МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІГІВСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ
КАФЕДРА ІНФОРМАЦІЙНИХ ТА КОМП'ЮТЕРНИХ СИСТЕМ

АДМІНІСТРУВАННЯ ОПЕРАЦІЙНИХ СИСТЕМ СІМЕЙСТВА UNIX

Методичні вказівки
до лабораторних робіт з дисципліни
"Системне програмування і адміністрування
операційних систем"
для студентів напряму підготовки
6.050102 "Комп'ютерна інженерія"

Затверджено
на засіданні кафедри
інформаційних та комп'ютерних систем

Протокол № 10 від «18» квітня 2013 р.

Чернігів ЧДТУ 2013

Адміністрування операційних систем сімейства UNIX. Методичні вказівки до лабораторних робіт з дисципіни “Системне програмування та адміністрування операційних систем” для студентів напряму підготовки 6.050102 “Комп’ютерна інженерія” / Укл. Є. В. Риндич, С.С. Стасюк, Д.І. Мельниченко. – Чернігів: ЧДТУ, 2013. – 50 с.

Укладачі: Риндич Євген Володимирович, кандидат технічних наук, доцент кафедри інформаційних та комп’ютерних систем

Стасюк Сергій Станіславович, старший викладач кафедри інформаційних та комп’ютерних систем

Мельниченко Дмитро Іванович, старший викладач кафедри інформаційних та комп’ютерних систем

Відповідальний за випуск: Казимир Володимир Вікторович, завідувач кафедри інформаційних та комп’ютерних систем, доктор технічних наук, професор

Рецензент: Нікітенко Євгеній Васильович, кандидат фізико-математичних наук, доцент кафедри інформаційних та комп’ютерних систем Чернігівського державного технологічного університету

ЗМІСТ

ВСТУП

Операційні системи сімейства UNIX давно зарекомендували себе як надійні та гнучкі в експлуатації. В той самий час, підтримка цих систем потребує ґрунтовних знань з архітектури операційної системи(ОС), навичок адміністрування і системного програмування, що не під силу рядовому користувачу. Адміністратор ОС або систем, в загальному розумінні – це на сьогодні доволі поширена професія.

Знання операційної системи, її організації та структури, передусім, необхідно адміністратору, який відповідає за налаштування та підтримку. Задачі адміністратора багаточисельні – від реєстрації користувачів та конфігурації мережі, до резервного копіювання системи та тонкого налаштування і оптимізації її роботи.

ОС сімейства UNIX – вільно поширювані Linux, FreeBSD і подібні їм довели надійність вільно поширюваного ПЗ з відкритими текстами. Вони стабільні і повнофункціональні, як і їх комерційні аналоги, при цьому — безкоштовні і відкриті. Ще одна перевага безкоштовних і відкритих систем - на їх прикладі значно легше розуміти внутрішню структуру і роботу системи самостійно.

Дані методичні вказівки складаються із 9 лабораторних робіт. Курс призначений для ознайомлення з особливостями адміністрування ОС на прикладі ОС Linux, виділити круг задач адміністрування ОС, навчитися втілювати рішення, які в майбутньому дозволять легко розширювати і ускладнювати структуру системи, вивчити комплекс основних прийомів роботи з системним програмним забезпеченням.

Матеріал лабораторних робіт побудований таким чином, щоб надати можливість студентам з точки зору системного програміста розглянути задачі з адміністрування систем сімейства UNIX.

Методичні вказівки не повинні використовуватися як довідник команд ОС Linux. Оскільки система постійно розвивається і оновлюється, тому приклади можуть бути видозмінені з часом, або замінені на новіші аналоги.

1 Лабораторна робота №1. Встановлення ОС GNU/Linux на персональний комп’ютер

1.1 Мета роботи

На прикладі ОС Linux вивчити процедуру інсталяції ОС сімейства UNIX на персональні комп’ютери. Ознайомитися з задачами адміністрування файлових систем, а також з сучасними файловими системами в Linux.

1.2 Теоретичні відомості

Файлові системи призначені для зберігання, доступу і управління доступом до даних(файлам) в структурованому виді. Різноманіття файлових систем в Linux обумовлене тим, що кожну з них обслуговує спеціальний драйвер в ядрі. Для користувача ж інтерфейс з будь-якою файловою системою виглядає однаково завдяки концепції VFS (Virtual File System).

В дистрибутивах Linux підтримуються наступні файлові системи: Minix, Ext2, Ext3 (тільки для ядра 2.4.x), ReiserFS (3.5 – для ядра 2.2.x, 3.5 и 3.6 для ядра 2.4.x), XFS (тільки для ядра 2.4.x), JFS (тільки для ядра 2.4.x), VFAT, NTFS (тільки читання), ISO9660, UDF, інші (менш розповсюджені).

1.2.1 Файлова система ext3

Файлова система Ext3, розроблена Dr. Stephen Tweedie, сформована на структурах свого прототипу – ext2; фактично, ext3 дуже схожа на ext2, але вона підтримує журналювання (journaling). Ведення журналу транзакцій значно підвищує стійкість до збоїв. Дисковий формат ext2 і ext3 ідентичний; з цього виходить, що при необхідності ext3 можна монтувати як ext2 без яких небудь проблем. Системна утиліта tune2fs дозволяє конвертувати наявну ext2 в journaling ext3.

Ext3 пропонує на вибір три режими журналювання при монтуванні файлової системи: data=writeback, data=ordered и data=journal.

Спочатку ext3 розроблялася для журналювання як своїх структур, так і даних користувача(metadata & data). У цьому режимі(називається "data=journal" mode) відбувається збереження усіх змін в системі. У режимі data=writeback, файлова система ext3 не виконує якого-небудь журналювання призначених для користувача даних і не захистить від руйнування дані в оновлюваних файлах у разі несподіваного перезавантаження. У режимі data=ordered файлова система ext3 "офіційно" журналює тільки метадані, але логічно блоки metadata і data групуються в єдиний модуль, званий transaction. Перед записом нових метаданих на диск, пов'язані з ними блоки файлів записуються першими.

1.2.2 Файлова система reiserfs

ReiserFS – нова файлова система з журналюванням, розроблена Hans Reiser і його командою Namesys. Hans припускає, що краща файлова система

та, яка формує єдине загальнодоступне середовище, або namespace. У такому середовищі додатки можуть взаємодіяти гнучкіше і ефективно.

ReiserFS використовує спеціально оптимізовані b*-balanced tree(одне на ФС) для організації усіх даних файлової системи. Це дає значне збільшення продуктивності, а також знімає ряд штучних обмежень на розміщення файлової системи. ReiserFS, подібно до більшості інших ФС нового покоління, динамічно виділяє inodes замість їх статичного розподілу, вироблюваного при створенні "традиційної" файлової системи. Це дає велику гнучкість і оптимальність у формуванні простору зберігання.

ReiserFS також має ряд особливостей, націлених спеціально на поліпшення роботи з маленькими файлами. ReiserFS не пов'язана обмеженням в розподілі пам'яті для файлу цілим числом 1,2 або 4 KB блоків. З потреби для файлу може бути виділений точний розмір. ReiserFS також включає деякі види спеціальної оптимізації файлових "хвостів"(notail) для зберігання кінцевих частин файлів, менших, ніж розмір блоку. Для збільшення швидкості, ReiserFS здатна зберігати вміст файлів безпосередньо усередині b*tree, а не у вигляді покажчика на дисковий блок (в ext2 є поняття fastlink, коли вміст "м'якого" посилання до 60 байт зберігається в inode).

1.2.3 Файлова система XFS

XFS – SGI's free, 64-bit – високопродуктивна файлова система, що журналюється, для Linux. Файлова система XFS спочатку була розроблена Silicon Graphics, Inc. на початку 90-х років. Зовсім нещодавно XFS стала доступна і для Linux. Поява підтримки XFS в Linux подія значима.

Тестування показало, що XFS – дуже швидка файлова система. XFS постійний лідер в тестах з маніпуляціями великими файлами. Приємна особливість XFS – вона не створює(втім, як і ReiserFS) зайву дискову активність. XFS намагається кешірувати якомога більше даних і "основою" для скидання на диск являється заповнення пам'яті, а не інтервал часу. Коли відбувається скидання даних на диск, це не робить помітного впливу на інші IO операції. Як протилежність – в ext3(режим "data=ordered") періодичне скидання даних на диск породжує проблеми з інтерактивністю та, при високій інтенсивності операцій IO, навіть disk thrashing. Перевага майже у всьому, крім маленької деталі – видалення старих файлів, "незручної" для XFS операції (ReiserFS та ext3 видаляють файли швидше, ніж XFS).

1.2.4 Файлова система Tmpfs

Tmpfs, ще відома як virtual memory(VM) filesystem. Tmpfs – ймовірно краща RAMDISK-like система. Подібно ramdisk, tmpfs використовує оперативну пам'ять, але, окрім цього, може використати swap devices. Тоді як традиційний ramdisk цей блоковий пристрій і перед його використанням необхідно відформатувати розділ командою mkfs з опціями, то файлова система tmpfs – пристрій не блоковий, готовий до використання відразу після монтування.

Tmpfs може використати і RAM, і swap, і тому відома як "virtual memory filesystem". Файлова система tmpfs просить сторінки у підсистеми VM для зберігання файлів. При цьому сама tmpfs не знає, чи знаходяться ці сторінки в swap або в RAM, це – "проблема" підсистеми VM.

На відміну від більшості "нормальних" файлових систем(наприклад, ext3, ext2, XFS, ReiserFS) tmpfs не є "надбудовою" над блоковим пристроєм. Оскільки tmpfs безпосередньо "вбудована" в VM, її можна монтувати відразу при створенні, командою:

```
sh# mount tmpfs /mnt/tmpfs -t tmpfs
```

Після виконання команди ви отримаєте нову tmpfs, змонтовану в /mnt/tmpfs і готову до використання. /mnt/tmpfs спочатку має дуже маленький розмір, але, у міру копіювання і створення файлів драйвер tmpfs просить у VM додаткову пам'ять, динамічно збільшуючи місткість. Інша перевага tmpfs – її "блискуча" швидкість. Оскільки файлова система tmpfs постійно завантажена в оперативну пам'ять, операції запису – читання відбуваються майже миттєво. "Безінерційність" може вважатися як плюсом, так і мінусом. Як можна здогадатися, дані в tmpfs після перезавантаження будуть загублені (операційна пам'ять енегрозалежна по своїй природі).

1.2.5 Команди роботи з файловими системами

`badblocks [опції] пристрій`

– використовується для пошуку поганих блоків на пристрої.

`/etc/init.d/autofs start|stop|reload`

– сценарій управління автоматичним монтуванням. В системі Linux `autofs`

– керується роботою демонів automount(8).

`dumpe2fs [опції] пристрій`

– вивід інформації про суперблок та групу блоків файлової системи на пристрої.

`e2image [ -r ] пристрій image-file`

– зберігання критичних даних файлової системи у файл *image-file*.

`fsck [опції] [пристрій | каталог]`

– перевірка файлової системи на помилки (filesystem check).

`mkfs [опції] пристрій`

– створення нової файлової системи (make filesystem)

`mkswap [опції] пристрій [розмір]`

– створення області swap в Linux

```
mount [ключи] [-t vfstype] [-o options] пристрій каталог
umount пристрій | каталог
```

– монтування та розмотування файлової системи.

`tune2fs device`

– конвертування файлової системи в ext2.

1.2.6 Система управління логічними томами (Logical Volume Manager)

LVM або менеджер логічних томів – система управління дисковим простором, що дозволяє абстрагуватися від управління фізичними пристроями. Вона дозволяє ефективно використати і легко управляти дисковим простором. У логічних томів, створених за допомогою LVM, можна легко змінити розмір, а їх назви можуть нести деяке смислове навантаження, на відміну від традиційних "/dev/hda" або "/dev/sda".

Система управління логичесикми томами особливо корисна в роботі з серверами, оскільки забезпечує масштабованість. Вона спрощує планування дискового простору і запобігає проблемам, що виникають при несподівано швидкому зростанні зайнятого місця в розділах.

Реалізація менеджерів логічних томів існує практично в усіх UNIX-подібних операційних системах. Часто вони сильно відрізняються в реалізації, але усі вони засновані на однаковій ідеї. Одна з поширених реалізацій була виконана Open Software Foundation.

Фізичний том – зазвичай, це розділ жорсткого диска(чи увесь жорсткий диск) або пристрій, який працює аналогічно розділу, наприклад облаштування RAID(у тому числі і програмний).

Логічний том – один або більше фізичних томів утворюють логічний том. Логічний том LVM ідеологічно подібний до розділу жорсткого диска не-LVM системи. Логічний том може містити файлову систему, наприклад /home або /usr.

Група томів – один і більше логічних томів утворюють групу томів. Група томів LVM ідеологічно подібна до жорсткого диска в не-LVM системі. Група томів формує з безлічі логічних томів адміністративну одиницю.

