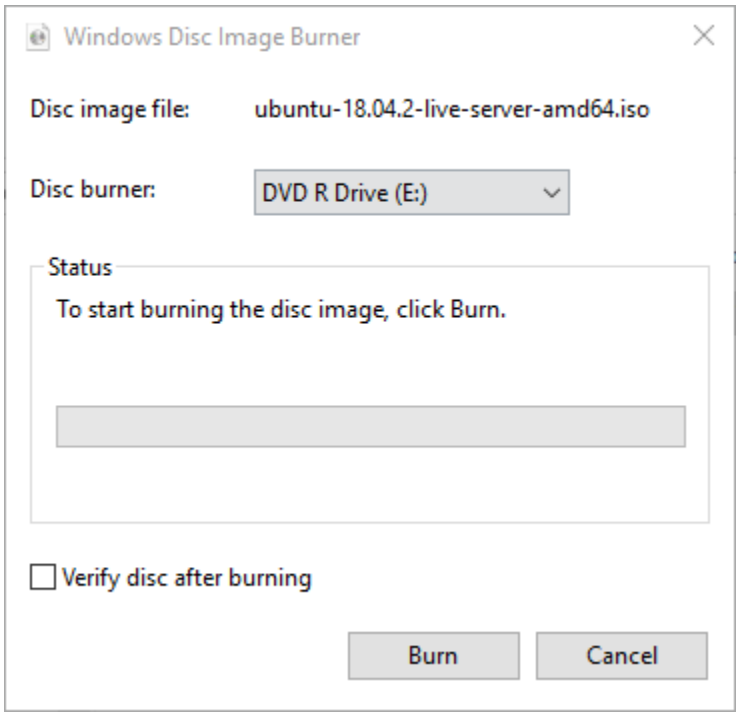
Ако инсталацията ще се извършва чрез DVD ROM, изображението, изтеглено на предната стъпка, трябва да се „изгори“ на носител. Последните няколко версии на Windows могат да извършват това без инсталирането на допълнителен софтуер.

Процедурата в Windows 10 започва със стартирането на файловия мениджър. Във файловия мениджър трябва да се щракне веднъж с левия бутон на мишката върху файла с изображението, който трябва да има разширение ***.iso***. При това във менюто на файловия мениджър ще се появи контекстна секция ***Disk Image Tools / Manage***, която ще съдържа бутон „***Burn***”. Вж. фиг. A.1.



Фиг. А.1. Бутони за „изгаряне“ на ISO изображение на диск във файловия мениджър на Windows 10

При щракване на бутона, ще се появи диалоговия прозорец за запис на файл, който е показан на фиг. А.2. Потребителите следва да отметнат ***Verify disc after burning,*** което ще помогне за откриване на грешки при записа, след което трябва да натиснат бутона ***Burn***.

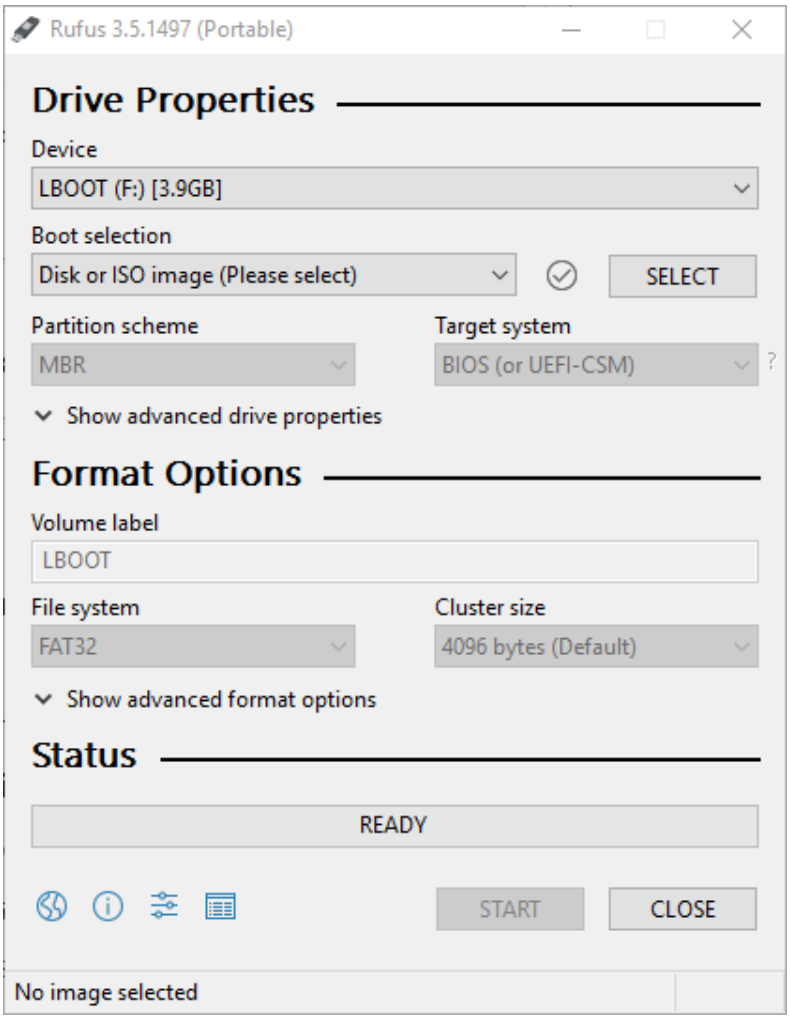


Фиг. А.2. Диалогов прозорец за изгаряне на диск в Windows 10.

2b. Създаване на Flash памет с Ubuntu

Възможно е системата на която се инсталира Ubuntu да няма DVD-ROM устройство. В такъв случай изтегленото изображение може да се запише на USB флаш диск. За целта могат да се използват множество инструменти. Един от тях е Rufus - ***https://rufus.ie/***.

Програмата има преносима версия, която не изисква инсталация. При стартирането и, потребителят вижда диалоговия прозорец, показан на фиг. А.3.



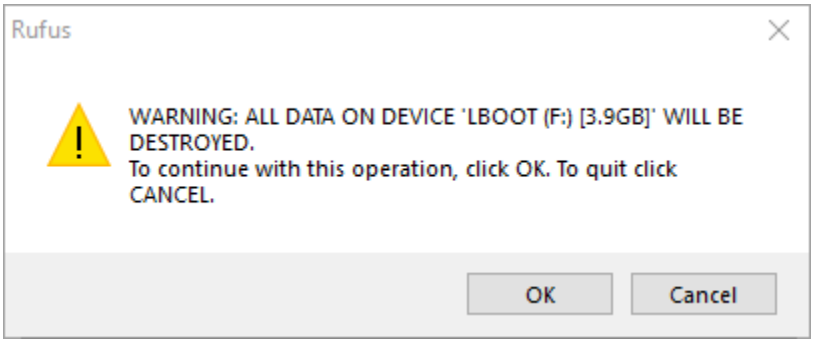
Фиг. А.3. Главен прозорец на Rufus

В него той трябва внимателно да избере от падащия списък ***Device*** това USB устройство, което ще бъде форматирано. След това потребителят следва да щракне на бутона ***Select*** за да избере файла с ISO изображението, който ще бъде записан на USB устройството.

След извършване на тези операции, потребителят може да натисне бутона ***Start***, което ще започне процеса на запис. Възможно е преди стартиране на записа, потребителят да трябва да отговори на няколко въпроса и да премине през няколко потвърждения.

По-новите дистрибуции на Ubuntu са ISOHybrid, което позволява те да се запишат в един от два режима – т.нар. ***ISO Image mode*** или ***DD Image mode***. Всеки от режимите би следвало да работи, но при евентуални проблеми, винаги следва да се изпробват и двата.

На фиг. А.4. е показан и последния диалог, който предупреждава потребителя, че след началото на процеса по запис, съществуващите данни на USB устройството



Фиг. А.4. Главен прозорец на Rufus

За. Стартиране на инсталационната програма

За да започне инсталацията, системата трябва да бъде инструктирана да зареди операционна система от подготвения по-рано инсталационен носител. Това се извършва по различен начин на различните компютри в зависимост от инсталирания BIOS.

По принцип повечето от модерните BIOS системи имат графичен интерфейс за конфигурация и по подразбиране стартират операционната система от CD или USB при положение, че такива, съдържащи валидна операционна система, са поставени при начално стартиране на системата.

Важно е да се знае, че най-голямата промяна в BIOS системите в последните години бе преминаването към UEFI. UEFI е нов, несъвместим

стандарт за BIOS системи, който се отразява и на начина, по който разпределението на дисковите дялове се записва на диска.

Ако компютърът, на който ще се инсталира Ubuntu, няма инсталирана операционна система и поддържа UEFI, потребителят следва да конфигурира BIOS за UEFI зареждане на системата. Ако на системата има инсталирани други дялове на вече инсталирани операционни системи, потребителят следва да използва старата схема за записване на информацията на дисковете, известна като MBR.

3b. Стартиране на инсталационната програма във виртуална машина на емулатора VirtualBox

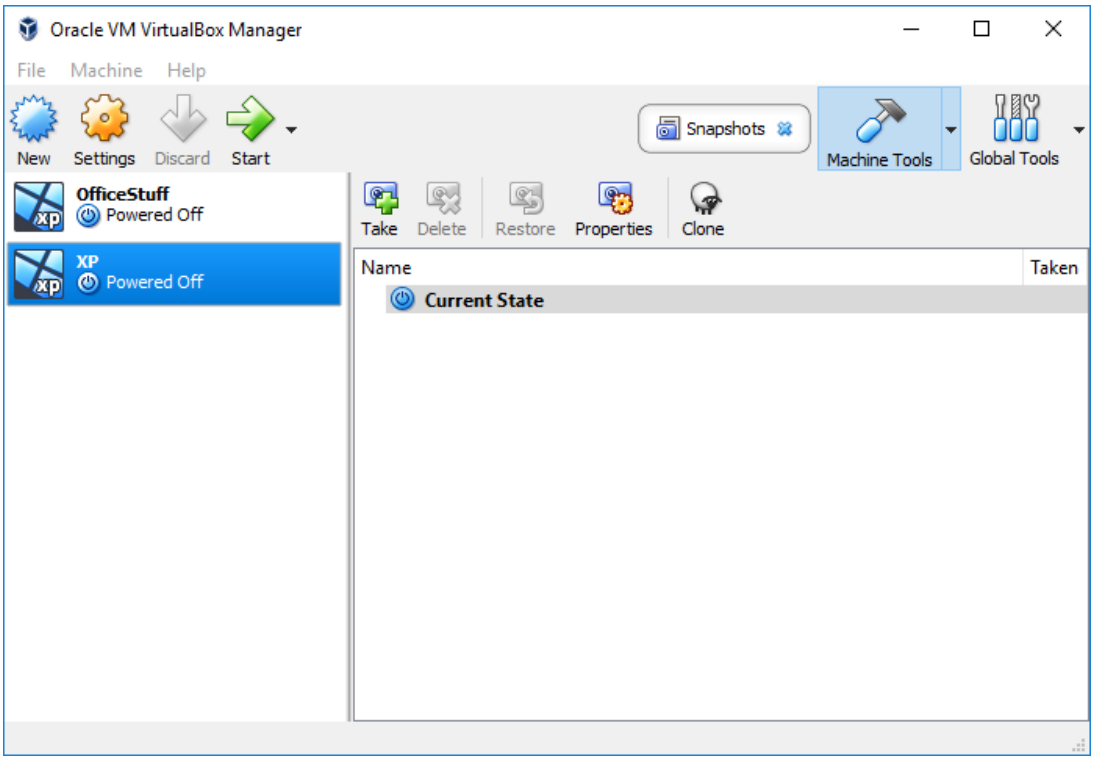
При невъзможност за инсталиране на Ubuntu на хардуерен сървър, той може да се инсталира като виртуална машина под управлението на редица емулатори, като например VmWare Player, HyperV и VirtualBox.

Настоящата точка ще демонстрира най-важните моменти при инсталация на системата върху VirtualBox под Windows.

Актуална версия на VirtualBox може да се изтегли от адрес:

https://www.virtualbox.org/wiki/Downloads

Процесът по инсталация не е особено сложен и включва инсталирането на няколко специализирани драйвера. След завършването на инсталацията, програмата следва да бъде стартирана, при което ще се покаже главният ѝ прозорец (фиг. А.5).

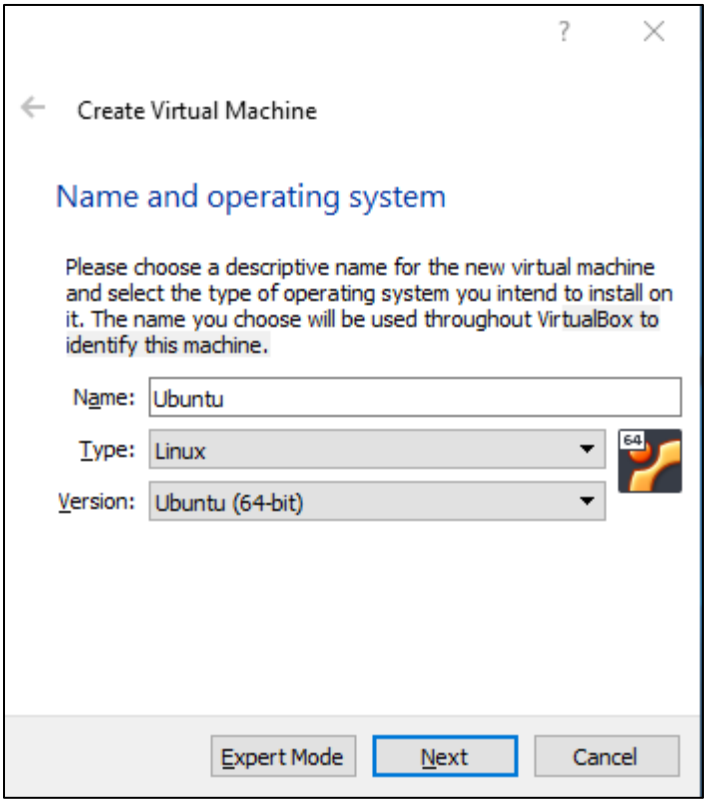


Фиг. А.5. Главен прозорец на VirtualBox с инсталирани две виртуални машини

За създаване на виртуална машина, от лентата с инструменти на главния прозорец следва да бъде натиснат бутона ***New***. Щракването върху този бутон стартира съветника за създаване на виртуална машина, който преминава през няколко стъпки.

В първата стъпка (показана на фиг. А.6.) потребителят трябва за зададе типа на операционната система (***Linux***) и версията на дистрибуцията (***Ubuntu 64-bit***), както и произволно избрано, описателно име, за виртуалната машина, което трябва да бъде попълнено в полето ***Name***.

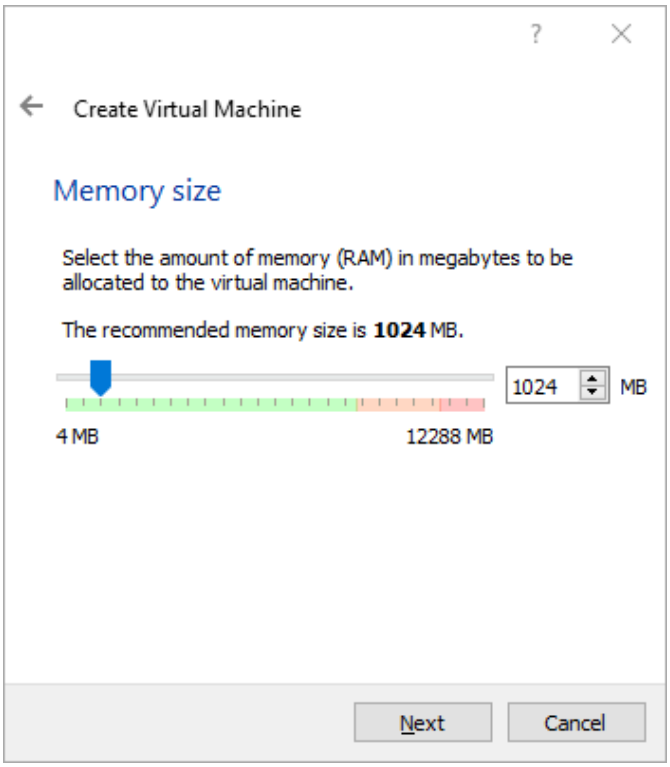
След приключване на въвеждането, потребителят трябва да натисне бутона ***Next***.



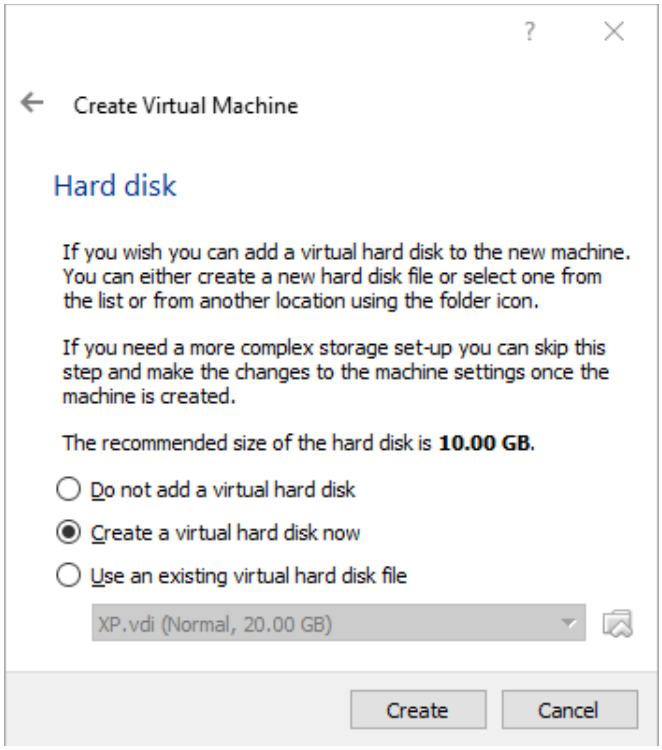
Фиг. А.6. Съветник за създаване на нова виртуална машина – първа страница

На втората страница на съветника, потребителят трябва да укаже размера на оперативната памет, която ще бъде заделена за виртуалната машина. За Ubuntu, VirtualBox ще задели по подразбиране 1 GB памет, което, както бе споменато, е достатъчно за учебна инсталация. (Фиг. А.7).

На следващите стъпки, VirtualBox ще изиска от потребителя да укаже детайли за дисковете на виртуалната машина. По подразбиране за Ubuntu, VirtualBox ще предложи на потребителя да създаде нов виртуален диск с размер от 10 GB (фиг. А.8). Потребителят следва да щракне бутона ***Create*** и да приеме това предложение.

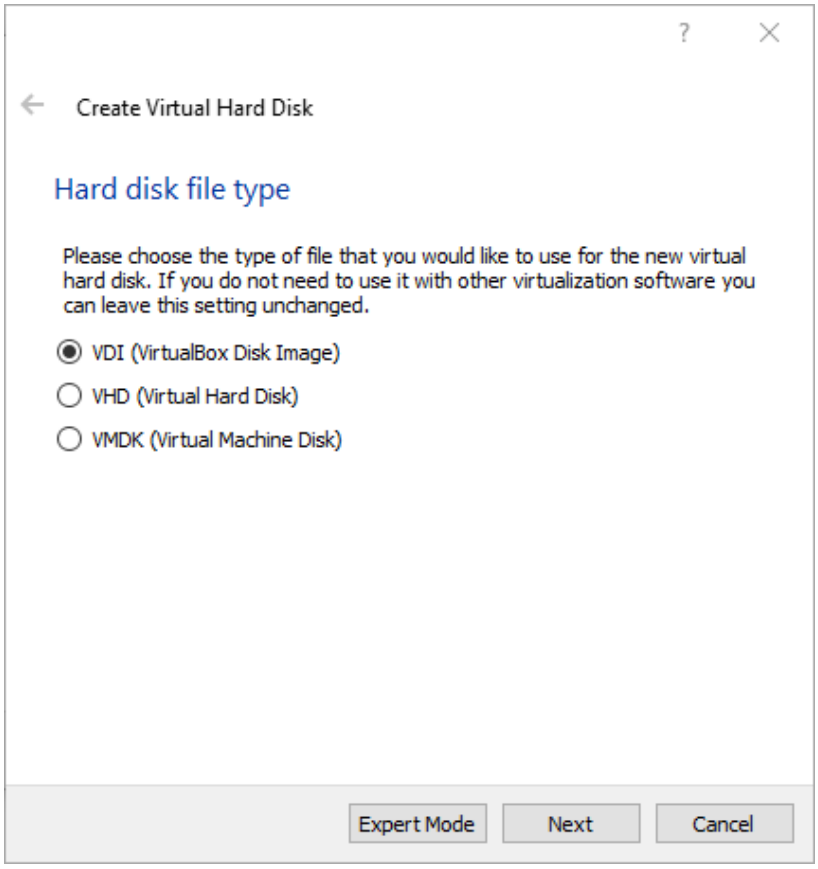


Фиг. А.7. Съветник за създаване на нова виртуална машина – втора страница



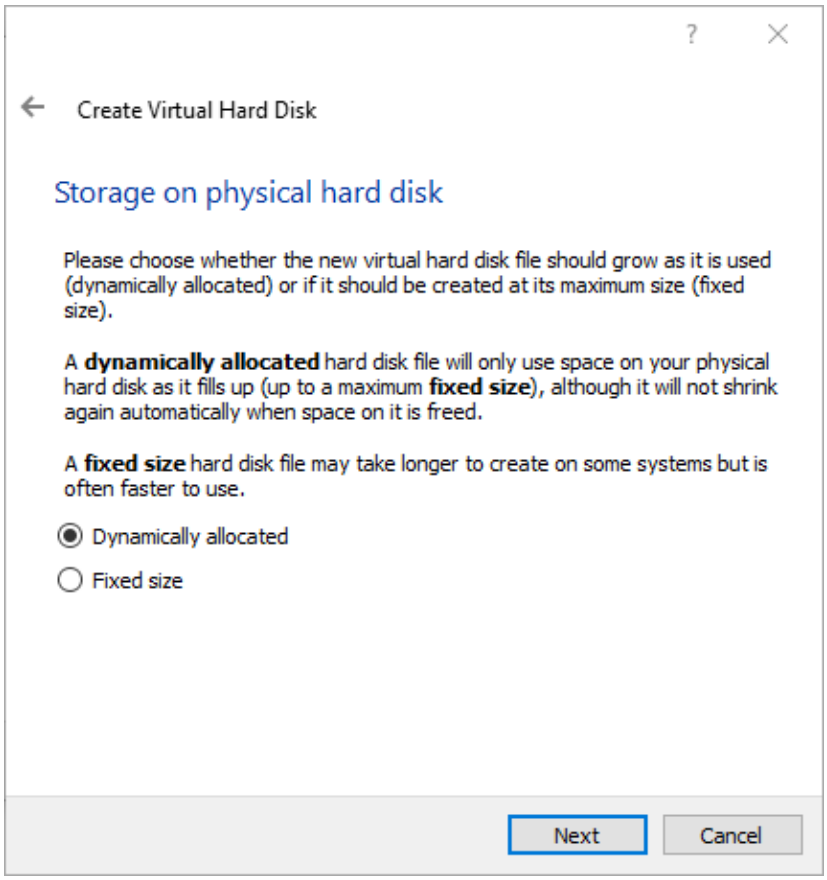
Фиг. А.8. Съветник за създаване на нова виртуална машина – трета страница

Щракването на бутона ***Create*** ще стартира нов съветник, специално насочен към създаването на виртуални дискове. На първата страница потребителят ще бъде запитан за типа на диска. Типът на диска има значение ако виртуалният диск ще се споделя или мигрира между различни емулатори. Това едва ли е притеснение на тепърва обучаващите се, поради което тук следва да се остави изборът по подразбиране – VDI и да се щракне на бутона ***Next***. (Фиг. А.9).



