Въведение в Линукс. Възникване на Линукс. История на Линукс. Основни дистрибуции. Основни концепции в Линукс- ядро, потребители, root. Принципи на свободния софтуер. Лицензионни правила GPL

Линукс, каквато се разбира в повечето случаи когато се говори за нея, е операционна система. Операционната система на кратко е програма, която управлява вашия компютър и неговите устройства. Освен това тя помага да общувате с него – да му задавате команди и да получавате резултатите на монитора или друго външно устройство. Линукс е създаден на базата на друга много популярна в миналото операционна система – UNIX. UNIX е създаден през далечната 1969 година в Bell Laboratories като наследник на друга многопотребителска операционна система – Multics. По това време UNIX е бил уникална операционна система. Написан на изцяло на C, UNIX е бил лесно преносим между различните платформи за разлика от другите операционни системи по онова време, които били писани на асемблер.
Точно тази преносимост е била и основния двигател на UNIX и появилите се след него клонинги. Освен това UNIX е бил различен и в друго отношение – той е изграден от множество малки и бързи програми, които взаимодействали помежду си чрез скриптове. Този подход се пази и досега и е различен от подхода предприет от Windows – многофункционални, големи и тежки програми.

Линикс води началото си от MINIX – един от многобройните клонинги на UNIX. MINIX е била също безплатна операционна система и нейният създател Андрю Таненбаум също предоставил нейния кода. Точно на базата на този код е през 1991 е бил създадено и първото ядро на Линукс от студента Линус Торвалдс.
Версии на това ядро още може да се намерят за свободно сваляне на адрес **http://www.kernel.org/pub/linux/kernel/Historic**

Основната разлика между Линукс и Windows е принципа на отворения код, на базата на който се разработва Линукс ядрото. Софтуер, който се разпространява с отворен код спазва правилата дефинирани в Open Source лиценза.

Първото изискване на OSD (Open Source Definition – Дефиниция на отворения код) е, че всеки пакет с отворен код трябва да се разпространява напълно свободно, като това не изключва негова продажба.
Второто изискване е, че освен програмата, вие трябва да предоставите и изходния код на тази програма. На това се базира и модела на отворения код за подобрение и развитие на софтуера. Получавайки изходения код хиляди програмисти могат да работят над тази програма, да оправят грешки и да добавят нова функционалност.
Третото изискване е променения изходен код да се разпространява по същия начин както и оригиналния.
Четвъртото изискване е, че не може една програма компилирана от променен изходен код да се разпространява под същото име и номер на версията както и оригиналната.
Това основните изисквания на отворения код. Пълния текст може да се намери на адрес **http://opensource.org/docs/definition.php**

Като цяло основната идея е свободното разпространение и развитие на софтуера. Последното което ще разкажа от историята на Линукс и отворения код е историята на GNU обществото. Самото означение GNU означава "GNU's Not UNIX" или преведено "ГНУ не е UNIX".
GNU е анонсирана от Ричард Сталман към края на Септември, 1983 година, с идеята да се създаде UNIX – подобна операционна система, която може да се разпространява свободно. Фактически GNU проекта официално стартира през Януари 1984г.

До тогава разпространението на свободен софтуер е ставало без никакви лицензи, което позволило на много компании да използват свободен софтуер за направата на комерсиални продукти. Именно това била причината да се замисли нов лиценз, който би могъл да предотврати комерсиализация и затваряне на продуктите.
Това е така наречения GNU General Public License (GPL) лиценз, който изисква всяка модификация на GPL софтуер също да бъде разпространявана под този лиценз. Това гарантирало, че свободния софтуер, каквото и развитие да търпи, ще си остане свободен.
GPL лиценза е един от няколкото лиценза прилагани при разпространението на свободен софтуер. Другите по често използвани са Artistic License и BSD лиценз.
Общото между тях е следното:
Софтуера може да се инсталира на неограничен брой компютри.
Броят на хората използващи софтуера също е неограничен.
Софтуера може да се копира неограничено (свободно или отворено редистрибутиране).
Няма ограничения по модифицирането на софтуера с изключение на запазването на базовата функционалност, за който е бил създаден.
Няма ограничение по разпространението, дори продажбата на софтуера.

След историята на Линукс и отворения код, нека да изясним какво се разбира под операционна система. Операционна система са нарича програма, който управлява компютъра и неговите периферни устройства, като мишка, клавиатура, принтер и др. Операционната система освен това се грижи и за разпределение на ресурсите между отделните програми и потребители.
В повечето случаи обаче обикновения потребител не може да прави нищо само с операционна система. За да може той да работи с нея са необходими допълнителни програми, наречени приложен софтуер или просто приложения.
При Windows и DOS част от тези приложения идват с операционната система, но при Линукс не е така. Това е така, защото Ликукс е само ядрото на операционната система. Той може да управлява устройства и задачи, но сам по себе си е абсолютно неизползваем.
Именно за това съществуват Линукс дистрибуции, които съдържат в себе си както ядрото на операционната система, така и приложен софтуер. Всяка една дистрибуция идва с базов набор от най-необходимите програми, като те са еднакви в повечето Линукс дистрибуции. Освен това много Линукс дистрибуции идват и с огромен набор от приложения, които включват браузъри, текстообработващи програми, офис пакети, среди за разработка на приложения, набор от графични среди и т.н.
Сега ще се запознаем с някои от по-известните Линукс дистрибуции:

Debian:
Debian е една уникална дистрибуция различаваща се от другите по това, че се издържа от дарения. Той е подходящ за опитни потребители и е една от най-големите (цели 7 диска). Друго характерно при Дебиан е начина на управление на пакетите (това с програмите преди да се инсталират). Пакетната система на Дебиан е толкова развита, че той може да се инсталира от две дискети и стабилна Интернет връзка.

Red Hat:
Red Hat е една от най-разпространените дистрибуции. Нейната инсталация е особенно лесна и е подходяща за начинаещи потребители. Red Hat е първата комерсиална дистрибуция. Едновременно с покупката на Red Hat потребителя получава и часове за поддръжка. Предимство на Red Hat е и това, че се поддържа от повечето производители на софтуер за Линукс. Наскоро обаче Red Hat преустанови свободното изтегляне и поддръжката на не сървърните си версии. За домашни потребители компанията пусна друга дистрибуция наречена Fedora.

SUSE:
Това е немска дистрибуция, която е изцяло локализирана на немски (това не означава, че не може да се използва и на други езици, вкл. и български). Инсталацията е елементарна и подходяща за начинаещи. Като цяло SUSE е една много добра дистрибуция. Тази година SUSE бе закупена от Novel и засега бъдещето на тази дистрибуция не е много ясно.

Slackware:
Slackware е предпочитан предимно от по-напредналите Линукс потребители, които искат да имат максимален контрол над всичко и това да става лесно и бързо. Под "лесно" и "бързо" обаче в Slackware се разбира ръчна промяна на множество конфигурационни файлове. Това прави дистрибуцията не особенно подходяща за начинаещи, въпреки нейната стабилност и сигурност. Освен това Slackware е иденичната дистрибуция от изброените, която не притежава вградено средство за ъпгрейд.

Mandrake:
Mandrake е френска дистрибуция базирана на Red Hat. Тя притежава елементарно средство за инсталация, която освен всичко друго може да работи на български. Като цяло Mandrake се характеризира с лесна администрация и удобен потребителски интерфейс. От изброените тя е най-подходяща за начинаещи.

Въпреки, че всяка една от дистрибуциите има различна целева група, всяка от тях е базирана на Линукс ядрото. Както и всяка програма разработчиците на ядрото непрекъснато добавят нещо ново или усъвършенстват съществуващи функции.
Това налага строга и добре дефинирана номерация на версиите на ядрата. Принципа на номерация е следния: всеки номер се състои от три цифри. Първата от тях се променя много рядко, само при принципно различни ядра. Това засега е станало само веднъж – при преминаването от версия 1 към версия 2. Скоро не се очаква преминаване към версия 3. Второ число се променя при съществени промени в текущото ядро. За стабилните версии на ядрата, то винаги е четно, а за нестабилните (testing kernels) – нечетно. Последното число обозначава малко промени в ядрото.
В момента се разработват две стабилни версии на ядрата – по-старата 2.4.x и съвременната 2.6.x. Последните версии в момента на писане са 2.4.24 и 2.6.3.
Всяко Линукс ядро е многопотребителско и многозадачно.
Това означава, че в един и същ момент на една Линукс машина може да се стартират множество процеси от множество потребители едновременно, без това да влияе по-някакъв начин на изпълнението на програмите. Тъй като Линукс ядрото е наистина многозадачно, то всяка програма се стартира в собствена област от паметта, като евентуален срив в нея не причинява срив в операционната система.

Въведение в системата. Многозадачност и многопотребителност на операционната система. Структура на файловата система и представяне на физическите и логически устройства – директория /dev. Зареждане на операционната система

Както бе отбелязано в предишната лекция, Линукс е многопотребителска и многозадачна операционна система. Като многозадачна операционна система Linux във всеки един момент от работата си поддържа няколко активни програми в паметта. Най-важната и същевременно задължителна програма е ядрото (kernel).
То се занимава с управлението на хардуера, паметта, разпределя ресурсите на компютъра между обикновените програми, управлява мрежовия трафик, файловата

система и прави още куп важни неща.Останалите активно работещи програми се наричат процеси (process).
Процесите могат да си предават един на друг информация по няколко начина, най-важният от които е мрежовата комуникация (networking). Подобно на телефонните разговори между хора, всеки процес може да се опита да се свърже с друг процес, работещ на същия или на отдалечен компютър. Ако другият процес 'вдигне телефона' (приеме TCP връзка), започва обмен на информация.
Ако компютърът има един процесор, във всеки един момент може да работи най-много един процес (или ядрото). Но ядрото, използвайки особеностите на хардуера така превключва активните процеси, че създава илюзията за едновременна работа на всички стартирани процеси. Тук новото ядро 2.6 въведе понятието приоритетна многозадачност (preemptible). В предишните версии, ядрото не можеше да бъде прекъсвано по време на работа. При Линукс 2.6 ядрото в повечето случаи може да бъде прекъснато по време на задача, така че другите приложения могат да продължат работата си дори когато нещо работи на много ниско ниво.
Освен многозадачна Линукс е и многопотребителска операционна система.
Тя може да съдържа набор от потребители, които се идентифицират пред нея чрез потребителско име и парола. Всеки един потребител има собствена директория, в която може да пази своите файлове (включително и конфигурационните файлове на различните програми), като тази директория е недостъпна за другите потребители.
Всяка една Линукс система има един свръхпотребител наречен root. Той администрира система и има пълните права върху нея. Този свръхпотребител се създава при инсталирането на дистрибуцията, като след това той създава другите потребители.
Цялата тази информация(а и още много друга) се пази на твърдият диск на компютъра.
Файлова система, най-общо казано се нарича набор от файлове, структурирани по някакъв начин и съхранени върху запомнящо устройство.
Едно от основните предимства на Linux е, че поддържа множество файлови системи. Това го прави особено гъвкав при съвместно съжителство с други операционни системи. Linux без проблем поддържа ext и ext2, msdos, ntfs и vfat, xia, minix, umsdos, iso9660, ufs, proc, sysv, ncp, smb, affs, hpfs, raiserfs и др.
Linux, както и UNIXTM не използват идентификатори (например C:\ или D:\) на устройствата за различните файлови системи, а организират всичко в една единствена йерархична дървовидна структура. Всяка нова файлова система, която трябва да бъде достъпна за Linux се добавя към тази дървовидна структура като нейно под-дърво. Мястото където присъединяваме (монтираме) добавяната файлова система се нарича точка на монтиране и задължително трябва да съществува във файловата система на Linux.
Всеки хард диск трябва да е разделен на дялове, като всеки дял е форматиран на дадена файлова система. Под Линукс тези дялове не са видими докато не се монтират.
Дяловете в Линукс се именуват по различен начин спрямо дяловете във Windows. Следната таблица показва точно как се именуват различните дискове в компютъра:

IDE0
IDE1
master
hda
hdc
slave
hdb
hdd

Дяловете на съответните дискове се номерират с името на диска последвано от номера на дяла. Номерата на дяловете започва от 1 до 4 за основните дялове и от 5 за логическите.
Например първия дял на master диска закачен на първия IDE канал (IDE0) е hda1.

Файловата система може да бъде разгледана в два аспекта. На високо ниво под файлова система се разбира разположението на директориите и файловата йерархия в един дял, а на ниско ниво представлява начина на форматиране на блоковото устройство.
Тук ще разгледаме файловата система само от високо ниво, тъй като ниското ниво обикновено е в интереса на тесен кръг от специалисти.

Една типична организация на директориите в една Linux система е следната:

```
/
|-- bin
|-- boot
|-- dev
|-- etc
|-- home
| '-- dino
|-- lib
|-- lost+found
|-- mnt
| |-- cdrom
| |-- fat
| '-- floppy
|-- proc
|-- root
|-- sbin
|-- tmp
|-- usr
| |-- X11R6
| |-- bin
| |-- doc

| |-- etc
| |-- games
| |-- include
| |-- lib
| |-- local
| | |-- bin
| | |-- doc
| | |-- etc
| | |-- games
| | |-- lib
| | |-- man
| | |-- sbin
| | '-- src
| |-- man
| |-- sbin
| |-- src
| | |-- linux -> linux-2.2.15-5
| | |-- linux-2.2.15-5
| |-- tmp
|-- var
```

В зависимост от конкретната Линукс дистрибуция горната структура може повече или по-малко да се отличава. Монтирането на различните файлови системи може да стане както автоматично при зареждането на системата, така и ръчно с командата mount. Някои дистрибуции даже притежават програма, която следи за прикачването на нови

устройства и ги монтира автоматично. Тази програма се нарича automounter и е стартирана непрекъснато. Такива програми, които са стартирани непрекъснато и следят за събития на които реагират се наричат демони.
Както може би прави впечатление, с устройствата под Линукс се работи малко по-различно отколкото под Windows. Тук всяко едно устройство представлява файл от файловата система. Споменатите hda, hdb и т.н всъщност са файлове в директорията /dev. В тази директория се намират и останалите устройства като звукови карти (/dev/dsp), мрежови карти (/dev/eth0 и т.н.), USB устройства и т.н. Тази схема е изключително гъвкава, тъй като операциите по четене и запис в устройствата се свеждат до операциите четене и запис от файлове (опростено казано). Всяко едно устройство може да е или block device или character device. Всеки един файл, който сочи устройство притежава два номера – major и minor. Например едно SCSI устройство /dev/sda2 (втория дял на първия открит SCSI диск), получава major номер 8, който е общ за всички SCSI устройства и minor номер 2, който указва втория дял на диска. Различните видове устройства си разпределят основните и второстепенните номера различно, затова не може да се определи със сигурност колко точно устройства може да има една Линукс система. В ядра до 2.4.x ограничението на основните и второстепенните номера е 255 и в съвременните системи понякога се оказват недостатъчни. Тези ограничения бяха повдигнати с новото 2.6.x ядро – 4096 major устройства и над милион minor.