Кожен фізичний том ділиться на частини, які називаються фізичними екстентами (Physical Extents, PE). Розмір фізичних екстентів може варіюватися, але однаковий в межах групи томів.

В межах фізичного тому кожен фізичний екстент має унікальний номер. Фізичний екстент – мінімальний блок простору, який може бути адресований системою LVM на фізичному сховищі. Аналогічно, кожен логічний том складається з мінімальних адресованих блоків, що носять назву логічних екстентів (Logical Extents, LE). В межах групи томів розмір логічного екстента дорівнює розміру фізичного. Очевидно, що розмір логічних екстентів однаковий для усіх логічних томів групи. Існує взаємно-однозначне відображення LE на PE. При кожному доступі до сховища використовується ідентифікатор LE для реальної роботи з фізичним пристроєм.

Тепер виникає питання, де зберігаються усі ці мета-дані про логічні томи і групи томів. Як відомо, в не-LVM системах, дані про розділи зберігаються в таблиці розділів. У LVM системі існує область дескрипторів групи томів (Volume Group Descriptor Area, VGDA), працююча аналогічно таблиці розділів. Вона зберігається на початку кожного фізичного тому, керованого за допомогою LVM.

1.3 Хід роботи

1. Створити віртуальну машину. У віртуальній машині необхідно створити два жорстких диска.
2. Встановити ОС.
3. Розбити другий жорсткий диск згідно з варіантами завдань, представленими в таблиці 1.1.

Варіанти завдань для лабораторної роботи вибираємо за останньою цифрою залікової книжки, SWAP розділ обов'язково повинен бути підключений, як і інші розділи. Кожному варіанту необхідно створити 3 розділи.

Таблиця 1.1 – Варіанти завдань

Номер варіанту	Завдання
0	I — ext2 50MB II — VFAT 100MB III — SWAP, залишок
1	I — ext4dev 70MB II — ntfs 200MB III — SWAP, залишок
2	I — ext3 90MB II — reiserfs 200MB III — SWAP, залишок
3	I — SWAP 250MB II — SWAP 100MB III — ext2 100MB
4	I — SWAP 100MB II — reiserfs 100MB III — SWAP, залишок
5	I — minix 200MB II — reiserfs 100MB III — SWAP, залишок
6	I — cramfs 100MB II — reiserfs 120MB III — SWAP, залишок
7	I — bfs 120MB II — reiserfs 250MB III — SWAP, залишок
8	I — bfs 110MB II — VFAT 00MB III — SWAP, залишок
9	I — cramfs 3 00MB II — ntfs 100MB III — SWAP, залишок

1.3.1 Створення віртуальної машини і жорстких дисків

Для виконання циклу лабораторних робіт пропонуємо вам встановити віртуальну машину. Це збереже вас від можливої неправильної конфігурації системи чи випадкового видалення інформації на вашому жорсткому диску.

Для цього пропонуємо встановити програму VirtualBox. VirtualBox можна встановити на будь-яку операційну систему. Завантажити її можна з офіційного сайту http://www.virtualbox.org/.

Після створення машини необхідно додати ще один жорсткий диск. Переходимо в налаштування машини (Settings), вкладка носії (Storage). Додаємо ще один контролер дисків (наприклад, SATA Contoller) та створюємо новий диск. Необхідно створити диск розміром приблизно 500Мб.

Для зручності роботи встановіть в налаштуваннях машини на вкладці Мережа (Network) тип підключення (Attached to) Мережевий міст (Briged Adapte) і виберіть ім'я адаптера, це дозволить дистанційно працювати із віртуальною машиною через SSH. Можна запускати.

1.3.2 Інсталяція

Рекомендуємо перед інсталяцією прочитати інструкції і рекомендації для вашого конкретного дистрибутиві, адже програма-інстолятор може, наприклад, працювати або з образом CD-ROM, або із звичайною файловою системою, чи при інсталяції необхідно буде щось додаткове.

При першому старті віртуальної машини програма запропоную вам встановити ОС на віртуальну машину. Погоджуйтесь. Якщо цього не відбулося, то VirtualBox повинен повідомити щось на зразок «FATAL: No bootable medium found! ….». Ідемо в Devices та вибираємо наш носій, з якого відбудеться встановлення, а далі в Machine → Reset.

Потрібно вибрати носій встановлення: CD-ROM, образа диску, або Flash-носій. Вибираємо. Сучасні інсталятори не потребують багато зусиль чи знань. В якості дистрибутиві для прикладів був вибраний Ubuntu 11.04 і всі приклади будуть приводитися для нього. Спочатку інсталятор запропонує вам Спробувати чи встановити ОС. Вибираємо мову і встановлення ОС. Ми вибрали English. Install Ubuntu.

Далі нам буде запропоновано встановити оновлення з інтернету — пропускаємо. Даля розмітка дисків. Зупинимося тут більш докладніше. Як ви пом'ятаєте жорстких дисків в нас два. Вибираємо Something else, Forward. Перед нами два не розмічені диски. Будемо працювати поки-що з першим, він повинне бути більший ніж другий. На ньому необхідно створити кореневий розділ «/» і розділ підкачки «Swap».

Другий жорсткий диск залишаємо не розміченим. Тиснемо Install Now. Під час інсталяції нас попросять вибрати годинниковий пояс, вказати час, розкладку та створити користувачів. Далі перезавантаження і ОС встановлена. ОС час-від-часу прохатиме о встановленні оновлення — воно для цих лабораторних робіт буде зайвим, тому можете відключити перевірку оновлення.

1.3.3 Репозиторії

Для роботи з операційною системою необхідно встановлювати програмне забезпечення. Зазвичай, нові програми завантажують з мережи Інтернет із спеціально створених репозиторіїв. Дізнайтеся, як додати до списку репозиторіїв вашої ОС нові і як вибрати її при пошуку нового програмного забезпечення.

1.3.4 Таблиці розділів

Найперший сектор жорсткого диска (сектор 1, доріжка 0, голівка 0) містить так званий головний завантажувальний запис (Master Boot Record). Цей запис займає не весь сектор, а тільки його початкову частину. Сам по собі головний завантажувальний запис є програмою. Ця програма під час початкового завантаження операційної системи з НМД поміщається за адресою 7C00h: 0000h, після чого їй передається керування. Завантажувальний запис продовжує процес завантаження операційної системи.

Наприкінці найпершого сектора жорсткого диска розташовується таблиця розділів диска (Partition Table). Ця таблиця містить чотири елементи, що описують до чотирьох розділів диска. В останніх двох байтах сектора перебуває значення 55AAh. Це ознака таблиці розділів (сигнатура таблиці розділів). Для перегляду та зміни вмісту таблиці розділів НМД використовується програма fdisk.exe. Елемент таблиці розділів диска – це структура розміром 16 байт, яка відповідає тій частині диску, що називається розділом. У структурі розташовується інформація про розташування та розмір розділу в секторах, а також про призначення розділу. Розділи диска бувають активними або неактивними. Активний розділ може використовуватися для завантаження операційної системи. Зауважимо, що диск може містити одночасно кілька активних розділів, які можуть належати різним операційним системам.

В самому першому секторі активного розділу розташований завантажувальний запис (Boot Record), який не слід плутати з головним завантажувальним записом (Master Boot Record). Завантажувальний запис зчитується в оперативну пам'ять головним завантажувальним записом, після чого йому передається керування. Саме завантажувальний запис виконує завантаження операційної системи.

1.3.5 Завантаження операційної системи

Завантаження операційної системи з жорсткого диска – двоступінчастий процес. Спочатку модулі ініціалізації BIOS зчитують головний завантажувальний запис в пам'ять за адресою 7C00h: 0000h і передають йому управління. Головний завантажувальний запис переглядає таблицю розділів і знаходить активний розділ.

Якщо активних розділів декілька, на консоль виводиться повідомлення про необхідність вибору активного розділу для продовження завантаження.

Після того як активний розділ знайдено, головний завантажувальний запис зчитує найперший сектор розділу в оперативну пам'ять. Цей сектор містить завантажувальний запис, якому головний завантажувальний запис і передає управління. Завантажувальний запис активного розділу виконує завантаження операційної системи, що перебуває в активному розділі. Такий двоступінчастий метод завантаження операційної системи необхідний з тієї причини, що спосіб завантаження залежить від самої операційної системи. Тому кожна операційна система має свій власний завантажувач. Фіксованим є тільки розташування завантажувального запису – найперший сектор активного розділу.

Байт зі зсувом 0, є прапором активного розділу і може приймати одне з двох значень – 0 або 80h, відповідно, для неактивного і активного розділів диска. Слово, що має розмір 2 байта і розташоване зі зміщенням 8, містить відносний номер першого сектора розділу. Як він обчислюється? Значення 0 відповідає доріжці 0, головці 0, сектору 1. При збільшенні відносного номера сектора спочатку збільшується номер сектора на доріжці, потім номер головки, і, нарешті, номер доріжки. Для обчислення відносного номера сектора можна використовувати наступну формулу: RelSect = (Cyl * Sect * Head) + (Head * Sect) + (Sect -1) У цій формулі Cyl - номер доріжки, Sect – номер сектора на доріжці, Head – номер головки. Зауваження щодо меж розділів диска: зазвичай розділи починаються з парних номерів доріжок, за винятком самого першого розділу. Цей розділ може починатися з сектора 2 нульової доріжки (головка 0), так як найперший сектор диска зайнятий головною завантажувальної записом. Байт зі зсувом 4 – це код системи, що використовує розділ диска. Для MS-DOS зарезервовані значення 0, 1, 4, 5. Значення 0 відповідає вільному розділу диска.

Якщо код системи в елементі таблиці розділу дорівнює 1 або 4, це означає, що розділ використовується MSDOS в якості первинного розділу (Primary Partition). Цей розділ зазвичай є активним і з нього виконується завантаження операційної системи. В залежності від того, який код системи вказано для первинного розділу (1 або 4) змінюється одна з характеристик логічного диска – розмір елемента таблиці розміщення файлів (FAT). Код 1 використовується для позначення 12-бітової FAT, 4 – для 16-бітової FAT. Значення коду системи, рівне 5, позначає розширений розділ MS-DOS (Extended DOS Partiton). Неважко помітити, що навіть використовуючи всі елементи таблиці розділів для створення логічних дисків, неможливо створити більше чотирьох дисків. У розширеному розділі MS-DOS ви можете створити будь-яку кількість логічних дисків. Програма fdisk.exe дозволяє вам створити один первинний розділ MS-DOS і один розширений розділ. Первинний розділ повинен бути активним, він використовується як диск C: і з нього виконується завантаження операційної системи. Розширений розділ розбивається програмою fdisk.exe на логічні диски D:, E: і т. д. Розширений розділ не може бути активним, отже, неможливо виконати завантаження операційної системи з логічних дисків, розташованих в цьому розділі. Якщо байт коду системи має значення 5, то на початку відповідного розділу

розташовується сектор, що містить таблицю логічних дисків. Фактично ця таблиця є розширенням таблиці розділів диска, розташованої в самому першому секторі фізичного диска.

Таблиця логічних дисків має формат, аналогічний таблиці розділів диска, але містить тільки два елементи. Один з них вказує на перший сектор логічного диска MS-DOS, він має код системи 1 або 4. Другий елемент може мати код системи 5 або 0. Якщо цей код дорівнює 5, то елемент вказує на наступну таблицю логічних дисків. Якщо код системи дорівнює 0, то відповідний елемент не використовується. Зі сказаного вище випливає, що таблиці логічних дисків пов'язані в список, на початок цього списку вказує елемент таблиці розділів диска з кодом системи, рівним 5. Для таблиці логічних дисків є відмінність у використанні полів кордонів логічних дисків. Якщо код системи дорівнює 1 або 4, ці межі обчислюються щодо початку розширеного розділу. Для елемента з кодом системи 5 використовується абсолютна адресація (щодо фізичного початку диска). Наведемо конкретний приклад. Нехай на диску створено два розділи - первинний і розширений. Первинний розділ використовується для завантаження MS-DOS (диск C :), розширений розділ містить логічні диски D:, E:. Нижче на рисунку 1.1 показано розміщення цих розділів.

Рисунок 1.1 – Розміщення розділів диску

GUID Partition Table (GPT) є стандартним форматом розміщення таблиць розділів на фізичному жорсткому диску. На відміну від MBR(Master Boot Record), який починається з виконуваної двійковій програми, яка ідентифікує і завантажує активний розділ, GPT(GUID Partition Table) спирається на розширені можливості EFI(Extensible Firmware Interface) для здійснення цих процесів. Однак MBR присутній на самому початку диска як для захисту, так і з метою сумісності. Власне GPT починається з заголовком таблиці розділів (Partition Table Header). GPT використовує сучасну систему адресації логічних блоків (LBA), забезпечує дублювання – зміст і таблиця розділів записані як на початку, так і в кінці диска. GPT дозволяє створювати розділи диска розміром до 9.4 ЗБ (9.4 × 1021 байт), у той час як MBR може працювати тільки з 2.2 ТБ (2.2 × 1012 байт).