Фиг. А.9. Съветник за създаване на нов виртуален диск – първа страница

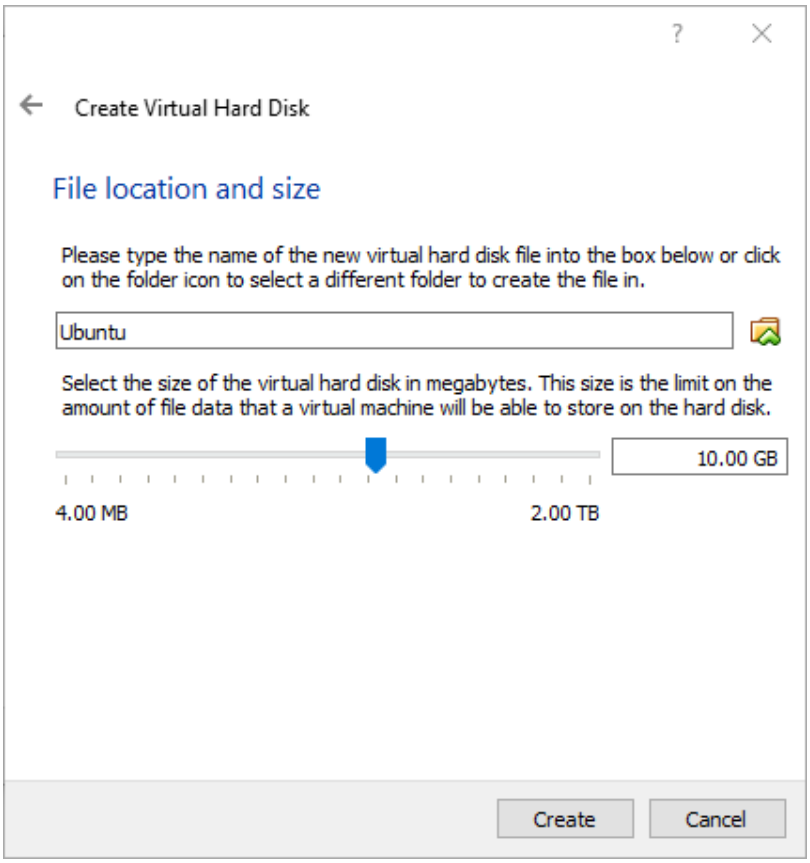
Втората страница от съветника за създаване на дискове пита потребителя дали цялото място за диска ще бъде заделено предварително или вместо това ще се заделя динамично (фиг. А.10).



Фиг. А.10. Съветник за създаване на нов виртуален диск – втора страница

Разликата между динамичното заделяне и предварителното заделяне е минимална и може да бъде усетена най-вече в понятията на свободно пространство на истинския диск на машината. При предварителното създаване, файлът, който съдържа виртуалния диск, ще заеме толкова място, колкото потребителят е указал независимо дали на виртуалния диск има записани някакви данни. В контраст, файлът на виртуален диск, който е динамично заделен ще расте само когато върху виртуалния диск се записват данни.

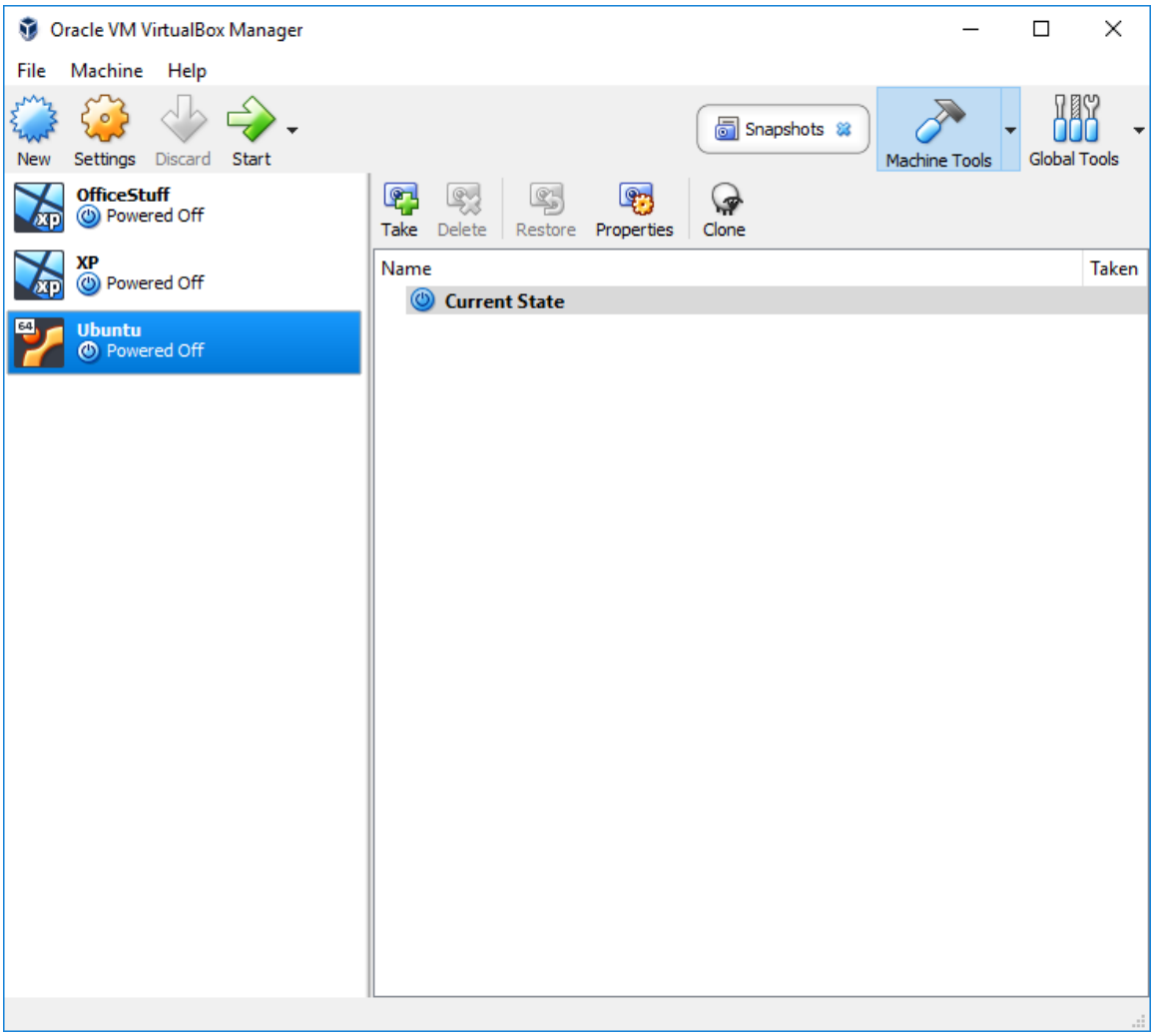
За учебна машина – е напълно подходящ изборът по подразбиране – ***Dynamically allocated***.



Фиг. А.11. Съветник за създаване на нов виртуален диск – трета страница

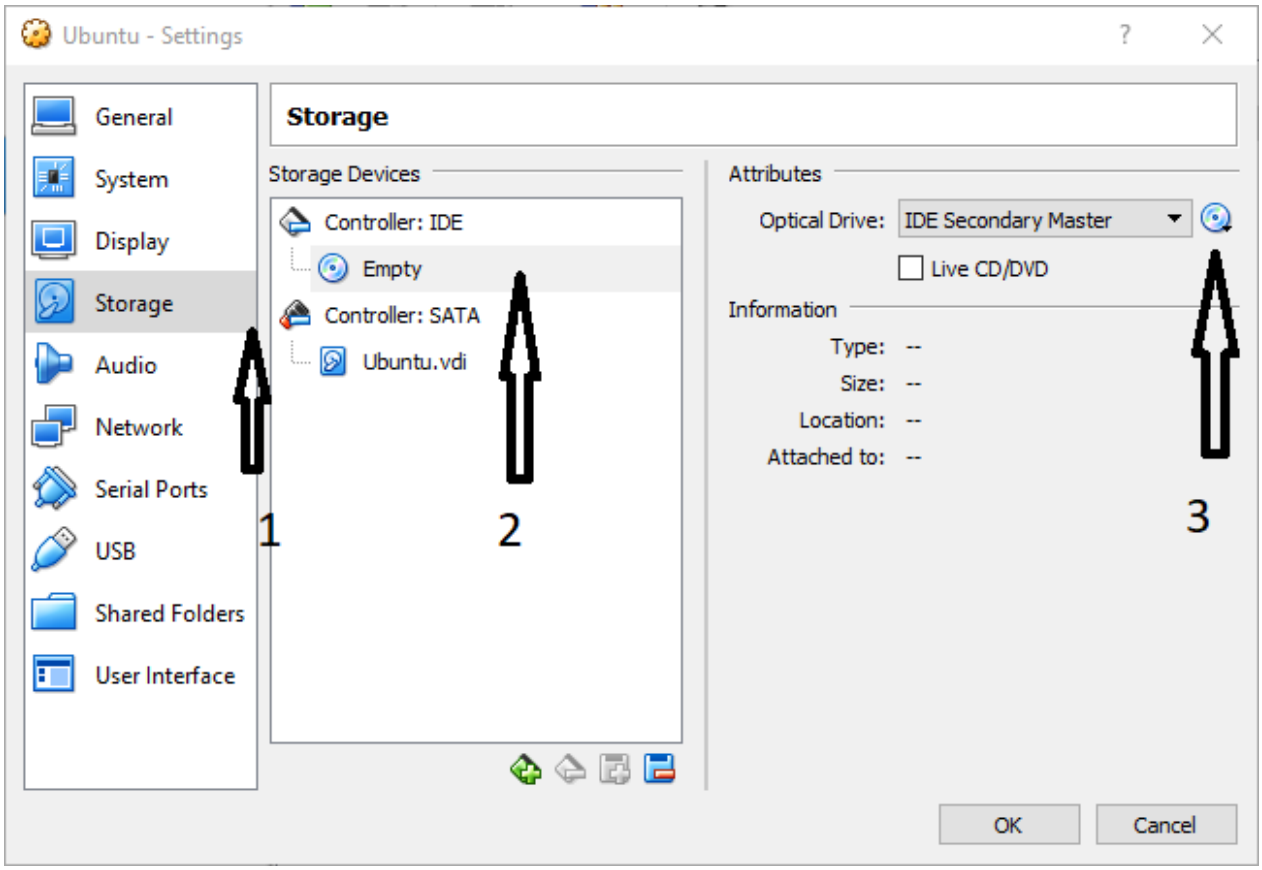
Третата и последна страница на съветника за създаване на дискове позволява на потребителя да промени големината, предложена от VirtualBox, както и да зададе име на файла, който ще съдържа виртуалния диск. За учебна машина не е необходимо да бъдат променяни настройките по подразбиране в този прозорец. Потребителят следва да щракне на бутона ***Create***, което ще завърши процеса по създаване на диск.

С приключването на съветника за създаване на дискове приключва и съветника за създаване на виртуални машини. Единствената операция, която остава, е прикачването на ISO изображение с инсталационните файлове на Ubuntu.



Фиг. А.12. Маркирана виртуална машина с име ***Ubuntu*** и бутони ***Start*** и ***Settings*** в главното меню на VirtualBox

За целта потребителят трябва да избере виртуалната машина и да щракне на бутон ***Settings*** (фиг. А.12). Това ще отвори диалога с настройки (фиг. А.13), който е организиран в множество страници, достъпни от списъка, поставен вляво. От този списък потребителят трябва да избере страница ***Storage***, след което да щракне върху иконата на CD в списъчното поле ***Storage Devices***. Това ще визуализира в най-дясната половина настройки за съдържанието на емулираното от VirtualBox DVD.



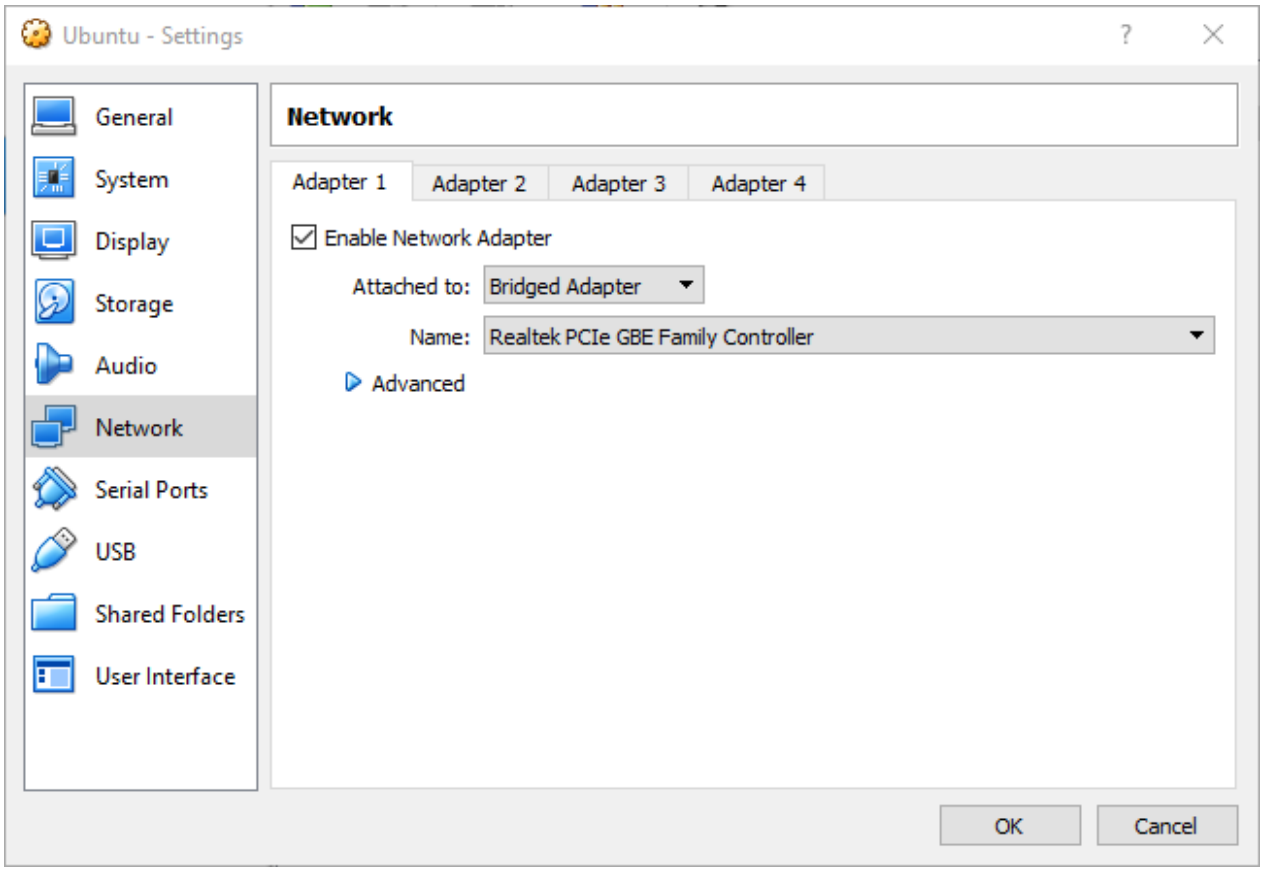
Фиг. А.13. Стъпки за визуализиране на настройките на виртуалното DVD устройство

Ако читателят е следвал предните стъпки и е записал ISO изображението на DVD носител, той може да избере да използва истинския носител, в който случай в настройките трябва да бъде отметнато полето ***Live CD/DVD***. Ако читателят разполага само с ISO изображение, той следва да щракне върху бутона с иконата, представляваща на CD (Вж. стрелка 3 на фиг. А.13), и да избере ***Choose Virtual Optical Disk File***. Това действие ще отвори стандартен диалог за избор на файл, в който потребителят трябва да посочи местоположението и името на файла, който съдържа ISO изображението.

За да улесни настройката на мрежата, потребителите, които използват домашни рутери с DHCP сървър следва да направят още една стъпка – от същия диалог с настройки, те трябва да изберат секция ***Network*** и в полето

Attached to на виртуалния мрежови адаптер, те трябва да изберат ***Bridged Adapter*** (фиг. А.14).

Избирането на ***Bridged Adapter*** ще свърже виртуалната мрежова карта на емулираната машина директно с мрежата, към която е свързана истинската машина. Това ще позволи на операционната система във виртуалната машина да вземе мрежови настройки директно от DHCP сървъра. В същото време, това означава, че всички потребители на мрежата ще виждат виртуалната машина като отделен компютър, свързан към мрежата и ще могат да се свързват директно към услугите, предлагани от виртуалната машина. Ако локалната мрежа не е домашна, това може да породи проблеми със сигурността.



Фиг. А.14. Настройки на виртуалния мрежови адаптер на VirtualBox

След като потребителят приключи с настройките, остава да натисне бутона ***OK***. На този етап виртуалната машина е готова за стартиране.

За да се стартира машината, потребителят трябва да маркира виртуалната машина и да избере от главното меню на VirtualBox бутона ***Start*** (вж. фиг. А.12.).

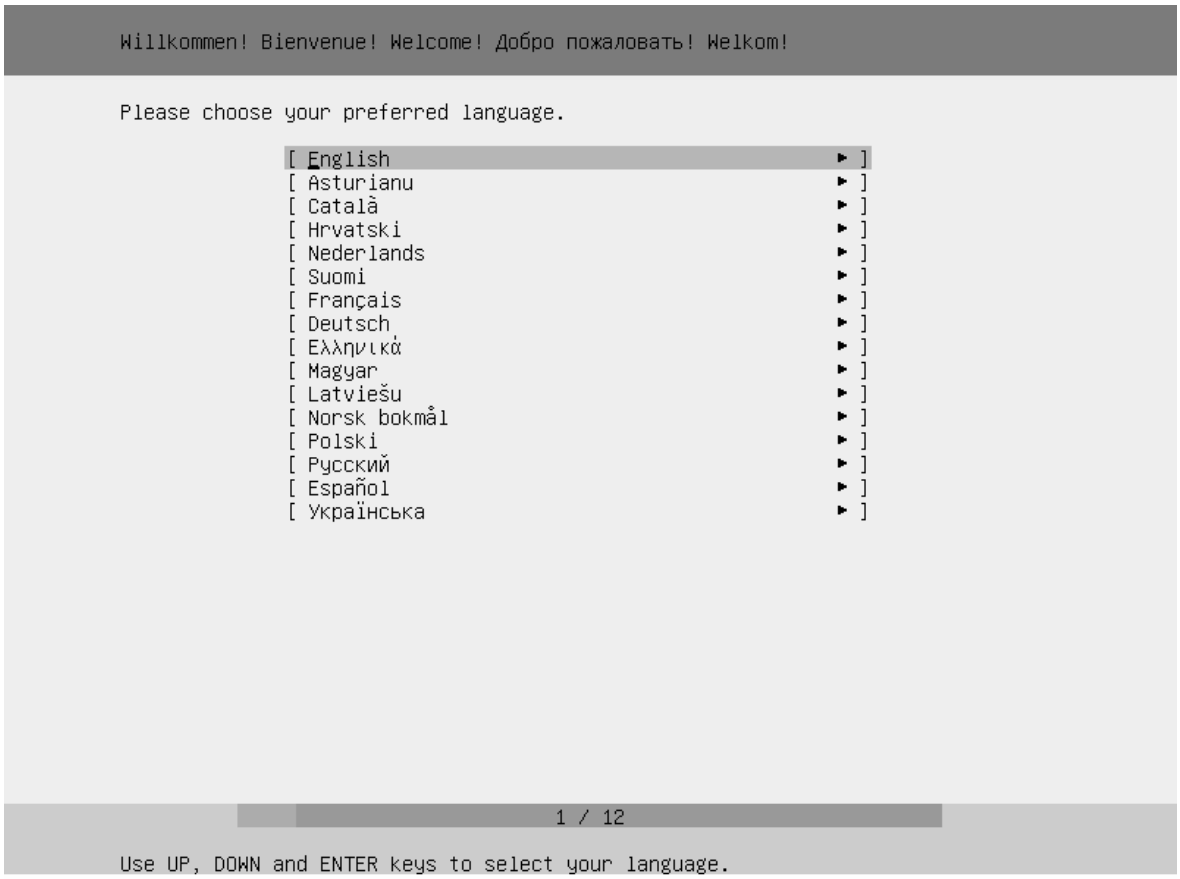
4. Инсталационна програма на Ubuntu Server

Ако потребителят е поставил инсталационен носител и е инструктирал правилно системата да зареди от него, след процеса на начално зареждане ще бъде стартирана инсталационната програма на Ubuntu. Тя започва работа с въпрос към потребителя за езика на инсталацията. Както е показано на фиг. А.15, изборът по подразбиране е английски език.

На следващата стъпка от инсталацията (фиг. А.16) инсталационната програма пита потребителите за разположението на тяхната клавиатура. Този въпрос е важен, ако потребителят разполага с национализирана клавиатура, която не съблюдава изцяло стандарта QWERTY и/или възнамерява да използва други езици, които са различни от английски..

За целите на учебната инсталация (а и по принцип, тъй като повечето сървъри не се управляват директно от системната конзола), изборът по подразбиране – стандартна QWERTY клавиатура за английски език - е доста уместен вариант.

Иначе в тази стъпка на инсталацията, инсталационната програма предлага два пътя за настройка на клавиатурата. Първият е ръчен, при който потребителят е подканен да укаже ***Layout*** (т.е. език на клавиатурата), след което да укаже вариант на разположението на клавишите за този език (***Variant***).



Фиг. А.15. Начален екран на инсталатора на Ubuntu Server

Вариантът е важен, тъй като за повечето езици има различни разположения на клавиатурата. Някои от разположенията следват стандарт от далечното минало (по-точно 19-ти век), наречен QWERTY, докато другите следват логиката на разработения през 40-те години на 20-ти век DVORAK стандарт, който улеснява писането по десетопръстна система на пишеща машина. За български език двата варианта са известни като PHONETIC (QWERTY) и BDS (DVORAK).