Важни устройства са:
/dev/hdx(y) – устройствата закачени на IDE контролера на дънната платка. Това може да са хард дискове, CD/DVD устройства и т.н.
/dev/sdx(y) – устройства закачени на SCSI конпролер. може да са хард дискове, CD/DVD устройства и т.н. Понякога някои USB и CD/DVD устройства емулират SCSI, за да работят.
/dev/tty(x) – това са терминалите на Линукс.
/dev/ttyS(x) – серийните устройства. Например /dev/ttyS0 е COM1 под DOS/Windows.
/dev/null – специално устройство от което не може да се чете. Целта му е ненужната информация да се пренасочва към него тъй като не води никъде.
/dev/zero – устройство при четенето от което потребителя може да запълни файл с нули.

Както бе отбелязано по-горе стартирането на Линукс започва от стартирането на ядрото. Това става благодарение на малка част от него записана на специално място на дисковия носител. Тази част сама по себе си не може да управлява компютъра, но тя знае къде да погледне, за да зареди пълното ядро. След като бъде заредено, ядрото започва хардуерна проверка на различните устройства на компютъра и присвояване на определени параметри на тези устройства. Когато хардуерната проверка завърши, Линукс ядрото предава управлението на специална програма наречена init. В този момент тя започва зареждане на стартиращите скриптове, откъдето разбира в какъв режим се стартира системата (графичен, конзолен, многопотребителски или еднопотребителски) и необходимите програми за нормалната работа на операционната система. На този етап се стартират и услугите на системата, които може да са различни видове сървъри, демон за отдалечено управление и т.н. В края на зареждането init зарежда програмата login, която подканя потребителя да въведе потребителско име или парола. Ако е зададен графичен режим, то вместо login се зарежда програмата xdm (X Display Manager), която стартира графичния сървър и графичната среда.

Описание на директориите и тяхната функция. Типове файлове. Битове за права на файловете, типове права. Задаване и отнемане на права.

Всяка една операционна система има определена структура на директориите и всяка директория има определена функция.

В Линукс има два типа директории: системни и потребителски. Системни са тези директории, които се създават при инсталирането на операционната система. Такива например са директориите /dev, /bin, /lib и т.н. Потребителски са директориите, които се създават от потребителите на операционната система с цел да пазят собствена информация в тях. Такива са директориите /home/student и всички поддиректории вътре в нея.

Основните системни директории са:

/bin – съдържа
основните програми на операционната система, без които тя не би функционирала правилно. Повечето програми от тази директория може да се изпълняват от всички потребители. Тя е присъства и в пътя по подразбиране на всички потребители.

/boot –
съдържа ядрото на операционната система, както и други файлове, необходими за правилното функциониране на ядрото.

/dev – съдържа
файловете отговарящи за хардуерните устройства на компютъра. Файловете в тази директория са достъпни само за четене от потребителите.

/etc – съдържа
глобалните конфигурационни файлове на операционната система и нейните програми. Някои от програмите може да пазят конфигурационни файлове и на други места.

/home –
съдържа домашните директории на потребителите. Тук за всеки потребител се създава отделна директория в която може да се съхраняват конфигурационни файлове или документи от различен тип. Специфичното в тези директории е начина на разграничаването на конфигурационните файлове и поддиректории от обикновените, а именно чрез добавяне на точка в началото им. Повечето файлови мениджъри скриват файл или директория, който започва с точка.

/lib –
директорията съдържа най-важните споделени библиотеки на операционната система – glibc-solibs. Тези библиотеки се наричат споделени (shared object), защото повечето програми ги използват при изпълнението си.

/lib/modules –
също много важна директория. Тук се съхраняват модулите на ядрото, т.е. драйверите на операционната система. Туй като всяка Линукс система може да има повече от едно ядро, то в тази директория се създават поддиректории с номера на всяко едно инсталирано ядро.

/mnt –
незадължителна системна директория. Тя се създава автоматично и нейната цел е обединяването на всички монтирани устройства на едно място. Линукс обаче не ви ограничава да монтирате устройства в други директории.

/opt – тук
повечето дистрибуции инсталират графичните среди като KDE и GNOME.

/proc – това е
специална директория в която няма реални файлове. Посредством тази директория вие имате достъп до различни параметри на ядрото, както и до различна информация пряко предоставяна от ядрото на операционната система.

/root –
домашната директория на свръхпотребителя. За обикновения потребител тази директория е заключена и той няма никакъв достъп до файловете в нея.

/sbin – важна
системна директория с множество програми за управление на операционната и файловата система. Тази директория не е включена в пътя по подразбиране на обикновения потребител. Всички програми в тази директория искат root привилегии, за да работят.

/tmp –
директория в която програмите записват временните си файлове. Тя има неограничен достъп за всички потребители. В нея обаче може да има поддиректории създадени от програми, които да са недостъпни за обикновените потребители.

/usr –
директория с множество поддиректории. Тук се пазят повечето инсталирани програми, помощната информация и др.

/usr/X11R6, /usr/X11
– в тази директория се пазят файловете на графичния сървър, шрифтовете му и неговите библиотеки.

/usr/bin –
съдържа бинарните файлове на инсталираните програми, които не са важни за нормалното функциониране на операционната система.

/usr/doc –
част от документацията на програмите. Тук се съхранява Linux-HOWTO.

/usr/include –
тази директория е необходима само ако ще се инсталират програми от изходен код. Тук се пазят заглавните C файлове на инсталираните библиотеки и програми, които са необходими за инсталиране на други програми.

/usr/info –
част от документацията на Линукс. Достъпа до нея се осъществява чрез специална програма – info.

/usr/local –
директория в която по подразбиране се инсталират програмите

компилирани от изходен код. Тази директория копира структурата на /usr.

/usr/man –
най-мащабната част от Линукс документацията. Тук се съхранява ръководство (manual page) за почти всички инсталирани програми. Достъпа до тази информация се осъществява с програмата man.

/usr/lib –
важна директория, в която се съхраняват повечето инсталирани библиотеки. Тези библиотеки не са необходими за работата на операционната система, а за работата на допълнително инсталирания софтуер.

/usr/share –
също важна директория. Тук програмите инсталират файловете, необходими за тяхната правилна работа.

/usr/src – в
тази директория се пази изходният код на Линукс ядрото. Той е необходим само когато ядрото ще се прекомпилира. И тук подобно на /lib/modules може да има код на повече от едно ядро. Те се съхраняват в директории, различаващи се с номера на версията. В директорията /usr/src присъства и линк linux, който сочи към една от директориите съдържащи изходен код.

/var –
системна директория съдържаща предимно логове на програмите и система, както и опашка с отложените или предстоящи задачи за автоматично изпълнение. Директория съдържа и друга информация, като пристигнали писма, файлове заключващи дадени процеси и сокети на работещи в момента програми.

В повечето директории обикновения потребител има права за четене, но не и за запис. Някои директории (като директорията /root и някои от директориите с логове) обикновения потребител няма никакви права. Обикновения потребител има пълни права единствено над файловете в собствената му директория и над собствените права в /tmp директорията. Тази организация затруднява значително инсталирането на зловредни програми (например вируси), тъй като те ще имат достъп единствено до файловете на потребителя, който ги изпълнява.

Досега многократно се спомена понятието права на потребителя и сега ще изясним какво е това право на потребителя на извърши определено действие, видовете права и това как се задават. Във всяка Линукс система има няколко вида потребители организирани в групи. Всеки файл може да задава три вида права за потребителя, три за групата и три за всички останали.

Тези права са:
право за четене (r)
право за запис (w)
и право за
изпълнение (x)

Ето казаното в таблична форма:

Собственик -> rwx **Група -> rwx** **Всички останали -> rwx**

Правата на всеки файл може да се видят с командата ls -l.
Ето пример на нейният изход:

#ls -l
drwx------ 40 dino users
1024 Sep 8 14:51 .drwxr-xr-x 7 root root
1024 Aug 18 19:23 ..-rw-r--r-- 1 dino users
30 Sep 8 18:17 sample.txtdrwxr-xr-x 2 dino users
1024 Sep 8 18:19 sample
Тук файловете, които започват с d всъщност са директории, а sample.txt е единствения истински файл.
В този пример файла sample.txt има следните права:
собственика на файла е dino и той може както да чете, така и да записва в него.
Тук трябва да се отбележи, че групата към която принадлежи един файл може да не съвпада с групата на собственика на файла.

групата на файла е
users и всички в тази група могат да четат от файла.

всички останали могат
да четат файла.

Ако разгледаме директориите, ще видим че те имат флаг за изпълнение (execution). Тук обаче този флаг има друго значение, а именно – право за търсене (search). Ако един потребител няма това право, то той не може да отваря директорията, но все още има достъп до файловете в нея. За да може да прочете един файл в този случай обаче, е необходимо потребителя да знае името на файла и пълният път до него. Освен тези права файла може да има още един флаг – SUID и GUID. Файл с такива флагове се изпълнява с привилегиите на потребителя или с привилегиите на групата, която изпълнява файла. Тези флагове се считат за много опасни и тяхното използване трябва да се ограничава.
Командите, които променят собственика и/или групата на файла и неговите права са съответно chown и chmod. Ако трябва да се променят правата на sample.txt, така че да бъде изпълним от всички, то трябва да се напише следното:
#chmod +x ./sample.txt
Синтаксиса на командата е
много прост: + включва, а – изключва даден флаг. Чрез r, w и x се уточнява кой флаг трябва да се промени.

По този начин се променя даденото право глобално. А ако трябва да се промени само за собственика или групата например? В този случай пред знака + или – се постява опция чии права трябва да се променят, като опциите са следните:
u – собственика(user)
g – групата(group)
o – всички останали (others)
a – всички (all), като това е опцията по подразбиране.

Ето и пример за премахване на правото за изпълнение от файла за всички останали потребители.
#chmod o-x ./sample.txt

Командата позволява свободно комбиниране на опциите, например:
#chmod u+rwx ./sample.txt
Нека отново разгледаме правата на един файл, като където флага е включен ще поставим еденица, а там където е изключен – 0.
#ls
-l ./sample.txt
-rwxr-xr--
1 dino users 30 Sep 8 18:45 sample.txt

Собственик -> r w x -> 1 1 1
Група -> r w x -> 1 0 1
Всички останали -> r w x -> 1 0 0

Сега нека разгледаме тази последователност от нули и еденици като три двоични числа –
111, 101 и 100:

Собственик -> r w x -> 1 1 1 -> 7
Група -> r w x -> 1 0 1 -> 5
Всички останали -> r w x -> 1 0 0 -> 1

Двоичната аритметика показва, че това са числата 7, 5 и 4. Причината за всичко това е, че оригиналния синтаксис на командата chmod е (от UNIX):
#chmod 744 ./sample.txt
Това ще установи на файла правата в таблицата в долната таблица:
Собственик Група Всички останали
r w x r w x r w x
1 1 1 1 0 1 1 0 0
111 101 100
7 4 1
Този синтаксис е по-сложен, но и много по-гъвкъв от типичния Линукс стил.
Чрез този метод е възможна промяната на всички права едновременно, нещо което не може да се постигне по първият начин. Начина на образуване на правата е следния:

4 – четене
2 – запис
1 – изпълнение/търсене

След това е достатъчно да се съберат числата отговарящи за съответните права.
Например право за четене и запис се дава като се съберат 4+2 =6.
Освен промяна на правата, понякога се налага да се променят и собственика или групата на файла. Това става с командата chown.

#chown gosho ./sample.txt
#ls -l ./sample.txt
-rwxr-xr-- 1 gosho users 30 Sep 8 18:55 sample.txt
След тази промяна потребителя dino
вече не притежава файла и съответно не може да си върне неговата собственост. Това може да направи или gosho или root.

Промяната на групата става чрез командата chgrp:

#chgrp root ./sample.txt
#ls -l ./sample.txt
-rwxr-xr-- 1 gosho root 30 Sep 8 19:02 sample.txt

Освен с chgrp и промяната на групата може да стане и чрез chown. В този случай собственика и групата се променят едновременно:

#chown dino:users ./sample.txt
#ls -l ./sample.txt
-rwxr-xr-- 1 dino users 30 Sep 8 19:02 sample.txt

Типове потребители. Администратор(root) и нормален потребител. Файлове с пароли. Скриване на паролите. Команди за създаване и изтриване на групи и потребители

Достъпът до всяка Линукс система става чрез потребителски акаунт, установен от системния администратор. Това означава, че на всеки потребител се задава потребителско име за идентификация, парола, собствена директория и му се задават права за достъп до системата.
Всичко това се прави чрез администраторския акаунт, наречен root.
Потребителя root се създава автоматично при инсталацията на дистрибуцията. Този потребител се нарича още свръхпотребител (суперпотребител), защото няма нещо, което този потребител да не може да направи. **Пред този потребител няма никакви забрани или ограничения и затова работата с него е много удобна, но също толкова опасна.** Поради значимостта на този потребител, командния интерпретатор променя обичайния prompt на командния ред.
Важно е да се отбележи, че името на този потребител не е задължително да бъде root, но това е името по подразбиране.
Освен суперпотребителя всяка Линукс система може да има и потребителски акаунти. Всеки акаунт има собствена директория в дървото на операционната система (изключение правят някои системни акаунти). Тя се нарича home (домашна) директория и потребителя попада винаги в нея след своя вход. Директорията се задава при създаването на потребителя и по подразбиране е : /home/ime_na_potrebitelia .
Обикновено потребителя има пълни права в тази директория, но това не е задължително. Например може да се ограничат правата на потребителя върху определени файлове в тази директория. Класически пример за това е ограничаването на правата върху файла profile, който се изпълнява след включване на потребителя в системата.