GPT складається з:

1. Перші 512 байт HDD – захист MBR. 64-байти області містять один 0xEE тип запису розділу, визначеного на всьому розмірі диска.
2. Далі 512 байт – первинний заголовок GPT. Містить унікальний диск GUID, розміщення основної таблиці розділів, кількість можливих записів в таблиці розділів, контрольну суму, основну таблицю розділів, місце вторинного (або резервного) GPT заголовку.
3. Наступні 16 Кб (за замовчуванням) – основна таблиця GPT – 128 записів кожного розділу з кожним записом розміром 128 байт (а значить, 16 КБ). Цей розмір може бути змінений, щоб вмістити більше записів в таблиці (> 128), а також може бути зменшений. Кожен розділ має тип розділу GUID.
4. Останній 512 байт жорсткого диска – вторинний заголовок GPT. Містить унікальний диск GUID, розміщення вторинної таблиці розділів, кількість можливих записів в таблиці розділів, контрольну суму та вторинну таблицю розділів, положення первинного заголовку GPT. Цей заголовок може бути використаний для відновлення GPT даних у випадку, якщо основний заголовок пошкоджений.
5. Далі 16 КБ – вторинна таблиця GPT. Всі дані ідентичні первинній талиці. Використовується в основному для відновлення.

1.3.6 Записи даних про розділи

Перші 16 байт визначають GUID типу розділу. Скажімо, GUID системного EFI розділу має вигляд «C12A7328-F81F-11D2-BA4B-00A0C93EC93B». Наступні 16 байт містять GUID, унікальний для даного конкретного розділу. Далі записуються дані про початок і кінець 64-бітних LBA, якщо вони є. Останнє місце відводиться інформації про імена і атрибутах розділів. На рисунку 1.2 приведена діаграма, яка схематично пояснює формат GUID Partition Table.

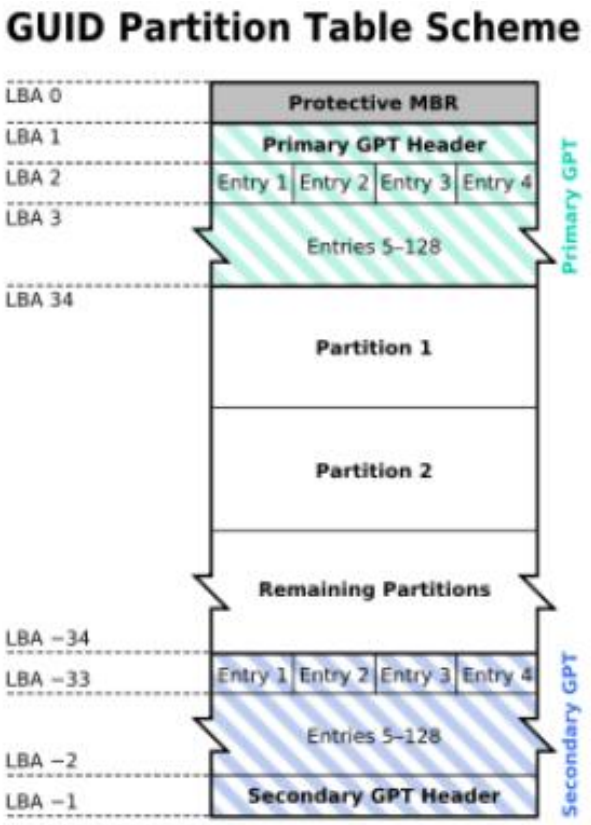

Рисунок 1.2 – Діаграма, що пояснює формат GUID Partition Table

1.3.7 Створення розділів і файлових систем

Disk Druid — здатний створювати і видаляти розділи диску, згідно з вимогами користувача, і працювати с точками монтування для кожного розділу.

fdisk — традиційний інструмент Linux для роботи з розділами диску. Він більш гнучкіший і вимагає досвіду від користувача роботи з дисками і розділами.

Parted — програма дозволяє маніпулювати таблицею розділів, створювати, видаляти, змінювати та діагностувати розділи і файлові системи.

Після встановлення у нас залишився ще один не розмічений диск. Перший, розмічений і той що використовується ОС, скоріш всього підключений як /dev/sda, а другий, не розмічений як /dev/sdb.

Для комфортної роботи можна віддалено підключитися до віртуальної машини через SSH. Для цього дізнаємося її адресу. В терміналі віртуальної машини :

```
ifconfig # дізнаємося IP-адресу (192.168.0.103)
```

Встановлюємо ssh:

```
sudo apt-get install openssh-server # встановлюємо  ssh сервер
sudo reboot # перезавантажимося про всяк випадок
```

Зараз в терміналі реальної машини ми можемо написати:

```
ssh <ім’я користувача в віртуальній машині>@<адреса>
```

```
ssh dimas@192.168.0.103
```

Приклад створення жорсткого диску за допомогою fdisk.

```
Sudo su
fdisk /dev/sdb
```

n — створити новий розділ,
p — primary partition,
1 — номер початкового циліндру (початок диску),
+100M — створюємо диск розміром 100Мб (кінець диску).

Також можна створити ще пару дисків аналогічним чином. Для виходу із збереження обов'язково необхідно натиснути «w». Розділи ми створили, але треба ще створити файлові систему на ньому і промонтувати диск. Створюємо Ext2 файлову систему

`mkfs.ext2 /dev/sdb` # створюємо файлову систему.

`mkdir /media/newdisk` # створюємо каталог, до якого примонтуємо розділ.

`mount /dev/sdb1 /media/newdisk/` # після примонтування їм можна користуватися.

1.3.8 Зберігання копії диску та наступне її використання

Для того, аби зберегти копію диску (наприклад, CD-ROM), необхідно зробити наступне:

1. Переконайтеся в наявності в каталозі достатньо вільного місця.
2. Виконати команду dd if=/dev/cdrom of=cdrom.iso bs=1M.
3. Після цього можна продивитись вмісь файду cdrom.iso, змонтувати його, наприклад, так:

```
mount -o loop cdrom.iso /mnt/cdrom
```

В якості вихідного пристрою для копіювання також може виступати будь-який дисковий пристрій, наприклад дискета або жорсткий диск. Крім того, образ CD, що вийшов, - ROM можна записати на матрицю CD - R/RW з використанням програми cdrecord, оскільки файл cdrom.iso є повним образом диска.

1.3.9 Шифрування даних на рівні ФС

Процедура створення зашифрованої файлової системи може виглядати наступним чином:

1. Необхідно створити файл необхідного розміру, для 8 Мб:

```
sh# dd if=/dev/zero of=test_file count=8 bs=1M
```

2. Необхідно настроїти алгоритм шифрування:

```
sh# modprobe cryptoloop 20 losetup -e blowfish /dev/loop0
test_file
```

3. Програма запитає розмір ключа:

```
D1 Available keysizes (bits): 128 160 192 256 Keysize:
```

4. Далі буде запитан пароль. Після введення пароля алгоритм шифрування blowfish буде підключений до пристрою /dev/loop0. Дані в зашифрованому виді будуть зберігатися у файлі test_file.

5. Необхідно створити файлову систему:

```
sh# mke2fs /dev/loop0
```

6. Змонтувати зашифрований пристрій:

```
sh# mount /dev/loop0 /mnt/disk
```

Після цього можна працювати з `/mnt/disk` як із звичайним пристроєм, який після закінчення роботи потрібно розмонтувати. Для подальшого використання даних необхідно повторити кроки 2 і 4.

Таким чином можна організувати роботу із зашифрованими файловими системами.

1.4 Вміст звіту

Робота вважається виконаною, якщо система встановлена і диски розмічені. Звіт повинен містити в собі повний набір команд, які використовувалися в ході лабораторної роботи.

1.5 Контрольні запитання

1. Перерахуйте способи встановлення дистрибутиву сімейства UNIX.
2. Чим можна розбити диск на розділи?
3. Яких тип має розділ Linux? Яких тип має розділ підкачки Linux? Для чого і де вказується тип розділу?
4. Як встановити програмний пакет після інсталяції і як його видалити?
5. Що необхідно зробити, щоб можна було завантажувати інші ОС на цьому ж комп’ютері і як це зробити?
6. Опишіть можливості програм parted, FIPS, fdisk та їм подібним.
7. Назвіть відмінності основних файлових систем для ОС Linux.
8. Як користуватися mount ?

2 Лабораторна робота № 2. Конфігурування і сбір ядра операційної системи

2.1 Мета роботи

На прикладі ОС Linux вивчити послідовність сбіру, конфігурування і встановлення нового ядра в ОС UNIX.

2.2 Теоретичні відомості

Головне досягнення в ОС UNIX — система дуже мобільна. Це виражається в тому, що вся операційна система, її ядро, порівняно дуже просто переносимо на різні апаратні платформи. Всі частини система, не включаючи ядро, повністю машинно-незалежні. Ці компоненти написані на С, і для їх переносу но нову платформу (принаймні, в класі 32-розрядних комп'ютерів) потребує лише перекомпіляції вихідних текстів в коди цільового комп'ютера.

Найбільші проблеми пов'язані з ядром системи, які повністю скривають специфіку комп'ютера,що використовується і залежать від його специфіки. В результаті розділення машино- залежних і машинно-незалежник компонентів ядра (з точки зору розробників операційної системи, в цьому і скривається основне досягнення ядра ОС Unix) вдалося добитися того, що більша частина ядра не залежить від архітектурних особливостей платформи.

Але порівняно не велика частина ядра машинно-незалежна. При переносі системи на нову платформу потрібно переписати цієї частини з використанням асемблеру і врахуванням кінцевої апаратури. Машинно-незалежні частини ядра добре ізольовані ві основної машинно-залежної частини і при гарному розумінні кожного машино-залежного компоненту переписування залежних компонентів є суто технічною задачею, хоча і потребує високої кваліфікації програміста.

Машинно-залежна частина традиційного ядра ОС UNIX включає наступні компоненти:

— розкрутка і ініціалізація системи на низькому рівні (поки що це залежить від апаратури);
— первинне опрацювання внутрішніх і зовнішніх переривань;
— управління пам'яттю (в тій частини, котра відноситься до апаратної обробки віртуальної пам'яті);
— перемикання контексту процесів між режимами користувача і ядра;
— пов'язанні з особливостями платформи частини драйверів пристроїв.

Ядро ОС сімейства UNIX може бути конфігуровано для оптимальної роботи конкретної машини і/чи окремих програм. Наприклад, можна змінити системні константи, такі як число задач чи максимальний обсяг пам'яті, яка використовується і т. п., можна змінити параметри чи алгоритми диспетчеризації, включаючи ядро чи виключити із нього підтримку тих чи інших системних функцій (зазвичай називають їх загальним терміном 'features').

Загальна послідовність для ОС UNIX послідовність зборки нового ядра виглядає наступним чином:

1) Знайти вихідні тексти ядра.

2) Зібрати інформацію о параметрах машини і периферійних пристроях.

3) Прочитати файли 'README', 'INSTALL' або 'Makefile' в кореневому каталозі дерева вихідних текстів ядра.

4) Додати чи модифікувати, вихідні файли драйверів або інших підпрограм ядра в відповідних каталогах дерева вихідних текстів.

5) Продивитись и коректувати файли конфігурації (склад) ядра.

6) Виділити проміжні результати останньої зборки і оновити залежності між вихідними файлами такою послідовністю команд:

```
make clean; make dep | | make depend
```

7) Побудувати ядро за допомогою програми 'make'. Зазвичай вона має вигляд команди: sh# make unix 2>&1 | tee errlog.txt.

8) Відкомпілювати і встановити, якщо треба додаткові модулі.

```
make modules; make modules_install.
```

9) Встановити ядро в системі, попередньо зберігши старе, на випадок якщо нове буде працювати не коректно.

10) Перезавантажити систему.

Самий простий спосіб зборки — команда 'make install', якщо така команда підтримується в Makefile, вона може змінити пп. 7-9.

2.3 Хід роботи

1. Ознайомитися з деревом каталогу вихідних текстів ядра ОС Linux.
2. Ознайомитися з поточною конфігурацію ядра, налаштувати параметри і набір драйверів для вашого ПК (скоріш за все необхідно коректувати тип процесора, файлові системи і драйвери периферійних пристроїв і мережевих протоколі — все зайве вимкніть).
3. Додайте до файлу init/main.c свій підпис на зразок «Це Васі ядро» додавши функцію 'printk()'. Продумайте як Ви зможете продемонструвати, що підпис виводиться при завантаженні ядра.
4. Ввімкніть відлагоджувальні повідомлення ядра для операційної згідно із своїм варіантом. Для цього, наприклад, вставте рядок #define DEBUG в начало вихідного файлу на мові C і рядок #undef DEBUG в його кінець, змініть Makefile в кореневому каталозі, замінив рядок

```
CFLAGS = -Wall –Wstrict-prototypes –O2 –fomit-frame-pointer
```

на рядок

```
CFLAGS = -Wall –Wstrict-prototypes –O2 –g
```

тим самим, включивши в ядро відлагоджувальну інформацію.