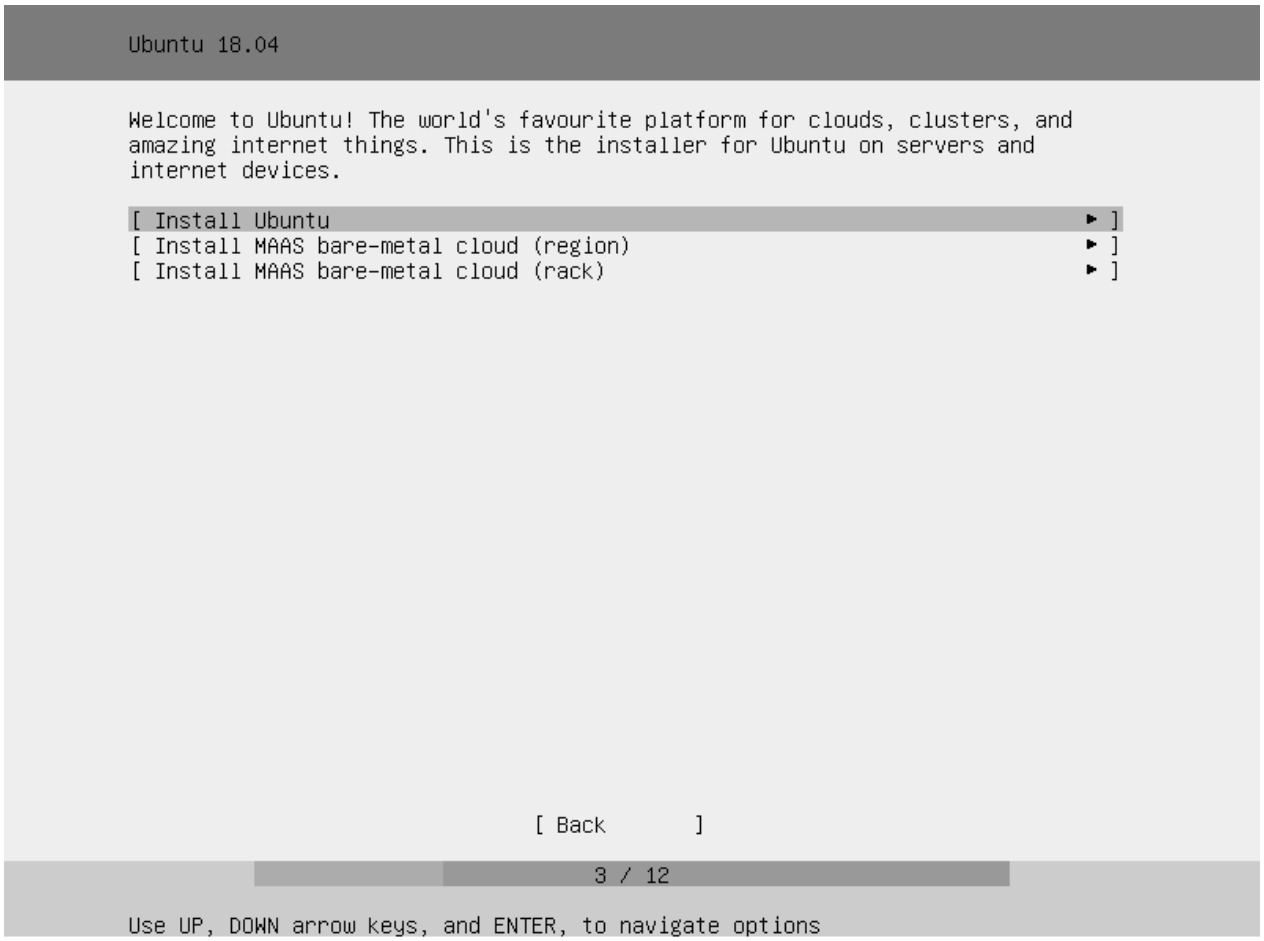
Инсталационната програма на Ubuntu Server предлага и друг вариант – ако потребителят не е наясно с разположението на клавиатурата си, системата ще помоли той да натисне клавиши с определени изобразени символи, по което Ubuntu ще познае разположението на клавиатурата.

За целта потребителят трябва да избере ***Identify Keyboard***.

Фиг. А.16. Начален екран на инсталатора на Ubuntu Server

На третата стъпка (фиг. А.17) потребителят трябва да избере какво точно възнамерява да инсталира от инсталационното DVD с Ubuntu Server. Заедно с Ubuntu Server, Cannonical – компанията, която съставя дистрибуцията, представя и решение за изграждане на облачни дейтацентрове. Ако на тази стъпка потребителят избере ***Install MAAS bare-metal cloud***, той ще инсталира това решение.

За инсталиране на отделен сървър, потребителят трябва да остави опцията, която е активна по подразбиране – ***Install Ubuntu***.

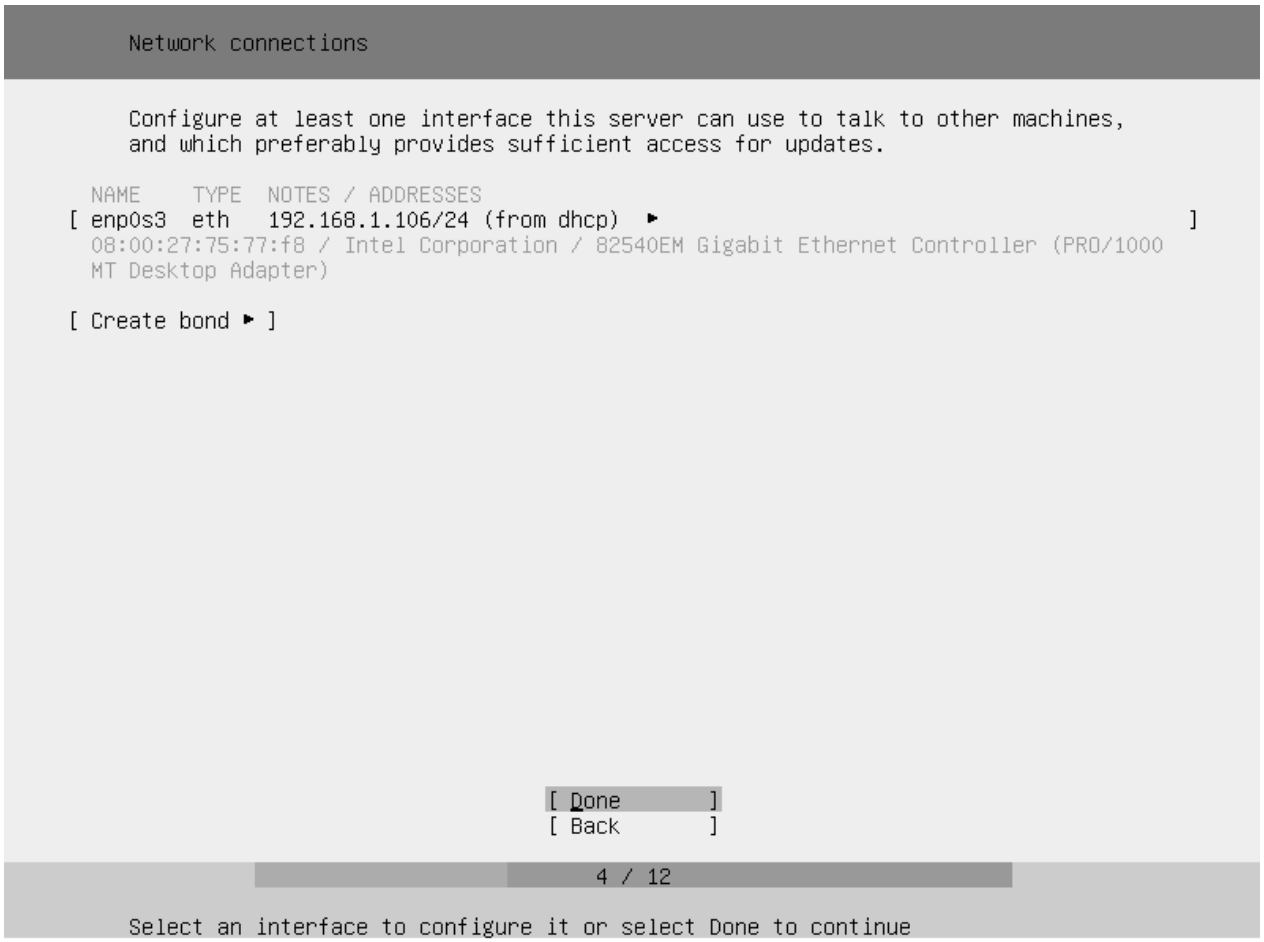


Фиг. А.17. Избор на платформата, която ще бъде инсталирана

Стъпка 4 от инсталационната програма (фиг. А.18) е от голямо значение, тъй като на тази фаза ще бъдат конфигурирани мрежовите карти на системата, като за целта ще им бъде зададен IP адрес.

Най-лесният вариант на конфигурация на мрежата е когато в локалната мрежа функционира DHCP сървър. При това условие, адресът ще бъде получен от DHCP сървъра автоматично. В противен случай, потребителят ще трябва да въведе IP адрес, мрежова маска и шлюз (рутер) по подразбиране, както и адресите на един или повече DNS сървъри.

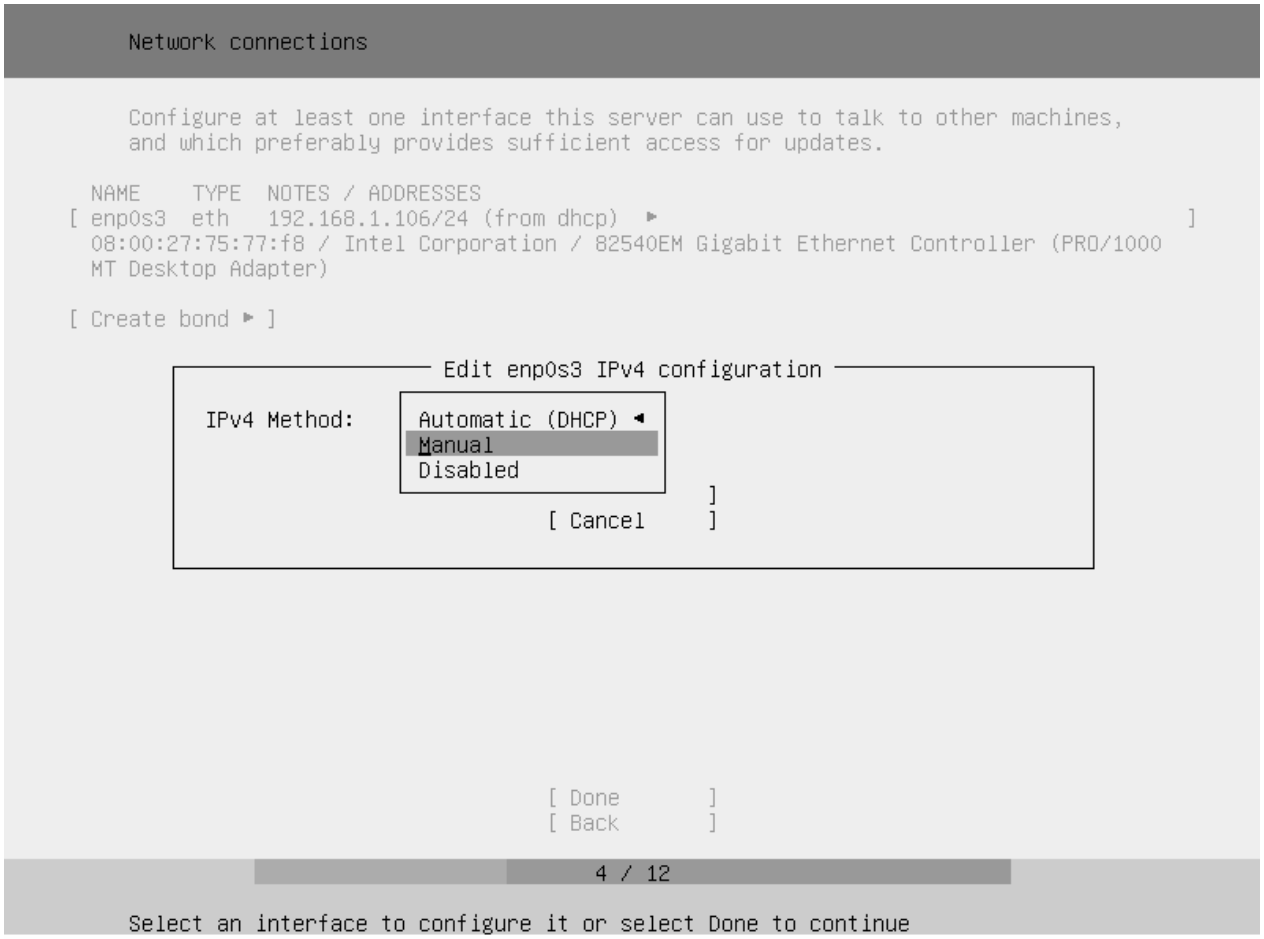
На фиг. А.18.а. е показан резултатът при използване на DHCP сървър, който е присъединил на машината адрес и маска 192.168.1.106/24.



Фиг. А.18.а. Автоматична конфигурация на мрежата чрез DHCP

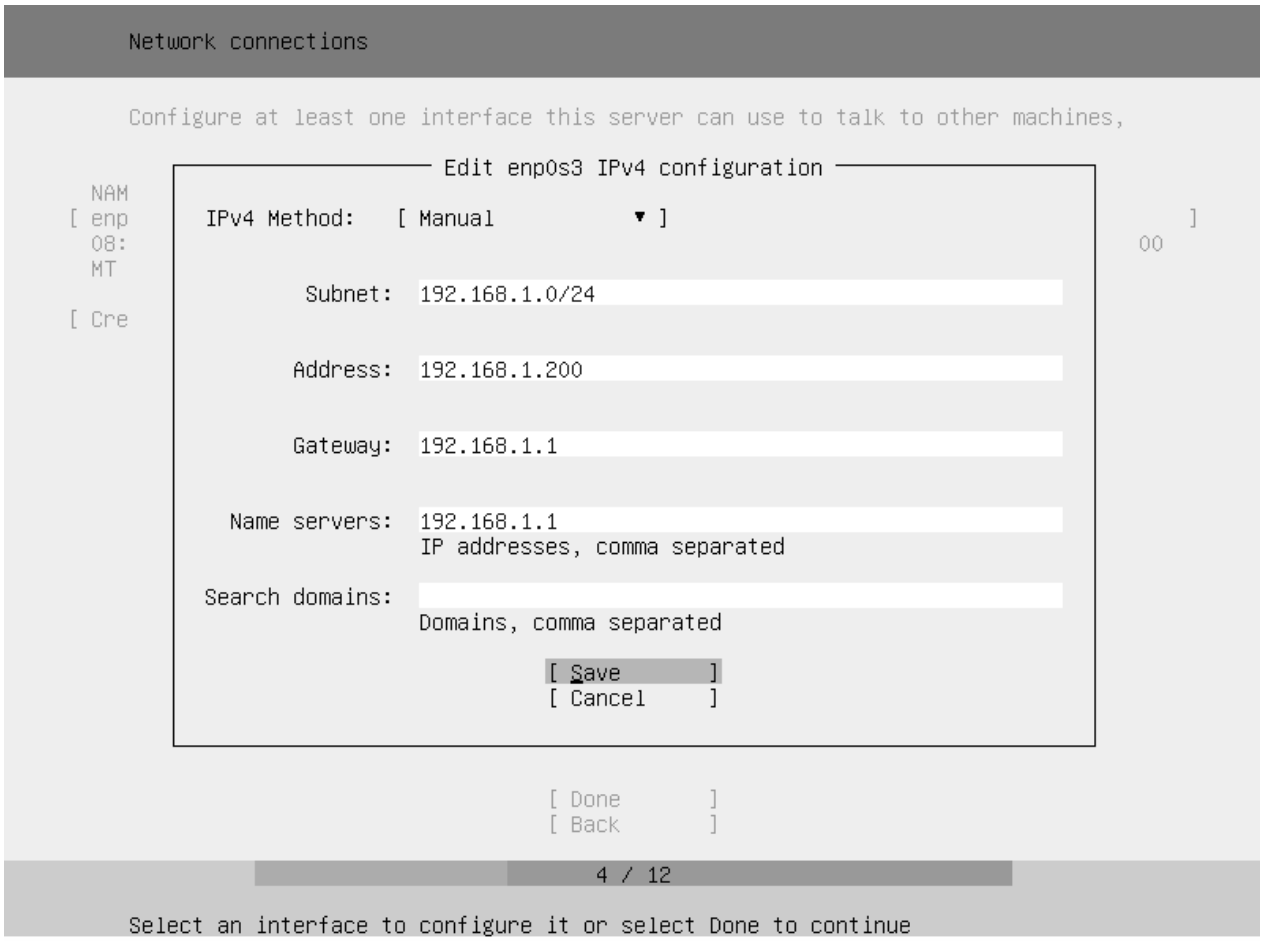
Ако в мрежата няма DHCP сървър или по някаква причина, дадените от DHCP сървъра настройки трябва да бъдат модифицирани, потребителят трябва да се придвижи върху реда в списъка с мрежовата карта и да натисне клавиша ***Enter***. На фиг. А.18.а. има един ред за единствената мрежова карта на системата, ограден в квадратни скоби, но при други системи е възможно да има множество редове за многото на брой мрежови карти, които тези системи ще имат. След това, от появилото се меню, той трябва да избере ***Edit IPv4*** или ***Edit IPv6*** в зависимост от това какъв адрес трябва да бъде настроен. Изборът на адрес в подобни мрежи ще изисква консултация със системния администратор, тъй като грешно зададен IP адрес със сигурност ще предизвика проблеми с

връзката, а в някои случаи може да предизвика и конфликт, който да попречи на други потребители на компютърната мрежа.



Фиг. А.18.б. Ръчна настройка на мрежова карта

След избора на вида адресна конфигурация, потребителят ще трябва да избере дали да избере (фиг. А.18.б) автоматична DHCP конфигурация (изборът по подразбиране), да направи ръчна конфигурация (***Manual***) или да забрани напълно мрежовата карта, при която операционната система няма да я използва (***Disabled***).



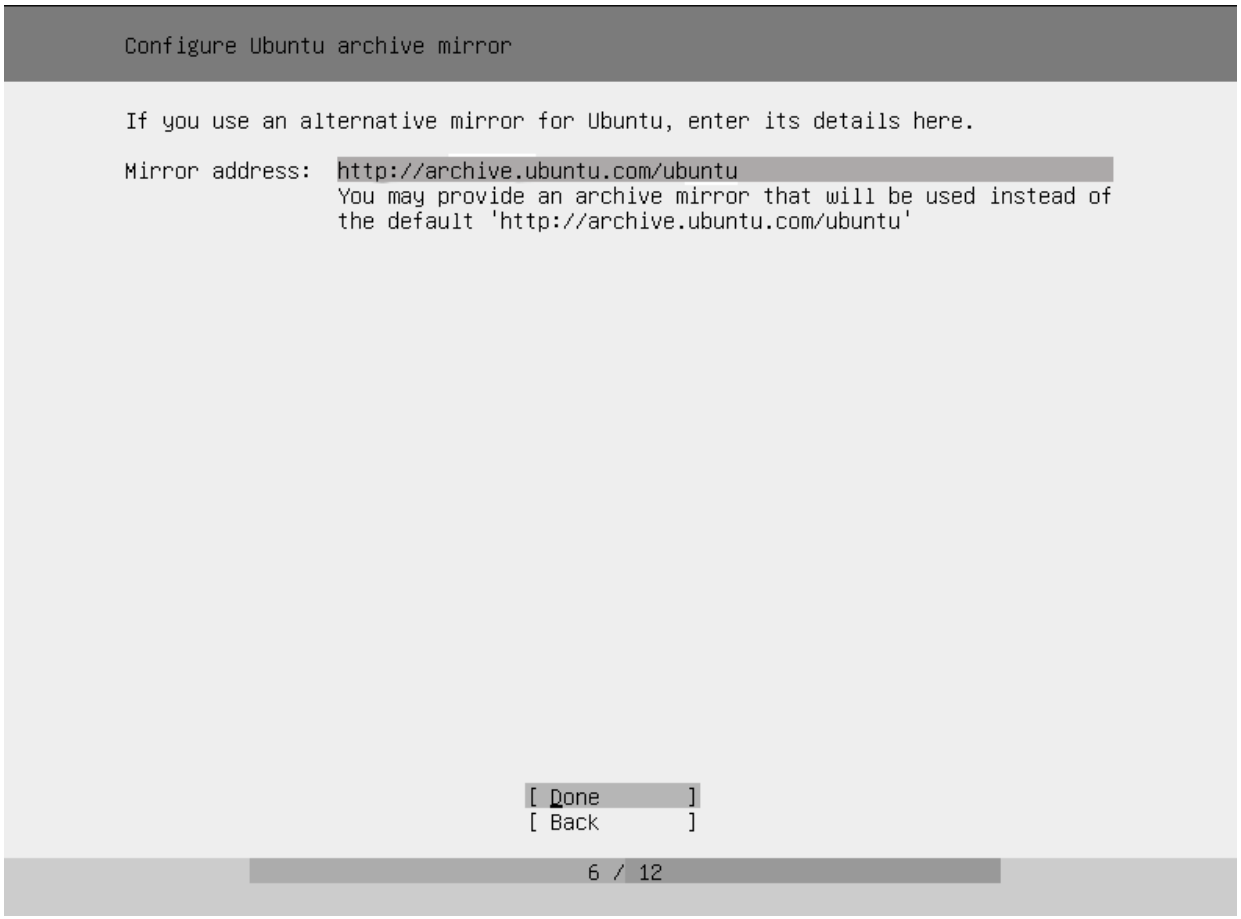
Фиг. А.18.в. Ръчна настройка на мрежова карта

След приключване на основната мрежова конфигурация, инсталационната програма на Ubuntu ще изведе запитване за въвеждане на адресите на евентуален прокси сървър. Потребителят трябва да се осведоми от администратора на мрежата дали мрежата използва прокси сървър. Повечето модерни мрежи по света не използват прокси сървъри, поради което обикновено на тази стъпка (фиг. А.19), не се налага въвеждането на каквото и да било.

На следващата по ред стъпка (фиг. А.20), Ubuntu ще попита потребителят дали желае да използва специфичен архив на Ubuntu за инсталацията.

Фиг. A.19. Конфигуриране на прокси сървър

Софтуерът на Ubuntu е организиран в пакети, които се съхраняват на централни сървъри в Интернет – т.нар. „архиви“. Част от тези пакети се копират на различните видове инсталационни DVD-та. За разлика от самите архиви, които се обновяват своевременно с поправки, обновявания на сигурността и новодобавени пакети, DVD-тата веднъж изтеглени и записани, не могат да се обновяват. Поради тази причина инсталацията на Ubuntu ще използва Интернет, за да се свърже с един от тези архиви и да провери дали няма обновени версии на някои от пакетите, които ще бъдат инсталирани. Ако такива има, те ще бъдат изтеглени от Интернет, вместо да бъдат използвани остарелите им версии от DVD носителя.



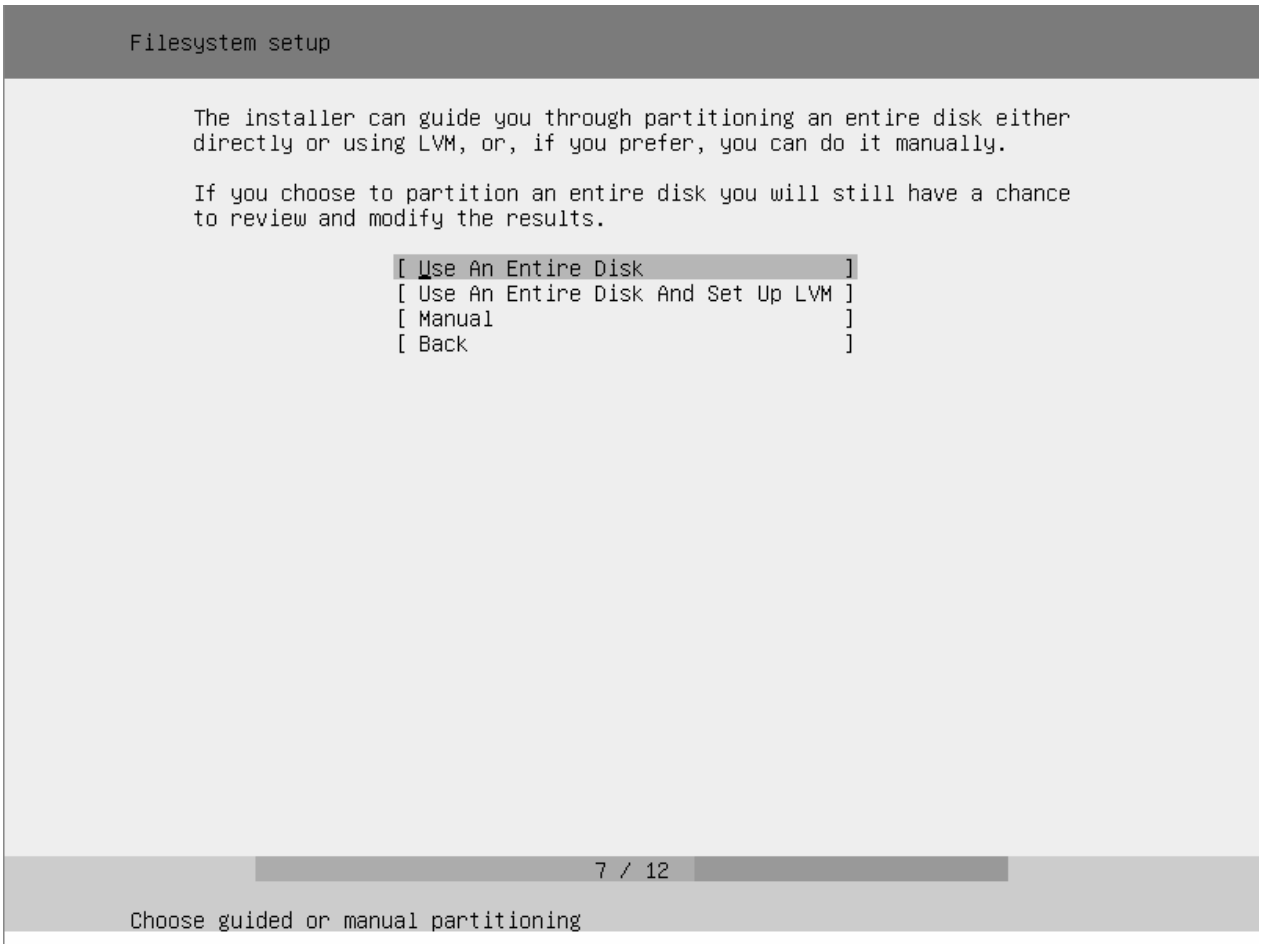
Фиг. А.20. Конфигуриране на прокси сървър

Изборът по подразбиране ***http://archive.ubuntu.com/ubuntu*** е напълно адекватен за учебна система.