Цялата информация за потребителите се съхранява във файла /etc/passwd.
Това е най-обикновен текстов файл, чийто собственик е суперпотребителя.
Само той може да редактира съдържанието му, а останалите потребители имат права само за четене. Редовете в този файл имат строго определен формат:
username:password:userID:groupID:comment:home_directory:login_comm

Нека разгледаме един примерен файл:

root:x:0:0::/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/log:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/:
news:x:9:13:news:/usr/lib/news:

```
uucp:x:10:14:uucp:/var/spool/uucppublic:
operator:x:11:0perator:/root:/bin/bash
games:x:12:100:games:/usr/games:
ftp:x:14:50::/home/ftp:
smmsp:x:25:25:smmsp:/var/spool/clientmqueue:
mysql:x:27:27:MySQL:/var/lib/mysql:/bin/bash
rpc:x:32:32:RPC portmap user:/:/bin/false
gdm:x:42:42:GDM:/var/state/gdm:/bin/bash
pop:x:90:90OP:/:
nobody:x:99:99:nobody:/:
sshd:x:33:33:sshd:/:
dino:x:1000:100inko,,,:/home/dino:/bin/bash
www:x:1001:102:,,,:/home/www:/bin/bash
ffsearch:x:1002:103::/home/ffsearch:
stinger:x:1003:100:,,,:/home/stinger:/bin/rzsh
postfix:x:1004:100::/home/postfix:
```

Всеки ред от този файл е съставен от седем полета, разделени с двоеточия. Ако нищо не се въведе между две двоеточия полето остава празно, но двоеточията задължително трябва да съществуват. Ето и обяснение на седемте полета:
username – уникалното потребителско име
password – паролата на потребителя в кодиран вариант
userID(UID) – уникалното число, което идентифицира потребителя пред системата
groupID(GID) – уникално число, което определя групата на потребителя
comment – коментар (може да истинското име на потребителя и т.н.)
home_directory – това е директорията в която попада потребителя след своето включване
login_comm – това е команда, която се изпълнява след вход на потребителя. Най-често това е някаква шел команда.

Трябва да се отбележи, че тази структура на файла е еднаква не само за Линукс, но и UNIX базираните операционни системи.
Както бе казано по-горе, операционната система записва паролата във второто поле. Повечето съвременни версии, не използват този подход поради проблеми със сигурността. Те пазят паролите в друг файла, наричан shadow password fail, който се намира в /etc/shadow. Когато паролите са записани в този файл, полето за парола съдържа x. Ето как изглежда един shadow файл:

```
root:$1$EnI1gPEs$LwH91OIgMRGwPcdqlleU21:12209:0:::::
bin:*:9797:0:::::
daemon:*:9797:0:::::
adm:*:9797:0:::::
lp:*:9797:0:::::
sync:*:9797:0:::::
shutdown:*:9797:0:::::
halt:*:9797:0:::::
mail:*:9797:0:::::
news:*:9797:0:::::
uucp:*:9797:0:::::
operator:*:9797:0:::::
games:*:9797:0:::::
ftp:*:9797:0:::::
smmsp:*:9797:0:::::
mysql:*:9797:0:::::
rpc:*:9797:0:::::
```

```
gdm:*:9797:0:::::
pop:*:9797:0:::::
nobody:*:9797:0:::::
sshd:*:9797:0:::::
dino:$1$Vd11bYBs$3NV2tT0FK48OR/CYGM/rD0:12209:0:99999:7:::
www:$1$m4as.9uy$sOJvGN/fJKeSItUDNxl3e0:12215:0:99999:7:::
ffsearch:!:12313:0:99999:7:::
stinger:$1$u8M/xs7r$YgtlAmXbGsppdUNk6/onw/:12400:0:99999:7:::
postfix:!:12411:0:99999:7:::
```

Когато потребителя въведе паролата си, програмата login прави проста проверка и сравнява двете пароли. Ако те съвпаднат потребителя се допуска до системата, в противен случай достъпа му се отказва. Въпреки, че преобразуването на паролата до нейния кодиран вариант е кратковременна операция, то обратната операция отнема много време. Полето за парола може да се използва и за ограничаване на достъпа. Това става, като се въведе * на мястото на кодираната парола. Това се използва за потребителски имена като nobody или lp. Обратно, ако това поле се остави празно, то дадения потребител ще може да се включва в системата без парола.

Всеки потребител има асоцииран уникален номер (UID), като чрез него операционната система идентификацира потребителя. Повечето UNIX™ системи използват идентификаторите от 0 до 99 за специални UID, а числата от 100 нагоре за потребителите. Slackware използва различна схема, а именно – от числата от 1000 нагоре се използват за потребителите. И в двата примерни файла има около дузина потребителски имена създадени автоматично от операционната система.

Предназначението на по-важните от тях е:
Потребителя root е суперпотребителя (UID 0)
demon се използва за системни процеси. Използва се само, за да притежава някои процеси и за да се установяват правилно техните права.
bin е собственика на изпълнимите файлове
adm притежава журналните и лог файловете
uucp се използва за UUCP комуникация

Повечето от тях имат звездичка в полето за парола, което забранява използването им при логин.
Всички потребители са организирани в групи. Всяка група подобно на потребителските имена притежава уникален идентификационен номер (GID).
Тя представлява обединение на няколко потребителя по някакви причини. Например в една група могат да бъдат хората от един отдел, а от друга – вашите приятели или клиенти. Така е по-лесно организирането на правата върху определени файлове или други устройства. Всеки потребител може да принадлежи на няколко групи, но в определен момент той е член само на една от тях.
Това е свързано с правата на достъп, тъй като Линукс може да присъедини на потребителя само един идентификатор на група.

Информацията за групите се съдържа във файла /etc/group, който има следния вид:

```
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root,adm
lp::7:lp
```

```
mem::8:
kmem::9:
wheel::10:root
floppy::11:root
mail::12:mail
news::13:news
uucp::14:uucp
man::15:
games::20:
slocate:x:21:
smmsp::25:smmsp
mysql::27:
rpc:x:32:
gdm::42:
ftp::50:
pop::90:pop
nobody::98:nobody
nogroup::99:
users::100:nobody
console:x:101:
sshd::33:sshd
www:x:102:dino,www
ffsearch:x:103:www
utmp::22:
shadow::43:
postdrop:x:104:
```

Формата на файла е следния:

име_на_групата:парола:GID:списък_с_потребители

Името на групата е уникално символно име.
Паролата на групата трябва да бъде въведена от потребителя, ако той пожелае да се присъедини към нея. Не всички дистрибуции използват това поле.
GID е уникалния групов идентификатор
Списъкът на потребителите съдържа всички потребители, които принадлежат на тази група

Всяка Линукс система има няколко системни групи – такива са bin, mail, uucp, sys и др. Присвояването на потребители към тези групи е лоша идея, тъй като това ще предизвика даването на прекалено големи права на тези потребители.
Всяка Линукс дистрибуция притежава вградени програми за добавяне, премахване и промяна на потребителите. Най-простия начин е ръчното редактиране на passwd файла. Това обаче може да предизвика непредвидими ситуации, ако се допусне грешка.
В този случай полето за парола се оставя винаги празно, а паролата се добавя след това.
При ръчна промяна на passwd и group може да се използват съответно командите vipw и vigr. Те извикват специална версия на редактора vi, която следи промените във файловете да бъдат направени така, че да не се достига до повреда на файловете.
Освен това може да се използват командите adduser или useradd (за добавяне на потребители), userdel (за премахване на потребители) и usermod (за промяна на детайлите на потребителя).
Повечето дистрибуции притежават и програми за манипулация с групи.
Това са groupadd, groupdel и groupmod. Командата groups показва към кои групи принадлежи дадения потребител.

В една добре конфигурирана Линукс система не би трябвало да се работи с root потребителя. В някои случаи обаче се налага от неговото използване.
Логичния начин е потребителя да се излезе от системата и след това да влезе като root (ако притежава неговата парола, разбира се). Този подход не е много удобен и затова има алтернатива на него. Това е използването на командата su. При извикването ѝ без допълнителни параметри, то тя се опитва да превключи към root потребителя (разбира се ще се наложи да въведете неговата парола). Освен този неявен начин на употреба, командата позволява превключването към който и да е съществуващ потребител:
#su dino
$

От този момент нататък суперпотребителя работи с привилегиите на потребителя dino. В този случай парола не се изисква, но когато обикновен потребител иска да превключи на друг, система винаги изисква парола. Излизането от този режим става чрез натискането на Ctrl+D или чрез командата exit.

Описание на командния интерпретатор. Видове командни интерпретатори

В тази лекция ще разгледаме няколко основни терминологични термина, които стоят в основата на всяка Linux и UNIX™ система.
Първото от тях е понятието конзола. Това е исторически термин зад който стои дългогодишната история на компютъра въобще. В началото под конзола се е разбирало терминала на който е стоял администратора на машината и на който са излизали системните съобщения. Най-общо казано конзолата е текстово изходно устройство, което прихваща системните съобщения идващи от ядрото на операционната система или от програмата за системни съобщения. В големите компютри, конзолата най-често е била свързана със серийна връзка, работеща по стандарта RS-232 (серийния порт на компютъра). В съвременните персонални компютри под конзола се разбира монитора на компютъра. В Линукс съществува и понятието виртуална конзола, като това са шесте терминални прозореца, които може да се превключват с Alt+F1 до F6.
Важно е да се отбележи, че системните съобщения излизат на първа виртуална конзола даже и никой да е логнат в нея. Типичен пример за това са системните съобщения на ядрото, които се появяват при стартирането на системата.

Друго важно понятие е терминал. Терминал с устройство (най-често хардуерно), което позволява комуникацията с компютъра. По принцип терминала е комбинация от монитор и клавиатура. Това не е така, обаче когато се говори за отдалечен терминал. Отдалечения терминал представлява специална програма съставена от клиент и сървър, чрез която потребителя може да се свързва отдалечено към компютър в който има валидно потребителско име и парола.

Като за начало трябва да знаете, че най-често използваните програми са telnet и ssh.
Най-общо терминала е това което стои между потребителя и шела (командния интерпретатор). Шел се нарича програмата с помощта, на която системата комуникира с потребителя. Тя чете въведените от терминала редове и изпълнява различни операции в зависимост от това което е въведено. След това, шела се опитва да преобразува въвежданите конструкции в инструкции, които ядрото е в състояние да разбере.
Всеки потребител при логическото си включване в системата стартира свое копие на шел в паметта. Това се прави, за да може той да работи без да пречи на останалите потребители на системата.
Шела възприема всичко до първия интервал като команда, а всичко останало, като аргументи на тази команда, като аргументите също се разделят с интервали.
Всички (или поне повечето) Линукс системи имат повече от един валиден шел.
Те са описани във файла /etc/shells. Той има следния прост вид:

/bin/bash
/bin/tcsh
/bin/csh
/bin/ash
/bin/ksh
/bin/zsh
/bin/rbash

Линукс система може да има повече инсталирани командни интерпретатори, но те са недостъпни за използване ако не са описани в този файл. Всеки потребител може да промени своя шел чрез командата chsh. Без параметри командата влиза в интерактивен режим на работа, а чрез опцията -s, този режим се изключва. Формата на командата е следния:

#chsh -s login_shell user
или само
#chsh

В някои дистрибуции е възможно проверка на валидните шелове чрез ключа -l към командата chsh, но Slackware не поддържа този ключ.
Въпреки, че съществуват множество командни интерпретатори, то не всички се използват масово. Най-често използваните командни интерпретатори са:
bash – bash е команден интерпретатор съвместим с шела sh. Той може да изпълнява команди, както от ред, така и от файл. Bash притежава мощен скриптов език, чрез който може да се пишат програми, изпълнявани директно от командния интерпретатор. Освен това bash притежава и функции, които са присъщи на други командни интерпретатори като ksh и csh.
csh – това е шел, който обединява типичните функции за един команден интерпретатор като автоматично завършване на файлов имена при натискането на Tab, управление на задачите и поддръжка на история на командите със синтаксиса на C.
tcsh – подобрена версия на csh, включваща допълнително корекция на синтаксиса и др., като запазва пълна съвместимост с оригиналния csh. Повечето дистрибуции създават символична връзка csh, сочеща към tsch.
ksh – KornShell е команден интерпретатор, поддържащ собствен език за програмиране и позволява изпълнение на команда както от команден ред, така и от файл.
zsh – подобрена версия на ksh. Притежва множество предимства пред него, като функции, вградена синтактична проверка. За разлика от tcsh и csh, то zsh не е напълно съвместим с ksh.

Повечето от изброените командни интерпретатори (bash, ksh, zsh) създават символична връзка, която започва с r и завършва с името на шела (rbash, rksh, rzsh), която указва на командния интерпретатор да превключи в защитен режим.
В този режим потребителя е силно ограничен, като ограниченията зависят от конкретния шел. Описанието на restricted шеловете в /etc/shells не се препоръчва, защото след преминаване към такъв шел, потребителя не може да се върне към обичайния за него команден интерпретатор.
Най-използвания от гореизброените командни интерпретатори е bash. Той е и шела по-подразбиране в Linux. Bash е абревиатура от Bourne Again Shell, създаден от Free Software Foundation на базата на Bourne shell от UNIX™.
Всяка обвивка (обкръжение) има свои променливи на обкръжението. Техните стойности се установяват от командния интерпретатор при стартирането му и имат различни стойности за различните потребители. Ето ви още една причина за всеки потребител да се стартира отделно копие на интерпретатора. Стойностите на тези променливи могат да се променят в процеса на работа.

Например ако не ви харесва знака $ за промпт достатъчно е да смените променливата на обкръжението PS1.
Променливите на обкръжението може да се променят чрез командата:

$export PS1=%
%

само за текущата сесия или чрез промяна на файла .bash_profile или само .profile, като в този случай промяната става активна след като потребителя се логне и е постоянна. Slackware използва файла .bash_profile за въвеждане на команди или настройка на bash. Този файл се зарежда винаги когато потребителя влезе в системата. По-важните опции на bash са:
-r, --restricted – указва на шела да влезе в ограничен режим. В този режим на потребителя са забранени следните операции: промяната на директории с командата cd; промяна на променливите на обкръжението SHELL, PATH, ENV и BASH_ENV; изпълнението на команди съдържащи знака /; задаването на / като аргумент на вградената команда .; задаване на файл съдържащ / като аргумент на вградената команда hash; импортиране на дефиниции на функции при стартиране; пренасочване на изхода с >, >|, <>, >& и >>; изпълнението на командата exec; изключване на режима на ограничения чрез командите set +r или set +o restricted. Освен по този начин режима на ограничения може да се включи и чрез изпълнението на rbash.
-rcfile file – изпълнява командите от файла, вместо от стандартния .bashrc.

Всички командни интерпретатори има множество променливи на обкръжението. Тези променливи описват различни характеристики на системата. Те се четат от останалите програми, които ги използват за настройка на вътрешните си параметри, като език на интерфейса, локализация, тип на системата, версия на операционната система, текуща директория и др.