5. Виконайте сбір і встановлення ядра. Додайте йому власну мітку.
6. Завантажте систему з новим ядром і продемонструйте вивід відлагоджувальних повідомлень.

2.3.1 Приклад зборки ядра для ОС Ubuntu

Ця інструкція була взята із офіційного сайту Ubuntu https://help.ubuntu.com/community/Kernel/Compile перекладена, адаптована і модифікована. Протестована на Ubuntu 11.04.

Збирати ядро краще за все не на віртуальній машині — зекономите кучу часу. Зібране ядро можна буде встановити як на віртуальній так и на стаціонарній машині, в залежності від конфігурації. Ніяких ризикових помилок під час зборки не повинно виникнути. Якщо ж зібране ядро ви не зможете встановити, можна завантажитися із старим, якщо ви вручну його не видалите.

1. Скачаємо вихідні коди ядра.

```
sudo apt-get install linux-source device-tree-compiler #
зкачуемо
mkdir ~/src  # каталог для ядра
cd ~/src
##### введіть версію ядра в поле при распаковці вихідних файлів
tar xjvf /usr/src/linux-source-<version-number-here>.tar.bz2
#### тут також версія
cd linux-source-<version-number-here>
```

Зараз ми перейшли в каталог із вихідними текстами ядра.

2. Конфігурація ядра. Тепер нам потрібно скопіювати поточну конфігурацію вашого ядра.

```
cp -vi /boot/config-`uname -r` .config
```

Перед конфігурацією ядра встановимо необхідні бібліотеки і програми. Далі є два варіанта работи в оконному режимі (make menu xconfig) налаштування чи консольному (make menuconfig). Різниці немає, робота виглядає однаково. Виберемо старий метод роботи.

```
sudo apt-get install libncurses5 libncurses5-dev # потрібні
бібліотеки
Make menuconfig
```

Для новішого же варіанту (в оконому режимі) потрібно встановити

```
sudo apt-get install qt3-dev-tools libqt3-mt-dev #
make xconfig
```

Далі за своїм бажанням чи необхідністю додаємо чи прибираємо модулі, які вважаємо за потрібними. Конфігуруємо у відповідності до конфігурації машини. Наприклад, якщо у мене ноутбук Asus, то мені немає сенсу модулі для підтримки обладнання HP. Уважно продивляємося и конфігуруємо, чим більше приберемо зайвого, тим менше будемо чекати компіляції. Зі структурою ядра можна ознайомитися, прочитавши http://wiki.kryukov.biz/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D1%8B_%D1%8F%D0%B4%D1%80%D0%B0_Linux або http://kernelnewbies.org/LinuxChanges#head-82b14f7727c01c4fb5dee39d6eae42facd4bc047

Після конфігурації зберігаємося!

3. Додаємо свої мітки.

Заходимо в ./init/main.c і в першу сподобану процедуру пишемо свій автограф-привітання. Вибір функції потрібно обґрунтувати.

```
gedit ./init/main.c
kernel_init{
…
printk (KERN_WARNING "Warning: Dimas was here.\n");
...
 }
```

Зберігаємося.

4. Додаємо відлагоджувальну інформацію.

У нас є вихідні тексти. Каталоги відповідають дереву конфігурації. Я додам відглагоджувальні інйормацію о роботі з CD-ROM'ом.

```
cd ./drivers/cdrom/
gedit ./cdrom.c
```

Зараз вибираємо «гарну фунцкію», вона повинна визиватся часто, чи хоча б визиватися, щоб було видно вивід інформації. Наприклад, функція запису диску визивається не досить часто, як читання з диску чи відкривання дисководу. Я обрав собі

```
void cdrom_release(struct cdrom_device_info *cdi, fmode_t mode)
 {
 …
 printk(KERN_WARNING "Warning: CD have opened. Dimas.\n")
 …
}
```

Також добавимо в початок файла #define DEBUG і строку #undef DEBUG в його кінець.

5. Заходимо в Makefile

```
gedit ./Makefile
```

і замінюємо рядок

```
CFLAGS = -Wall -Wstrict-prototypes -O2 -fomit-frame-pointer
```

на рядок

```
CFLAGS = -Wall –Wstrict-prototypes -O2 -g
```

6. Приступаємо до компіляції ядра

```
make-kpkg clean #
```

Цей пункт можна (і навіть краще) не виконувати. Адже він очищає скомпільовані раніше модулі. Якщо цього не виконати, то не модифіковані та відкомпільовані модулі не будуть заново компілюватися — перевагою є менший час компіляції ядра.

`Some-string-here` — тут назвемо неше ядро.

```
 fakeroot make-kpkg --initrd --append-to-version=-some-string-
here kernel-image kernel-headers
```

7. Встановлення нового ядра.

Після компіляції будуть створені два файли: ***image та ***headers (що це за файли подивіться і дізнайтеся самостійно)

```
sudo dpkg -i linux-image-*.deb
```

`sudo dpkg -i linux-headers-*.deb *` — назва ваших файлів.

При встановленні Grub автоматично пропише нові ядра в файл налаштувань завантажень. Якщо нове ядро не завантажується, необхідно завантажити попереднє, видалити неробоче (наприклад за допомогою Synapric'a), виправити помилки, зібрати і встановити заново.

Якщо при загрузці нового ядра ви отримуєте повідомлення "Kernel panic - not syncing: VFS: Unable to mount root fs on unknown-block(0,0)" однією з проблем може бути збій при створенні файлу initrd.img-'kernel-name'.

Самим прости рішенням цієї проблеми може бути:

— перезавантажитись з робочим ядром;

— виконати команду "sudo update-initramfs -c -k 'kernel name'";

— редагувати /boot/grub/grub.cfg, дописавши "... initrd 'шлях до створенного initrd.img-xxx' ..."

Наприклад, ви маєте:

```
menuentry 'Ubuntu, c Linux 2.6.32-38-generic' --class ubuntu --
class gnu-linux --class gnu --class os {
recordfail
insmod ext2
set root='(hd0,1)'
search --no-floppy --fs-uuid --set 57b04421-2d65-4e21-9b66-
22e5925bf4a4
linux /boot/vmlinuz-2.6.32-38-generic root=UUID=57b04421-2d65-
4e21-9b66-22e5925bf4a4 ro quiet splash
 }
```

Редагуємо на:

```
menuentry 'Ubuntu, c Linux 2.6.32-38-generic' --class ubuntu --
class gnu-linux --class gnu --class os {
recordfail
insmod ext2
set root='(hd0,1)'
search --no-floppy --fs-uuid --set 57b04421-2d65-4e21-9b66-
22e5925bf4a4
linux /boot/vmlinuz-2.6.32-38-generic root=UUID=57b04421-2d65-
4e21-9b66-22e5925bf4a4 ro quiet splash
initrd /boot/initrd.img-2.6.32-38-generic
 }
```

— виконуєте 'sudo update-grub', хоча це і не обов’язково;

— перезавантажуєтесь;

— завантажуєтесь з новим ядром.

2.3.2 Варіанти завдань

Необхідно зібрати своє ядро ОС, додати вивід своєї інформації при завантаженні ОС, включити вивід відлагоджувальної інформації для операції з пристроєм згідно варіанту завдання з таблиці 2.1.

Таблиця 2.1 – Варіанти завдань

Номер	Вивід інформації при роботі з
1	USB
2	Bluetooth
3	WiFi
4	Ethernet
5	Ext4

6	Ramfs
7	SCSI
8	Power Leds
9	CD-ROM
0	VFAT

Можна вибрати інший варіант за умови, якщо даного пристрою немає або є інші вагомі причини.

Корисні посилання до виконання даної лабораторної роботи наведені у джерелах [1-5].

2.4 Вміст звіту

1. Дерево каталогів першого рівня вихідних текстів ядра і коротке їх призначення кожного каталогу.
2. Послідовність дій для встановлення та зборки нового ядра на Вашій машині.
3. Перелік конфігурацій що ви залишили чи прибрали.

2.5 Контрольні запитання

1. Навіщо потрібна перекомпіляція ядра?
2. Як зібрати ядро для сімейства UNIX?
3. Що таке модулі і навіщо вони потрібні в ОС LINUX?
4. Чому можу не працювати система з новим ядром?
5. Що зробити, щоб нове ядро завантажилося и як це зробити?
6. Які способи відлагоджування ядра ОС LINUX?

3 Лабораторна робота №3. Створення завантажувального пристрою

3.1 Мета роботи

Ознайомитися з методами створення завантажувального пристрою. Навчитися створювати завантажувальний пристрій.

3.2 Теоретичні відомості

Що таке завантажувальна флешка, для чого вона потрібна і які її можливості? Незабаром CD і DVD накопичувачі втратять свою популярність унаслідок громіздкості і порівняно невеликої ємності. Їм на зміну приходять компактні флеш-карти або USB-накопичувачі, об'ємом від 2 Гб і більше.

USB-флеш-накопичувач (сленг. флешка, флешка, флеш-драйв) — запам'ятовуючий пристрій, що використовує як носій флеш-пам'ять і підключається до комп'ютера або іншого пристрою, що зчитує по інтерфейсу USB.

Основне призначення USB-накопичувачів — зберігання, перенесення і обмін даними, резервне копіювання, завантаження операційних систем (LiveUSB) та ін.

За допомогою завантажувальної флешки можна форматувати й відновлювати жорсткі диски комп'ютера, тестувати оперативну пам'ять, підключатися до інтернету або локальній мережі, в залежності від програм, які встановлені на флеш-карті.

Флеш пам'ять була винайдена Фудзі Масуока (Fujio Masuoka), коли він працював у Toshiba в 1984 році. Ім'я «флеш» було придумано також у Toshiba колегою Фудзі, Седзі Аріізумі (Shoji Ariizumi), тому що процес стирання вмісту пам'яті йому нагадав фотоспалах (англ. flash). Масуока представив свою розробку на IEEE 1984 International Electron Devices Meeting (IEDM), що проходила в Сан-Франциско, Каліфорнія. Intel побачила великий потенціал у винаході і в 1988 році випустила перший комерційний флеш-чіп NOR-типу.

NAND-тип флеш-пам'яті був анонсований Toshiba в 1989 році на International Solid-State Circuits Conference. У нього була більше швидкість запису і менше площа чипа.

Переваги:

— мала вага, безшумність роботи і портативність;
— універсальність: сучасні комп'ютери, телевізори та DVD-програвачі мають USB-роз'єми;
— низьке енергоспоживання (так як не є механізмом на відміну від CD, DVD і жорстких дисків);
— працездатність в широкому діапазоні температур;
— більш стійкі до механічних впливів (вібрації і ударів) в порівнянні з жорсткими дисками;

— не схильні до дії подряпин і пилу, які були проблемою для оптичних носіїв і дискет;
— здатні зберігати дані повністю автономно до 5 років. Найбільш перспективні зразки - до 10 років.

Недоліки:

— обмежене число циклів запису-стирання перед виходом з ладу. Чіпи пам'яті, зроблені за технологією MLC (більшість) найчастіше витримують не більше 5000 циклів перезапису. Крім цього обмежений ресурс USB-коннектора - близько 1500 підключень;
— швидкість запису і читання обмежені пропускною спроможністю USB;
— на відміну від компакт-дисків, мають недоліки, властиві будь електроніці;
— чутливі до електростатичного розряду — звичайне явище в побуті, особливо взимку;
— чутливі до радіації.

Завантажувальний USB-носій часто використовують для установки самих різних операційних систем, сама по собі флеш-карта завантажувальних секторів не має, необхідна установка спеціальної програми.

Розглянемо простий приклад створення завантажувальної флешки і її використання для установки операційної системи лінукс. Вам будуть потрібні — звичайна порожня флешка, об'ємом 1 Гб.

3.3 Хід роботи

1) Скачуємо початковий код ядра лінукс
   http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.37.tar.bz2

Також нам потрібні початковий код GRUB (ftp://alpha.gnu.org/gnu/grub/grub-0.97.tar.gz) і BusyBox (http://www.busybox.net/downloads/busybox-1.18.3.tar.bz2)

2) У своєму домашньому каталозі створюємо робочий каталог, розпаковуємо початковий код в окремі каталоги в робочому каталозі розпакування архівів.
3) Збираємо ядро з початкового коду.

В консолі переходимо в каталог, куди ми розпакували початковий код ядра Лінукса, викачуємо сюди. config файл, який можна скачати (http://upload.com.ua/get/902372703/.config) (Перед складанням перевірте ім'я файлу конфігурації він повинен називатися «.config» саме з крапкою на початку!)

(Ті хто хочуть самостійно конфігурувати ядро, можуть набрати $ make nconfig) і набираємо

```
$ Make
```

Якщо при конфігуруванні нічого не було змінено то, зібране ядро (файл bzImage) знаходиться в <початковий код ядра> / arch/x86/boot /

4) Збираємо BusyBox. В консолі переходимо в каталог, куди ми розпакували початковий код BusyBox.