На следващата стъпка от инсталацията се заделят дискови дялове. Ubuntu предлага две алтернативи – използване на целия диск за Ubuntu, **което ще изтрие всички съществуващи дялове и операционни системи**, или ръчен режим на създаване на дялове.

Начинаещите потребители вероятно ще искат да посветят цяла система на Ubuntu, тъй като ръчното разделяне на дялове и зареждането на повече от една операционна система от тях изисква специфични знания за начина на разделяне на дисковете, както и за редица особености в

процеса по зареждане на отделните системи. Неправилно изпълнен, този процес крие рискове от загуба на информация. Поради тази причина е силно препоръчително начинаещите потребители, които нямат специална система, която да заделят за Ubuntu Server, да използват емулатор, както бе очертано в предните точки.



Фиг. A.21. Конфигуриране на дисковите дялове за Ubuntu Server

На фиг. A.21. е показан диалогът за избор на стратегия за разделяне на дисковете с двете опции, които водят до това целият диск да бъде изтрит и заделен за Ubuntu. ***Use An Entire Disk*** е изборът по подразбиране и указва на

инсталатора да използва целия диск. ***Use An Entire Disk And Set LVM*** указва на инсталатора да използва целия диск и да създаде ***LVM*** дисков масив.

Фиг. А.22. Избор на диск за инсталацията на Ubuntu Server

LVM (от “Logical Volume Manager” – “Мениджър на логически томове”) е специална система за управление на дискови дялове в Ubuntu, която поддържа разрастващи се и смаляващи се дялове, като в същото време играе важна роля изграждането на софтуерни RAID масиви. Поради тази причина LVM е по-скоро опция, която се използва от истинските сървъри.

При инсталирането на учебна машина тя спокойно може да бъде изпусната, като на тази стъпка спокойно може да се остави изборът по подразбиране - ***Use An Entire Disk***.

След като потребителят е избрал използването на цял диск, от него се изисква да покаже кой от дисковете на системата ще се използва, тъй като е възможно системата да има множество физически дискове. Това се извършва на следващата стъпка, която е показана на фиг. А.22.

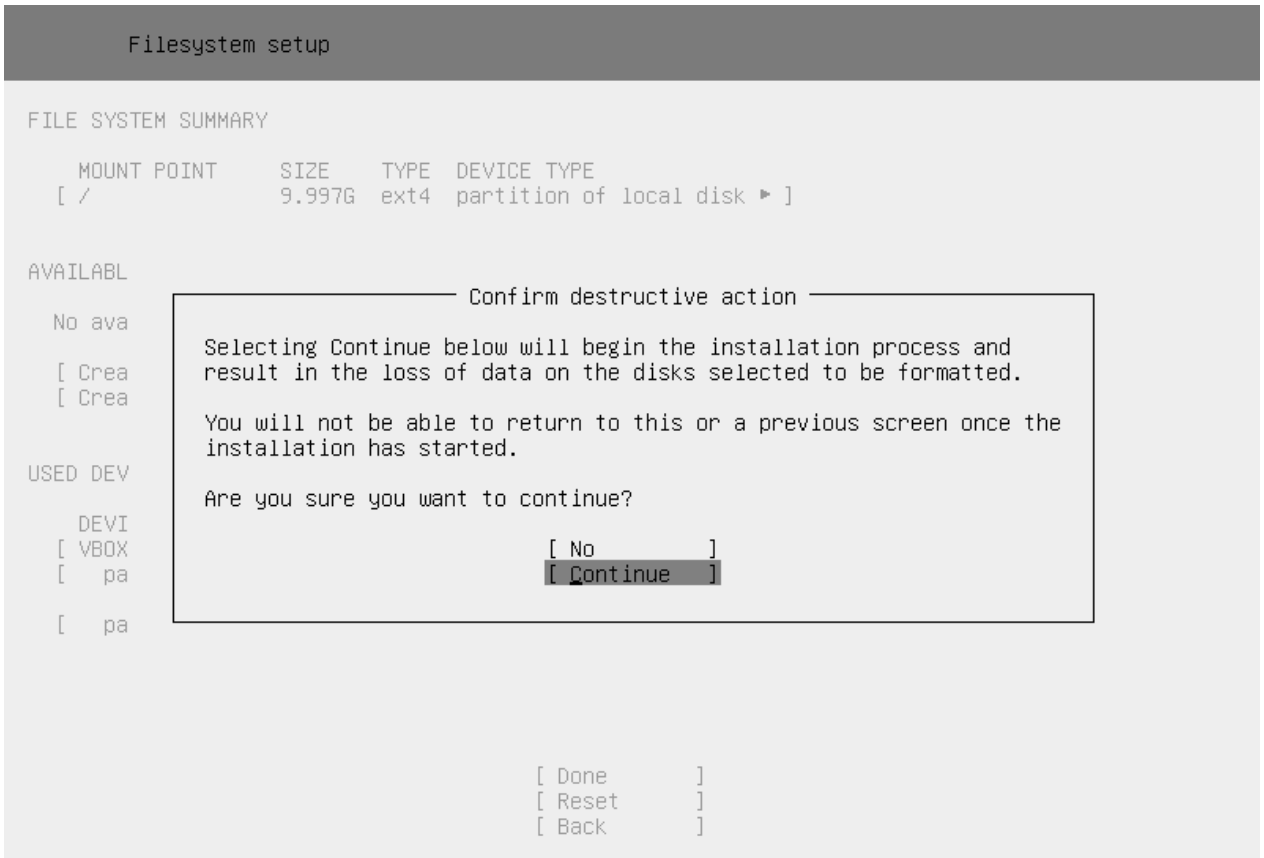
На фиг.А.22. е демонстрирана ситуация, при която системата има само един диск.

```
        Filesystem setup

FILE SYSTEM SUMMARY

    MOUNT POINT     SIZE    TYPE  DEVICE TYPE
  [ /               9.997G  ext4  partition of local disk ► ]


AVAILABLE DEVICES

  No available devices

  [ Create software RAID (md) ► ]
  [ Create volume group (LVM) ► ]


USED DEVICES

    DEVICE                                SIZE    TYPE
  [ VBOX_HARDDISK_VB793272dc-74e31d5b    10.000G  local disk ► ]
  [   partition 1                         1.000M (0%)        ► ]
        bios_grub
  [   partition 2                         9.997G (99%)       ► ]
        formatted as ext4, mounted at /



                                        [ Done       ]
                                        [ Reset      ]
                                        [ Back       ]

                                 7 / 12

        Select available disks to format and mount
```

Фиг. А.23. Обобщение на действията, които ще бъдат предприети от инсталационната програма

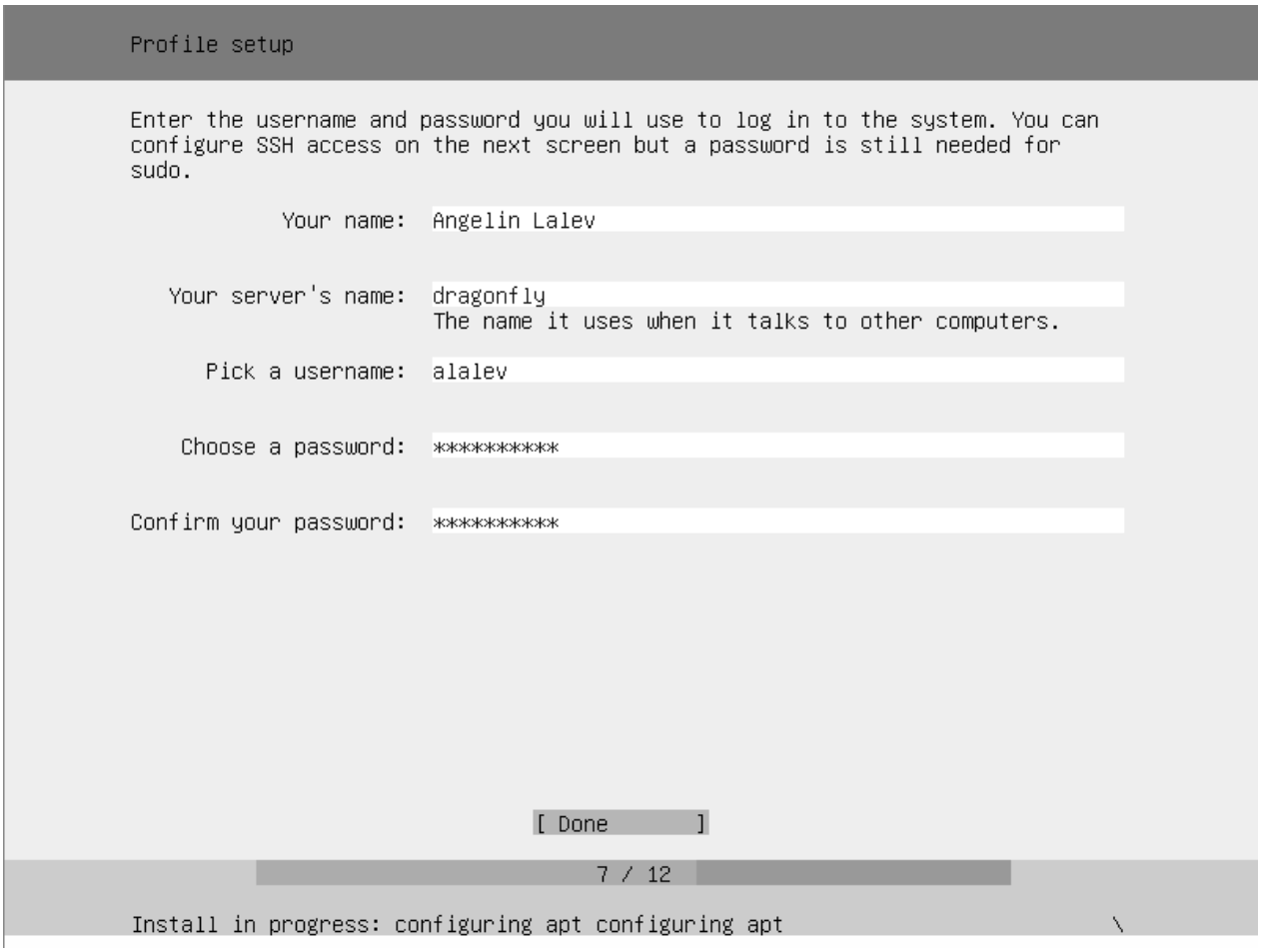
Конфигурацията на дисковете приключва с две стъпки (показани съответно на фиг. А.23 и А.24). Фиг. А.23. показва екрана за обобщение на действията, които ще бъдат предприети. На този екран са описани дисковете и дяловете, които ще бъдат създадени. Потребителят може да се откаже от описаните действия като избере ***Reset*** или ***Back***.

Ако потребителят избере ***Done***, той ще бъде запитан още веднъж дали иска да извърши действията, тъй като те ще доведат до изтриване на съществуващите дискови дялове, ако има такива. Диалогът е показан на фиг. А.24. В него потребителят трябва да се придвижи от ***No*** to ***Continue***, за да покаже съгласие с показаните на предния екран операции.



Фиг. А.24. Потвърждение на действията, които ще бъдат предприети от инсталационната програма

След тази стъпка инсталацията на Ubuntu Server ще започне, като докато тя се изпълнява, потребителят ще бъде запитан серия от въпроси, които касаят допълнителната настройка на бъдещата система.

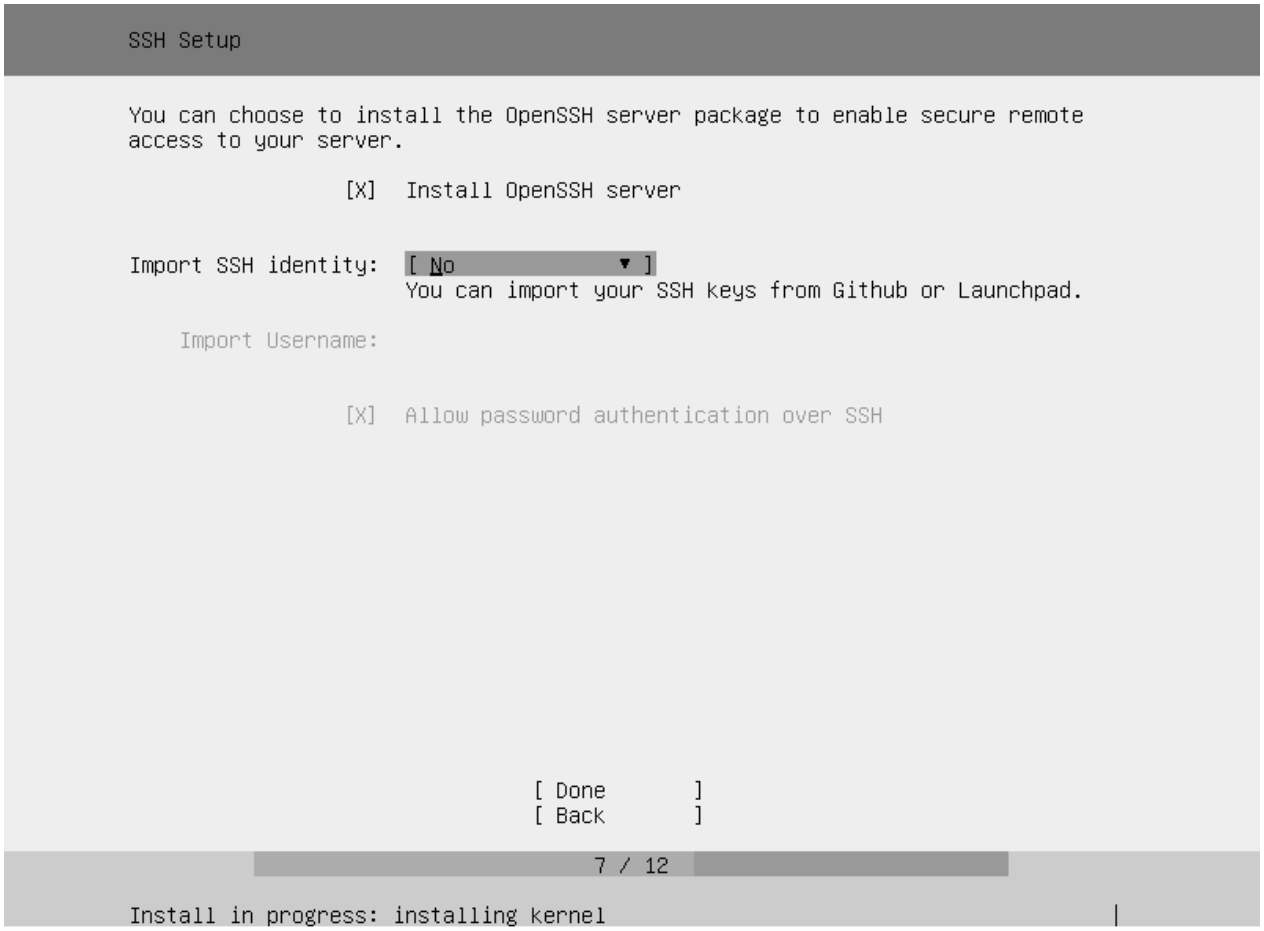


Фиг. А.25. Създаване на административен потребител

Първият от тези въпроси касае даването на име на системата, както и създаването на потребител, който ще бъде неин администратор. Този потребител ще бъде поставен от инсталационната програма в групите ***adm*** и ***sudoers***, което ще му позволи да администрира системата, като извиква команди с помощта на ***sudo***.

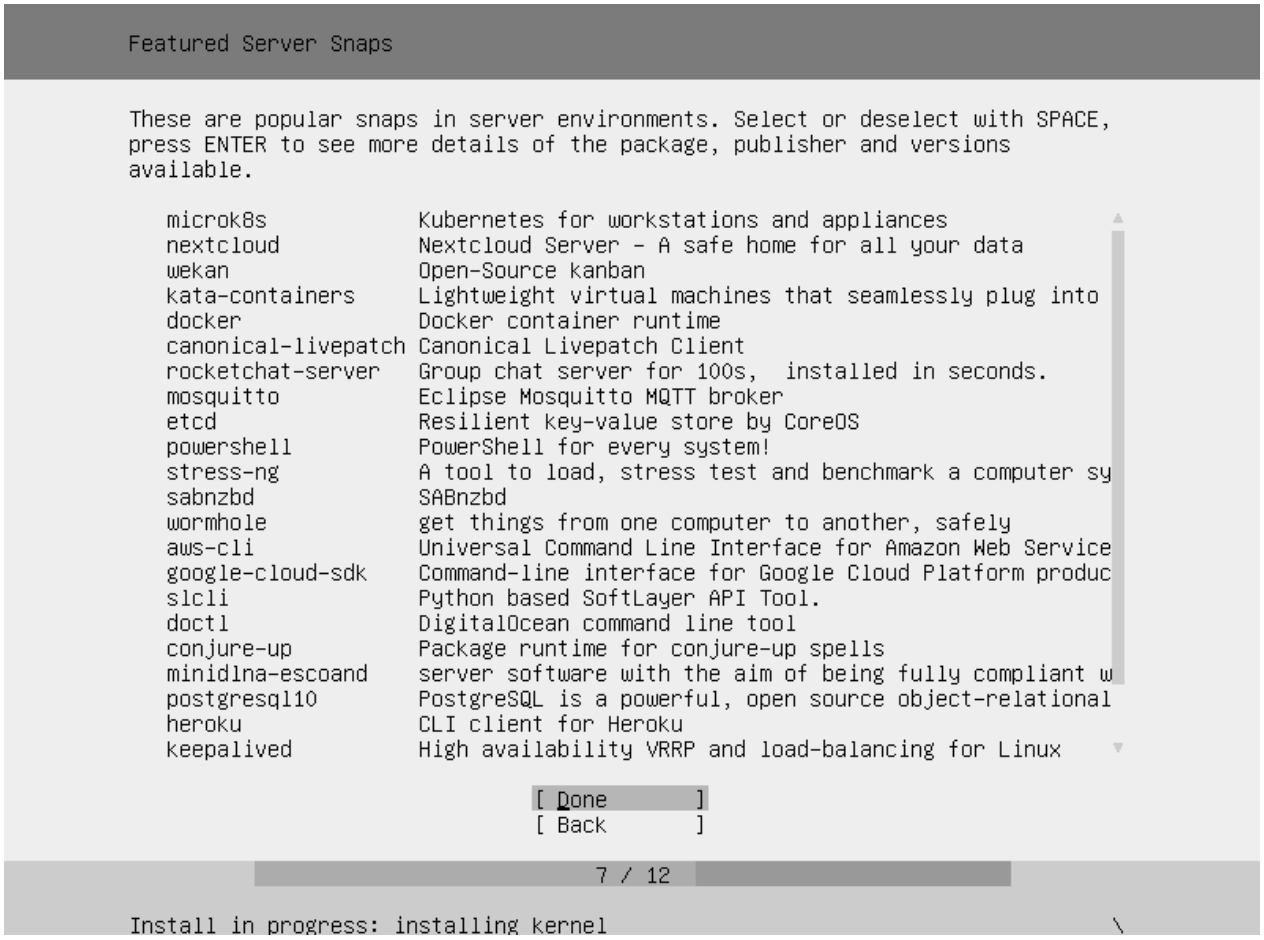
Основните полета са показани на фиг. А.25, и са както следва:

- ***Your name*** – пълно име (например име и фамилия или трите имена) на потребителя
- ***Your server's name*** – име на сървъра, който се инсталира. Потребителят може да избере произволно име тук и то ще бъде поставено в ***/etc/hostname***. За системи, които имат домейн име е най-добре това име да бъде написано тук.
- ***Pick a username*** - потребителско име, което потребителят следва да избере за себе си.



Фиг. А.26. Инсталиране и активиране на OpenSSH

- ***Choose a password*** – парола на потребителя
- ***Confirm your password*** – потвърждение на паролата на потребителя (двете стойности - ***Choose a password*** и ***Confirm your password*** – не се показват на екрана и трябва да са еднакви. Това е предпазна мярка против грешки при въвеждането на паролата, които биха предотвратили успешни влизания в системата по-късно).