Bash притежава множество променливи на обкръжението, по-важните от които са:

BASH – указва пълното име на файла използвано за извикването на това копие на bash.
BASH_VESRION – показва версията на командния интерпретатор.
GROUPS – показва списък с групите на които е член потребителя.
HOSTNAME – показва името на компютъра.
OSTYPE – показва на каква операционна система се изпълнява копието на bash
PWD – показва текущата директория
RANDOM – при всяко извикване този параметър се променя от 1 до 32767 по случаен признак.
UID – показва уникалния идентификационен номер на потребителя, който е стартирал копието на bash.
HISTFILE – дефинира и показва файла в който се съхраняват изпълнените команди. По подразбиране това е ~/.bash_history.
HISTSIZE – дефинира и показва максималното количество реда, които може да се съдържат във файла с историята на командите.
HOME – дефинира и показва домашната директория на потребителя.
LANG – дефинира локала на операционната система за всички категории. Отделните категории може да са различни от този параметър.
LC_MESSAGES – дефинира езика на който да се показват различните съобщения.
PATH – показва и дефинира пътя в който се търсят командите.

Четенето на тези променливи може да стане с командата echo:

$echo $HOME
/home/dino

$

Записа на нова стойност става чрез командата export:

$export HOME=/home/ftp
$echo $HOME
/home/ftp
$

Важна особеност е това, че при четене от променливата е необходимо пред нея да се сложи знак за долар, докато при нейна промяна това не е необходимо.
Освен променливите на обкръжението, които се дефинират автоматично от командния интерпретатор, почти всички програми дефинира собствени променливи на обкръжението.
Bash притежава мощен език за програмиране, който позволява писането на скриптове за администриране на системата или за други дейности, които могат да се извършават автоматизирано. Bash скрипта може да представлява или поредица от команди поставени във файл, които се изпълняват последователно, или скрипт написан на езика на bash в който се съчетават външни команди, променливи, цикли, условия и вътрешни bash команди. Пример за прост bash скрипт е следната последователност от команди:

$cat > simple_script
echo "Building file...."
ls -l /dev > dev_list
echo "Number of files:"
wc -l dev_list
rm dev_list
echo "Exiting"
Ctrl+D
$chmod +x simple_script
$./simple_scipt
Building file....
Number of files:
2633 dev_list
Exiting
$

Тази проста поредица от команди са обединени в един текстов файл наречен шел скрипт. Този файл трябва да има права за изпълнение, за да може командния интерпретатор да го изпълни. Повечето шел скриптове обаче не са толкова прости. Примери за сложни скриптове са командите installpkg, pkgtool и т.н, които служат за добавяне, премахване или обновяване на програмите под Slackware. Чрез използването на командите kdialog и dialog на шел скриптовете може да се направи и графичен интерфейс съответно за KDE и за текстова конзола.
Пример за такъв скрипт е pkgtool и netconfig.
Писането на шел скриптове е една от трудните аспекти на Линукс, но тяхното разбиране би довело до пълен контрол върху системата чрез прост набор от предварително написани скриптове.

Базова настройка на системата. Конфигурационна директория /etc . Структура на директорията. Скриптове за начално зареждане

Както всяка операционна система и Линукс се нуждае от настройка след неговото инсталиране. Едно от най-големите удобства на тази операционна система е лесното конфигуриране чрез прости текстови файлове. Тези файлове се намират в директорията

/etc. Тази директория съдържа множество файлове и поддиректории, като тези поддиректори обединяват конфигурационни файлове по някакъв признак. Например всички скриптове, чрез които се пускат и спират процесите, съответно при пускане и спиране на компютъра, се намират в поддиректорията rc.d. Поддиректорията X11 пък обединява конфигурационните файлове на графичния сървър и т.н. В тази лекция ще ви запозная с най-важните файлове и поддиректори намиращи се в конфигурационната директория /etc.
Ще разделя типовете конфигурационни файлове в пет основни групи: настройка на мрежата, настройка на шела, настройка на потребителите, настройка на модулите и настройка на пускането и спирането на операционната система.

1.НАСТРОЙКА НА МРЕЖАТА

Файловете с които се настройва мрежовата конфигурация на операционната система са няколко:
hosts – файла описва имената на компютрите, като ги свързва с техните IP адреси. На всеки компютър може да се присвои и кратко име или псевдоним. Формата на файла е следния:

```
#IP address computer name computer alias
127.0.0.1 localhost
192.168.7.102 server514.mgu.bg server514
192.168.7.103 class514pc1.mgu.bg class514-1
... ... ...
192.168.7.109 class514pc8.mgu.bg class514-8
```

Първият ред е задължителен и описва т.нар. loopback на компютъра.
Редове започващи с # се считат за коментар.

resolv.conf – използването на горния файл е възможно само за малки мрежи или за мрежи нямащи DNS сървър, който да транслира имената в IP адреси. Този метод не може да се използва за хостове в Интернет, където описанието им би било непосилна задача. Затова за Интернет се нуждаем от DNS сървър. Файла resolv.conf служи за описанието на DNS сървъруте към които трябва да се върже компютъра, за да транслира името в IP адрес. Формата на файла е следния:

```
search mgu.bg
nameserver 217.75.128.2
nameserver 217.75.128.9
```

След промяната му се налага изпълнението на командата
#killall -HUP inetd
за да влязат промените в сила.
HOSTNAME – описва името на компютъра. Състои се само от един ред, който има следния вид:
server514.mgu.bg
host.conf – управлява начина по който работи клиента транслиращ името в IP адрес. Състои се от няколко директиви. Формата е следния:

```
order hosts, bind
multi on
```

Тук order и multi са директивите, а hosts, bind и on, техни опции. Пълна информация за файла може да се намери на в man страниците на Линукс.
hosts.allow, hosts.deny, hosts.equiv – описват хостовете на които е позволено или не да се свързват към компютъра. Файла hosts.equiv описва доверените хостовете, на тях е позволено да извършват отдалечени операции чрез командите rlogin и rsh.
inetd.conf – файл в който се описват за кои услуги да отговаря inet демона. Разликата между използването му и нормалното стартиране на сървър се състои в това, че inet демона не стартира услугата, докато тя не се заяви. Това е полезно за услуги, които се използват рядко, тъй като е по-бавно от стандартния начин, но пък се пести оперативна памет. Формата на файла е следния:

#име_на_услуга тип_на_сокета прот. флагове потреб. път_към_сървъра арг.
ftp stream tcp nowait root /usr/sbin/tcpd proftpd
talk dgram udp wait root /usr/sbin/tcpd in.talkd
Име_на_услуга е името на услугата описана във файла services, тип_на_сокета зависи от протокола и е stream за TCP протокол и dgram за UDP. Флагове е поле за задаване параметри на inet демона. Потребител е поле задаващо с какви привилегии ще работи услугата. Път_към_сървъра е пълният път към сървъра обслужващ услугата. Поради проблеми със сигурността се използва специален wrapper наречен tcpd, вместо истинския сървър. Аргументи, са аргумента подаден на tcpd, като тук се задава истинския сървър отговарящ за дадена услуга.
./rc.d/rc.inet1 – шел скрипт, който чете файла rc.inet1.conf и настройва IP адреса на компютъра, IP адреса на гейтуея и мрежовата маска. Описанието им се извършва във файла rc.inet1.conf и е независимо за всяка една мрежова карта.
След промените трябва да се изпълни командата:
#./rc.inet1 stop
#./rc.inet1 start
за да влязат промените в сила.

2.НАСТРОЙКА НА ШЕЛА

Настройката на обкръжението се извършва основно от файла /etc/profile. Той се зарежда всеки път когато някой потребител влезе в системата. Чрез него може да се променят глобалните променливи на обкръжението и да се задават нови, които ще влязат в сила при следващ вход. Последната версия (9.1) на Slackware използва този файла за да зареди скриптовете намиращи се в поддиректорията profile.d.
Тази директория съдържа няколко шел скрипта, които отговарят за различна част от конфигурацията.
lang.sh, lang.csh – в този скрипт може да се променя езика на интерфейса на операционната система и нейната локализация. За да излизат съобщенията на български език е достатъчно да се промени променливата LANG=bg_BG.CP1251. По подразбиране тази променлива е en_EN.
gtk+.sh, gtk+.csh – скрипт който задава параметри отнасящи се за графичната библиотека GTK.
kde.sh, kde.csh – скрипт задаващ променливите на обкръжение, които се използват от KDE и неговите програми.
qt.sh, qt.csh - скрипт който задава параметри отнасящи се за графичната библиотека QT.
/etc/shells – показва валидните командни интерпретатори.

3. НАСТРОЙКА НА ПОТРЕБИТЕЛИТЕ

Файловете, които отговарят за това вече са разглеждани и затова тук ще бъдат само изброени:
passwd – файл в който се пазят регистрираните в системата потребители. В старите версии на Линукс този файл се използваше и за пазене на потребителските пароли.
passwd- е негово резервно копие.
shadow – файл в който се пазят паролите на потребителите. shadow- е негово копие.
group – файл чийто съдържание е групите регистрирани в операционната система.
group- е копие на файла.

4. НАСТРОЙКА НА МОДУЛИТЕ

Модулите са аналога на драйверите за Windows. В Линукс има два основни файла, които отговарят за зареждането и управлението на модулите.
Това са файловете /etc/rc.d/rc.modules, чрез който се задават модулите които трябва да се зареждат при всяко стартиране и файла /etc/modules.conf чрез който се задават параметри на модулите или определен модул се обвързва с определено име на устройство. Първият файл съдържа множество коментирани редове, като всеки ред отговаря на определен модул. Следния ред е типичен ред пример:

#/sbin/modprobe 8139too
Ако на този ред бъде премахнат коментара операционната система ще се опитва да зарежда модула 8139too при всяко стартиране (този модул отговаря за мрежовите карти с чип Realtek 8139). Чрез втория файл може този модул да се присвои към определено устройство. Например:
alias eth0 dmfe
alias eth1 8139too
alias char-major-195 nvidia
Първият ред присвоява към устройство отговарящо за първата мрежова карта модула dmfe (мрежови карти Davicom), а на втората мрежова карта модула 8139too. Последния ред указва на операционната система, че за char устройство с главен номер 195 отговаря модула nvidia (модула на видео картите с чип на Nvidia).

5. НАСТРОЙКА НА ПУСКАНЕТО И СПИРАНЕТО НА ОПЕРАЦИОННАТА СИСТЕМА

Пускането и спирането на операционната система се извършва от шел скрипове намиращи се в директорията /etc/rc.d/. Линукс има седем режима в които може да работи: режим на единичен потребител, многопотребителски режим, режим с графична среда, режим на рестартиране и режим на изключване.
Те са обозначени със следните номера – 1, 3, 4, 6, 0 наречени runlevels. Номерата 2 и 5 не се използват. За всеки един от режимите в директорията има по един шел скрипт:
rc.0 – това е символична връзка към rc.6. Отговаря за runlevel 0 (изключване на операционната система)
rc.K – отговаря за runlevel 1 (еднопотребителски режим)
rc.M – отговаря за runlevel 3 (многопотребителски режим)
rc.4 – отговаря за runlevel 4 (режим с графична среда)
rc.6 – отговаря за runlevel 6 (рестартиране на операционната система)
rc.S – не отговаря за никой от режимите тъй като се стартира преди тях. Използва се за инициализация на операционната система. Тук се прави проверка за коректно изключване на операционната система, стартиране на модулите и т.н.

Всеки един от тези скриптове стартира множество програми необходими за нормалната работа в дадения runlevel. По подразбиране системата влиза в runlevel 3. Това може да се промени чрез промяна на файла /etc/inittab.
id:3:initdefault:
трябва да се промени на
id:4:initdefault:

Освен тези скриптове, в тази директория има още няколко важни скрипта:
rc.local – първоначално този скрипт е празен. Тук администратора може да добавя собствени команди, които ще се изпълняват при всяко стартиране на операционната система.
rc.inet2 – от тук се стартират скриптовете на сървърите които са инсталирани.
rc.inetd – в последната версия, Slackware добавиха този прост скрипт който позволява да се рестартира inetd. Това в старите версии ставаше с командата killall -HUP inetd.
rc.sysvinit – позволява автоматично разпознаване и стартиране на скриптове предназначени за Red Hat. Тъй като Red Hat (Mandrake, Debian, Fedora, SUSE) използва Sys V тип на скриптовете, а Slackware BSD стил, чрез този файл се постига съвместимост.
rc.syslog – управлява системния процес наречен syslog. Той следи и записва възниканлите грешки и по-важни събития в определени журнални файлове.
rc.serial – управлява серийните устройства, като чете конфигурацията от файла /etc/serial.conf

Останалите файлове са скриптове за управление на инсталираните сървъри и зависят от направената инсталация.

Други важни файлове, които не попадат в условната подредба:
/etc/fstab – описва устройствата като хард дискове, камери, CD-ROM, Floppy и др. и опциите им които влизат в сила при монтиране. Освен това тук може да се опишат у-вата които се монтират автоматично при зареждането на операционната система.
/etc/mtab – описва всички монтирани устройства. При всяко закачане и разскачане на у-во, този файл се осъвременява.
/etc/lilo.conf – един от най-важните файлове. Чрез него се променя boot loader-а на Линукс. В някои дистрибуции не се използва LILO за тази цел, а GRUB. В този случай този файл не съществува. Този файл е разделен на секции – по една за всяка инсталирана операционна система. Тук Линукс леко излиза извън тази рамка, понеже един инсталиран Линукс може да има повече от една секция.
Причина за това е, факта че Линукс може да има инсталирани няколко ядра, като за всяко ядро е необходима отделна секция. Ето формата на този файл:

```
# LILO configuration file
# generated by 'liloconfig'
#
# Start LILO global section
boot = /dev/hda
#compact # faster, but won't work on all systems.
prompt
timeout = 300
# End LILO global section
# DOS bootable partition config begins
other = /dev/hda1
label = WinXP
table = /dev/hda
# DOS bootable partition config ends
# Linux bootable partition with ACPI config begins
#image = /boot/kernel-2.4.22-acpi
image = /boot/latest
root = /dev/hda7
label = Linux-2.4
read-only # Non-UMSDOS filesystems should be mounted read-only for checking
append="hdc=ide-scsi"
vga = 791
# Linux 2.6.0 bootable partition with ACPI config begins
#image = /boot/kernel-2.6.3
image = /boot/bzImage-2.6.3
root = /dev/hda7
label = Linux-2.6
read-only # Non-UMSDOS filesystems should be mounted read-only for checking
append="hdc=ide-cd"
vga = 791
# Linux bootable partition config ends
```

Освен тези най-важни файлове и директории, директорията /etc съдържа още множество по-малко критични конфигурационни файлове.
Въпреки това ето някои от тях:

/etc/X11 – директория в който се пази конфигурацията на графичния сървър. Най-важния файл е XF86Config или XF86Config-4. Това е основния конфигурационен файл на X сървъра.
/etc/fonts – в тази директория се пазят конфигурационни файлове отнасящи се към инсталираните шрифтове. Най-важен от тях е файла fonts.conf, който описва директориите с инсталирани шрифтове (не само).
cron.daily, cron.hourly, cron.weekly, cron.monthly – във всяка една от тези директории може да се добавят шел скриптове, програми или линкове към програми, които ще се изпълняват съответно веднъж дневно, веднъж на час, веднъж на седмица или веднъж на месец.
/etc/ld.so.conf – файл описващ директориите в които операционната система пази своите библиотеки. След промяната му трябва да се изпълни командата ldconfig, за да влязат промените в сила. Във файла ld.so.cache се пази кеширана информация за инсталираните библиотеки.
/etc/lpd – файл за конфигурация на демона позволяващ принтиране.
Промяната на всички тези файлове става единствено с обикновен текстов редактор. Повечето Линукс дистрибуции притежават шел скриптове, чрез които тази промяна е по-удобна и лесна. За съжаление няма Линукс дистрибуция в която да има шел скриптове за всички възможни промени. Затова универсалния и най-гъвкав начин за конфигуриране на операционната система е простия редактор.