Можливо в систему потрібно встановити libcurses для правильної роботи menuconfig! Встановити можна за допомогою менеджера пакетів вашої системи. (Якщо menuconfig для BusyBox не викликається і Ви не знаєте як встановити libcurses то можна скачати конфігураційний файл тут http://upload.com.ua/get/902458997/config і покласти його в каталог BusyBox, не забудьте перейменувати його в «. Config» , після цього $ make menuconfig робити НЕ ПОТРІБНО!).

5) Конфігуруємо збірку:

```
$ Make menuconfig
```

Відзначаємо пункт:

```
Busybox Settings ---> Build Options ---> [*] Build BusyBox as a static binary (no shared libs)
```

Повертаємося назад в перше меню конфігуратора, зберігаємо конфігурацію.

```
Save Configuration to an Alternate File
```

Після того як ми маємо файл конфігурації в каталозі Busybox, можна його зібрати, для цього набираємо:

```
$ Make
```

Зібраний бінарний файл називається busybox і знаходиться в каталозі з його початковим кодом.

6) Збираємо Grub.

Для складання необхідно щоб в системі були встановлені:

```
GNU Bison 2.3 or later
GNU gettext 0.17 or later
GNU binutils 2.9.1.0.23 or later
```

В консолі переходимо в каталог, куди ми розпакували початковий код Grub. Конфігуруємо збірку, і збираємо.

```
$ Cd <шлях до каталогу з розпакованим початковим кодом Grub>
$. / Configure
$ Make
```

Якщо. / Configure лається на binutils (GRUB requires a working absolute objcopy ;) то необхідно модифікувати. / configure

```
echo $ ECHO_N "checking whether $ {OBJCOPY} works for absolute addresses ...
.... 
.... for link_addr in 0x2000 0x8000 0x7C00; do
.... 
.... if {ac_try = '$ {OBJCOPY-objcopy} - only-section =. text-O binary conftest.exec conftest' ....
```

Якщо компіляція завершується з помилкою «kern / device.c: 80: error: generating trampoline in object (requires executable stack)» треба виправити Makefile:

— виконуємо `Sudo chmod a + w. / Makefile` ;

— відкриваємо Makefile будь-яким текстовим редактором, видаляємо параметр «-Werror» (в кінці найдовшої строчки);

— зберігаємо зміни, запускаємо збірку заново `($ make)`.

Із збіркою Grub як виявилося виникло безліч проблем, звичайно краще все-таки зібрати вручну, але якщо зовсім ніяк, то вже зібрану нову версію для архітектури x86, скачати можна звідси:

http://upload.com.ua/get/902459022/grub -0.97.zip

(архів необхідно тільки розпакувати, $. / configure і $ make виконувати не потрібно. Розпакувати в окремий каталог і далі по тексту замість шляху до каталогу з початковими файлами Grub підставляти шлях до каталогу з розпакованим Grub або просто розпакувати у каталог з початковим кодом.

7) Приступаємо до форматування флешки.

Визначаємо файл пристрою флешки. (У мене це / dev / sdb) і в подальших командах підставляємо шлях до своєї флешки.

Увага! Якщо не правильно вказати пристрій, то у Вас є чудова можливість вбити розділи на жорсткому диску і втратити всю інформацію.

Upd: визначити файл пристрою флешки можна так: виймаємо всі флешки і інші переносні пристрої зберігання інформації і до кінця лабораторної роботи ніяких нових пристроїв окрім флешки, яку відформовуємо, краще не підключати, дивимося які файли жорстких дисків є в системі:

```
$ Ls / dev | grep sd
$ Ls / dev | grep hd
```

Підключаємо флешку, і дивимося знову, має з'явитися ще один пристрій у мене це sdb. Хто не знає пристрої відрізняються останньою буквою, цифри біля це логічні диски цього пристрою, тобто sdb1 це перший логічний диск на моїй флешці, sdb2 - другий і т.д.

```
parted / dev / sdb
```

Створюємо на флешці нову таблицю розділів:

```
(Parted) mklabel
New disk label type? msdos
Warning: The existing disk label on / dev / sdb will be destroyed and all data on this disk will be lost. Do you want to continue?
Yes / No? y
Створюємо на флешці 1-й розділ:
(Parted) mkpart
Partition type? primary / extended? p
File system type? [Ext2]? ext2
Start? 512B
End? 256MB
Warning: The resulting partition is not properly aligned for best performance.
Ignore / Cancel? i
(Parted) p
Model: Flash Drive UT_USB20 (scsi)
Disk / dev / sdb: 2022MB
Sector size (logical / physical): 512B/512B
Partition Table: msdos
Number Start End Size Type File system Flags
1 512B 256MB 256MB primary
```

Покидаємо parted:

```
(Parted) quit
Створюємо на флешці 2-й розділ:
$ Sudo fdisk / dev / sdb
Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4, default 2): 2
First sector (500001-3948542, default 500001):
Using default value 500001
Last sector, + sectors or + size {K, M, G} (500001-3948542,
default 3948542):
Using default value 3948542
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): c
Changed system type of partition 2 to c (W95 FAT32 (LBA))
Command (m for help): a
Partition number (1-4): 1
Command (m for help): p
Disk / dev / sdb: 2021 MB, 2021654016 bytes
63 heads, 62 sectors / track, 1010 cylinders, total 3948543
sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical / physical): 512 bytes / 512 bytes
I / O size (minimum / optimal): 512 bytes / 512 bytes
Disk identifier: 0x000082f5
Device Boot Start End Blocks Id System
/ Dev/sdb1 * 1 500000 250000 83 Linux
/ Dev/sdb2 500001 3948542 1724271 c W95 FAT32 (LBA)
Command (m for help): w
The partition table has been altered!
Calling ioctl () to re-read partition table.
WARNING: If you have created or modified any DOS 6.x
partitions, please see the fdisk manual page for additional
information.
Syncing disks.
```

8) Створюємо на розділах флешки файлові системи:

```
mkfs.ext3 / dev/sdb1
mkdosfs-F 32 / dev/sdb2
```

9) У робочому каталозі створюємо каталог для монтування флешки.

```
$ Cd <шлях до робочого каталогу>
$ Mkdir mnt
```

10) Монтуємо перший розділ флешки:

```
$ Sudo mount / dev/sdb1. / Mnt /
```

11) Для автоматичного підхоплення завантажувачем розділу створеної флешки додаємо мітку розділу

```
$ Sudo e2label / dev/sdb1 RESCUE
```

12) Переходимо на змонтований розділ і створюємо в ньому каталоги / boot / grub /

```
$ Cd. / Mnt /
$ Sudo mkdir boot
$ Cd boot
$ Sudo mkdir grub
$ Cd. / Grub
```

13) Встановлюємо Grub на флешку.

Копіюємо файли завантажувача, в каталог grub:

```
$ Sudo cp <шлях до каталогу з розпакованим початковим кодом
Grub> / stage1/stage1. /
$ Sudo cp <шлях до каталогу з розпакованим початковим кодом
Grub> / stage2/stage2. /
```

Установку завантажувача і створення файлів проводимо від імені супер користувача (команда $ su). Для того щоб вийти з режиму супер користувача використовується Команда # exit.

(Зверніть увагу запрошення командного рядка в режимі супер користувача змінилося з $ на # далі для стислості всі команди для виконання яких недостатньо можливостей; sudo будуть починатися з символу #).

Зверніть увагу ЦС автоматично замінює символ подвійні лапки (виглядають як два ') на символи «і». Перевіряйте консольні команди після копіювання!

```
echo-e "default \ n # \ n # \ n # \ n # \ n # \ n # \ n # \ n #
\ n # \ n # \ n # WARNING: If you want to edit this file
directly, do not remove any line \ n # from this file, including
this warning. Using \ `grub-set-default \ 'is \ n # strongly
recommended.»>. / Default
echo-e "(hd0) \ tLABEL = RESCUE»>. / device.map
echo-e "default 0 \ ntimeout 2 \ ntitle Linux Rescue \ nroot
(hd0, 0) \ nkernel / boot / bzImage root = LABEL = RESCUE init =
\" / bin / busybox sh \ "\ ninitrd (hd0, 0) / boot /
rescue_initramfs.cpio.gz »>. / menu.lst
```

В консолі переходимо в каталог, куди ми розпакували початковий код Grub.

```
cd <шлях до каталогу з розпакованим початковим кодом Grub> /
grub / grubGNU GRUB version 0.97 (640K lower / 3072K upper
memory) [Minimal BASH-like line editing is supported. For the
first word, TABlists possible command completions. Anywhere else
TAB lists the possiblecompletions of a device / filename. ]
Grub> device (hd0) / dev / sdb grub> root (hd0, 0) Filesystem
type is ext2fs, partition type 0x83grub> setup (hd0) Checking if
«/ boot/grub/stage1» exists ... yesChecking if «/ boot /
grub/stage2 »exists ... yesChecking if« /
boot/grub/e2fs_stage1_5 »exists ... noRunning« install /
boot/grub/stage1 (hd0) / boot/grub/stage2 p / boot / grub /
menu.lst «... succeededDone.grub > quit.
```

14) Копіюємо на флешку ядро Лінукса.

```
$ Cd <шлях до робочого каталогу> / mnt / boot /
$ Sudo cp <початковий код ядра> / arch/x86/boot/bzImage. /
```

15) Створюємо образ Initramfs

```
$ Cd <шлях до робочого каталогу> /
$ Sudo mkdir initramfs
```

```
$ Cd initramfs
$ Mkdir. / Bin
$ Sudo cp <шлях до каталогу з розпакованим початковим кодом
Busybox> / busybox. / Bin /
```

Якщо крім Busybox Вам потрібні інші утиліти, то також копіюємо їх сюди, не забуваючи про їх динамічні бібліотеки, які копіюємо в підкаталоги як зазначено утилітою ldd, тільки замість кореня «/» використовуємо «. /»

Створюємо файл ініціалізації, щоб оболонка сама запускалася, та кілька потрібних файлових систем монтувалися.

Щоб була можливість працювати з пристроями потрібно скопіювати, файли потрібних пристроїв з каталогу / dev / з працюючої ОС, в однойменний каталог в корені образу Initramfs. Від цього може постраждати універсальність, працювати буде і без цього але файли пристроїв потрібно створювати вручну.

```
echo-e '#! / bin / busybox sh \ nmount-t proc none / proc \
nmount-t sysfs none / sys \ n / bin / busybox sh'> init $ sudo
chmod + x. / init
find. -Print0 | cpio - null-ov - format = newc | gzip -9> <шлях
до робочого каталогу> / mnt / boot / rescue_initramfs.cpio.gz
```

16) Тепер у нас є завантажувальна флешка з лінуксом. Акуратно размонтуємо і можна користуватися.

3.4 Вміст звіту

Звіт повинен містити короткий опис використовуваних в роботі утиліт, а також детальний опис дій з виконання роботи (команди і результат їх виконання). Корисні посилання для даної лабораторної роботи знаходяться у джерелах [6-10].

3.5 Контрольні запитання

1. Що таке флеш накопичувач? Навіщо він потрібний?
2. Навіщо потрібний BusyBox?
3. Як монтувати флеш носій?
4. Як створити новій розділ на флешці?
5. Що необхідно зробити, аби додати мітку розділу?

4 Лабораторна робота № 4. Управління обліковими записами і процесами користувачів

4.1 Мета роботи

Отримати практичні навички адміністрування користувачів і контролю над процесами в ОС сімейства UNIX.

4.2 Теоретичні відомості

4.2.1 Загальні задачі управління користувачами

Багато сучасних застосувань вимагають підтримки колективної роботи, авторизації і автентифікації користувачів, природно, в основному ці функції покладаються на розраховану на багато користувачів операційну систему, в якій можна розмежувати ресурси і права між користувачами і/або групами користувачів.

Концепція захисту, прийнята в сучасних ОС, базується на апаратному захисті коду операційної системи і ресурсів ПК за допомогою розділення процесів на привілейовані і призначені для користувача. Доступ до усіх ресурсів мають тільки привілейовані процеси, призначені для користувача процеси повинні звертатися до сервісних підпрограм ОС. Надання призначеним для користувача програмам прав на безпосереднє звернення до ресурсів потенційно небезпечно для будь-якої системи!

Операційна система на основі внутрішніх структур даних, що називаються 'матрицею прав доступу' приймає рішення про правомочність тих або інших дій користувача. У UNIX матриця прав доступу задається файлами 'passwd' і 'group ', а також індексними дескрипторами файлів.

Адміністрування користувачів включає наступні завдання:

— створення і видалення користувачів і груп;

— призначення і зміна паролів;

— налаштування прав доступу;

— контроль цілісності системної бази даних про користувачів;

— контроль використання дискового простору;

— контроль призначених для користувача процесів.

Усі перелічені вище завдання в тому або іншому ступені піддаються автоматизації.

4.2.2 Стандартні системні файли

1. /etc/passwd – файл, що визначає основні призначені для користувача параметри.

2. /etc/group – файл, що визначає групи користувачів.