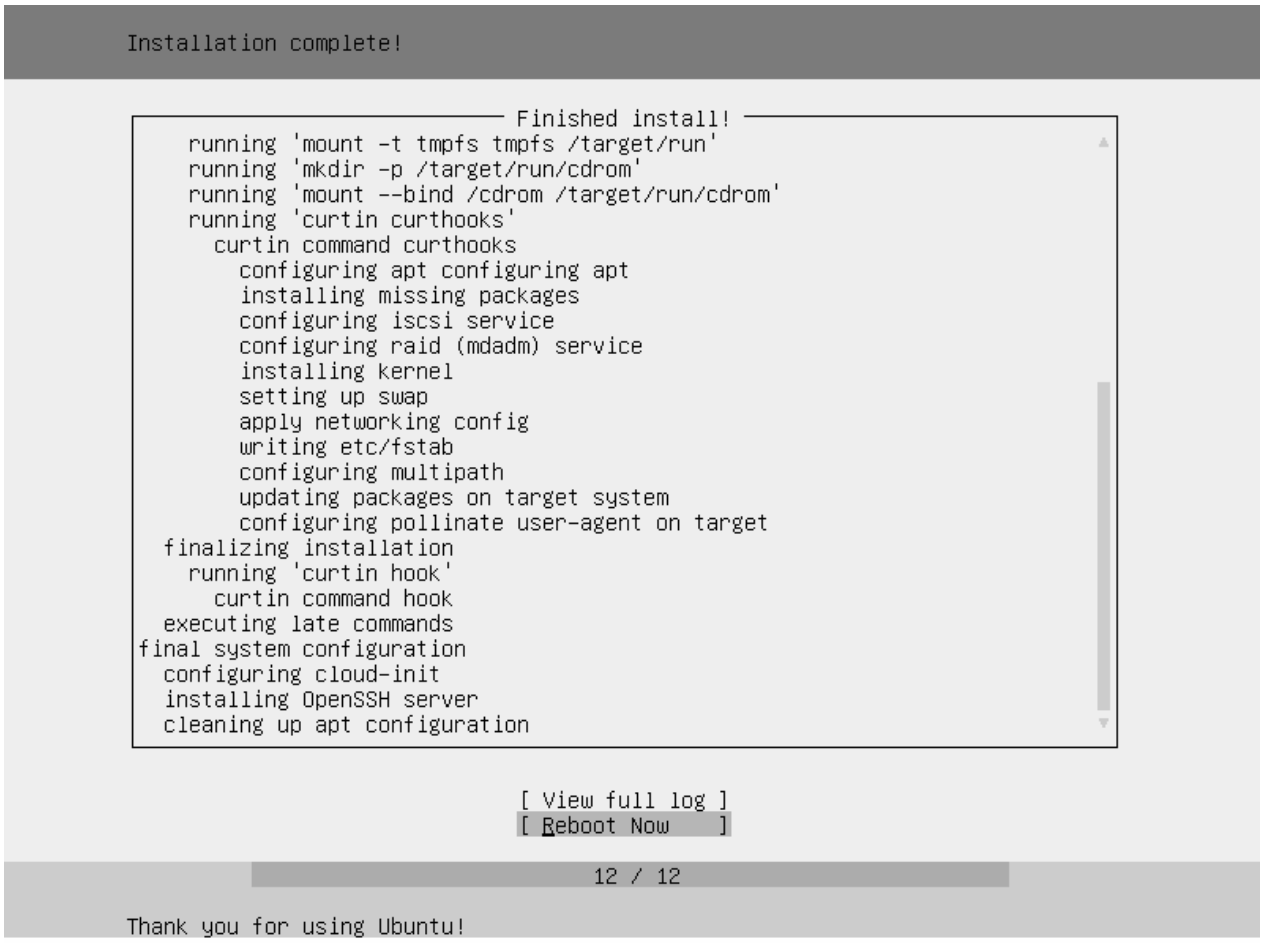
Фиг. А.27. Избор на снапове, които да бъдат инсталирани

На следващата стъпка (фиг. А.26), потребителят трябва да укаже дали възнамерява да се свързва отдалечено към системата с помощта на ***OpenSSH***.

Ако инсталацията на учебната машина е във виртуална машина и потребителят възнамерява да използва само системната конзола за свързване, ***OpenSSH*** не е задължителен.

При желание за тестване на посочените в гл. 1 методи за свързване или когато Ubuntu се инсталира върху физическа система, OpenSSH следва да бъде инсталиран. За целта потребителят трябва да отметне полето ***Install SSH server***.

Полето ***Import SSH identity*** позволява на системата да импортира ключове на потребителя, които са въведени в Github, или Ubuntu Launchpad. Начинаещите потребители вероятно няма да имат такива ключове, поради което тук могат да оставят избора по подразбиране – ***No***.



Фиг. А.28. Приключване на инсталацията и рестартиране на системата.

Като последен въпрос, инсталаторът на Ubuntu Server предлага инсталирането на някои стандартни платформено-независими пакети с готови приложни програми – т.нар. “снапове”. Инсталирането на такива снапове със системата не е необходимо. Ако възникне нужда, те могат да бъдат инсталирани по-късно.

След като потребителят отговори и на този въпрос и избере ***Done*** (фиг. А.27), инсталационната програма ще покаже журнала на инсталацията, който се попълва на живо докато евентуално инсталацията приключи. Когато това се случи, потребителят ще види бутон ***Reboot Now*** (фиг. А.28). Преди да го натисне, потребителят трябва да премахне инсталационния носител, за да предотврати повторно зареждане на инсталационната програма.

ПРИЛОЖЕНИЕ В – ИНСТАЛИРАНЕ НА CENTOS 7.6

CentOS е безплатна дистрибуция, която води началото си от Red Hat Enterprise Linux (RHEL). RHEL е водещата платена дистрибуция на Линукс в света, която се предлага от Red Hat Inc. Тъй като Red Hat, по силата на GPL лиценза, са задължени да предоставят изходния код на операционната си система, групата около CentOS започна съществуването си чрез компилирането и комплектуването на този код в безплатна дистрибуция. Резултат от това е, че днес CentOS е функционално съвместима с RHEL и се отличава от RHEL основно по това, че RHEL има комерсиална поддръжка, докато CentOS не предлага такава.

Основните версии на CentOS следват тези на RHEL, като последната актуална версия на CentOS към момента на написване на това приложение е CentOS 7.6. Въпреки че в най-скоро време се очаква излизането на нова главна версия на RHEL – версия 8, което ще доведе до евентуалното излизане на CentOS 8, настоящата версия ще получава обновявания за сигурността до 2024 година.

3. Проверка на хардуерните изисквания

За разлика от много други дистрибуции, стандартният инсталационен носител на CentOS поддържа едновременно 32 и 64 битови системи.

Официалната документация не посочва точни препоръки за минималната памет на системата, на която ще бъде инсталиран CentOS 7, като дава за 64 битовата версия ориентировъчната цифра от 1 GB памет на логически процесор/ядро.

CentOS 7 също се нуждае от препоръчителните 10-20 GB свободно дисково пространство за инсталация на системата.

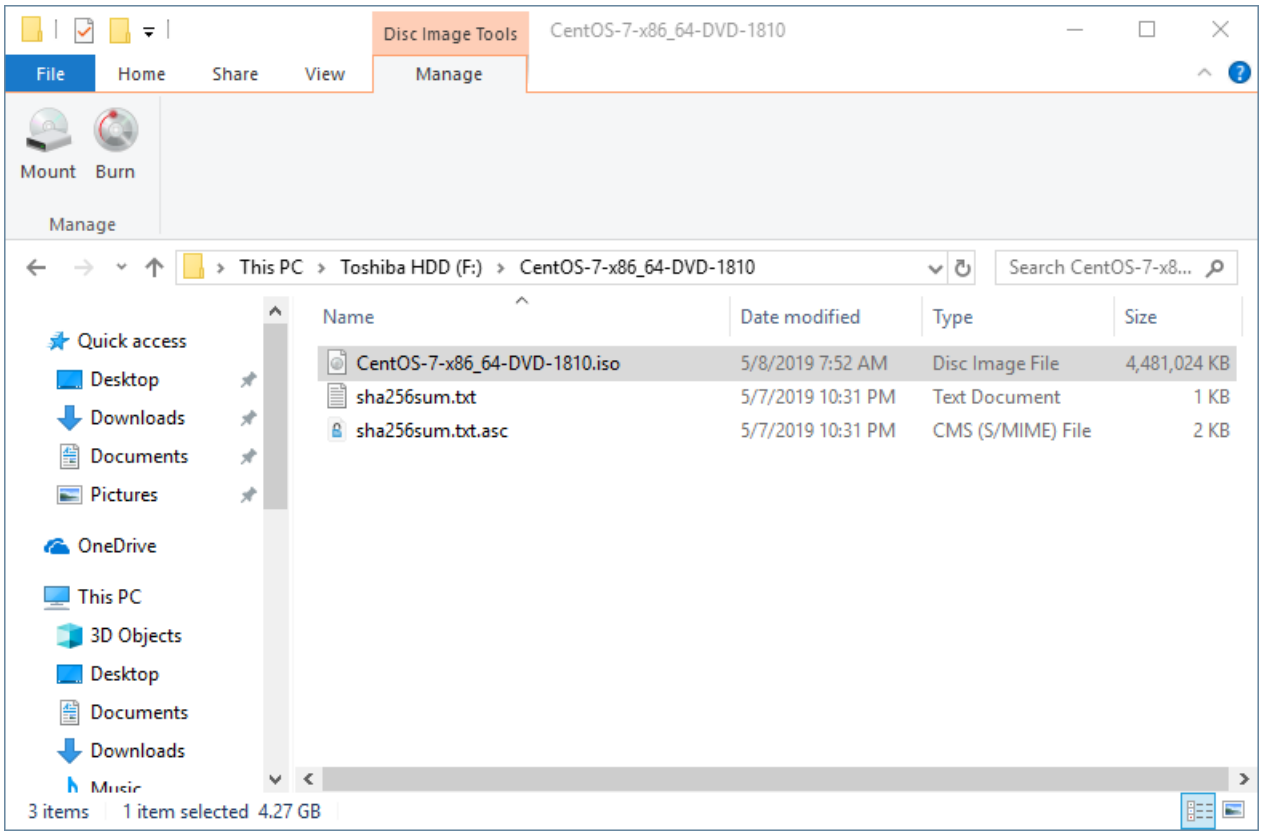
4. Изтегляне на ISO изображенията

Инсталацията на CentOS, независимо дали става въпрос за инсталация чрез DVD диск или USB флаш памет, започва с изтеглянето на едно от стандартните ISO изображения. Потребителят трябва да избере между минимална и пълна версия на DVD носителя. Двата носителя са достъпни на страницата ***https://www.centos.org/download/***. Пълният носител предлага повече пакети, като разликата спрямо използването на минималния е в това, че при минималния може да се наложи допълнително изтегляне на пакети от Интернет в процеса на инсталацията.

2а. Създаване на DVD-ROM

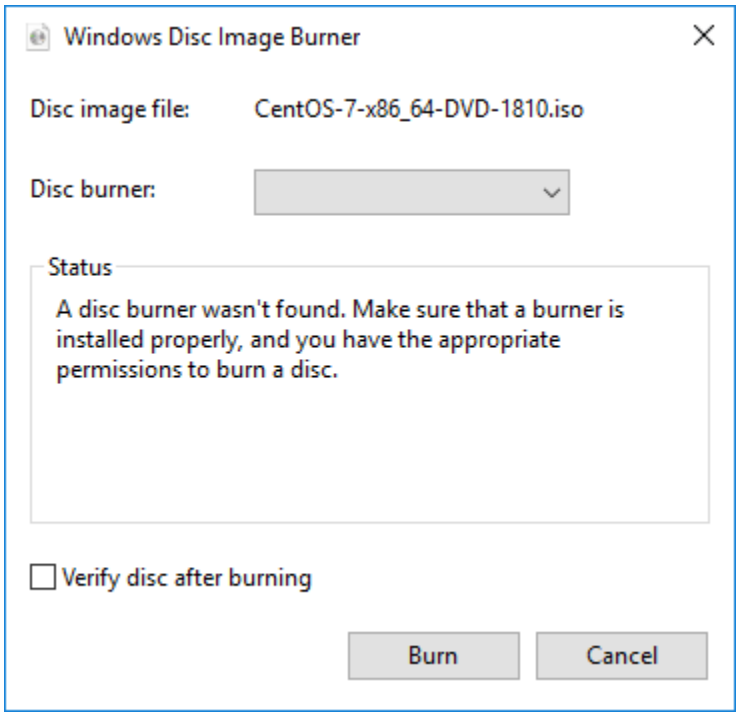
Ако инсталацията ще се извършва чрез DVD ROM, изображението, изтеглено на предната стъпка, трябва да се „изгори“ на носител. Последните няколко версии на Windows могат да извършват това без инсталирането на допълнителен софтуер.

Процедурата в Windows 10 започва със стартирането на файловия мениджър. Във файловия мениджър трябва да се щракне веднъж с левия бутон на мишката върху файла с изображението, който трябва да има разширение ***.iso***. При това във менюто на файловия мениджър ще се появи контекстна секция ***Disk Image Tools / Manage***, която ще съдържа бутон „***Burn***”. Вж. фиг. B.1.



Фиг. В.1. Бутони за „изгаряне“ на ISO изображение на диск във файловия мениджър на Windows 10

При щракване на бутона, ще се появи диалоговия прозорец за запис на файл, който е показан на фиг. В.2. Потребителите следва да отметнат ***Verify disc after burning,*** което ще помогне за откриване на грешки при записа, след което трябва да натиснат бутона ***Burn***.

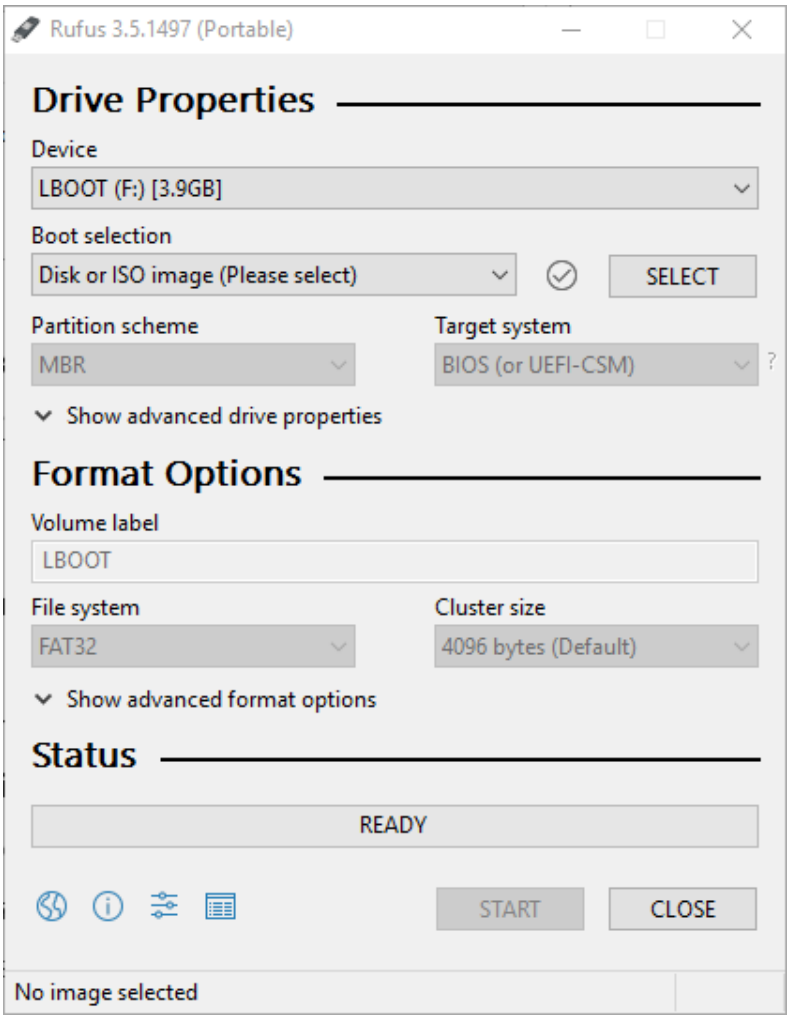


Фиг. В.2. Диалогов прозорец за изгаряне на диск в Windows 10.

2b. Създаване на Flash памет с CentOS

Възможно е системата на която се инсталира CentOS да няма DVD-ROM устройство. В такъв случай изтегленото изображение може да се запише на USB флаш диск. За целта могат да се използват множество инструменти. Един от тях е Rufus - ***https://rufus.ie/***.

Програмата има преносима версия, която не изисква инсталация. При стартирането и, потребителят вижда диалоговия прозорец, показан на фиг. В.3.



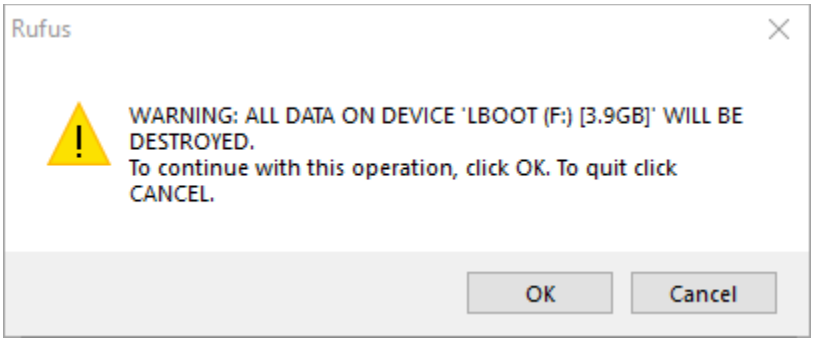
Фиг. В.3. Главен прозорец на Rufus

В него той трябва внимателно да избере от падащия списък ***Device*** това USB устройство, което ще бъде форматирано. След това потребителят следва да щракне на бутона ***Select*** за да избере файла с ISO изображението, който ще бъде записан на USB устройството.

След извършване на тези операции, потребителят може да натисне бутона ***Start***, което ще започне процеса на запис. Възможно е преди стартиране на записа, потребителят да трябва да отговори на няколко въпроса и да премине през няколко потвърждения.

По-новите дистрибуции на CentOS са ISOHybrid, което позволява те да се запишат в един от два режима – т.нар. ISO Image mode или DD Image mode. Всеки от режимите би следвало да работи, но при евентуални проблеми, винаги следва да се изпробват и двата.

На фиг. В.4. е показан и последния диалог, който предупреждава потребителя, че след началото на процеса по запис, съществуващите данни на USB устройството



Фиг. В.4. Главен прозорец на Rufus

За. Стартиране на инсталационната програма

За да започне инсталацията, системата трябва да бъде инструктирана да зареди операционна система от подготвения по-рано инсталационен носител. Това се извършва по различен начин на различните компютри в зависимост от инсталирания BIOS.

По принцип повечето от модерните BIOS системи имат графичен интерфейс за конфигурация и по подразбиране стартират операционната система от CD или USB при положение, че такива, съдържащи валидна операционна система, са поставени при начално стартиране на системата.

Важно е да се знае, че най-голямата промяна в BIOS системите в последните години бе преминаването към UEFI. UEFI е нов, несъвместим

стандарт за BIOS системи, който се отразява и на начина, по който разпределението на дисковите дялове се записва на диска.

Ако компютърът, на който ще се инсталира Ubuntu, няма инсталирана операционна система и поддържа UEFI, потребителят следва да конфигурира BIOS за UEFI зареждане на системата. Ако на системата има инсталирани други дялове на вече инсталирани операционни системи, потребителят следва да използва старата схема за записване на информацията на дисковете, известна като MBR.

3b. Стартиране на инсталационната програма във виртуална машина на емулатора VirtualBox

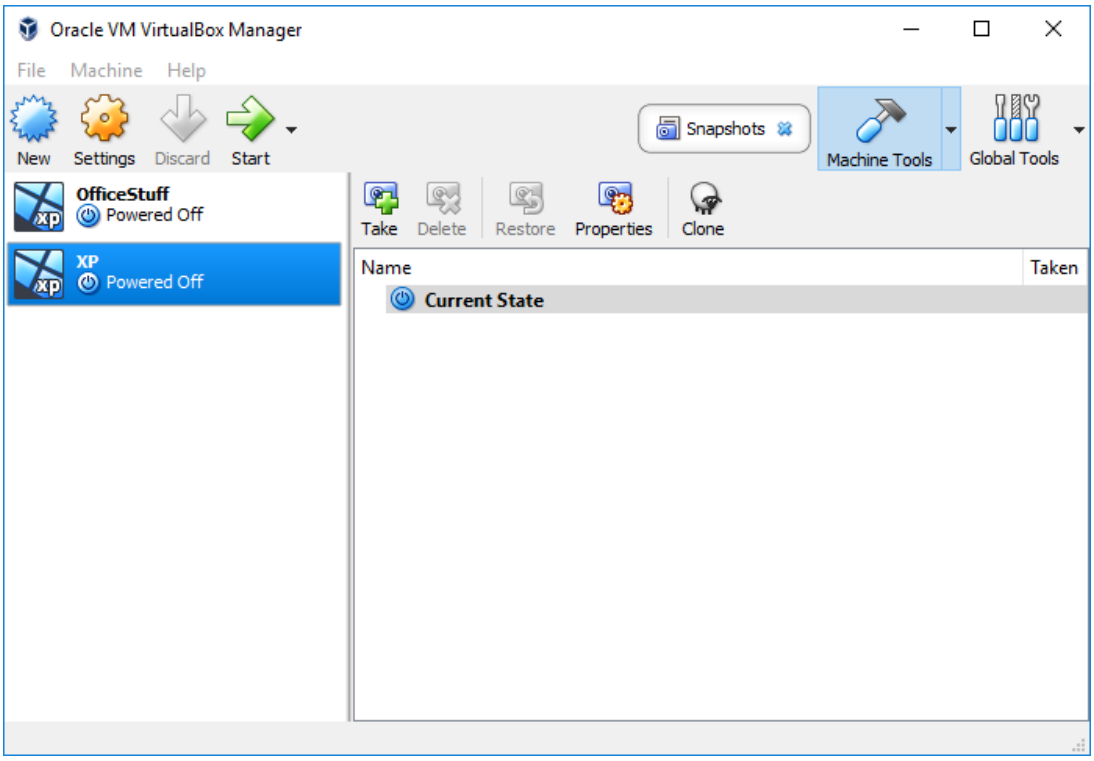
При невъзможност за инсталиране на CentOS на реален хардуер, дистрибуцията може да се инсталира като виртуална машина под управлението на редица емулатори, като например VmWare Player, HyperV и VirtualBox.

Настоящата точка ще демонстрира най-важните моменти при инсталация на системата върху VirtualBox под Windows.

Актуална версия на VirtualBox може да се изтегли от адрес:

https://www.virtualbox.org/wiki/Downloads

Процесът по инсталация не е особено сложен и включва инсталирането на няколко специализирани драйвера. След завършването на инсталацията, програмата следва да бъде стартирана, при което ще се покаже главният ѝ прозорец (фиг. В.5).

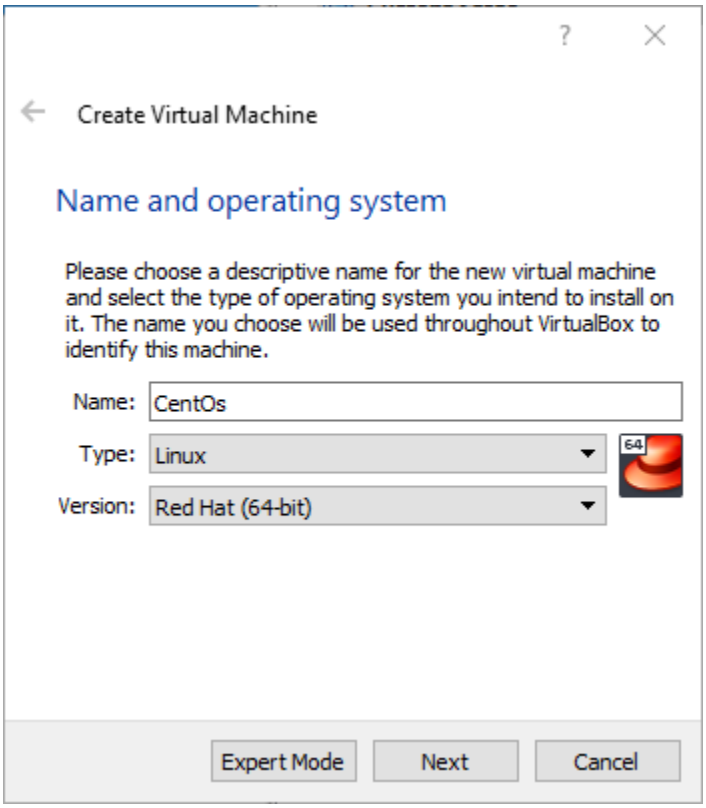


Фиг. В.5. Главен прозорец на VirtualBox с инсталирани две виртуални машини

За създаване на виртуална машина, от лентата с инструменти на главния прозорец следва да бъде натиснат бутона ***New***. Щракването върху този бутон стартира съветника за създаване на виртуална машина, който преминава през няколко стъпки.

В първата стъпка (показана на фиг. В.6.) потребителят трябва за зададе типа на операционната система (***Linux***) и версията на дистрибуцията (***RedHat 64-bit***), както и произволно избрано, описателно име, за виртуалната машина, което трябва да бъде попълнено в полето ***Name***.