Конфигуриране на системата. Мрежова конфигурация

Един от най-важните аспекти на Линукс е работата му в мрежа. Основаната команда, която служи за конфигурирането на мрежата е ifconfig. Чрез нея може да се задават IP адресите на мрежовите карти, мрежовата маска и т.н. Най-простия начин за нейното използване е:
#ifconfig
Това ще покаже всички интерфейси , които са активни в момента.

Типичен изход от тази команда е:

```
eth0 Link encap:Ethernet HWaddr 00:48:54:51:96:36
inet addr:217.75.146.24 Bcast:217.75.146.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:167735 errors:27 dropped:24 overruns:23 frame:2
TX packets:33803 errors:0 dropped:0 overruns:0 carrier:0
collisions:0
RX bytes:63976099 (61.0 Mb) TX bytes:2873645 (2.7 Mb)

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:1187 errors:0 dropped:0 overruns:0 frame:0
TX packets:1187 errors:0 dropped:0 overruns:0 carrier:0
collisions:0
RX bytes:94999 (92.7 Kb) TX bytes:94999 (92.7 Kb)
```

eth0 и lo са идентификаторите на мрежовите интерфейси като eth0 е първата мрежова карта, а lo е интерфейса осигуряващ loopback. Този интерфейс съществува винаги, даже и ако компютъра няма инсталирана мрежова карта. inet addr показва IP адреса, който е присвоен на съответния интерфейс; Bcast – адреса за броадкаст на мрежата; Mask - мрежовата маска. Link encap показва типа на интерфейса. Ако интерфейса е мрежова карта Hwaddr показва нейния хардуерен адрес (MAC адрес).

Друг начин за извикване на командата ifconfig е изричното задаване на интерфейса, който искаме да видим:

#ifconfig eth0
eth0 Link encap:Ethernet HWaddr 00:48:54:51:96:36
inet addr:217.75.146.24 Bcast:217.75.146.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:210005 errors:46 dropped:41 overruns:40 frame:6
TX packets:49380 errors:0 dropped:0 overruns:0 carrier:0
collisions:0
RX bytes:102780349 (98.0 Mb) TX bytes:3836101 (3.6 Mb)

Освен разглеждането на мрежовите интерфейси, командата позволява и манипулиране с тях. Например спирането и пускането на мрежовия интерфейс eth0 става по следния прост начин:
#ifconfig eth0 down
#ifconfig eth0 up

След изпълнението на първата команда, всякакъв трафик от и към eth0 се прекъсва. Той се възстановява след изпълнение на втората команда.
Основната функция на командата, обаче е задаването на настройки на мрежовите интерфейси.
По-важните параметри на командата са:
interface – името на интерфейса, който ще се настройва. Например eth0. Вместо interface се слага това име.
up – „вдига“ интерфейса с име interface. След тази команда интерфейса става активен.
down – „сваля“ интерфейса с име interface. След тази команда интерфейс се деактивира.
[-] arp – позволява или забранява използването на ARP протокол за този интерфейс.
[-]promisc – чрез този флаг се спира или пуска т.нар. promiscuous mode.

В този режим всички пакети в мрежата ще се прихващата от интерфейса.
metric N – задава метриката на интерфейса.
mtu N – задава големината на MTU (Maximum Transfer Unit). Това е големината на пакета, която се предава на един път. По подразбиране за Ethernet карти, големината е 1500 байта.
netmask addr – задава мрежовата маска на интерфейса.
[-] broadcast [addr] – ако се зададен аргумент (addr), се променя адреса за броадкаст на мрежата. Ако не е зададен се включва или изключва флага IFF_BROADCAST на интерфейса.
[-]pointopoint [addr] – ако е зададен аргумент (addr) се активира режима „от-точка-до-точка“ (point-to-point). Този режим означава, че се осъществява директна връзка между два компютъра без никой между тях. Този режим се използва от модемите при осъществяване на връзка.
address – на мястото на този флаг се поставя IP адреса на интерфейса.

Ето няколко примерни конфигурации:
#ifconfig eth0 192.168.9.100 – задава на интерфейса eth0 IP адрес 192.168.9.100
#ifconfig ppp0 mtu 512 – задава размер на MTU 512 байта. Удачно при dial-up връзки.
#ifconfig eth0 broadcast 192.168.9.0 – задава като броудкаст адрес, мрежовия адрес.
По подразбиране броудкаст адреса е 192.168.9.255
#ifconfig eth0 netmask 255.255.255.32 – задава мрежова маска на интерфейса eth0.
По подразбиране маската ма мрежа клас C е 255.255.255.0.

Командата ifconfig може да се използва и за присъединяване на няколко IP адреса към една и съща мрежова карта. Те образуват „псевдоними“:

#ifconfig eth0:1 192.168.10.100 up
Друга важна команда е route. Чрез нея администратора може да манипулира маршрутизиращата таблица на ядрото на операционната система.

Най-простото използване на командата е:
#route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
localnet * 255.255.255.0 U 0 0 0 eth0
loopback * 255.0.0.0 U 0 0 0 lo
default moria2.online.b 0.0.0.0 UG 1 0 0 eth0
Ако се използва ключа -n се спира търсенето името на хоста, като вместо него се показва IP адреса:
#route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
localnet * 255.255.255.0 U 0 0 0 eth0
loopback * 255.0.0.0 U 0 0 0 lo
default 217.75.146.1 0.0.0.0 UG 1 0 0 eth0

При този начин на използване на командата се визуализира текущата маршрутизиращата таблица. Премахването на ред от таблицата става с командата:
#route del default
Това ще премахне gateway-а от таблицата. Ако искаме да добавим нов, то трябва да изпълним следната команда:
#route add default gw 192.168.7.1 или #route add default gw gateway2.compsc.mgu.bg
Разбира се, за да се изпълни коректно втория запис е необходимо да има DNS сървър, който да преведе името gateway2.compsc.mgu.bg в IP адреса 192.168.7.1.
Чрез използването на тези две команди може да се конфигурира мрежовата карта и gateway на един компютър, така че той да има достъп до локалната мрежа.
Ако се зададе и gateway, компютъра ще получи и достъп до Интернет. Достъпът до Интернет изисква и задаване на DNS сървър във файла /etc/resolv.conf. Въпреки, че бе разгледан, ето формата на този файл:
search axelbg.com
nameserver 217.75.146.1
nameserver 217.75.128.9
Тук може да се задават неограничен брой DNS сървъри, като вградения DNS клиент (наречен resolver) ще ги обходи всичките, докато не намери такъв, който да върне IP адреса на търсения хост.
Въпреки, че е възможно, задаването на тези параметри всеки път е неудобно. Затова Линукс притежава конфигурационни файлове, в които те се пазят.

В миналата лекция те бяха бегло спомената, но тук ще бъдат разгледани по-подробно. Slackware притежава вграден скрипт, който пуска, спира и задава необходимите настройки на мрежата. Този скрипт се намира в директорията /etc/rc.d/ и се нарича rc.inet1. Той чете конфигурацията от файла rc.inet1.conf намиращ се в същата директория.
Този файл има следния вид:
# /etc/rc.d/rc.inet1.conf
# Config information for eth0:
IPADDR[0]="217.75.146.24"
NETMASK[0]="255.255.255.0"
USE_DHCP[0]=""
DHCP_HOSTNAME[0]=""

```
# Config information for eth1:
IPADDR[1]=""
NETMASK[1]=""
USE_DHCP[1]=""
DHCP_HOSTNAME[1]=""
...
GATEWAY="217.75.146.1"
DEBUG_ETH_UP="no"
```

Тук за всеки един мрежови интерфейс има отделна секция. По-подразбиране тези секции са три, като при добавяне на нова секция е необходимо и промяната на файла rc.inet1.
Обяснените на всеки един от параметрите е следното:
IPADDR – IP адреса на мрежовия интерфейс. Ако такъв интерфейс не съществува полето трябва да е празно (виж IPADDR[1])
NETMASK – мрежовата маска на интерфейса.
USE_DHCP – Ако се присвои стойност „yes“ се активира динамичното търсене на IP адрес чрез DHCP протокол. Ако не се използва DHCP трябва да се остави празно.
DHCP_HOSTNAME – това поле задава IP адреса на DHCP сървъра. Ако не се използва DHCP трябва да се остави празно.
GATEWAY – задава gateway-а по подразбиране. Скрипта rc.inet1 чете този ред и добавя адреса на този gateway в маршрутизиращата таблица.
DEBUG_ETH_UP – включва режим на debug. В този режим всички съобщения се показват на екрана на компютъра (или на стандартния изход, ако е зададен друг), вместо в журналните файлове.

Активирането на всички интерфейси става с командата:
#/etc/rc.d/rc.inet1 start или #/etc/rc.d/rc.inet1,
а деактивирането с командата:
#/etc/rc.d/rc.inet1 stop

Използването на DHCP налага инсталирането на специален клиент със собствен конфигурационен файл. Този файл отново се намира в директорията /etc и се нарича dhcpd.conf. В началото той е празен, тъй като настройките по подразбиране, в повечето случаи са достатъчни. Затова той трябва да се променя само ако се забележат проблеми с DHCP сървъра.
Други важни конфигурационни файлове се hosts, HOSTNAME и inetd.conf.

Те бяха разгледани в миналата лекция.
За удобство на потребителя повечето дистрибуции притежават приложения, които улесняват конфигурирането на мрежата. Някои от тях са конзолни шел скриптове, а други са графични приложения.

Ето и приложенията с които става това при по-популярните дистрибуции:

Red Hat – Red Hat (отскоро Fedora) е оборудван както с графично, така и с конзолно приложение за конфигуриране на мрежата. Графичното приложение може да се стартира от System Settings -> Network или чрез командата redhat-config-network. Същото приложение може да работи и в текстов режим, като за тази цел то трябва да се извика чрез командата redhat-config-network-tui.

SUSE – SUSE притежава мощно средство за цялостно конфигуриране на системата наречено YaST. YaST се интегрира в Control Panel на KDE и може да бъде стартиран от там. В конзолен режим YaST се стартира с командата yast2.

Mandrake – Mandrake също притежава програми за удобно конфигуриране на мрежата. Можете да ги извикате с командите drakconnect или drakconf. Под конзола може да използвате netconf за целта.

Knoppix – администрирането на мрежата става с командата netcardconfig.

Slackware – Slackware няма графично приложение за конфигуриране на мрежата, но притежава шел скрипт, който прави това. Той се стартира чрез командата netconfig. Този скрипт попълва необходимите файлове и се опитва да „вдигне“ интерфейса eth0. След неговото изпълнение е необходимо да се изпълнят командата killall -HUP inetd или /etc/rc.d/rc.inetd restart (в последната версия на Slackware), за да влезе в сила промяната на файла resolv.conf.

Конфигуриране на системата. Конфигурационен файл /etc/fstab. Прикачване и разкачване на устройства и опции на командата mount. Пакетни системи. Видове пакетни системи – tgz, rpm, deb. Инсталиране на бинарни пакети. Пакетна система на Slackware Linux. Команди за инсталиране, премахване и обновяване на пакети. Програми на трети страни за инсталация и създаване на пакети. Обновяване на системата през Интернет.

Както бе споменато още в началото, всеки хард диск или дял от хард диск представляват файлове в директорията /dev. За да може да се използва хард диска е необходимо преди това той да се монтира или прикачи. Командите, които монтират или демонтират блоково устройство (а такива са всички хард дискове, CD-ROM и др.) са mount (за монтиране) и umount (за демонтиране). Mount и umount използват за своята работа два важни файла – fstab и mtab. Fstab служи за описание на файловите системи, които може да се монтират, а в mtab се описват монтираните в момента файлови системи. Това описание се извършва автоматично от mount и umount. mount има множество опции, по-важните от които са:
-a – монтира всички файлови системи описани във файла /etc/fstab
-n – монтира дадено устройство без да прави запис във файла /etc/mtab. Тази опция е полезна когато файла mtab се намира на файлова система, която е само за четене.
-r – монтира файловата система само за четене. Синоним на тази опция е -o ro
-w – монтира файловата система за четене и писане. Синоним на тази опция е -o rw
-t – тип на файловата система, която се монтира. По-важните от тях са ntfs, vfat, raiserfs, msdos, ext2, ext3 и iso9660. По принцип ядрото, в повечето случаи, само установява правилно типа на файловата система, но в понякога се налага нейното ръчно задаване. За да може да се монтира даден тип файлова система, трябва ядрото да е компилирано с поддръжка за нея.
-o – задава множество опции на командата mount. Тези опции са различни за разлините файлови системи. Освен това някои от тях имат смисъл само ако са описани във файла fstab.
defaults – задава опциите по подразбиране, а именно rw, suid, exec, auto, nouser и async
async – задава асинхронен режим на работа на файловата система.
auto – файловата система може да бъде монтирана чрез ключа -a.
exec – позволява изпълнението на файлове от тази файлова система.
ro – монтира файловата система само за четене.
rw – монтира файловата система за четене и писане.
suid – позволява флаговете SUID и GUID да имат ефект за дадената файлова система.
user – позволява на обикновения потребител да монтира файловата система. Името на потребителя монтирал файловата система се записва във файла mtab, така че той да може после да я демонтира.
users – позволява на всички потребители да монтират и демонтират дадената файлова система.

Повечето файлови системи имат собствени опции, които се различават както по описание, така и по значение. Това, както и техния брой (над 30) прави запомнянето на всяка опция трудна задача. Информация за отделните файлови системи и техните опции може да се намери на man страницата на командата mount.
Командата umount има значително по-малко опиции от mount. По-важните от тях са:
-n – демонтира файловата система без да пише във файла mtab.
-r – ако демонтирането пропадне, umount пробва да превключи файловата система в режим само за четене.
-a – демонтира всички файлови системи описани в mtab.
-f – принудително демонтиране на файловата система. Тази опция изисква ядро с версия 2.1.116 или по-висока.
-l - „мързеливо“ демонтиране. Отделя файловата система от йерархията веднага и изчиства всички връзки към нея, веднага щом тя се освободи. Опцията изисква ядро с версия 2.4.11 или по-късна.