3. /etc/aliases – поштові псевдоніми користувачів. Після зміни цього файлу необхідно виконувати команду 'newaliases'.

4. /etc/profile – командний файл, що виконується при вході в систему будь-якого користувача.

4.2.3 Команди управління користувачами, їх файлами та процесами

Нижче приведено кілька корисних команд для роботи з користувачами, їх файлами і процесами:

adduser [опції] ім'я_ користувача
useradd [опції] ім'я_ користувача

- додати користувача.

userdel [-r] ім'я_ користувача

- видалити користувача з системи.

edquota [опції] ім'я_ користувача. … | ім'я_ групи… | -t

- назначити дискову квоту.

vipw [опції]

- дозволяє редагувати файл passwd з розрахуванням особливостей системи.

last [опції]

- показати інформацію про сеанси роботи користувачів.

fuser [опції] им'я_файлу …

- визначити процеси, які використовують вказані файли або сокети.

setfacl [опції] им'я_файлу …

- задати для файлу(ів) список управління доступом (ACL- Access Control List).

getfacl [опції] им'я_файлу …

- показати список управління доступом (ACL).

su [опції]

- запустити командний інтерпретатор з заміною ідентифікатора користувача і групи.

sudo [опції] команда

- запустити команду з правами іншого користувача.

kill [опції] PID …

- надіслати процесу сигнал.

strace [опції] команда

- запустити програму в режимі моніторингу системних викликів, сигналів, системних збоїв, и т. п.

top [опції]

- видати відсортований список процесів у системі.

ps [опції]

- видати інформацію про процеси.

Команда 'ps' має велику кількість опцій, що дозволяють використати її виведення в командних файлах(утилітах). Наприклад, отримати тільки номери процесів, що виконують заданий командний рядок "mc", можна так:
sh# ps –no-headers –C mc –o “%p”.

Вам відповідно до свого варіанту пропонується реалізувати будь-якими відомими засобами одне з приведених нижче завдань.

4.2.4 Управління користувачами. Користувачі з точки зору UNIX

Система реєструє наступну інформацію про кожного користувача.

Ім’я користувача (user name) Це ім’я повинно буди унікальним в рамках системи. В іменах можуть бути використані тільки англійські літери, числа і символи _ і .(крапка).

Ідентифікаційний номер користувача (User ID) . Цей номер, скорочено позначається як UID, є унікальним ідентифікатором користувача в системі, Взагалі кажучи, система відстежує користувачів за їх номерами UID, а не по іменах.

Ідентифікаційний номер групи (group ID). Цей номер (скорочено GID) позначає групу, до якої за замовчуванням відноситься користувач. Групи дозволяють регулювати доступ багатьох користувачів до різних ресурсів. Кожен користувач належить одній або кільком групам, і цю приналежність встановлює системний адміністратор.

Пароль (password) Це зашифрований (encripted) пароль користувача. Для створення і зміни пароля використовується команда passwd. В системі користувачам дозволено самостійно змінювати пароль за допомогою утиліти passwd, зазвичай знаходиться /usr/bin/passwd. Власником passwd є користувач root оскільки утиліта працює з файлом /etc/passwd який може модифікувати тільки користувач root. Щоб непривілейований користувач міг працювати з системним файлом (тобто змінити свій пароль) на утиліті passwd повинен бути встановлений біт SUID (при запуску на виконання файл отримує не права того хто запустив його, а права власника файлу). Якщо встановлені права доступу SGID, то це рівнозначно установці біта SUID, тільки успадковуються права не власника файлу, а групи власника. Якщо встановити SGID на каталог, то файли в цьому каталозі будуть мати такі ж установки як і установки каталогу.

Повне ім'я (full name) Крім системного імені користувача, в систему заноситься і зберігається ім'я (прізвище і т. д.) «реального» користувача. Наприклад, користувачеві smitj в реальному житті може відповідати людина на ім'я Jon Smit.

Домашній каталог (home directory) Ця назва каталогу, в який потрапляє користувач після того, як він увійшов в систему (зареєструвався, login), і де зберігаються його власні файли. Такий каталог є у кожного користувача, і всі такі каталоги зібрані в один каталог, зазвичай названий /home.

Початкова оболонка (login shell) Командна оболонка, яка запускається при вході в систему. Наприклад, /bin/bash або /bin/sh.

Вся ця інформація зберігається в файлі /etc/passwd. Кожен рядок у файлі має формат: ім'я користувача: зашифрований пароль: UID: GID: повне ім'я: домашній каталог: оболонка Наприклад:

```
kiwi:Xv8Q981g71oKK:102:100:Laura Poole:/home/kiwi:/bin/bash
```

У деяких системах є «тіньові паролі» (shadow passwords), коли інформація про пароль зберігається в файлі /etc/shadow. Така схема є дещо більш безпечною, оскільки файл /etc/passwd може читатися ким завгодно, а права доступу до файлу /etc/shadow набагато сильніше обмежені.

Тіньові паролі також забезпечують інші функції, наприклад, минання терміну дії пароля.

4.2.5 Створення нових користувачів

При створенні нових користувачів треба зробити послідовність з декількох дій. По-перше, на користувача заводиться запис у файлі /etc/passwd, де користувачеві даються унікальні ім'я і UID. UID звичайних користувачів повинні бути більше 100, оскільки низькі UID зарезервовані для системних цілей. Також вказуються GID, реальне ім'я та інша інформація. Далі створюється домашній каталог користувача, і права доступу встановлюються так, що цим каталогом володіє даний користувач. До каталогу поміщуються файли ініціалізації командної оболонки. Також у всій системі модифікуються конфігураційні файли (наприклад, сховище (spool) для присланих користувачам електронних листів).

Вручну створювати користувачів не так важко, проте коли експлуатується система з великою кількістю користувачів, може виявитися забутою якась деталь. Найпростіше в цьому випадку створювати нових користувачів за допомогою інтерактивної програми, яка автоматично оновлює вміст всіх потрібних системних файлів. Така програма називається useradd або adduser, залежно від того, яке програмне забезпечення встановлено.

У файлі /etc/default/ useradd міститься інформація про стандартну початкову конфігурацію для всіх нових користувачів. У цьому файлі задаються значення змінним, які використовує програма useradd. Крім того, цей файл вказує, де знаходяться конфігураційні файли містять налаштування за замовчуванням. Розташування цих файлів задається змінною SKEL. Файли, що містяться в цей каталог (такі, як файл. Profile, який встановлює режим за замовчуванням у всій системі, а також файли. Zshrc або. Bashrc), будуть автоматично скопійовані в домашній каталог створюваного користувача командою useradd.

4.2.6 Видалення користувачів

Видалення користувачів із системи може бути віконне командою userdel або deluser. Якщо потрібно тимчасово заборонити користувачеві вхід в систему, але не видаляти його домашній каталог та інші зроблені установки, можна просто поставити зірочку (символ *) в те поле файлу /etc/passwd, де знаходиться пароль. Наприклад, таким чином змінена рядок для користувача kiwi буде виглядати як

```
kiwi: * Xv8Q981g71oKK: 102:100: Laura Poole:
/home/kiwi:/bin/bash
```

При цьому вхід в систему користувача kiwi стане неможливим.

4.2.7 Установка атрибутів користувача

Після того, як створено ім'я нового користувача, може виявитися потрібним змінити атрибути цього користувача, наприклад, домашній каталог або пароль. Найпростіший спосіб зробити це – просто поміняти інформацію у

файлі /etc/ passwd. Для створення пароля потрібно використовувати команду passwd. Так, команда passwd larry змінить пароль користувача larry. Змінювати паролі будь-яких користувачів може тільки користувач root, однак свої паролі користувачі можуть змінювати самі, віддаючи команду passwd без параметрів.

4.2.8 Групи користувачів

Як зазначалося вище, кожен користувач належить одній або декількох груп. Єдине, що є істотним у приналежності до тієї чи іншої групи – це права доступу. Для кожного файлу визначений не тільки користувач-власник, а й група-власник, і набір прав доступу, які визначають, як користувачі з цієї групи можуть здійснювати доступ до цього файлу.

При створенні нового користувача створюється також група, ім'я якої збігається з ім'ям користувача і куди входить тільки він один.

Є декілька груп, визначених системою, наприклад, bin, mail, sys. Ці групи створені для оформлення прав доступу до системних файлів, і користувачі не повинні належати до цих груп. Для користувачів створюються спеціальні групи, наприклад, users. Для користувачів можна створити декілька груп, наприклад, student, staff, faculty.

Інформація про групи міститься у файлі /etc/group. Формат кожного рядка такий:

```
назва групи: пароль: GID: інші члени групи
```

Приклади груп:

```
root: *: 0:
users: *: 100: mdw, larry
guest: *: 200:
other: *: 250: kiwi
```

Перша група – root – спеціальна група для користувача root. Друга група –users – містить звичайних користувачів. GID цієї групи дорівнює 100, і в неї входять користувачі mdw і larry. Нагадаємо, що у файлі /etc/passwd кожному користувачеві визначена його група за замовчуванням. Тим не менш, користувачі можуть належати до більш ніж одній групі, і це здійснюється за допомогою перерахування їх імен у файлі /etc/group. Команда groups перераховує список груп, яких стосується (має доступ) даний користувач.

Третя група називається guest і призначена для відвідувачів. Для інших користувачів створено групу other; в цю групу занесений користувач kiwi.

Іноді в файлі /etc/group заповнюється поле password (пароль) для того, щоб встановити пароль на груповий доступ. Це потрібно рідко. Для того, щоб не дозволяти користувачам проникати в привілейовані групи (командою newgroup), в це поле треба поставити символ *.

Для створення нових груп користувачів можуть бути використані команди addgroup або groupadd. Зазвичай легше внести вручну нову сходинку в файл /etc/group, оскільки ніякого іншого конфігурування не потрібно. Для видалення групи можна просто видалити відповідний рядок у файлі /etc/group.

4.2.9 Процеси

ОС сімейства UNIX є багатозадачною операційною системою. Це означає, що одночасно може бути запущена більш ніж одна програма. Кожна програма, що працює в деякий момент часу, називається процесом. Кожна команда, яку ви запускаєте, породжує хоча б один процес. Є кілька системних процесів, запущених весь час і підтримують функціональність системи.

У кожного процесу є унікальний номер, названий process ID, або PID, як і у файлів, у кожного процесу є власник і група. Інформація про власника і групі процесу використовується для визначення того, які файли і пристрої можуть бути відкриті процесом з урахуванням прав на файли. Також у більшості процесів є батьківський процес. Наприклад, при запуску команд з оболонки, оболонка є процесом і будь-яка запущена команда також є процесом. Для кожного запущеного таким шляхом процесу оболонка буде батьківським процесом. Винятком з цього правила є спеціальний процес, названий init. init завжди перший процес, його PID завжди 1. init запускається автоматично ядром під час завантаження ОС.

Дві команди дуже корисні для перегляду працюючих в системі процесів, це ps і top. Команда ps використовується для отримання списку запущених процесів і може показати їх PID, скільки пам'яті використовують, команду, якій вони були запущені і т.д. Команда top показує запущені процеси і оновлює екран кожні кілька секунд, що дозволяє спостерігати за роботою комп'ютера в реальному часі.

За замовчуванням, ps показує тільки ті процеси котрі вам налажат. Якщо використати ключ –A то виведеться таблиця всіх процесів. Наприклад:

```
PID   TTY      TIME     CMD
2643  pts/0    00:00:00 su
2644  pts/0    00:00:00 bash
3092  pts/0    00:00:00 ps
```

Як ви можете бачити в даному прикладі, висновок ps організований в кілька колонок. Ідентифікатор процесу PID обговорювалося раніше. PID призначаються з 1 до 99999 і знову з початку, якщо останнє число буде перевищено. Колонка TTY показує термінал (tty), на якому запущена програма. TIME це кількість часу центрального процесора, використане програмою - це звичайно не час, що минув з запуску програми, оскільки більшість програми проводять багато часу в очікуванні деякого події перед тим, як зайняти час процесора. Нарешті, COMMAND це команда, якій програма була запущена.

У ps є безліч різних опцій, які впливають на виведену інформацію. Один з найбільш корисних наборів опцій це auxww. a дозволяє показати інформацію про всі запущені процеси, а не тільки тих, якими ви володієте. u показує ім'я користувача, що володіє процесом, та інформацію про використовуваної пам'яті. x показує інформацію про процеси-даемони і ww вказує ps показати весь командний рядок, замість обрізання її, коли вона стане занадто довгою, щоб умістится на екран.