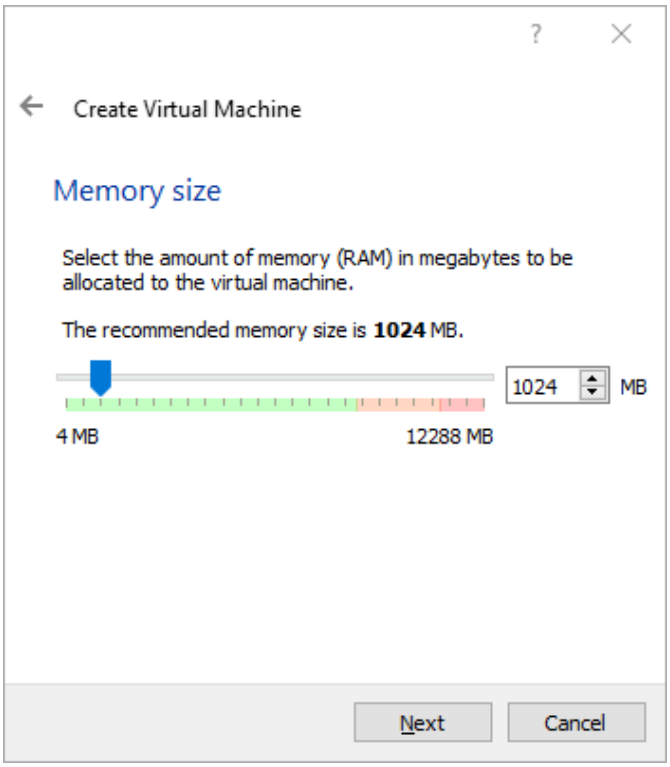
След приключване на въвеждането, потребителят трябва да натисне бутона ***Next***.



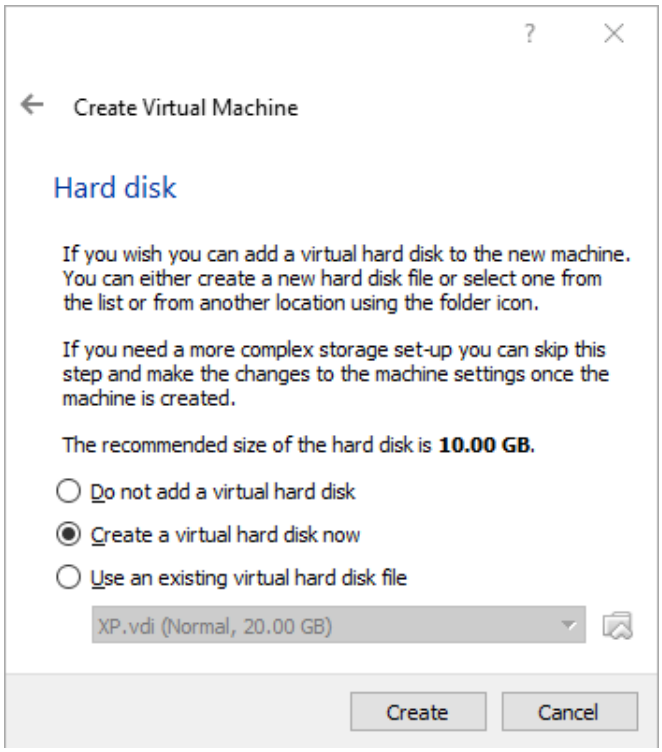
Фиг. В.6. Съветник за създаване на нова виртуална машина – първа страница

На втората страница на съветника, потребителят трябва да укаже размера на оперативната памет, която ще бъде заделена за виртуалната машина. За CentOS, VirtualBox ще задели по подразбиране 1 GB памет, което, както бе споменато, е достатъчно за учебна инсталация. (Фиг. В.7).

На следващите стъпки, VirtualBox ще изиска от потребителя да укаже детайли за дисковете на виртуалната машина. По подразбиране за CentOS, VirtualBox ще предложи на потребителя да създаде нов виртуален диск с размер от 8 GB (фиг. В.8). Потребителят следва да щракне бутона ***Create*** и да приеме това предложение.

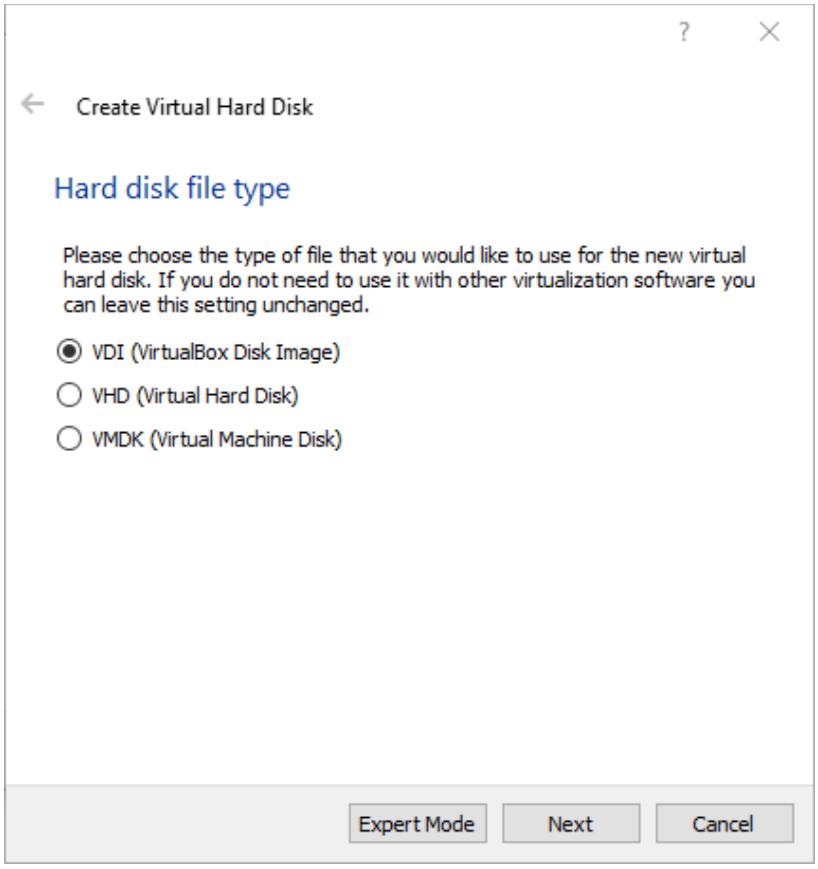


Фиг. В.7. Съветник за създаване на нова виртуална машина – втора страница



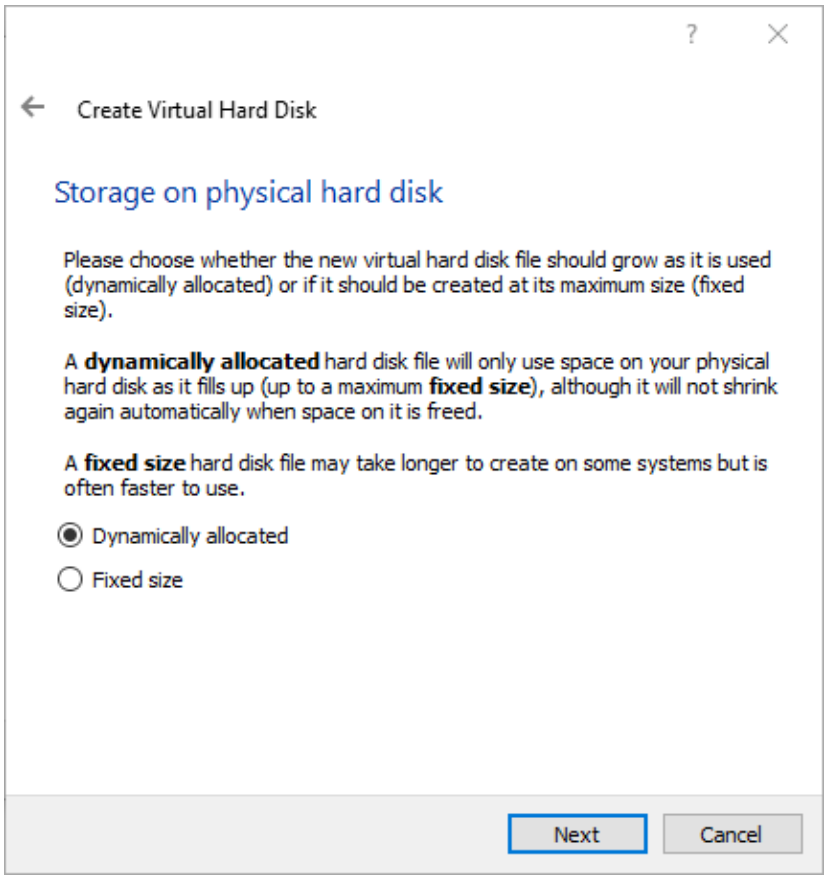
Фиг. В.8. Съветник за създаване на нова виртуална машина – трета страница

Щракването на бутона ***Create*** ще стартира нов съветник, специално насочен към създаването на виртуални дискове. На първата страница потребителят ще бъде запитан за типа на диска. Типът на диска има значение ако виртуалният диск ще се споделя или мигрира между различни емулатори. Това едва ли е притеснение на тепърва обучаващите се, поради което тук следва да се остави изборът по подразбиране – VDI и да се щракне на бутона ***Next***. (Фиг. В.9).



Фиг. В.9. Съветник за създаване на нов виртуален диск – първа страница

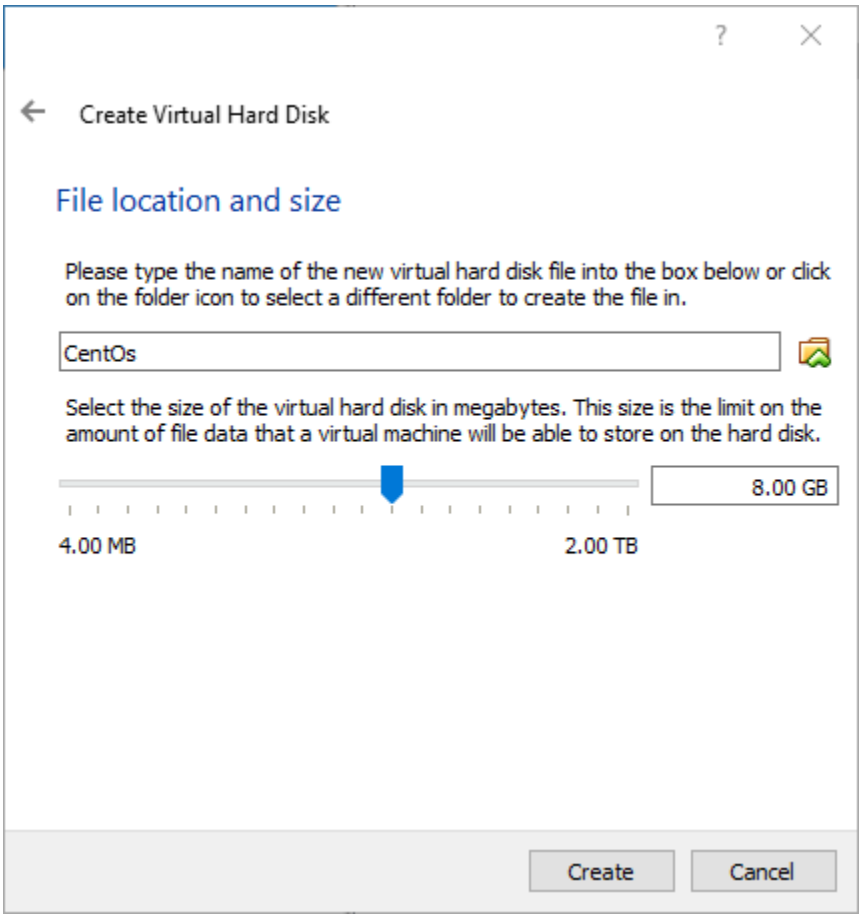
Втората страница от съветника за създаване на дискове пита потребителя дали цялото място за диска ще бъде заделено предварително или вместо това ще се заделя динамично (фиг. В.10).



Фиг. В.10. Съветник за създаване на нов виртуален диск – втора страница

Разликата между динамичното заделяне и предварителното заделяне е минимална и може да бъде усетена най-вече в понятията на свободно пространство на истинския диск на машината. При предварителното създаване, файлът, който съдържа виртуалния диск, ще заеме толкова място, колкото потребителят е указал независимо дали на виртуалния диск има записани някакви данни. В контраст, файлът на виртуален диск, който е динамично заделен ще расте само когато върху виртуалния диск се записват данни.

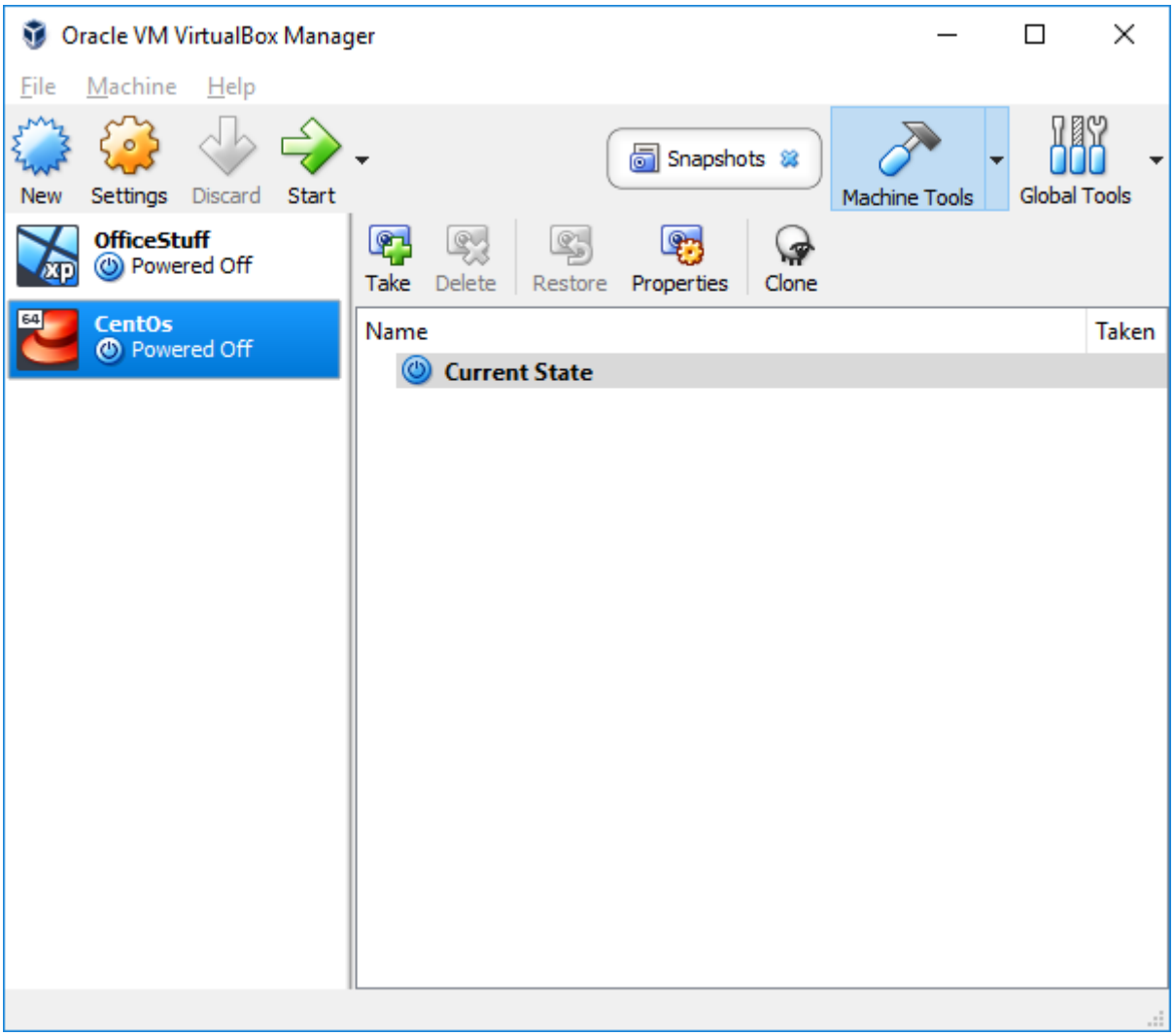
За учебна машина – е напълно подходящ изборът по подразбиране – ***Dynamically allocated***.



Фиг. В.11. Съветник за създаване на нов виртуален диск – трета страница

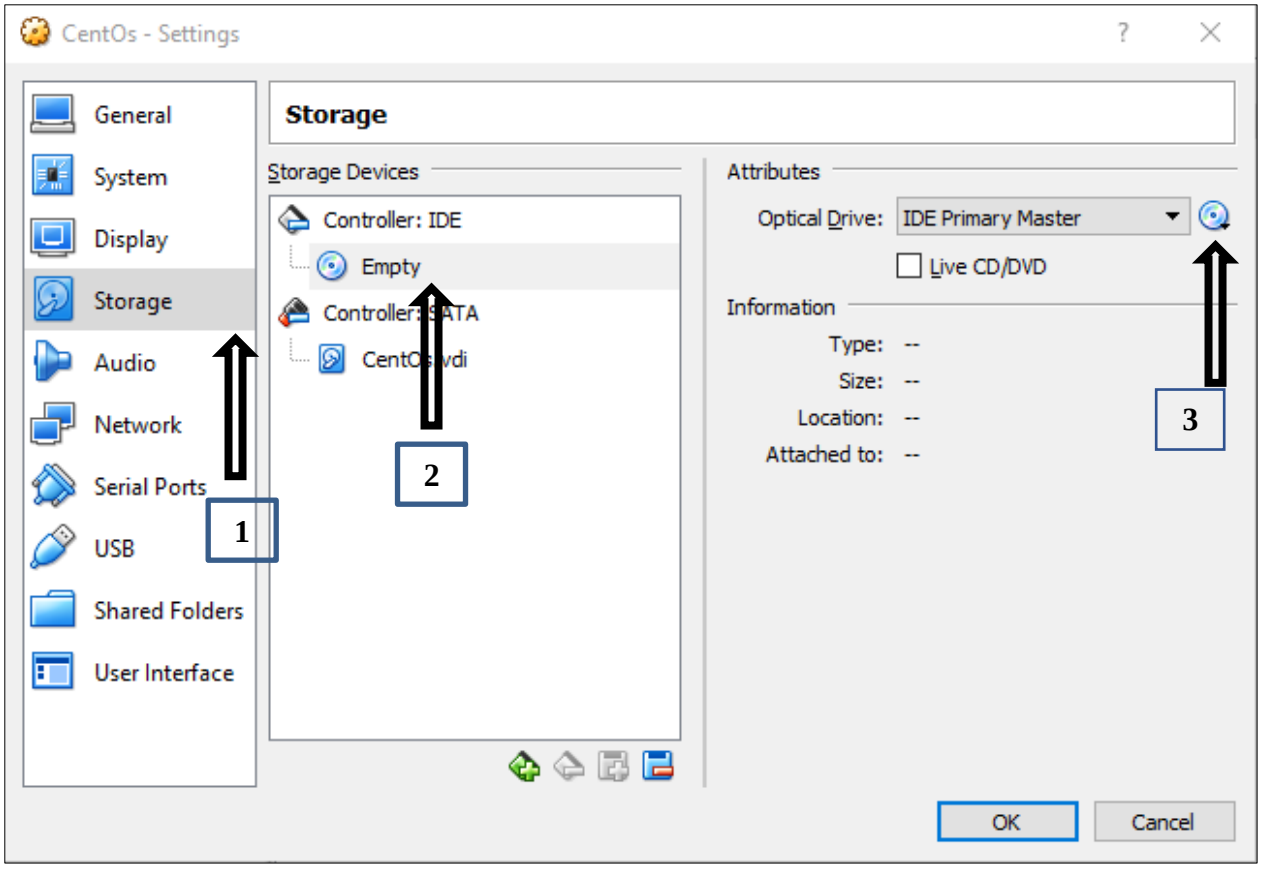
Третата и последна страница на съветника за създаване на дискове позволява на потребителя да промени големината, предложена от VirtualBox, както и да зададе име на файла, който ще съдържа виртуалния диск. За учебна машина не е необходимо да бъдат променяни настройките по подразбиране в този прозорец. Потребителят следва да щракне на бутона ***Create***, което ще завърши процеса по създаване на диск.

С приключването на съветника за създаване на дискове приключва и съветника за създаване на виртуални машини. Единствената операция, която остава, е прикачването на ISO изображение с инсталационните файлове на CentOS.



Фиг. B.12. Маркирана виртуална машина с име ***Ubuntu*** и бутони ***Start*** и ***Settings*** в главното меню на VirtualBox

За целта потребителят трябва да избере виртуалната машина и да щракне на бутон ***Settings*** (фиг. B.12). Това ще отвори диалога с настройки (фиг. B.13), който е организиран в множество страници, достъпни от списъка, поставен вляво. От този списък потребителят трябва да избере страница ***Storage***, след което да щракне върху иконата на CD в списъчното поле ***Storage Devices***. Това ще визуализира в най-дясната половина настройки за съдържанието на емулираното от VirtualBox DVD.



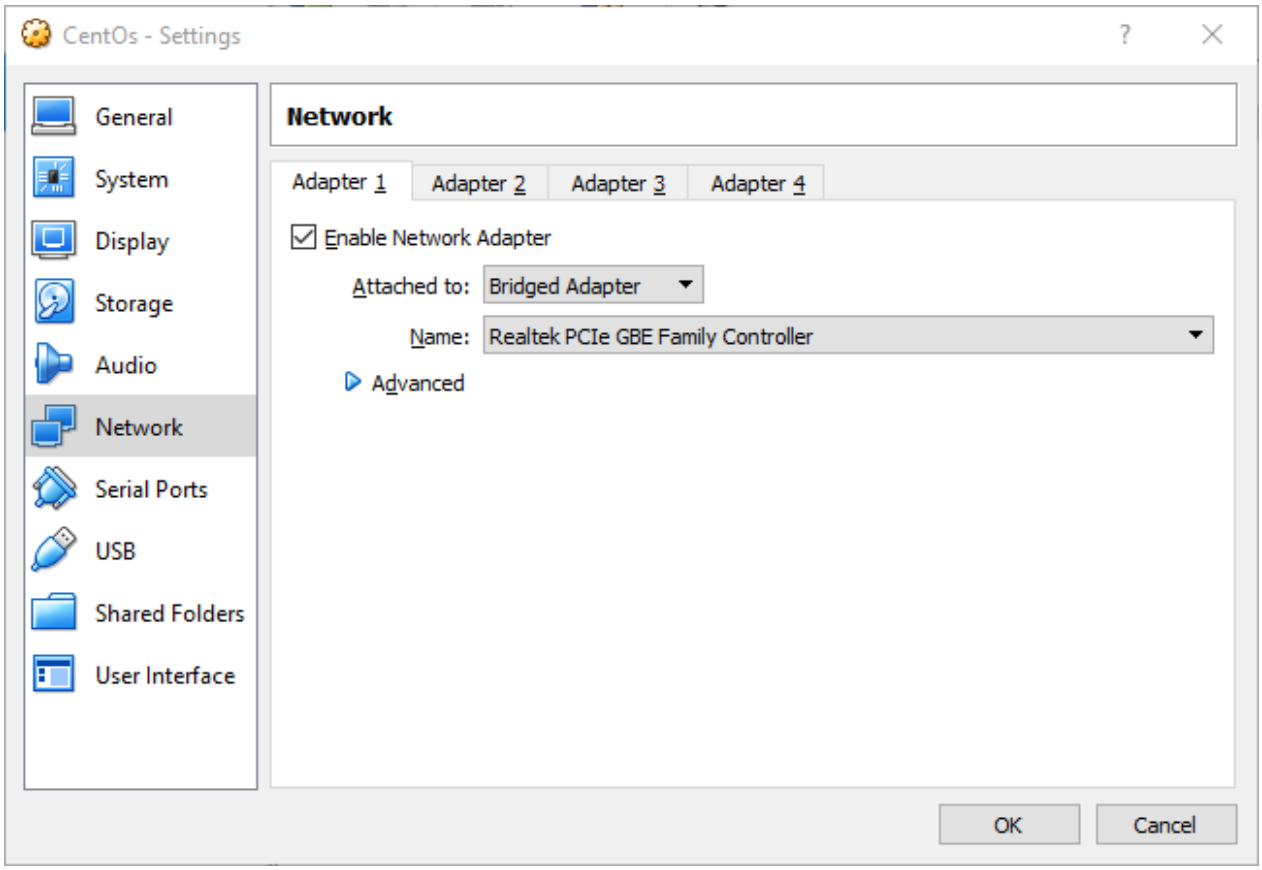
Фиг. В.13. Стъпки за визуализиране на настройките на виртуалното DVD устройство

Ако читателят е следвал предните стъпки и е записал ISO изображението на DVD носител, той може да избере да използва истинския носител, в който случай в настройките трябва да бъде отметнато полето ***Live CD/DVD***. Ако читателят разполага само с ISO изображение, той следва да щракне върху бутона с иконата, представляваща на CD (Вж. стрелка 3 на фиг. В.13), и да избере ***Choose Virtual Optical Disk File***. Това действие ще отвори стандартен диалог за избор на файл, в който потребителят трябва да посочи местоположението и името на файла, който съдържа ISO изображението.