Ето няколко примера за използването на двете команди:

#mount /dev/hdc1 /mnt/hd – монтира /dev/hdc1 в директорията /mnt/hd
#mount -t ntfs /dev/hda1 /mnt/hd – монтира /dev/hda1 в директорията /mnt/hd, като задава и типа на файловта система
#mount /dev/hda1 /mnt/hd -o ro – монтира файловата система в режим само за четене
mount -r /dev/hda1 /mnt/hd – същото, но записано по друг начин
#umount /mnt/hd – демонтира файловата система прикачена към директорията /mnt/hd

Командата mount може да се използва и за монтиране на побитово копирани от дискета, CD-ROM или хард диск файлове. Пример за такива файлове са свалените от Интернет дистрибуции на Линукс с разширение iso.
Ето как става това:
#mount /home/dino/knoppix-3.4.iso /mnt/tmp -o loop
След тази команда вие ще може да разгледате съдържанието на файла knoppix-3.4.iso преди да го запишете на CD-ROM.
Един от най-важните файлове в Линукс е файла fstab. Тук се описват файловите системи които трябва да се монтират при стартиране на операционната система и точките в които те се монтират. Освен това тук се задават и опции специфични за дадената файлова система. Един типичен fstab файл има следния вид:

```
/dev/hda6 swap swap defaults 0 0
/dev/hda7 / reiserfs defaults 1 1
/dev/hda8 /mnt/storage vfat
uid=1000,users,iocharset=cp1251,codepage=866,exec,umask=000 0 0
/dev/hda1 /mnt/hd ntfs users,exec,uid=1000,iocharset=cp1251,ro 0 0
/dev/sr0 /mnt/cdrw iso9660 user,unhide,noauto,owner,users,iocharset=cp1251,exec 0 0
/dev/hdb /mnt/cdrom iso9660 user,unhide,noauto,owner,ro,users,iocharset=cp1251,exec 0 0
/dev/fd0 /mnt/floppy auto noauto,owner 0 0
devpts /dev/pts devpts gid=5,mode=620 0 0
proc /proc proc defaults 0 0
```

fstab е разделен на 5 колони. Първата от тях указва за кое устройство става дума. Втората колона задава т.нар. точка на монтиране, третата – типа на файловата система. Четвъртата колона е за задаване на опции към файловата система. Последната колона се състои от две числа. Първото от тях оказва дали на файловата система е позволено да се прави dump, а второто указва дали файловата система да се проверява за грешки при стартиране на операционната система. То трябва да бъде 1 за root файловата система (там където е инсталиран Linux, като в случая това е /dev/hda7), 2 за останалите и 0 за тези, които не трябва да се проверяват. Всички файлови системи зададени във

fstab ще се монтират автоматично при стартиране на операционната система. Освен това за описаните тук файлови системи може да се използва и съкратен запис на командата mount:
#mount /mnt/storage
вместо
#mount /dev/hda8 /mnt/storage

Файла mtab се попълва автоматично и неговия формат не е толкова важен, затова няма да бъде разгледан.
Всяка Линукс дистрибуция се разпространява с множество по-малки програми и приложения. Всички те са организирани в пакети. Пакета е набор от файлове, които са архивирани и имат определена структура на архива. Тази структура е различна за различните видове пакетни системи. Тя е най-проста в пакетите на Slackware и най-сложна за пакетите на Debian. По средата стои пакетната система на Red Hat. Видовете пакетни системи се разпознават по разширението на файла – tgz за Slackware, rpm за Red Hat, Fedora, Mandrake ,SUSE и deb за Debian, Knoppix. От тях единствено tgz не притежава проверка на зависимости и обновяване през Интернет. Тук ще наблегна на пакетната система на Slackware, но в края ще бъдат разгледани и пакетните системи rpm и deb.
Пакета на Slackware представлява архивен файл с разширение tgz. Освен файловете на програмата във файла се намира и една директория наречена install. Тук се може да има следните файлове:
doints.sh – това е шел скрипт, който се изпълнява веднага след копирането на файловете.
slack-desc – файл описващ предназначението на програмата. Това описание се показва по време на инсталирането на пакета.
slack-required – официално не се поддръжа от инсталатора на Slackware.
Служи да окаже на някои външни програми какви програми и/или библиотеки са необходими за правилната работа на програмата.
Всеки пакет има следния вид на името:
coreutils-5.2.1-i486-1.tgz

Тук coreutils е името на пакета, а 5.2.1 е неговата версия. i486 е архитектурата за която е създаден пакета. В случая това пакета е създаден за Intel процесор (или съвместим) и ползва оптимизации залегнали в архитектурата на 486 поколението процесори. Това означава, че този пакет няма да работи на машини с процесори по-ниски от 486 (386 например). Много дистрибуции предлагат своите пакети в няколко варианта – например за 586,486 и 386. Използването на пакет компилиран за ниска версия на процесора води до намалено бързодействие, особено за натоварващи приложения като сървъри. Последната цифра е т.нар. build номер. Разширението показва, че става дума за пакет на Slackware.
Slackware притежава вградени приложения за инсталиране, премахване и обновяване на пакети. Това съответно са installpkg, removepkg и upgradepkg. Опциите на installpkg са:
-warn – показва кои директории и/или файлове ще бъдат премахнати или презаписани при инсталирането на дадения пакет. Инсталация на пакета не се извършва.
-root /root_dir – инсталира пакетите в различна от подразбиращата се директория (а тя е /).
-infobox – използва програмата dialog, за да показва съобщенията в текстов диалогов прозорец.
-menu – показва диалогов прозорец с избор дали потребителя желае да инсталира пакета или не.
-tagfile – указва различен от подразбиращия е се файл с приоритети.

Опциите на removepkg са:

-warn – генерира отчет кои директории и файлове ще бъдат премахнати, без да ги премахва в действителност.
-preserve – запазва цялостната структура на пакета при премахването му в /tmp/preserved_packages/packagename
-copy – копира структурата на пакета в /tmp/preserved_packages/packagename без да го премахва
-keep – запазва определена информация, създадена в процеса на премахване на пакета.

И тук може да премахнете пакет инсталиран в различна от / директория. Това става така:
#ROOT=/mnt removepkg package_name

Опциите на upgradepkg са:
-dry-run – показва отчет кои пакети ще инсталирани или обновени, без да ги обновява в действителност.
--install-new – по принцип upgradepkg обновява само съществуващи вече пакети, като пренебрегва тези които не са инсталирани. Тази опция модифицира поведението на upgradepkg и той започва да инсталира нови, неинсталирани пакети.
--reinstall – по принцип upgradepkg обновява пакети с по-нова версия или build номер. Тази опция модифицира това поведение, като кара upgradepkg да инсталира пакети със същия номер на версия.

Ето и няколко примера за работа с пакети:
#installpkg coreutils-5.2.1-i486-1.tgz
#installpkg -root /mnt k3b-0.11.9-i686-1.tgz
#upgradepkg coreutils-5.2.2-i486-1.tgz
#removepkg apache-1.3.29-i486-2.tgz

Освен тези програми за работа с пакети Slackware притежава и приложението наречено pkgtool. Това е приложение, което използва текстови диалогови кутии и менюта. Друга възможност за улеснено манипулиране с пакети е kpackage. Това е вградено в KDE приложение, което може да работи с пакетната система на Slackware.
Тъй като пакетната система на Slackware не разполага с обновяване през Интернет и проверка на зависимости в Мрежата се появиха приложения, които запълниха тази ниша. Някои от тях влязоха в EXTRA диска на Slackware, което означава, че са достатъчно сигурни за работа. Това са приложенията swaret и slapt-get. И двете притежават възможност за автоматично търсене и обновяване на пакети, както и поддръжка на проверка на зависимости. Разбира се, те са далеч от функционалността на приложения като apt-get на Debian, но все пак работата която вършат е много добра за техния размер. И двете приложения има зачатъци на графичен интерфейс, който прави работата с тях по-удобна.
Например обновяването на дистрибуцията със swaret става по следния елементарен начин:
#swaret --update
swaret 1.6.2-1

[ftp://ftp.mirrors.unixsol.org/slackware/slackware-current/]
Fetching CHECKSUMS List File... DONE!
FILELIST List File... DONE!
Packages Descriptions... DONE!
Extra Packages Descriptions... DONE!
ChangeLog... DONE!
#swaret –upgrade
След това програмата ще ви запита, кои от намерените нови пакети да инсталира като можете да избирате пакетите един по един или да кажете да се инсталират всички.

Освен това преди да започенете обновяване е добре да добавите български огледала на версията current на Slackware. Това става като промените файла /etc/swaret.conf.
Ето и линкове на два огледални сайта на Slackware в България:
ftp://ftp.mirrors.unixsol.org/slackware/slackware-current/
ftp://mirrors.evrocom.net/slackware/slackware-current/
Чрез swaret може да инсталирате, обновявате и премахвате пакети точно както и с вградените инструменти, но в единен интерфейс. Не по-малко важен е и факта, че swaret притежва интерфейс на български.
Можете да намерите swaret на официалната му страница в Интернет:http://swaret.org/
Българския превод може да намерите на: http://swaret.org/i18n/swaret.lang.BULGARIAN
Една от най-широко разпространените пакетни системи е rpm (Red Hat Package Manager) на Red Hat. Тя се използва в множество популярни дистрибуции като Red Hat, Mandrake, SUSE, Fedora и др. rmp поддържа обновяване през Интернет и проверка на зависимости. Програмата чрез която се инсталират, премахват и обновяват пакети се нарича rpm. Ето примери за нейното използване:
#rpm -i foobar-1.0-1.i386.rpm – инсталиране на пакета foobar-1.0-1.i386.rpm
#rpm --force -i foobar-1.0-1.i386.rpm – принудително инсталиране на пакета. Използването на тази опция е опасно за дистрибуцията.
#rpm --nodeps -i foobar-1.0-1.i386.rpm – инсталиране на пакета без проверка на зависимости. Използването на тази опция също не се препоръчва.
#rpm -i ftp://ftp.redhat.com/pub/redhat/rh-2.0-beta/RPMS/foobar-1.0-1.i386.rpm – инсталиране на пакета от Интернет.
#rpm -e foobar – премахване на пакета foobar
#rpm --nodeps -e foobar – премахване на пакета foobar без проверка на зависимост. След това е възможно програмите, нуждаещи се от този пакет да не работят коректно или въобще да не работят.
#rpm -U foobar-1.1-1.i386.rpm – обновяване на пакета.
Това са много малко от опциите на програмата rpm (за повече информация man rpm).
Другия широко разпространен пакетен формат е deb. Приложението за работа с deb пакети се нарича dpkg. Ето примери за неговото използване:
#dpkg -i xpdf_2.02pl1-1_all.deb – инсталира пакета xpdf_2.02pl1
dpkg -r xpdf_2.02pl1-1_all – премахва пакета xpdf_2.02pl1
#dpkg -l – показва списък с инсталираните пакетите
Използването на dpkg е сложна и неприятна дейност, особено за пакети с много зависимости. Затова е по-добре да се използва инструмента apt. Той изцяло стъпва върху dpkg за основните дейности по премахването или инсталирането на пакетите, но тук самото apt се грижи да доставката на файловете необходими за инсталирането на пакета. Преди да се използва apt трябва да се конфигурира. Тази конфигурация включва задаване на източници на пакети във файла /etc/apt/sources.list. Той има следния вид:
deb http://security.debian.org/ stable/updates main contrib non-free
deb http://mirrors.ludost.net/debian stable main contrib non-free
deb http://ftp.bg.debian.org/debian stable main contrib non-free
След това е необходимо само да се изпълнят командите:
#apt-get update
#apt-get upgrade
Инсталирането на пакети става по следния начин:
#apt-get install xpdf
Тази команда инсталира пакета xpdf и всички пакети необходими за неговата работа.
Чрез apt-get може и да се премахват пакети:
apt-get remove xpdf
Това обаче не премахва пакетите инсталирани, за да се осигури нормална работа на пакета xpdf.
Информация за даден пакет може да се покаже с програмата apt-cache:
$ apt-cache show xpdf

Освен конфигурирането почти всички действия на apt-get могат да се извършат и с интерактивната програма aptitude. Употребата й е препоръчителна за не напреднали потребители.
Повече информация за пакетната система и програмите които работят с нея може да намерите на адрес http://debian-book.debian-nikola.ho...lit/node73.html като информацията е на български.

Конфигуриране на системата. Стартиране и конфигурация на SAMBA сървър. Достъп до споделени Windows ресурси под Линукс и обратно.

В много случаи мрежата в която работи вашата Линукс машина е хетерогенна, т.е. освен Линукс машини има и множество Windows базирани такива. Ако се налага споделяне на файлове или принтери, които да са видими за всички компютри в такава мрежа се налага използването на протокола Server Message Block(SMB). Този протокол се използва от Windows базирани компютри за организиране на достъп до файлове и принтери (това са т.нар. споделени файлове (shared files) и споделени принтери (shared printers)). Използвайки набора от помагала Samba на Андрю Тридгел, UNIX™ - машини могат да организират достъп до дисковете и принтерите за Windows машини.

Това което може да правите със Samba е :
да се дава достъп до дисковете на Linux Windows-машини.
да се получава достъп до дисковете на Windows за машини под Linux.
да се дава достъп до принтерите под Linux за Windows-машини.
да се получава достъп да принтерите под Windows от Linux-системи.
Преди да може да работите със Samba трябва вашата локална мрежа да работи с TCP/IP протокол, тъй като тя работи само с този протокол. Освен това Samba може да работи само в рамките на един сегмент от мрежата, тъй като използва броудкаст за разпращането на пакетите. Това не позволява на Samba да работи с маршрутизатори, освен ако не е настроено тунелно IP. И накрая... за да работите със Samba, то вие трябва да я имате инсталирана.

Можете да проверите дали имате инсталирана Samba със следната команда:
#whereis smbd
smbd: /usr/sbin/smbd /usr/man/man8/smbd.8.gz /usr/share/man/man8/smbd.8.gz
Ако изхода е подобен на показания, то вие имате инсталиран Samba пакета. В противен случай ще трябва да го инсталирате. Това може да стане както от предоставяните от дистрибуцията пакети, така и от изходен код. В момента на писане на лекцията последната версия на Samba е 3.0.2a и може да бъде изтеглена от http://www.samba.org.