Вивід top схожий на щойно описаний. Зазвичай він виглядає так:

```
last pid: 34218; load averages: 0.65, 0.36, 0.29 up 0+11:38:05 16:30:36
processes: 1 running, 2 sleeping
Mem: 82M Active, 93M Inact, 438M Wired, 112K Cache, 112M Buf, 1390M Free
Swap: 2048M Total, 2048M Free
PID USERNAME  THR PRI NICE   SIZE    RES   STATE   C   TIME   WCPU COMMAND
23805  root     1   106  10   6080K  3128K select  3  0:01   0.00%   sshd
23806  root     1    8   10   3136K  2008K  wait   2  0:00   0.00%   bash
34218  roou     1   106  10   2160K  1376K  CPU2   0  0:00   0.00%   top
```

— top – автоматично оновлює екран кожні дві секунди; це значення можна змінити опцією s;
— PID – ідентифікатор процесу;
— USERNAME - користувач, від якого запущений процес;
— THR – кількість потоків, запущених процесом;
— PRI – поточний пріоритет процесу;
— NICE – пріоритет, виставлений командою nice. Від -20 (найвищий) до 19;
— SIZE – розмір процесу (дані, стек і т. д.) в кілобайтах;
— RES – поточне використання оперативної пам'яті;
— STATE – поточний стан («START», «RUN» (тільки в цьому стані показує поточне навантаження програми на процесор), «SLEEP», «STOP», «ZOMB», «WAIT» або «LOCK»);
— C – номер процесора, на якому йде виконання (доступний тільки на SMP системах);
— TIME – час використання процесора в секундах;
— CPU – відсоток доступного часу процесора, яке використовувала запущена програма;
— WCPU – усереднене значення CPU;
— COMMAND – назва команди, під якою працює процес.

4.3 Хід роботи

1. Вивчіть базу даних про користувачів і команди управління цією базою у Вашій версії системи UNIX(назви, формати файлів і синтаксис команд необхідно привести в звіті).
2. Створіть користувача "user" с ім'ям "User U. Userovsky" і поштовим псевдонімом "u_ser". Встановіть для нього період обов'язкової зміни пароля в 30 днів і дискову квоту в 30 MB. Видаліть користувача і усі його файли. Послідовність дій із створення і видалення користувача запишіть в звіт.
3. Вивчіть синтаксис команди 'ps' і формат виведення інформації про процеси.
4. Ознайомтеся з вмістом файлової системи "/proc".
5. Відповідно до свого варіанту розробіть утиліту для ОС сімейства UNIX.

Варіант 1

Утиліта для перевірки цілісності бази даних про користувачів і контролю за програмами зі встановленим бітом 'setuid'. Утиліта повинна забезпечувати

наступні функції: контроль кількості суперкористувачів, перевірка відповідності файлу passwd і тіньового файлу паролів, пошук 'ненадійних' з точки зору захисту виконуваних файлів

Варіант 2

Діалогова утиліта для створення нових користувачів.

Варіант 3

Утиліта для розсилки листів групам користувачів(по файлу '/etc/group'). Передбачити різні режими відправки, наприклад, щоб кожен одержувач бачив або увесь список адрес, або тільки своя адреса.

Варіант 4

Утиліта, що знищує усі процеси вказаного користувача. Передбачити діалоговий режим або вибір з меню.

Варіант 5

Діалогова утиліта kill – вибір процесу(процесів) зі списку і посилка будь-якого сигналу зі списку сигналів.

Варіант 6

Утиліта контролю ресурсів, споживаних призначеними для користувача завданнями. Процес, що перевищив заданий ліміт використання пам'яті або процесорного часу автоматично знищується.

Варіант 7

Сповіщення заданих користувачів про вхід в систему нового користувача(і про вихід з системи, відповідно).

Варіант 8

Діалогова утиліта 'whodo' – список користувачів в динаміці, по вибору висновок повної інформації про процеси.

4.4 Вміст звіту

У звіті наводиться опис дій відповідно до пункту "Хід роботи", синтаксис і опції вивчених команд і початкові тексти розробленої утиліти з коментарями.

4.5 Контрольні запитання

1. Які категорії користувачів і права доступу існують в ОС сімейства UNIX?
2. Приведіть формат файлу 'passwd ', розкажіть про призначення кожного поля.
3. Навіщо потрібна команда sudo? Як призначити індивідуальне право на доступ до захищеного файлу?
4. Як призначити псевдонім користувачеві?
5. Як користувач може змінити своє повне ім'я і shell в 'passwd'?
6. Які права доступу і власник мають бути встановлені для утиліти 'passwd'?
7. Які способи перегляду інформації про процеси Ви знаєте?
8. Приведіть послідовність дій з установки дискових квот для користувачів в Linux.

5 Лабораторна робота № 5. Реєстрація подій у системі

5.1 Мета роботи

Вивчити стандартні для ОС сімейства UNIX засоби обліку і виведення налагоджувальної інформації.

5.2 Теоретичні відомості

Облік подій в системі або мережі є важливим завданням і робиться зазвичай для поточного контролю стану системи(logging), для налагодження системних сервісів(debug logging), а також для підрахунку споживаних користувачами ресурсів(accounting). Зазвичай система обліку реєструє події у вигляді текстових повідомлень, можна також накопичувати облікові дані в змінних-лічильниках, у базах даних і т. п. Стандартним для UNIX підходом до ведення обліку подій в системі являється використання підсистеми SYSLOG.

Підсистема syslog і однойменний мережевий протокол застосовуються для реєстрації будь-яких подій в системі або мережі. Щоб подія була записана за допомогою демона 'syslogd' в журнальному файлі, необхідно щоб програма, що зафіксувала подію, передала його в syslogd по мережі або через UNIX-сокет.

Переваги syslog в порівнянні з індивідуальним обліком в кожній програмі – це централізація обліку і уніфікація формату повідомлень. Облікова інформація розбивається по декількох каналах (FACILITY) і рівнях пріоритетності (PRIORITY). Налаштування демона syslogd записані у файлі /etc/syslog.conf (він може знаходитися в іншому каталозі – дивись 'man syslogd' і 'man syslog.conf'). Якщо змінюється файл syslog.conf, треба послати демонові syslogd сигнал SIGHUP, інакше зміни не набудуть чинності. За сигналом "відбій" (SIGHUP) демон syslogd закриває свої файли реєстрації, перечитує конфігураційний файл і знову запускає процес реєстрації. Для запису повідомлень в журнальні файли з командного рядка або командного файлу існує команда 'logger ', а в програмах на мові Сі можна використати бібліотечні виклики openlog(), closelog(), syslog() і setlogmask(). Далі наведений приклад файлу налаштувань syslog.conf для системи FreeBSD.

```
# приклад
#  $FreeBSD: src/etc/syslog.conf,v 1.13.2.3 2002/04/15 00:44:13
dougb Exp $
# Spaces ARE valid field separators in this file. However,
# other *nix-like systems still insist on using tabs as field
# separators. If you are sharing this file between systems, you
# may want to use only tabs as field separators here.
# Consult the syslog.conf(5) manpage.
*.err;kern.debug;auth.notice                    /dev/console
*.notice;kern.debug;lpr.info;mail.crit  /var/log/messages
security.*                                      /var/log/security
auth.info;authpriv.info                         /var/log/auth.log
mail.info                                       /var/log/maillog
```

```
lpr.info                                /var/log/lpd-errs
cron.*                                  /var/log/cron
*.emerg                                 *
# uncomment this to log all writes to /dev/console to
/var/log/console.log
#console.info                          /var/log/console.log
# uncomment this to enable logging of all log messages to
/var/log/all.log
# touch /var/log/all.log and chmod it to mode 600 before it will
work
#*.*                                   /var/log/all.log
# uncomment this to enable logging to a remote loghost named
loghost
#*.*                                   @loghost
# uncomment these if you're running inn
news.crit
     /var/log/news/news.crit
news.err                               /var/log/news/news.err
news.notice
     /var/log/news/news.notice
# this must help to extract logs from single programs
!startslip
*.*                                    /var/log/slip.log
!pp
*.*                                    /var/log/ppp.log
!mpd
*.*                                    /var/log/mpd.log
!pppoed
*.*                                    /var/log/pppoed.log
!named
*.*                                    /var/log/named.log
!drwebd
*.*                                    /var/log/drwebd.log
```

5.3 Хід роботи

1. Ознайомтеся з вмістом каталогу /var/log в Linux. Здійсніть пошук усіх журнальних файлів в системі.

Список знайдених файлів з іменами програм, яким вони належать, внесіть у звіт.

2. Налаштуйте syslog таким чином:
   — повідомлення про помилки повинні дублюватися на віртуальну консоль № 8;
   — усі повідомлення ... направити в окремий файл;
   — повідомлення про роботу програми sudo спрямуйте по мережі на машину beast;
   — задайте період ротації усіх файлів кожні 2 місяці, кількість архівів, що зберігаються, -- 5.
3. Файли конфігурації syslog і logrotate внесіть в звіт.
4. Перепишіть з сервера і встановіть у себе "secure syslog".

5. Напишіть на мові C або C++ скелетну програму, в якій відкривається реєстрація повідомлень в каналі "LOG_DAEMON" і видаються два повідомлення: про старт і завершення програми. Програма повинна перейти у фоновий режим і зупинитися через 10 секунд після старту, крім того, в ній має бути передбачена реакція на деякі сигнали (на Ваш розсуд).

5.4 Вміст звіту

Звіт повинен містити короткі описи вивчених програм, файли налаштувань і текст програми відповідно до п. 4 ходу роботи.

5.5 Контрольні запитання

1. Для чого треба вести журнальні файли?
2. Навіщо потрібний syslog? Які принципи роботи syslog?
3. Що таке FACILITY і PRIORITY в налаштуваннях syslogd? Наведіть приклади.
4. Як автоматично видаляти застарілі журнальні файли?
5. Що необхідно зробити для використання сервісу syslog з програми на C?
6. Що необхідно зробити для використання сервісу syslog з програми на shell?
7. Які переваги "secure syslog" в порівнянні з традиційним syslog?

6 Лабораторна робота № 6. Налаштування та тестування базових функцій роботи в мережах

6.1 Мета роботи

Вивчити базові мережеві засоби ОС сімейства UNIX, прийоми налаштування і адміністрування IP- мереж.

6.2 Теоретичні відомості

Комп'ютерні мережі прийнято розрізняти за декількома критеріями, наприклад, за територіальною ознакою - локальні і глобальні, по топології з'єднання машин, по використовуваних протоколах і т. п. Для користувачів мережа має бути прозорою, для адміністратора - надійною, легко керованою і по можливості однорідною. Заслуженою популярністю користуються мережі, побудовані на протоколах Internet. Локальні мережі, засновані на цьому протоколі, прийнято називати інтернет мережами.

Сімейство найбільш поширених на сьогодні протоколів називають абревіатурою TCP/IP (Transmission Control Protocol/Internet Protocol). TCP/IP базується на багаторівневій схемі взаємодії протоколів, яка складається з рівнів: фізичного, канального, мережевого, транспортного і прикладного.

Будь-яка сучасна ОС підтримує декілька різних протоколів передачі даних канального рівня і поставляється з набором драйверів для різних способів підключення до мережі (мережевих плат, модемів, і т. п.). Існує декілька популярних протоколів транспортного рівня, наприклад, TCP (Transmission Control Protocol), UDP (User Datagram Protocol). Розглянемо завдання по адмініструванню мереж TCP/IP:

1. Планування мережі(вибір топології, кількості і фізичного розміщення серверів, робочих станцій і іншого активного і пасивного мережевого устаткування, розбиття на підмережі, призначення IP- адрес і символічних імен, вибір типу маршрутизації, і т. п.).

2. Встановлення обладнання і програмного забезпечення для нього (встановлення мережевих плат, налаштування ядра, встановлення драйверів).

3. Запуск базових мережевих служб(налаштування мережевих інтерфейсів і маршрутизації, тестування з'єднань).

4. Установка і налаштування програм, що використовують протоколи передачі даних високого(прикладного) рівня.

6.2.1 Довідкові дані за системою адресації

Мережеві пакети можуть досягти пункту призначення тільки за наявності правильної адреси. У TCP/IP використовується поєднання декількох схем адресації:

— MAC-адреси мережевого устаткування;

— IP-адреси програмного забезпечення;

— текстові імена комп'ютерів.

Мережева плата має MAC- адреса, яка відрізняє її від інших мережевих плат цієї фізичної мережі. Розмір адреси в мережі Ethernet (MAC-адреси) дорівнює 6 байтам (старші три байти – ідентифікатор виробника, молодші – номер пристрою). Розмір адреси для IPv4 дорівнює 4 байтам і його прийнято записувати в десятковій нотації, відділяючи точкою числа, що представляють відповідні байти. Адресу мережі або машини доповнює мережева маска – вона потрібна для правильного вибору маршрутів в мережі (мережа IP є мережею з комутацією пакетів). Мережева маска показує, яка частина IP-адреси є адреса мережі (або підмережі), а яка – номером машини в цій підмережі. Маска зазвичай записується в десятковій формі, але може бути вказана після адреси через косу риску у вигляді кількості початкових одиниць в масці. Наприклад, IP-адреса 127.0.0.1 завжди посилається на поточну машину, її символічне ім'я – localhost. Адреса мережі для localhost 127.0.0.0/8, що відповідає мережевій масці 255.0.0.0.