За да улесни настройката на мрежата, потребителите, които използват домашни рутери с DHCP сървър следва да направят още една стъпка – от същия диалог с настройки, те трябва да изберат секция ***Network*** и в полето ***Attached*** to

на виртуалния мрежови адаптер, те трябва да изберат ***Bridged Adapter*** (фиг. B.14).

Избирането на ***Bridged Adapter*** ще свърже виртуалната мрежова карта на емулираната машина директно с мрежата, към която е свързана истинската машина. Това ще позволи на операционната система във виртуалната машина да вземе мрежови настройки директно от DHCP сървъра. В същото време, това означава, че всички потребители на мрежата ще виждат виртуалната машина като отделен компютър, свързан към мрежата и ще могат да се свързват директно към услугите, предлагани от виртуалната машина. Ако локалната мрежа не е домашна, това може да породи проблеми със сигурността.



Фиг. B.14. Настройки на виртуалния мрежови адаптер на VirtualBox

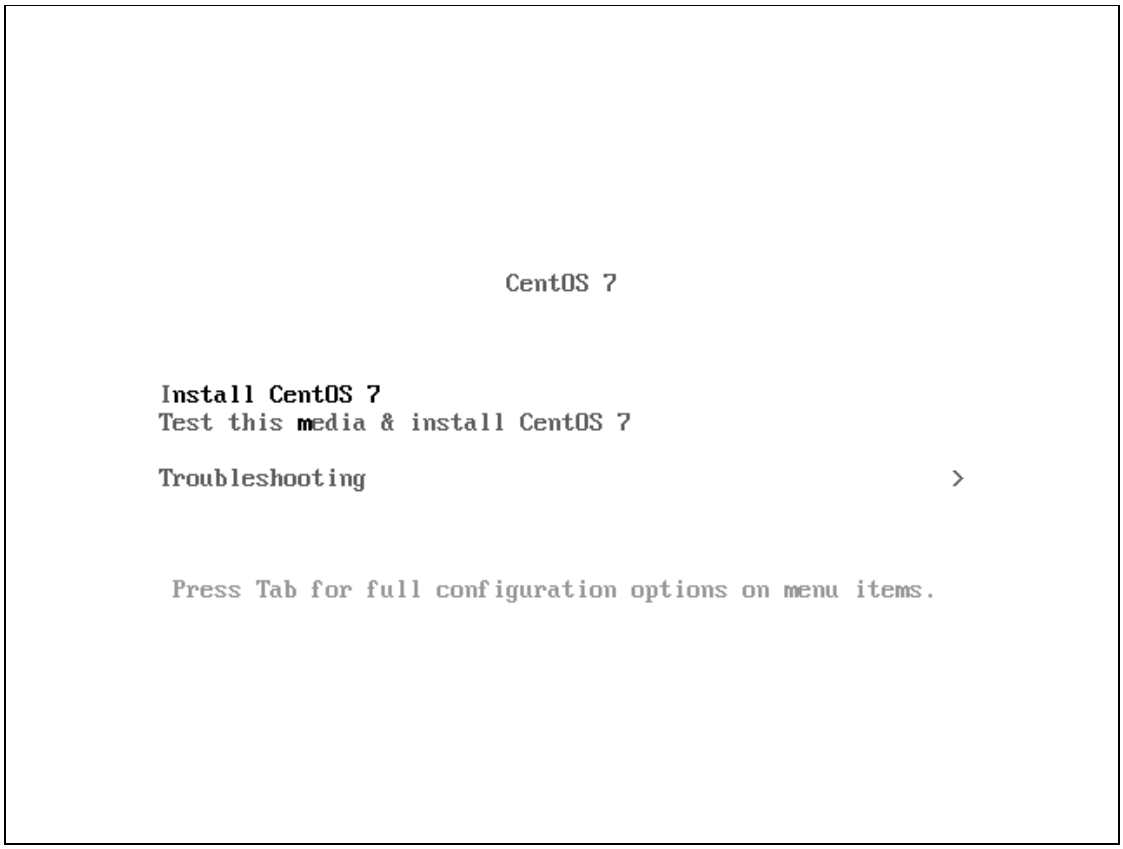
След като потребителят приключи с настройките, остава да натисне бутона ***OK***. На този етап виртуалната машина е готова за стартиране.

За да се стартира машината, потребителят трябва да маркира виртуалната машина и да избере от главното меню на VirtualBox бутона ***Start*** (вж. фиг. В.12.).

4. Инсталационна програма на CentOS

Ако потребителят е поставил инсталационен носител и е инструктирал правилно системата да зареди от него, след процеса на начално зареждане ще бъде стартирана инсталационната програма на CentOS.

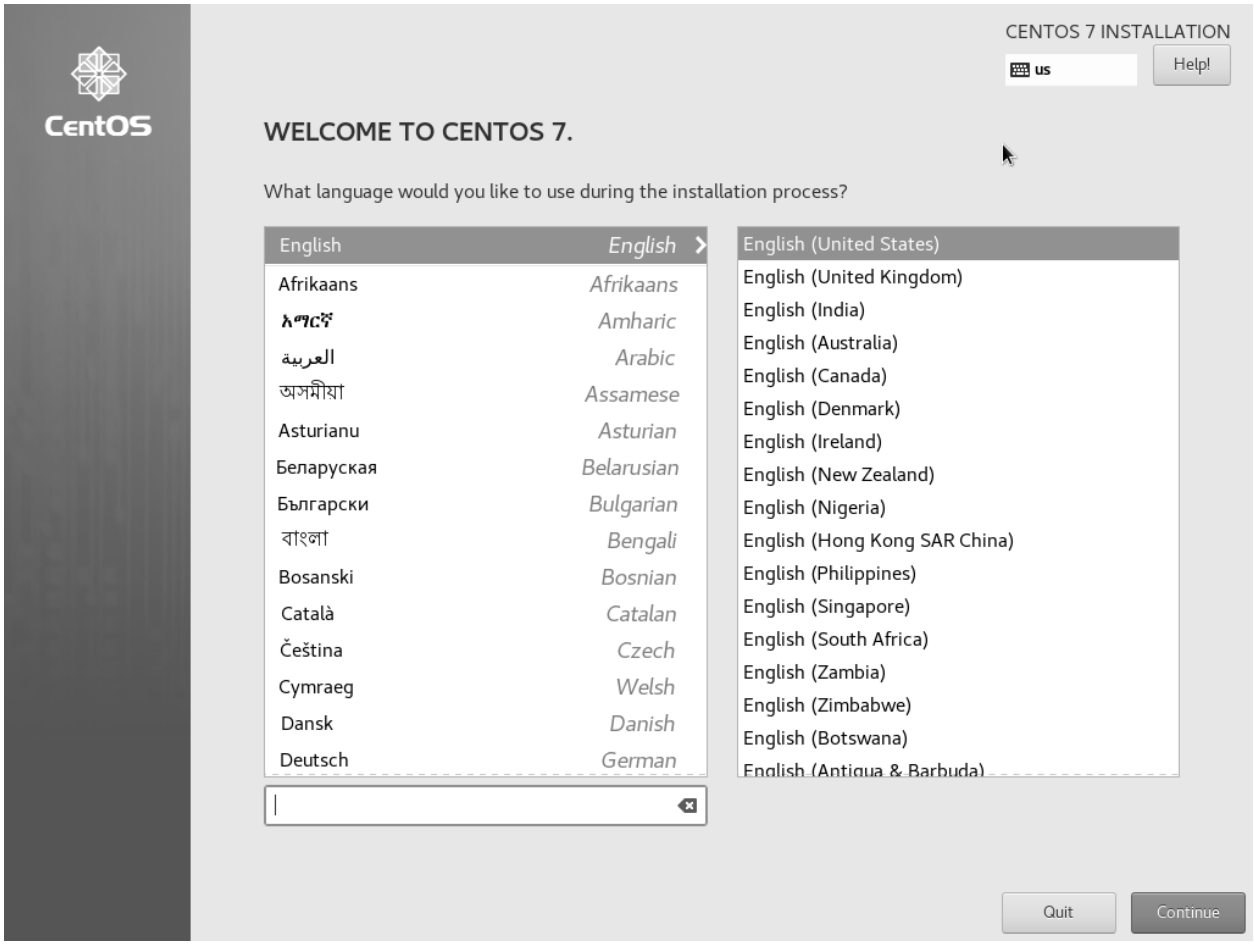
Първият въпрос, изведен от инсталационната програма, касае това дали потребителят иска да стартира инсталация (фиг. В.15).



Фиг. В.15. Начално меню на инсталационната програма на CentOS

За да стартира инсталация, потребителят трябва, разбира се, да избере ***Install CentOS 7***.

Инсталирането на CentOS 7 нормално продължава в графичен режим:



Фиг. B.16. Меню за избор на език в инсталационната програма на CentOS

Графичната част на инсталационната програма започва работа с въпрос към потребителя за езика на инсталацията. Както е показано на фиг. B.16, изборът по подразбиране е английски език.

Фиг. В.17. Меню за избор на език в инсталационната програма на CentOS

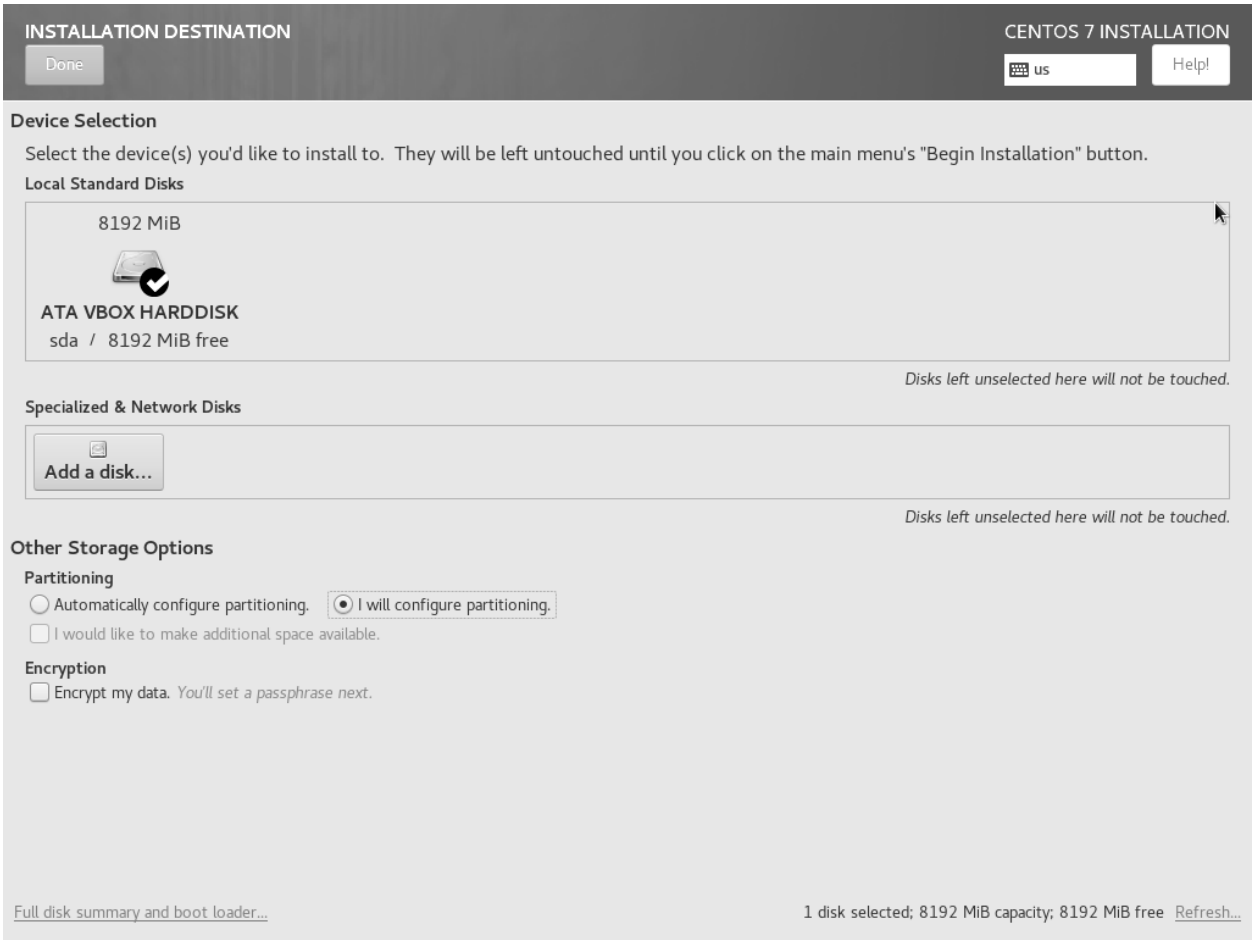
На следващата стъпка от инсталацията (фиг. В.17) потребителят вижда главното меню на инсталационната програма. Инсталацията на CentOS се отличава от тази на много други софтуерни продукти по това, че задаването на параметрите може да стане в ред, избран от потребителя, като за целта се използва именно показаното главно меню.

За учебни цели, елементите от секция ***Localization*** (***Date & Time***, ***Keyboard***, ***Language Support***), не се нуждаят от допълнителна модификация. Ако те не бъдат модифицирани, езикът на потребителският интерфейс ще бъде английски, а клавиатурната подредба ще бъде QWERTY. Тези настройки са

напълно приемливи, а и могат да бъдат променени по-късно, след инсталирането на системата.

Единствената стъпка от главното меню, без която инсталацията на системата няма да може да продължи, е избирането на елемент ***Installation Destination***. При щракването върху него, потребителят ще зададе настройки на дисковете и дисковите дялове, върху които ще бъде инсталиран CentOS.

На фиг. B.18 е показан диалоговият прозорец, който се отваря след избора на този елемент.



Фиг. B.18. Екран ***Installation Destination***

В горната секция на диалоговия прозорец, озаглавена ***Local Standard Disks***, инсталационната програма показва инсталираните на системата дискове. В конкретния пример този диск е 8 GB виртуален диск на VirtualBox.

Дисковете, които ще се използват от инсталационната програма, трябва да бъдат отметнати. За целта, потребителят трябва да щракне с мишката върху тях. При щракване с мишката върху даден диск, на неговата икона се показва отметка като показаната на фиг. В.18.

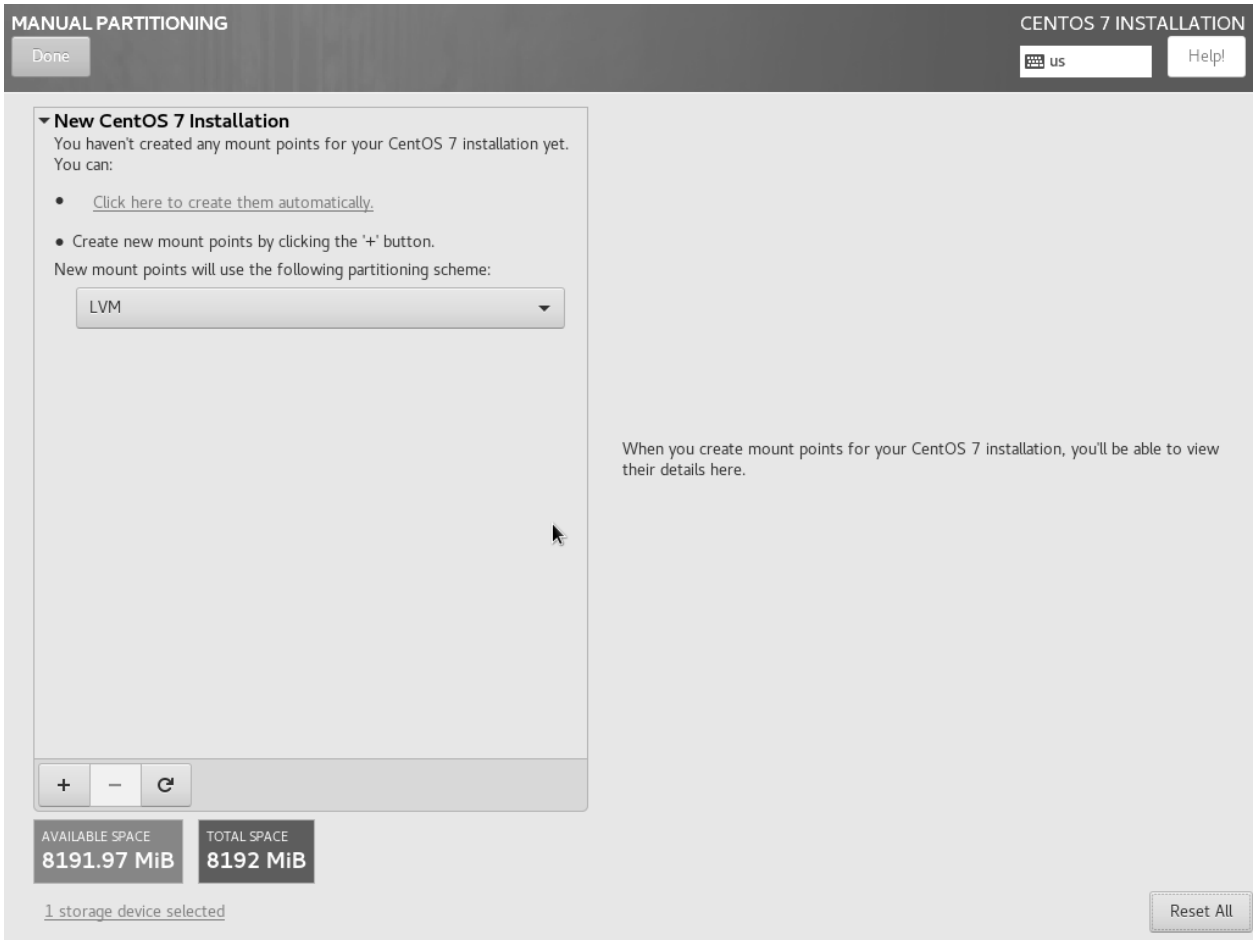
В средната част на диалога, секция ***Specialized and Network Disks*** предлага възможности за използване на iSCSI и други варианти, при които дисковете не се намират физически на системата, а са достъпни през мрежата. Тези опции, макар и безценни при построяване на професионални дейтацентрове, едва ли са от полза за начинаещи, поради което не са разгледани в настоящото приложение.

В долната част на диалоговия прозорец се намира секция ***Other Storage Options***. В подсекция ***Partitioning***, инсталационната програма предлага две опции – ***Automatically configure partitioning*** и ***I will configure partitioning***.

Automatically configure partitioning е подходяща в случаите, когато избраният диск/дискове нямат други операционни системи. Това е именно препоръчителният сценарий за инсталиране на Linux от начинаещи върху *празен* диск. **Избирането на тази опция ще изтрие наличните дискови дялове**.

I will configure partitioning дава възможност потребителят сам да създаде дяловете на системата. Тази опция трябва да се използва винаги когато потребителят смята да използва едновременно CentOS с друга операционна система.

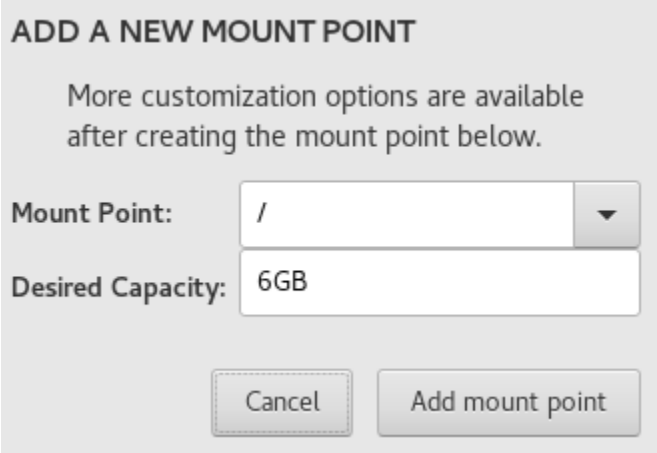
В настоящия пример с учебна цел ние ще демонстрираме използването на тази опция върху празен диск (нещо, което не е необходимо, тъй като в такъв случай автоматичното заделяне на дялове ще е напълно задоволително). За целта, след щракване с мишката върху ***I will configure partitioning***, потребителят следва да натисне бутона ***Done*** в горната част на диалога. При това ще се покаже диалогът за ръчно добавяне на дялове (фиг. В.19.)



Фиг. В.18. Екран ***Manual Partitioning***

В долния ляв ъгъл на диалога са поставени два бутона, обозначени със знаци плюс и минус. Бутонът със знак плюс се използва за добавяне на дискови дялове, докато бутонът със знак минус се използва за премахване на вече

създадени дялове. При натискане на бутона „плюс“ се появява диалогът, показан на фиг. В.19.



Фиг. В.19. Диалог за добавяне на нов дисков дял

Този диалог има две полета – ***Mount Point*** и ***Desired Capacity***. В първото поле се поставя директорията, в която ще се монтира новосъздадения дял, докато във второто поле се записва големината му на диска.

За учебна инсталация е достатъчно да се създадат чрез диалога три дяла. Те са:

1. Дял за директория ***/boot*** с размер не по-малък от 512MB);
2. ***Swap*** дял с големина, съобразена с наличното дисково пространство, като е препоръчително тя да е поне равна на оперативната памет;
3. Дял за главната директория, който трябва да заема останалата част от диска и да има големина, достатъчна за да се инсталират системните програми.

Когато дяловете бъдат въведени, следва да се натисне бутона ***Done*** в горната част на прозореца. При това действие ще се изведе диалог за потвърждение (Фиг. В.20.).