Основните файлове в Samba са:
smbd – това е демона който осигурява работата със SMB
nmbd – демон осигуряващ NetBIOS имената
smbclient – клиент за SMB за UNIX™ машини
smbprint – скрипт за печатане на принтер на SMB машина
smbstatus – показва информация за осъществените SMB връзки
smbmount и smbumount – монтират отдалечените SMB ресурси на локалната файлова система
smb.conf – конфигурационния файла на Samba сървъра
Двата основни демона може да работят както като автономни процеси, така и като процеси контролирани от inetd. Принципна разлика в конфигурацията няма, с изключение на това, че Samba ще работи малко по-бързо ако е пусната като автономен процес. Преди да пуснете Samba уверете се, че следните редове съществуват във файла /etc/services и ако не съществуват ги добавете:
netbios-ns 137/tcp nbns
netbios-ns 137/udp nbns

netbios-dgm 138/tcp nbdgm
netbios-dgm 138/udp nbdgm
netbios-ssn 139/tcp nbssn
Ако сте инсталирали Samba от пакета на Slackware, то сървъра може да се пусне по следния начин:
#/etc/rc.d/rc.samba start – пускане на сървъра
#/etc/rc.d/rc.samba stop – спиране на сървъра
etc/rc.d/rc.samba restart – рестартира сървъра (при промяна на конфигурационния файла се налага рестартиране)
Ако използвате Red Hat скрипта се намира в /etc/rc.d/init.d/ и се нарича smb.

За да може да използвате пълноценно Samba, то тя трябва да се настрои. Настройката се извърша в един единствен файла smb.conf, който в Slackware се намира в директорията /etc/samba. В други дистрибуции местоположението на файла може да е различно, като най-вероятно е да се намира в директорията /etc/.
Чрез файла smb.conf вие определяте към какви системни ресурси искате да дадете достъп за външния свят и какви ограничения искате да определите при използването на тези ресурси. Този файл е разделен на раздели, като всеки раздел започва с определено име – например [global], [homes], [printers] и т.н.
Секцията [global] определя някой променливи, които Samba ще използва за определяне на достъпа до всички ресурси.
Раздела [homes] позволява на отдалечените потребители да имат достъп до своите (и само до тях) домашни директории на локалната Linux-машина. Така че, ако потребителите на Windows се пробват да се включат към този раздел от своите Windows машини, то те ще бъдат включени към своите персонални домашни директории. Ще отбележим, че за да могат да направят това те трябва да са регистрирани на Linux-машината.

Нека сега започнем с прост smb.conf файла, който ще позволява на отдалечените потребители да имат достъп към техните домашни директории на локалната машина и да пишат във временна директория.

```
;/etc/smb.conf
[global]
; Разкоментирайте този ред , ако вие искате да дадете достъп на потребителя "гост"
; guest account = nobody
log file = /var/log/samba-log.%m
lock directory = /var/lock/samba
workgroup = Class514
netbios name = Server514
server string = Class 514 - server resources

[homes]
comment = Home Directories
browseable = no
read only = no
create mode = 0750

[tmp]
comment = Temporary file space
path = /tmp
read only = no
public = yes
```

Нека разгледаме секциите една по една.
Секцията [global] съдържа шест реда, първият от които е коментиран. Вторият указва къде ще се пази журналния файла на Samba сървъра, а третия - файла, който заключва

Samba от повторно стартиране. Четвъртия ред задава групата в която ще работи сървъра и петия – неговото име. Последният ред задава кратко обяснение на сървъра.
Секцията [global] може да съдържа и още параметри по-важните от които са:
interfaces – задава на кой мрежови интерфейс ще работи Samba сървъра. Например interfaces = 192.168.0.1/24 127.0.0.1/24
bind interfaces only – ако стойността на този параметър е yes, то заявките идващи от броадкаст адреси различни от тези на интерфейсите описани в interfaces, ще бъдат отказвани. Това, в комбинация с параметъра interfaces повишава сигурността на сървъра.
time server – ако стойността е yes, то nmbd се идентифицира като time сървър на Windows машините.
encrypt passwords – ако стойността е no, се деактивира криптирането на паролите. Трябва да имате в предвид, че Windows 98/NT и по-нови използват криптирани пароли.
socket option – задава специални параметри към сокета на който работи Samba сървъра. Чрез тях се контролира мрежовото ниво от ISO/OSI модела. Параметъра се използва за ускоряване работата на сървъра. Препоръчителни опции са TCP_NODELAY и SO_SNDBUF=8192.
preferred master – ако стойността на параметъра е yes, то Samba сървъра ще стане MASTER BROWSER за дадената работна група (workgroup). Това означава, че останалите компютри в групата ще се свързват към него, за да получат информация за другите компютри в групата.
security – задава как клиентите да отвръщат на Samba сървъра и е един от най-важните параметри. Възможните стойности са:
USER – използва се ако потребителите, които ще се свързват към сървъра имат същите потребителски имена под Windows, както тези под Линукс. Ако някое име не съвпадне, то достъпа му се отказва. Това е стойността по подразбиране.
SHARE – сървъра не изисква валидно потребителско име и парола, за да позволи на клиента да осъществи връзка.
DOMAIN – тази стойност на параметъра security трябва да се използва само ако машината е добавена в Windows NT Domain (чрез програмата net идваща с пакета Samba). Освен това тя изисква параметъра encrypt passwords да има стойност yes.
За да може Линукс да свърже потребителя е необходимо той да присъства като валиден потребител и в Линукс машината.
SERVER – в този режим Samba ще се опита да валидира потребителското име и паролата от друг SMB сървър, например NT машина. Ако това пропадне, Samba превключва на security = USER. Изисква се encrypt passwords да има стойност yes, с изключение в случаите когато другата машина не поддържа криптирани пароли.
ADS – в този режим Samba ще работи като член на домейн в ADS (Active Directory Service)област . Този режим изисква инсталиран и конфигуриран Kerberos. Освен това Линукс машината трябва да се добави към ADS областта чрез помощната програма net (man net).

В секцията [homes] се описват параметрите на достъп до домашните директории на потребителите. Първият ред указва коментара, който се появява срещу директорията. Вторият ред задава дали дадения споделен ресурс ще се вижда в списъка с достъпни споделени ресурси. Третия ред задава режим на достъп до ресурса, а последния ред – позволенията с които ще се създават файловете.
В секцията [tmp] параметъра public показва, че за достъп до този ресурс не се изисква парола, а параметъра path, задава пътя до директорията, която се споделя.
След промяна на файла е добре той да се провери за валидност. Това става с командата testparm. Ако тя не върне грешка, конфигурацията е валидна. За да влезе новата конфигурация в сила трябва да рестартирате сървъра с командите:
#/etc/rc.d/rc.samba restart – за Slackware
#/etc/rc.d/init.d/smb stop
/etc/rc.d/init.d/smb start – за Red Hat

Ето още един пример за споделяне на директория:
[Music]
comment = Music folder
path = /mnt/storage/Music
guest only = Yes
guest ok = Yes
Параметъра guesk ok е синоним на public. Параметъра guest only задава достъпа до споделения ресурс, като в случая никой освен guest потребителите нямат достъп до ресурса.
Това е най-основната конфигурация за осигуряване достъп на Windows машини до Линукс чрез SMB протокола.
Конфигурирането на smb.conf може да се улесни ако се използва помощното приложение SWAT (Samba Web Administration Tool). SWAT работи като сървър и очаква връзка към порт 901. За връзка се използва обикновен браузър. SWAT позволява цялостно конфигуриране на Samba сървъра чрез удобен и лесен WEB интерфейс.
Освен SWAT може да се използва и специален плъгин за KDE наречен ksambaplugin (http://ksambakdeplugin.sourceforge.net/), който се интегрира в Контролния панел на KDE и в неговите менюта.
Когато искате да осъществите достъп до Windows споделени ресурси от Линукс трябва да използвате програмата smbclient, която се разпространява с пакета Samba. Тя ви предоставя FTP подобен конзолен интерфейс. Достъпа до ресурса \\server514\shared се осъществява така:
#/usr/sbin/smbclient \\\\server514\\shared – ако ресурса е споделен без парола
#/usr/sbin/smbclient \\\\server514\\shared mypasswd – ако ресурса е споделен с парола
След като попаднете в обвивката на командата може да използвате help, за да видите възможните команди. Друга начин за достъп до споделени ресурси е възможността на тяхното прикачане към локалната файлова система.
Това става с командата smbmount:
#smbmount \\\\server514\\shared /mnt/smb
#smbumount /mnt/smb
Друг начин за монтиране на споделен SMB ресурс е чрез командата mount:
#mount -t smbfs \\\\server514\\shared /mnt/smb
#umount /mnt/smb
За да е възможно монтирането на SMB споделена директория към локалната файлова система е необходимо ядрото да е компилирано с поддръжка на SMBFS. Това може да проветите с командата:
#cat /proc/filesystems
Ако списъка, който тя връща включва smbfs, то ядрото ви е компилирано с тази поддръжка. В противен случай ще е необходимо за използвате програмата smbclient за достъп до споделените ресурси.
Освен файлове, чрез Samba може да се споделят и принтери.

Ето примерна конфигурация за споделяне на принтер от Линукс:
[global]
printing = bsd
load printers = yes
printcap name = /etc/printcap
max print jobs = 100
[printers]
comment = All printers
printable = yes
path = /var/spool/samba
browseable = no
guest ok = yes
public = yes

read only = yes
writable = no

Значението на отделните параметри е следното:
[global]
printing = bsd – казва на Samba да използва BSD стил на принтиране.
В новите дистрибуции се предпочита използването на CUPS.
load printers = yes – при използването на този параметър се избягва дефинирането на секция за всеки отделен принтер. Споделят се всички принтери описани в /etc/printcap.
max print jobs = 100 – задава максималния брой едновременни задачи.
printcap name = /etc/printcap – задава пътя на файла където са описани достъпните принтери. Ако се използва CUPS този файла трябва да има права за писане.
[printers]
printable = yes – ако този параметър не е със стойност yes, smbd ще откаже да се стартира.
path = /var/spool/samba – трябва да сочи към директория където Samba да съхранява пристигащите файлове. Тази директория трябва да е различна от тази зададена на системата за принтиране на Линукс.
browseable = no
Ако вашата дистрибуция използва CUPS (Common UNIX Print System), то параметрите printing и printcap name трябва да са със стойности cups. За да може да използвате CUPS, то Samba трябва да е компилирана с такава поддръжка.
Сега ще разгледаме случая на принтиране от Линукс машина на споделен принтер работещ на Windows. За да може да правите това трябва да отговаряте на следните условия:
Вие трябва да имате правилни записи във файла /etc/printcap (те трябва да съответстват на локалната структура на директориите за буферна директория и т.н.)
Трябва да имате скрипт /usr/bin/smbprint. Той се доставя заедно с изходните кодове на Samba, но не със всички двоични дистрибутиви на Samba (например в пакета на Slackware, smbprint не присъства). В по-новите дистрибутиви на Samba е заменен с smbspool.
Ако вие искате да преобразувате ASCII файлове в Postscript вие трябва да имате програмата nenscript, или неин еквивалент. nenscript това е конвертор на Postscript и той обикновено се разполага в директорията /usr/bin.
Нека разгледаме един примерен /etc/printcap, като файла е за принтер HP 5MP на сървър Windows 2000. Използват се следните полета на файла /etc/printcap:
cm – коментар
lp -име на устройството, отворено за въвеждане
sd - spool директория на принтера (на локалната машина)
af - файл за отчет за използването на принтера
mx - максималния размер на файла (нула - без ограничения)
if - име на входния филтър (скрипта)

```
# /etc/printcap
lp:\
:cm=HP 5MP Postscript Printer on Server514:\
:lp=/dev/lp1:\
:sd=/var/spool/lpd/lp:\
:af=/var/spool/lpd/lp/acct:\
:mx#0:\
:if=/usr/bin/smbprint:
```

Убедете се, че буферните директории и директорията използвана за отчет за ползването, съществуват и са с права за запис. Убедете се, че реда 'if' съдържа правилния път към скрипта smbprint (даден по-долу) и убедете се, че записите сочат към правилното устройство за въвеждане (специалния файл /dev).

След това може да използвате програмата smbspool за печат на споделени принтери през SMB протокола. Формата за използване на програмата е:
smbspool {job} {user} {title} {copies} {options} [filename] , където:
job – съдържа идентификационния номер на задачата и за момента не се използва от smbspool.
user – съдържа името на потребителя и за момента не се използва от smbspool.
title – съдържа името на задачата, което се интерпретира като име на отдалечения файл
copies – брой копия на принтирания файл.
options – задава различни опции. Не се използва от smbspool.
filename – задава името на файла, който ще се изпрати на отдалечения принтер. Този файл трябва да се създаден чрез „Print as file“ от съответната програма и трябва да бъде в Postscript формат. Повечето програми на KDE може да печат във файл с Postscript формат.

Конфигуриране на системата. Стартиране и конфигурация на Apache Web сървър

Apache Web Server е най-често използвания Web сървър в Интернет. Според Netcraft той се използва в над 60% от Интернет сайтовете. Причината за неговата популярност е че той е безплатен и много стабилен. Друго голямо предимство на Apache, че той е с отворен код, което позволява лесно модифициране на кода или писане на допълнителни модули. Apache поддържа следните основни функции:

Сигурност
Apache притежава набор от защити, които го правят труден за осъществяване на атаки. Този набор включва идентификационни механизми и Secure Socket Layer (SSL). Тъй като Apache поддържа широк списък от сървъри за бази данни като MySQL и Oracle, то те може да се използват за съхраняване на потребителите имащи достъп до Web сървъра.
Можете да добавите поддръжка на SSL на сървъра чрез модула mod_ssl. SSL използва цифрови подписи за криптиране на данните пренасяни през Интернет.

Поддръжка на HTTP/1.1
Apache поддържа най-новата спецификация на HTTP протокола, а именно 1.1. Това означава, че Apache поддържа виртуални хостове, постоянни връзки (persistent connection), кеширане на ресурсите, качване на файлове от страна на клиента и др.

Поддръжка на множество езици
Apache поддържа множество езици за създаване на Web съдържание. Езиците които той поддържа са: PHP (Hypertext Preprocessor), ASP (Active Server Pages), JSP (Java Server Pages), Perl, CGI (Common Gateway Interface), SSI (Server Side Includes) и много други.

Кеширане на информацията
Чрез модула mod_proxy, Apache може да кешира страниците, като при повторно заявяване тя се зарежда не от Интернет, а от локалния кеш. Това може да спести трафик и време за зареждане.

Dynamic Shared Objects
Apache поддържа механизъм за зареждане на модули, които добавят различна функционалност на Web сървъра. Това става без прекомпилиране на Web сървъра.
Поддръжка на Windows
Версия 1.x.x на Apache може да работи успешно и на Windows базирани машини. Тази версия обаче не е оптимизирана за работа под Windows и не трябва да се използва за

реални Web сървъри. След версия 2.x Apache вече може да работи и като реален сървър на Windows базирани машини.