При використанні класової адресації для зручності керування мережею Internet усі IP-адреси розбиті на п'ять базових класів відповідно до таблиці 6.1.

Таблиця 6.1 – Класи IP-адрес

Клас	Перший байт	Формат	Коментарі
A	1-126	C.M.M.M	Найрізноманітніші мережі або адреси, зареєстровані для Міністерства оборони США. Мережа приблизно з 1.6 мільйонів хостів.
B	128-19	C.C.M.M	Великі організації, зазвичай з підмережами. 16320 мереж, в кожній 65024 адреси.
C	192-223	C.C.C.M	Невеликі організації; адреси цього класу отримати легко. Майже 2 мільйони мереж, в кожній 254 адреси.
D	224-239		Групові адреси; не назначаються на постійній основі
E	240-254		Експериментальні адреси

Оскільки на машині може виконуватися декілька призначених для користувача процесів, то протокол TCP розширює концепцію IP-адрес, вводячи поняття порту. Порт є двобайтовим числом, що вказує конкретне застосування на машині-адресатові. Стандартним сервісам UNIX зіставлені конкретні порти, які визначені у файлі /etc/services.

6.2.2 Протоколи CIDR та IPv6

Для вирішення проблеми нестачі IP- адрес одночасно запропоновано два підходи. Перше з рішень – протокол CIDR (Classless Inter-Domain Routing – безкласова між доменна маршрутизація) – передбачало зміни принципів

управління існуючими чотирьохбайтовими адресами і спрощення таблиць маршрутизації за рахунок введення поняття суміжності. Протокол усуває потребу в системі класів, на підставі якої раніше визначалася мережева частина IP-адреси. Він є прямим розширенням ідеї підмереж, оскільки дозволяє задавати мережеву маску, що визначає межу між мережевою і машинною частинами адреси. В цілях маршрутизації допускається, щоб мережева частина була менше, ніж того вимагає клас адреси. Завдяки укороченій масці виникає ефект об'єднання декількох мереж. З цієї причини протокол CIDR іноді називають протоколом формування надмереж.

Друге довготривале рішення -- це стандарт IPv6. Він є наступним поколінням протоколу IP, в якому довжина адреси збільшена до 16-ти байтів і врахований ряд інших помилок, виявлених упродовж 25 років експлуатації протоколу IP. У ньому інтегровані засоби аутентифікації і забезпечення безпеки, а також усунена процедура фрагментації на проміжних маршрутизаторах.

6.2.3 Файли конфігурації

Для конфігурації мережі вносимо зміни в наступні файли:

— /etc/hosts - використовується для установки відповідності між іменами вузлів і IP- адресами в локальній мережі. Приклад змісту файлу hosts :

```
127.0.0.1 localhost
192.108.21.48 test.black.com test loghost
192.108.21.1 hate.black.com hate
192.108.21.254 chain.black.com fence
```

— /etc/networks - відображає ім'я мережі на мережевий номер і навпаки. У IP- адресах у файлі networks вказується тільки частина, що відповідає номеру мережі або підмережі.

— /etc/sysconfig/network - (специфічний для Linux, FreeBSD використовує /etc/rc.conf), використовується командними файлами завантаження для налаштування мережевих інтерфейсів і статичних маршрутів.

Наприклад:

```
NETWORKING=yes
HOSTNAME=test.black.com
FORWARD_IPV4=false
DOMAINNAME=black.com
GATEWAY=192.108.21.254
GATEWAYDEV=eth0
```

— /etc/sysconfig/networking/ifcfg-xxx - (специфічний для Linux, xxx - ім'я інтерфейсу), використовується для налаштування окремих інтерфейсів, у тому числі і псевдонімів(aliases). Приклад - вміст файлу 'ifcfg -lo ':

```
DEVICE=lo
IPADDR=127.0.0.1
NETMASK=255.0.0.0
NETWORK=127.0.0.0
```

```
BROADCAST=127.255.255.255
ONBOOT=yes
NAME=loopback
```

— /etc/resolv.conf - вміщує список серверів імен (DNS – Domain Name Service), котрим можна посилати запроси.

Файл resolv.conf має формат:

```
search и'мя_домену …
nameserver ip-адреса
```

У директиві search вказаний список доменів, які слід опитувати, якщо ім'я комп'ютера виявляється не повністю визначеним. Сервери імен підлягають опитуванню в тому порядку, в якому задані директиви nameserver.

— /etc/hosts.conf - задає порядок перетворення імен і адрес, зазвичай файл hosts продивляється перед зверненням до DNS.

6.2.4 Утиліти для роботи з мережею

`hostname [им'я_комп'ютера]` – задати/продивитись ім’я комп'ютера.

`ifconfig [інтерфейс]`

`ifconfig інтерфейс [сімейство_адрес] адреса [опції] up | down`

Команда ifconfig використовується для підключення і відключення мережевого інтерфейсу, завдання його IP-адреси і маски підмережі, а також інших опцій і параметрів. Вона зазвичай виконується під час початкового завантаження, але може застосовуватися і для внесення змін до працюючої системи. Приклад завдання альтернативної адреси на першому адаптері Ethernet:

```
sh# ifconfig eth0:0 128.138.240.1 netmask 255.255.255.0 up
```

Якщо аргументи не передані, ifconfig видає інформацію про стан активних інтерфейсів. Параметр інтерфейсу – зазвичай це ім'я (чи псевдонім, визначуваний в /etc/modules.conf) драйвера, за яким йде номер пристрою, наприклад, eth0 для першого інтерфейсу Ethernet.

Ключове слово 'up' викликає активізацію інтерфейсу. Воно задається неявно при привласненні адреси інтерфейсу. 'down' викликає зупинку роботи драйвера для цього інтерфейсу. 'netmask' встановлює маску мережі IP для цього інтерфейсу. За умовчанням використовується звичайна маска мережі класу A, B або C(що визначається по IP- адресі інтерфейсу), але можна встановити потрібне значення маски.

`ping [опції] ім'я_хосту | адреса`

Служить для перевірки працездатності окремих комп'ютерів і сегментів мережі. Вона посилає ICMP- пакет ECHO_REQUEST конкретному комп'ютеру і чекає відповіді. Повідомлення команди ping про втрату пакетів говорить лише про те, що якийсь компонент (протокол або драйвер) мережі працює неправильно.

`traceroute [опції] ім'я_хосту | адреса`

Дозволяє встановити послідовність шлюзів, через які проходить IP-пакет на шляху до пункту свого призначення. Якщо ця команда не працює (чи працює занадто повільно), це може бути пов'язано з помилкою при

визначенні імені комп'ютера через DNS. Можна використати 'traceroute -n', вона виводить тільки IP-адреси шлюзів. traceroute використовує символ '*' при неможливості визначити час проходження пакету або при тайм-ауті. Деякі версії обмежують кількість хопов (вузлів) в маршруті за умовчанням, див. документацію для установки цього параметру а також для встановлення значення тайм-ауту для очікування пакетів.

Знаки оклику у виведенні команди сигналізують про серйозні проблеми:

— ! – порт недосяжний (ICMP_UNREACH_PORT) та ttl менше 2 (час життя пакету, ttl, обчислюється у хопах);
— !F – потрібна фрагментація (ICMP_UNREACH_NEEDFRAG);
— !P – помилка протоколу (ICMP_UNREACH_PROTOCOL);
— !H – комп'ютер недосяжний (ICMP_UNREACH_HOST);
— !N – мережа недосяжна (ICMP_UNREACH_NET);
— !S – помилка використання адресації відправника (ICMP_ UNREACH_SRCFAIL).

`route add [-net | -host] адреса [netmask маска] [gw шлюз] [metric N] [dev xxx]`

`route del [-net | -host] адреса` – виводить та дозволяє змінити статичні маршрути.

Параметр add дозволяє задати маршрут до хосту або мережі по вказаній адресі через вказаний шлюз.

Параметр metric задає лічильник переходів (вагу) маршруту.

`netstat [опції]` – засіб для моніторингу мережі TCP/IP. Вона дозволяє переглядати таблицю маршрутизації, стан активних мережевих з'єднань і корисні статистичні дані по кожному мережевому інтерфейсу. Ось найчастіше використовувані опції цієї команди :

— -i – інформація про інтерфейси;
— -nr – про маршрутизацію;
— -a – про всі очікуючі (відкриті) сокети.

ip
tc
iptables

6.2.5 Системні параметри ядра

Для того, щоб перевірити, чи дозволено проходження пакетів через Вашу машину:

```
cat /proc/sys/net/ipv4/ip_forward
```

Якщо не дозволено, увімкніть:

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Для того, щоб зробити ці зміни постійними, потрібно відредагувати файл /etc/sysctl.conf.

6.2.6 Дефрагментація пакетів

Іноді пакет занадто великий, щоб пересилатися одним кадром. В такому випадку пакет поділяється на фрагменти і надсилається як декілька пакетів. Вузол отримувача в такому випадку перед передачею пакету збирає пакет з частинок і тільки потім передає його в сокет. Проблема з фрагментами полягає в тому, що іноді міжмережевому екрану потрібно проаналізувати початок пакету, який міститься тільки в першому фрагменті.

Якщо Ваш вузол – єдине з'єднання з зовнішньою мережею, можна налаштувати ядро Linux таким способом, щоб воно збирало всі пакети, які проходять через нього:

```
echo "1" > /proc/sys/net/ipv4/ip_always_defrag
```

Таке налаштування було актуальним для попередніх версій ядра, сучасні ядра в разі потреби намагаються зібрати пакет автоматично.

6.3 Хід роботи

1. Визначте MAC- адреса мережевої плати на Вашій машині і її IP- адресу і маску.
2. Задайте нову адресу IP для своєї машини, заздалегідь прибравши стару. Необхідно виділити підмережу з мережі класу "C" 10.10.10.0 так, щоб клас не займав зайвого адресного простору, номери машин вибираються від входу за годинниковою стрілкою, шлюз повинен мати перший номер.
3. Налаштуйте статичну маршрутизацію з маршрутами мінімум на два шлюзи інтрамережі і Інтернет.
4. Додайте колишню IP- адресу в якості псевдоніма до нової IP- адреси. Доповніть, якщо це необхідно, таблицю маршрутизації.
5. Проглянете кеш протоколу ARP на Вашій машині.
6. Запустіть обмін ICMP пакетами з локальним сервером, параметри пакетів: розмір 2048 байт, в тілі пакету рядок: "пакет с машини №xx"
7. Увімкніть режим перехоплення пакетів і переконайтеся, що сусіди теж передають ICMP-пакети відповідного змісту. Для цього скористайтеся утилітою 'tcpdump'.
8. Запишіть в звіт карту відкритих мережевих з'єднань і серверних сокетів для Вашої машини.
9. Налаштуйте міжмережний екран таким чином:
   — Перевірити поточний стан пакетного фільтру
   — З'ясувати, як запускається netfilter у Вашому дистрибутиві і де він зберігає свої правила.
   — Встановити політику таблиць у нехтування пакетами
   — Дозволити всі вхідні пакети, пов'язані з вихідними з'єднаннями
   — Дозволити вхідні ICMP пакети з запитами ECHO.
   — Дозволити вхідні з'єднання на 22 TCP порт.
   — За допомогою ULOG журналювати вхідні пакети на 80 TCP порт.
   — Зробити ці зміни постійними.

6.4 Вміст звіту

Звіт повинен містити короткий опис використовуваних в роботі утиліт, а також детальний опис дій з виконання роботи (команди і результат їх виконання).

6.5 Контрольні запитання

1. Формат IP-адреси и маски, вибір мережевої маски для підмережі.
2. Призначення і синтаксис виклика утиліти ifconfig.
3. Призначення та відмінності протоколів ARP та RARP.
4. Принципи та протоколи маршрутизації.
5. Налаштування статичної маршрутизації за допомогою утиліти route.
6. Використання утиліти netstat.
7. Використання утиліти tcpdump.
8. Призначення та використання утиліти ip.
9. Призначення та використання утиліти tc.
10. Робота з міжмережевим екраном iptables.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. http://citforum.ru/operating_systems/unix/glava_12.shtml
2. https://help.ubuntu.com/community/Kernel/Compile
3. http://wiki.kryukov.biz/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D1%8B_%D1%8F%D0%B4%D1%80%D0%B0_Linux
4. http://kernelnewbies.org/LinuxChanges#head-82b14f7727c01c4fb5dee39d6eae42facd4bc047
5. http://www.ibm.com/developerworks/ru/edu/l-lpic2201/index.html
6. http://informst.ucoz.ru/publ/kak_sdelat_v_virtualbox_zagruzochnuju_fleshku/41-1-0-173
7. http://informst.ucoz.ru/publ/25-1-0-67
8. http://informst.ucoz.ru/publ/25-1-0-66
9. http://cs.stu.cn.ua/post/904/
10. http://en.gentoo-wiki.com/wiki/Build_Your_Own_LiveCD_or_LiveDVD
11. UNIX. Пособие системного администратора. / Пер. с англ. Под ред. д – К.: BHV, 2002 р.