SUMMARY OF CHANGES

Your customizations will result in the following changes taking effect after you return to the main menu and begin installation:

Order	Action	Type	Device Name	Mount point
1	Destroy Format	Unknown	sda	
2	Create Format	partition table (MSDOS)	sda	
3	Create Device	partition	sda1	
4	Create Format	xfs	sda1	/boot
5	Create Device	partition	sda2	
6	Create Format	physical volume (LVM)	sda2	
7	Create Device	lvmvg	centos	
8	Create Device	lvmlv	centos-swap	
9	Create Format	swap	centos-swap	
10	Create Device	lvmlv	centos-root	
11	Create Format	xfs	centos-root	/

Cancel & Return to Custom Partitioning Accept Changes

Фиг. В.20. Диалог за обобщаване и потвърждаване на зададените промени

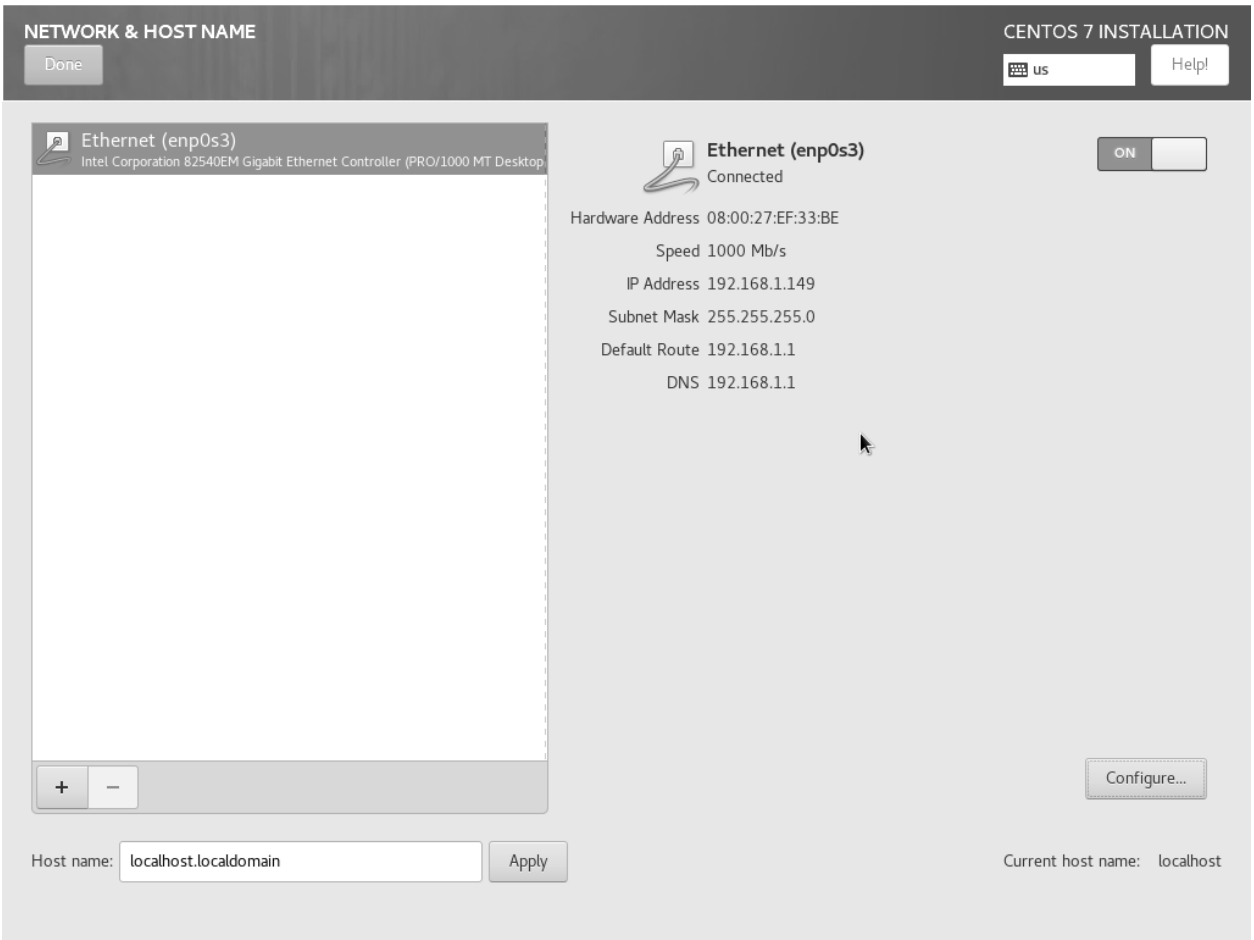
При натискане на ***Accept Changes***, действията ще бъдат изпълнени и потребителят ще бъде върнат към главното меню.

Следващият важен елемент от инсталацията касае настройката на мрежата. Без нея инсталационната програма няма да има достъп до обновявания на софтуерните пакети, които да бъдат инсталирани още по време на инсталацията.

Задаването на мрежова конфигурация се активира от ***Network & Host Name*** в главното меню на инсталацията. При това действие се показва екранът със същото име, показан на фиг. В.21. В списъка, разположен вляво, се намират разпознатите мрежови интерфейси (карти) на системата, всяка от които има уникалното име, дадено от ядрото. При щракване върху един от тези интерфейси (типичната домашна система вероятно ще има само един) в дясната част на екрана се показват конфигурационните му параметри. Най-важният от тези параметри е On/Off бутона, който показва дали даденият мрежови интерфейс ще бъде използван. Ако потребителят определи, че

даденият мрежови интерфейс трябва да бъде използван, той следва да установи този бутон така, че той да показва текста „On”

Ако в мрежата има DHCP сървър, инсталацията ще конфигурира свързания към мрежата мрежови интерфейс с данните, получени от сървъра. При това положение обаче, IP адресът на сървъра може да се промени при евентуално рестартиране и последващо включване.



Фиг. В.21. Диалог за задаване на мрежови настройки

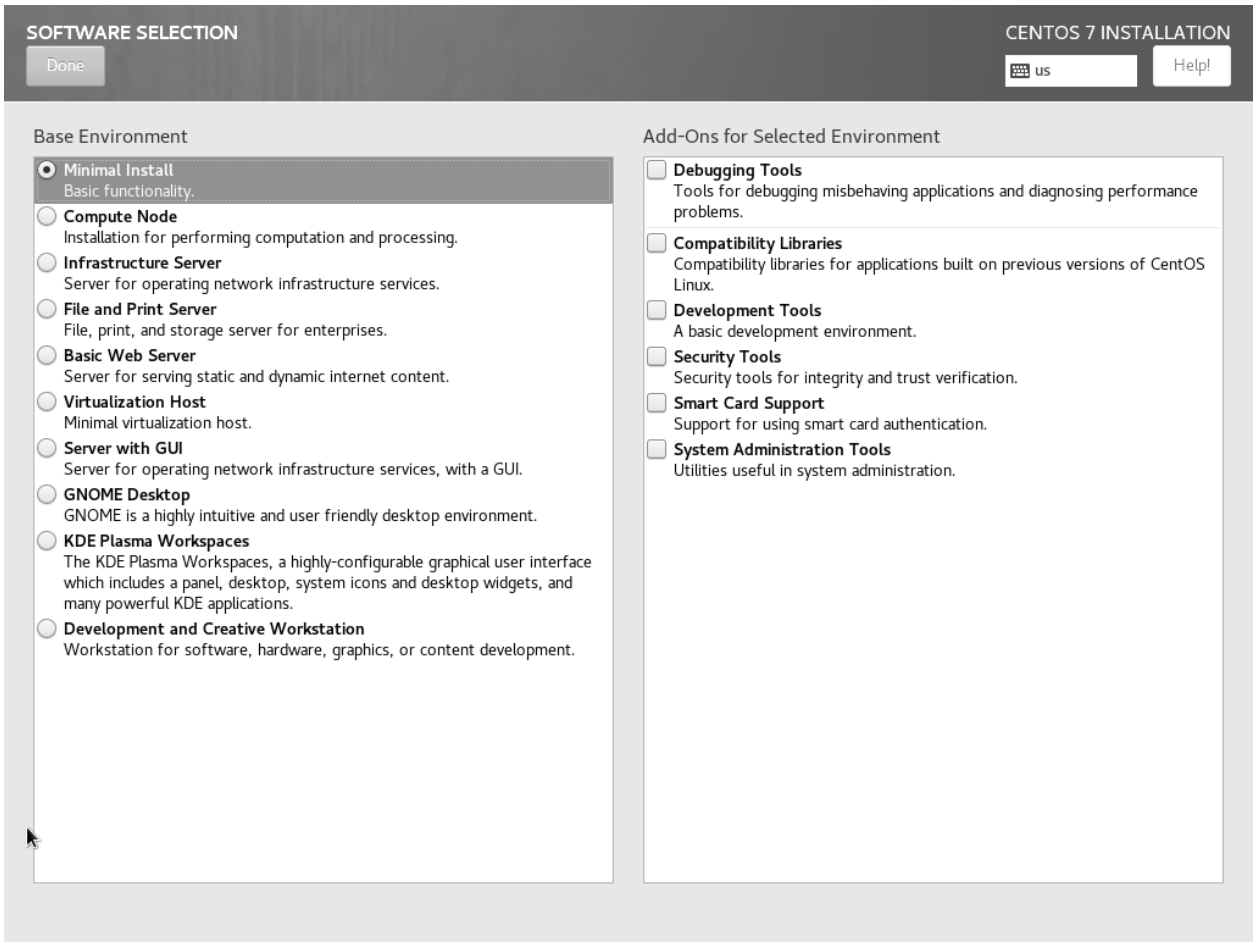
За начинаещи потребители, които все още не се интересуват от прецизно настройване на мрежовите параметри, автоматичното получаване на IP адрес от DHCP сървъра е адекватен вариант, като те трябва да обърнат внимание на

адреса, получен от сървъра. Ако той няма да се изключва продължително време, шансовете са много добри даденият IP адрес да остане непроменен.

Показаният на фиг. 21 екран информира потребителя, че системата е получила IP адрес 192.168.1.149. Ако системата има инсталиран OpenSSH на този адрес може да бъде използван за отдалечено свързване в рамките на локалната мрежа.

В някои случаи се налага IP адресът да се въведе ръчно. Начинаещите потребители са посъветвани да се консултират с администратора на мрежата, след което да въведат параметрите в екрана за настройка, който се отваря след натискане на бутона ***Configure***.

Последното действие, което начинаещите потребители евентуално биха желали да извършат от главното меню на инсталацията, е да определят какви програми ще се инсталират заедно с CentOS. Допълнителните програми идват под формата на инсталационни пакети, които са групирани в групи и категории. Настройката на това кои от тях ще бъдат инсталирани става чрез щракване върху ***Software Selection*** от главното меню на инсталационната програма. При това действие се показва екранът на фиг. B.22.



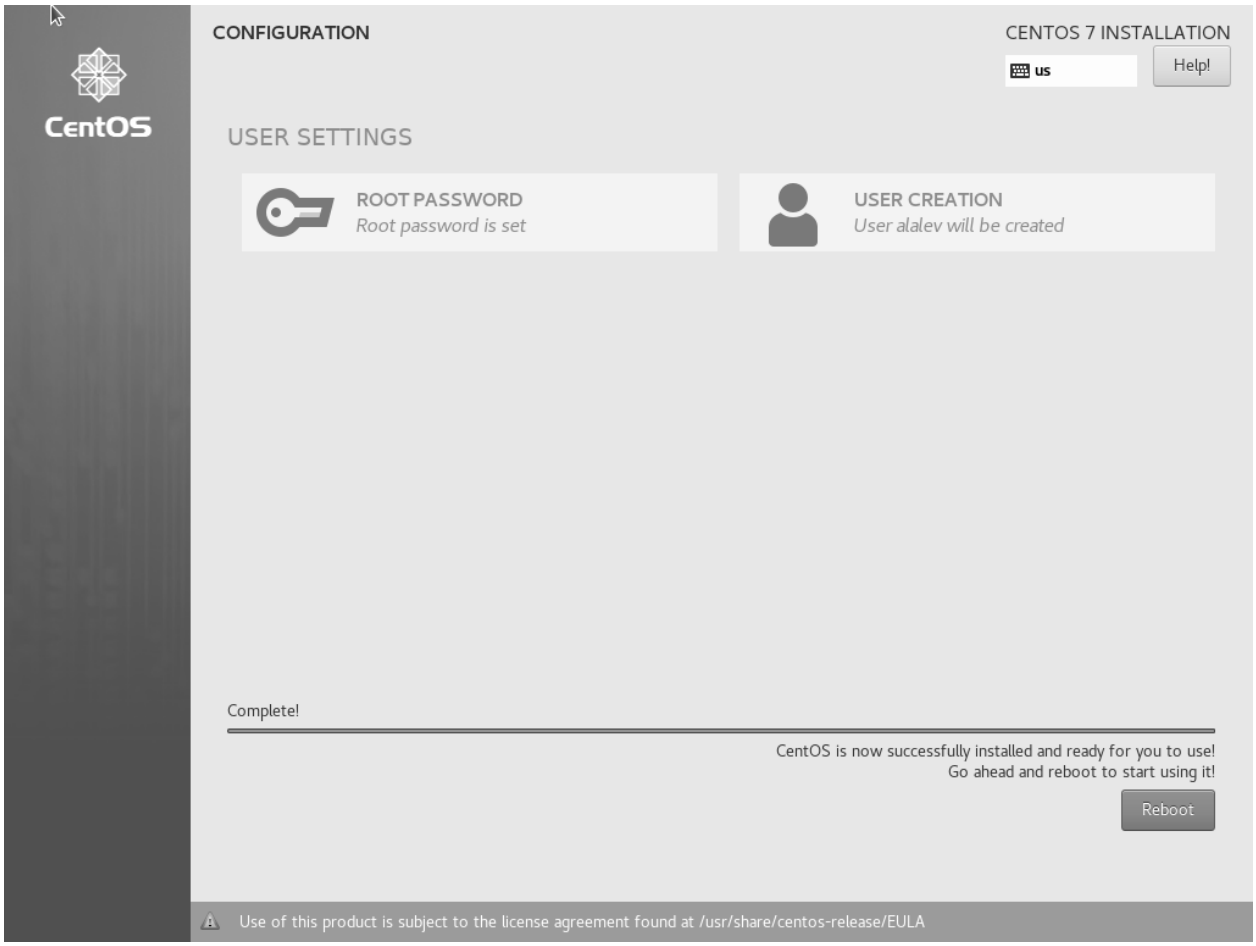
Фиг. В.22. Диалог за задаване на мрежови настройки

Изборът по подразбиране в този екран е ***Minimal Installation***, което е достатъчно за повечето учебни цели, още повече че по всяко време по-късно потребителят може да инсталира програми с помощта на пакетния мениджър ***yum***. Въпреки това, потребител, който желае бърза инсталация и настройка на системата може да избере от тук най-различни допълнителни опции и техните добавки.

Най-интересната за начинаещите потребители възможност в този екран най-вероятно е опцията ***Basic Web Server***, заедно с добавките ***PHP Support*** и ***Python***. Тази опция вероятно трябва да бъде активирана заедно с опцията

Infrastructure Server с една от двете добавки – ***MariaDB Database Server*** или ***PostgreSQL Database Server***.

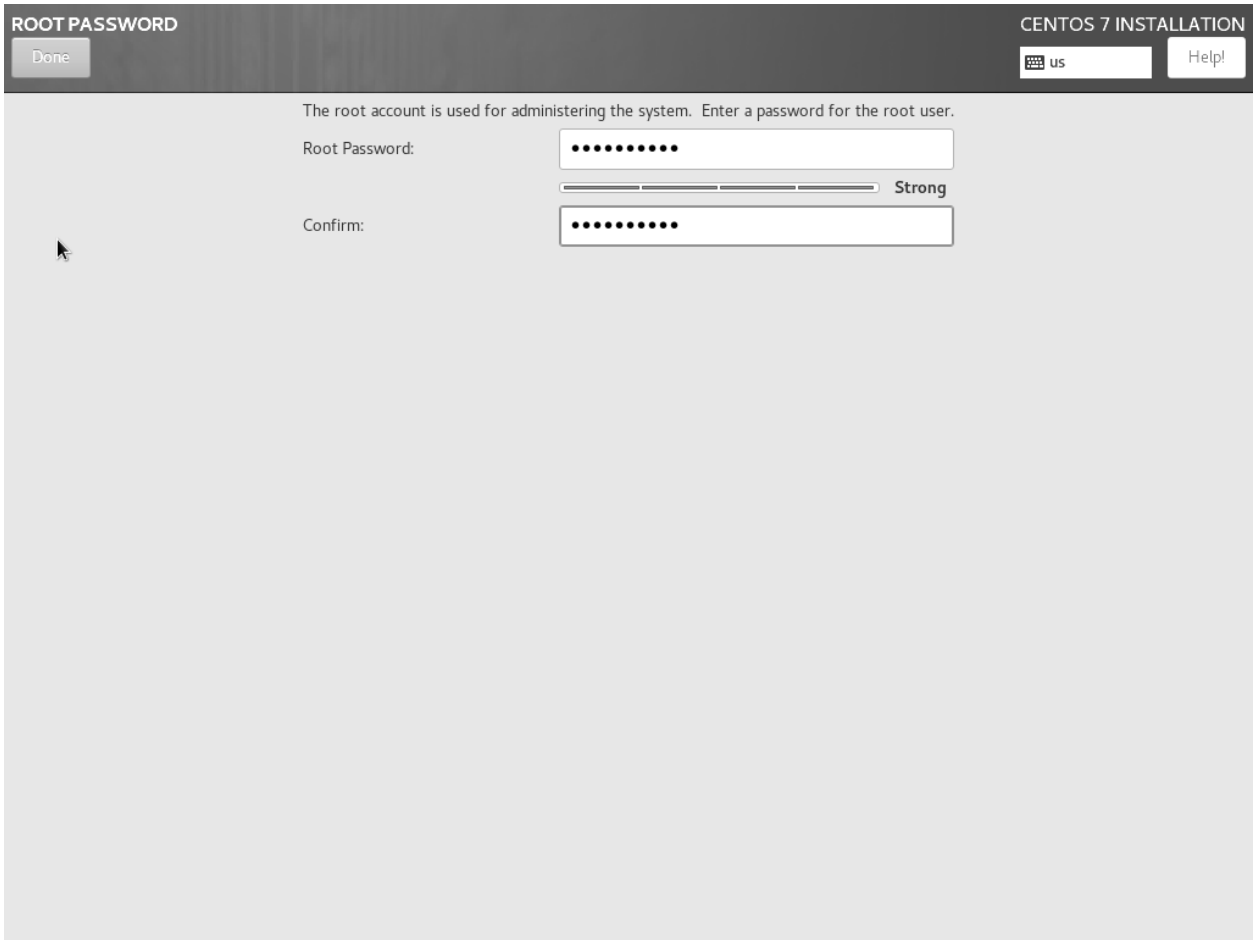
След завършване на всички предпоставки за инсталацията, потребителят следва да избере бутона ***Begin Installation*** от главното меню (вж. фиг. B.17). Натискането на този бутон стартира същинската част от инсталацията. Докато тази част от инсталацията тече (вж. фиг. B.23), CentOS прави опит да използва рационално времето, като възлага на потребителя две задачи по конфигурацията на системата. Първата от тях е задаването на паролата на административния акаунт ***root***, а втората е добавянето на допълнителни потребители на системата.



Фиг. B.23. Създаване на потребители и задаване на парола на администратора

Само една от двете задачи е задължителна, което дава на инсталиращия избор. Той може или да зададе парола директно на потребителя ***root*** или да създаде друг потребител с права за изпълнение на ***sudo***. В първия случай, влизането в системата ще е възможно директно като потребител ***root***. При втория случай, потребителят ще влиза с друг акаунт, а ще използва ***sudo*** за извършване на административни операции.

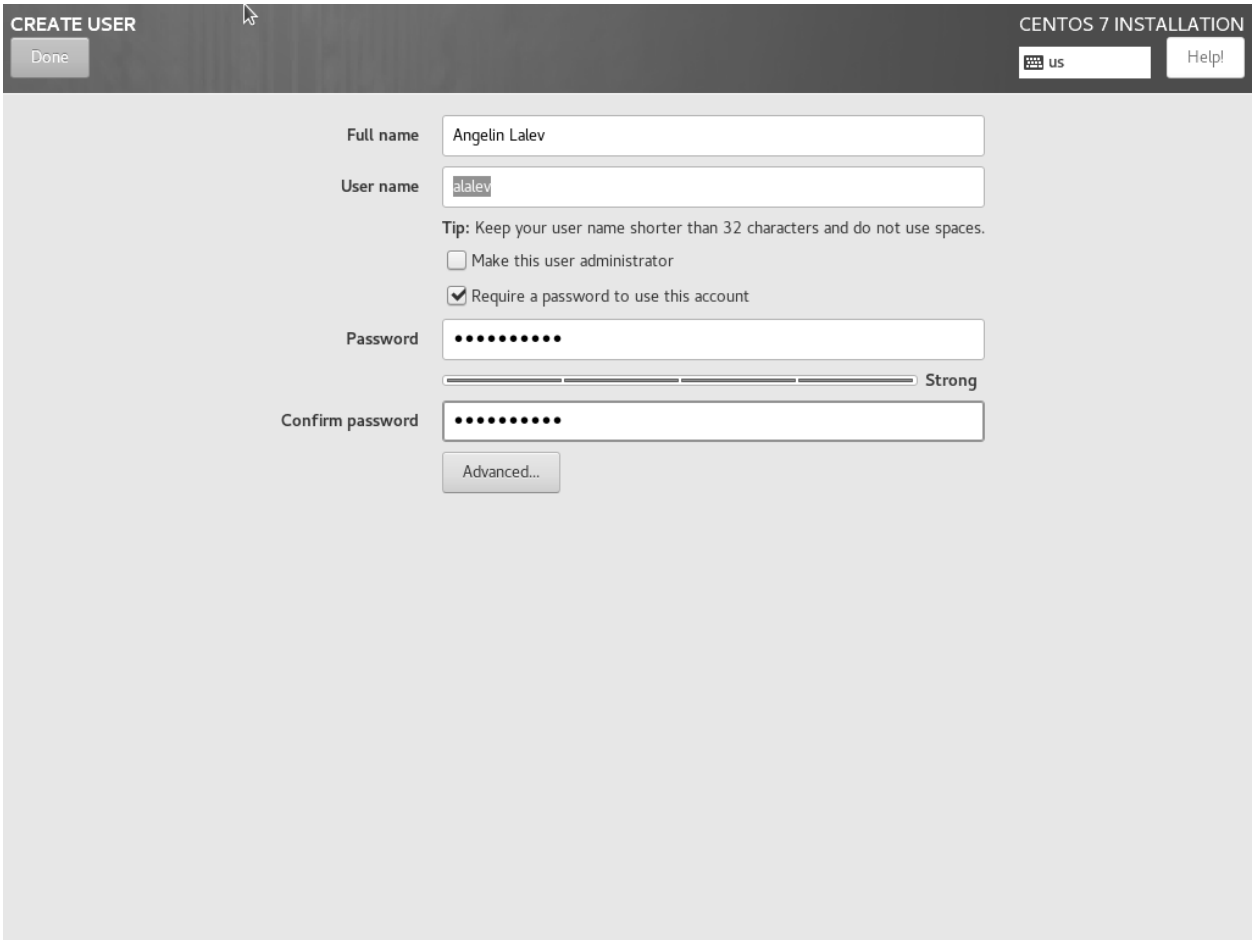
Задаването на администраторската парола става чрез избиране на елемента ***Root Password*** . Това отваря екрана на фиг. В.24.



Фиг. В.23. Задаване на парола на администратора

Естествено, двете пароли, въведени в ***Root Password*** and ***Confirm*** трябва да са еднакви, след което потребителят трябва да натисне ***Done***.

Задаването на допълнителни става чрез извикване на другия елемент – ***User Creation***, което отваря екрана, показан на фиг. B.24.



Фиг. B.24. Създаване на потребител с права за изпълнение на ***sudo***

За създаването на потребител е необходимо да бъдат въведени неговите пълни имена (***Full name***), потребителско име (***User name***), както и да бъде въведена паролата два пъти (***Password***, ***Confirm password***). За да може потребителят да изпълнява команди от името на потребителя ***root*** чрез извикване на ***sudo***, при създаването му трябва да се отметне полето ***Make this user administrator***.

След приключването на току-що описаните конфигурационни стъпки и след като системата е приключила инсталацията си, потребителят трябва да премахне инсталационния носител и да рестартира системата от бутона ***Reboot***.