Скаларност
Можете да конфигурирате Apache така, че множество сайтове да работят на един сървър. Това става чрез задаване на виртуални хостове.
След като разбрахме защо Apache е предпочитан Web сървър е време да разберем основите на неговата конфигурация.

Преди да започнем с конфигурацията трябва да проверите дали имате инсталиран Apache. Това може да стане със следната команда:
#whereis apachectl
apachectl: /usr/sbin/apachectl /usr/man/man8/apachectl.8.gz
/usr/share/man/man8/apachectl.8.gz
Ако Вие имате изход подобен на този, то Apache е инсталиран на вашата система.
В противен случай трябва да инсталирате Apache. Това може да стане както от бинарните пакети на вашата дистрибуция или от изходен код. В повечето случаи пакета е компилиран с повечето полезни опции и компилирането от изходен код е безсмислено. Ако все пак желаете да компилирате, то кода може да се свали от http://www.apache.org/dist/httpd/.
След като свалите кода трябва да го декомпресирате:
#tar zxvf apache_1.3.24.tar.gz
След това трябва да добавите група и потребител за работата на Apache:
#groupadd www
#useradd -g www www
След това чрез passwd -l трябва да заключите потребителя www, така че само root да има достъп до него.
След това трябва да се стартира скрипта ./configure , който да създаде файловете необходими за компилиране не сървъра. Този скрипт има множество полезни опции които може да се видят чрез:
#./configure --help
Ето и примерна конфигурация (възможно най-елементарна):
#./configure --prefix=/usr/local/apache --server-uid=www --server-gid=www \
>--htdocsdir=/home/www/htdocs \
>--cgidir=/home/www/cgi-bin \
>--enable-module=most \
>--enable-shared=max
След това трябва последователно да се изпълнят командите:
#make
#make install
След като е инсталиран, може да контролирате Apache с командата apachectl.
Тя има следните опции:
start – стартира сървъра.
stop – спира сървъра.
restart – рестартира сървъра при промяна на конфигурацията.
fullstatus – предоставя пълна информация за сървъра в момента.
Изисква инсталиран браузър lynx и активиран модул mod_status.
status – предоставя кратка информация за сървъра в момента.
Изисква инсталиран браузър lynx и активиран модул mod_status.
graceful – елегантно рестартиране на сървъра. Ако не е стартиран, то чрез тази команда той се стартира.
configtest – проверява валидността на конфигурацията.
help – кратка помощна информация.

Сега е време за кратък преглед на основните конфигурационни директиви.
Те се намират във файла /etc/apache/httpd.conf (в зависимост от това как е компилиран файла може да се намира и в друга директория). Той е разделен на няколко секции. Първата от тях е Global Environment. Директивите в тази секция са глобални и се отнасят за целия Web сървър и различните виртуални хостове.
Тя включва следните основни директиви:
ServerType – можете да настроите сървъра да работи в един от двата режима: standalone или inetd. inetd може да се използва само за UNIX/Linux базирани системи. В първия случай сървъра работи като самостоятелен сървър, а във втория – като услуга стартираща се от inet демона.
ServerRoot – тази директива задава основната директория в която е инсталиран сървъра. В нашия пример това е /usr/local/apache
StartServers – указва колко сървъра да бъдат стартирани едновременно.
Добре е тази директива да не се променя. За Windows версията на Apache тази директива няма никакво значение и не присъства в конфигурационния файл.
MaxClients – задава максималния брой на едновременно свързани клиента.
По подразбиране стойността е 150, което може да не е достатъчно за натоварени сървъри.
Listen – това е важна директива, която може сериозно да повлияе на бързината и сигурността на сървъра. Чрез нея се задава порта и IP адреса на който ще „слуша“ сървъра за заявени връзки. По подразбиране порта е 80 за локалната машина. Можете да задавате или само порт (Listen 5000) или комбинация от порт и IP адрес (Listen 192.168.7.102:80). Можете да задавате повече от една комбинация от IP адрес и порт, като подреждате Listen директивите последователно една след друга. Ако зададените IP адреси не са достъпни за сървъра, той ще откаже да се стартира.
LoadModule – също много важна директива, чрез която може да зареждате различни модули, като по този начин разширявате функциите поддържани от сървъра. Например : LoadModule mime_module libexec/mod_mime.so Тук libexec е директорията в която са инсталирани модулите на сървъра.
AddModule – след като сте добавили модула в LoadModule трябва да го направите и в тази тук. Например: AddModule mod_mime.c След това модула е активен и достъпен за използване.
Това са по-важните директиви в Global Environment секцията.
Следващата секция, която ще разгледаме е Main Server Configuration. Тази секция се отнася за основния Web сървър. Отделните виртуални хостове може да имат конфигурация различна от тази, ако дадена директива не е зададена изрично в конфигурацията на виртуалния хост, то се използва стойността й зададена тук. Main Server Configuration секцията съдържа следните по-важни директиви:
Port – задава порта на който „слуша“ сървъра. По подразбиране стойността е 80.
User – задава името на потребителя с които права работи сървъра.
Group – задава името на групата с които права работи сървъра.
ServerAdmin – задава електронния адрес, който се появява на страниците генерирани от сървъра при грешки или съобщения.
ServerName – задава името под което работи сървъра. По подразбиране се използва localhost. Той обаче е подходящ само за тестови цели. Зададеното име трябва да е регистрирано DNS име или поне да бъде описано във файла /etc/hosts.
DocumentRoot – важна директива задаваща къде сървъра ще търси Web документите. За нашия пример това е /home/www/htdocs .
DefaulType – чрез тази директива се задава на сървъра как да изпрати към клиента документ от тип, които не му е познат. По поддразбиране стойността е text/plain.
ErrorLog – тази директива указва пътя към журналния файл в който се пазят грешките на сървъра.
LogLevel – задава степента на подробност на грешките. По подразбиране стойността в warn, което означава, че към журналния файл се изпращат грешки от тип Warning или

по-сериозни. Възможните стойности са debug, info, notice, warn, error, crit, alert, emerg. Най-високо ниво на съобщаване на грешки е debug, а най-ниско - emerg
ScriptAlias – задава псевдоним на директорията, в която ще се изпълняват CGI или други скриптове. По подразбиране стойността е зададената при конфигурацията на сървъра, а за конкретния пример тя е: ScriptAlias /cgi-bin/ „/home/www/cgi-bin/“. Когато един клиент напише в адресното поле на браузъра с : http://www.localhost/cgi-bin/ , то той ще осъществи достъп до директорията /home/www/cgi-bin/
BrowserMatch – чрез тази директива се търси съвпадение в името на браузъра използван от клиента и зададеното в директивата. Ако се намери съвпадение, то се може да се укаже на сървъра да промени своето поведение така, че да избегне определени бъгове в даден браузър. Например: BrowserMatch „Mozilla/2“ nokeepalive.

Друга много важна директива е . Тук може да се задават различни опции за дадената директория. Задължително име поне две такива директиви – една за DocumentRoot и една за директорията зададена чрез ScriptAlias. В тази директива може да се задават различни поддирективи, които имат влияние само за дадената директория. Ето пример за използването на директивата:

Options Indexes FollowSymLinks
DirectoryIndex index.php index.html index.htm
AllowOverride All
Order allow,deny
Allow from all

Значението на различните поддирективи е следното:
Options – задава различни опции на директорията. Indexes означава, че ако няма някои от документите по подразбиране (зададени в DirectoryIndex), то ще се показва съдържанието на директорията. FollowSymLinks е опция указваща на сървъра да следва символичните линкове. Ако се зададе и ExecCGI, то в дадената директория ще може да се изпълняват CGI скриптове.
DirectoryIndex – указва реда в който ще се търсят документите по подразбиране в дадената директория. В конкретния случай сървъра първо ще потърси файла с име index.php, ако не го намери, ще потърси index.html и т.н.
AllowOverride – ако стойността е None, то всички директиви зададени във файла .htaccess се игнорират. В този случай стойността е All, което позволява на този файл да предефинира опциите зададени в директивата . Името на този файл може да се променя чрез директивата AccessFileName.
Order – указва в какъв ред ще се извършва проверката за това дали даден хост и дадено потребителско име има достъп до сървъра. Следващата директива Allow from с опция all указва , че всички хостове ще имат достъп до дадената директория.

Понякога се налага да се асоциира дадено разширение на файла към определен MIME тип. Например, за да знае сървъра какво да прави с PHP файловете (освен зареждането на модулите) е необходимо файловете с разширение .php да се асоциират към MIME типа application/x-httpd-php. Това може да стане с директивата AddType:
AddType application/x-httpd-php .php
Ето и обяснение на други директиви в :
AddEncoding – чрез тази директива можете да свържете дадено файлово разширение към даден тип.
AddLanguage – свързва файлово разширение към определен език.
AddHandler – свързва файлово разширение към определен изпълнител (handler).

Третата секция в httpd.conf е Virtual Hosts. Чрез правилното конфигуриране на тази секция Вие ще можете да пуснете няколко Web сайта, които се хостват на един единствен Web сървър. Тези сайтове може да бъдат част от основния домейн или да не

бъдат. Всеки отделен виртуален хост трябва да се поставя в своя собствена секция. Вътре в нея се дефинират отделните директиви специфични за дадения виртуален хост. Виртуалните хостове могат да бъдат два типа IP базирани (IP-Based) и Именно базирани (Name-Based). Разликата между тях е, че при първия тип е необходимо за всеки виртуален сървър да се конфигурира отделен IP адрес. За разлика от него, при втория тип можете да конфигурирате множество виртуални сървъри на един единствен IP адрес. Важно е да се знае, че втория тип виртуални хостове не могат да работят в версия на HTTP по-ниска от 1.1.
Ето примерна конфигурация на IP-Based виртуален хост:

```
ServerName steve.ulala.com
DocumentRoot /home/www/public_html/steve

ServerName pola.ulala.com
DocumentRoot /home/www/public_html/pola
```

За да работи това е необходимо да конфигурирате вашия мрежови интерфейс да има повече от едно IP едновременно. Това може да стане така:

```
#ifconfig eth0:0 172.17.68.222
#ifconfig eth0:1 172.17.68.223
```

Разбира се, трябва да имате и валидни записи в DNS сървърите които да свързват steve.ulala.com и pola.ulala.com с 172.17.68.222 и 172.17.68.223. Това най-лесно може да стана като добавите следните редове в /etc/hosts:

```
172.17.68.222 steve.ulala.com
172.17.68.223 pola.ulala.com
```

Конфигурацията на Name-Based виртуални хостове е подобна. Преди секциите с отделните виртуални хостове, трябва да промените директивата:

```
NameVirtualHost 172.17.68.220

ServerName steve.ulala.com
DocumentRoot /home/www/public_html/steve

ServerName pola.ulala.com
DocumentRoot /home/www/public_html/pola
```

Съществена разлика има в редовете добавени в /etc/hosts:

```
172.17.68.220 steve.ulala.com
172.17.68.220 pola.ulala.com
```

В самите секции на различните виртуални хостове, може да слагате различни директиви, които имат по-висок приоритет от тези в секцията описваща основната конфигурация.

Apache поддържа и т.нар. Dynamic Virtual Hosting. Това позволява на администратора създаде специфични темплейти за DocumentRoot и ScriptAlias . На тази база и на базата на клиента, Web сървъра създава сам необходимите директории и връзки. Това е подходящо да се използва в сайтове поддържащи множество виртуални хостове, чието администриране на ръка би било трудоемка задача. Dynamic Virtual Hosting може да направите както с IP-Based виртуални хостове, така и с Name-Based такива.
Повечето Web сървъри се използват за създаване на динамично съдържание, чрез определени програмни езици. Най-старата технология за създаване на динамично съдържание е използването на CGI. Чрез CGI можете да напишете програма на който език пожелаете и да визуализирате резултата на Web браузъра.
За да направите това, трябва да са спазени някои изкисвания, които ще бъдат разгледани по долу. Като първо, трябва да имате директория в която е позволено изпълнението на CGI скриптове. Ето пример за такава директория:

Options +ExecCGI

След това трябва да присвоите дадени разширения към CGI съдържанието:
AddHandler cgi-script cgi pl
След това трябва да напишете програма, чийто изход е форматиран по определен начин:
MIME заглавие – първото което трябва да връща вашата програма е заглавие от определен MIME тип, а именно text/html за HTML страници. Заглавието задължително завършва с два празни нови реда.
Изход в HTML формат – след заглавието трябва да следва валиден HTML код.

Ето и пример за такава програма писана на езика Perl:
#!/usr/bin/perl
#mycgitest.cgi
print „Content-type: text/html\n\n“;
print „First CGI program“;
За да се изпълни този файл трябва да са спазени следните условия:
да се намира в директорията /home/www/cgi-bin (която трябва да е собственост на потребителя с правата на който работи Web сървъра)
файла да има права за изпълнение от потребителя с които права работи сървъра
да има разширение .pl или .cgi

Накрая ще поговорим малко за сигурността на сървъра. На първо място е да укажете на сървъра да работи с права различни от тези на root потребителя.
Това може да стане чрез директивите User и Group. След това трябва внимателно да обмислите кои директории може да показват соите файлове при липса на index.html. Може да забраните показването като премахнете от директивата Options опцията Indexes. Винаги проверявайте CGI скриптовете които пишете за грешки и/или пропуски, тъй като CGI скриптовете са едни от най-големите дупки в сигурността на Web сървъра. За да намалите вероятността от проблеми с CGI трябва да използвате обща директория зададена чрез директивата ScriptAlias. Позволяването да се стартират CGI скриптове от всички директории е много лоша идея. Други проблеми носят SSI (Server Side Includes): значително по-голямо натоварване на сървъра, тъй като в този случай той проверява цялата страница за

SSI тагове
използването на 'exec cmd' в SSI, чрез което може да извикате произволен скрипт или програма. Тъй като тя се изпълнява с правата на потребителя с които работи сървъра трябва внимателно да следите какво се пуска с тови SSI таг.
Чрез файловете .htaccess можете да задавате различни права и достъп за всяка директория. Ако вашия сървър използва AllowOverride All, то потребителя може да запише в своята директория файл с такова име, който да компрометира сигурността на сървъра.

Стартирането на някои услуги на машина с работещ Web сървър може да даде повече права отколкото сте желали. Пример за това е използването на SAMBA сървър на машина с работещ Web сървър, туй като в определени случаи можете да предоставите достъп на целия свят до ресурсите на вътрешната мрежа. Стартирането на telnet и други подобни услуги не само ще намали сигурността на вашия Web сървър, но и на цялата машина като цяло.
По принцип спазването на девиза „колкото по-малко, толкова по-добре“ е желателно когато говорим за стартирани услуги.