МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ „ЛЬВІВСЬКА ПОЛІТЕХНІКА”

АДМІНІСТРУВАННЯ КОМП’ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

МЕТОДИЧНІ ВКАЗІВКИ ДО САМОСТІЙНОЇ РОБОТИ

для студентів першого (бакалаврського) рівня вищої освіти
спеціальності 123 “Комп’ютерна інженерія”

Затверджено
на засіданні кафедри
електронних обчислювальних машин.
Протокол № 1 від 28 серпня 2020 р.

Львів 2020

Адміністрування комп'ютерних систем і мереж: методичні вказівки до самостійної роботи для студентів першого (бакалаврського) рівня вищої освіти спеціальності 123 "Комп'ютерна інженерія" / Укл. М. О. Хомуляк – Львів: Видавництво Національного університету „Львівська політехніка", 2020. – 64 с.

Укладач — Хомуляк М. О., ст. викладач

Відповідальний за випуск — Мельник А.О., завідувач кафедри ЕОМ

Рецензенти — Ваврук Є. Я., канд. техн. наук, доцент,
Юрчак І. Ю., канд. техн. наук, доцент

ЗМІСТ

ВСТУП

Метою навчальної дисципліни „Адміністрування комп'ютерних систем і мереж" є засвоєння основних принципів системного адміністрування комп'ютерних мереж. Одержані при вивченні курсу знання необхідні для подальшої практичної діяльності спеціаліста і для засвоєння дисциплін „Засоби аналізу та управління мережами", „Адміністрування корпоративних інформаційних систем" та „Проектування засобів адміністрування комп'ютерних мереж".

В результаті вивчення дисципліни студент повинен знати методику встановлення та обслуговування сучасних мережних операційних систем та керування їх основними характеристиками, повинен вміти здійснювати адміністрування комп'ютерних систем і мереж та користуватись необхідними для цього апаратними і програмними засобами.

Відповідно до робочої навчальної програми дисципліни „Адміністрування комп'ютерних систем і мереж" передбачені такі види самостійної роботи студентів, як підготовка до лабораторних занять, виконання розрахунково-графічної роботи, підготовка до контрольної роботи, а також опрацювання додаткових теоретичних і практичних питань, що не увійшли до основного курсу:

– ознайомлення з функціонуванням текстових терміналів, послідовністю їх приєднання та налаштування;

– вивчення принципів об'єднання обчислювальних Linux-систем в TCP/IP-мережі;

– конфігурування ядра Linux для заданого периферійного оточення, вивчення класифікації периферійного обладнання обчислювальної системи та дисциплін обслуговування цього обладнання;

– ознайомлення з принципами логічної побудови твердих дисків на прикладі встановлення завантажувача операційних систем LILO;

– вивчення принципів з'єднання комп'ютерів послідовними каналами із застосуванням протоколів SLIP та PPP;

– ознайомлення з організацією служби Internet на Linux-системі, вивчення головних протоколів та послуг Internet, а також їх конфігурування;

– вивчення розподіленої файлової системи NFS для обчислювальних комплексів та конфігурування її на Linux-системі;

– ознайомлення з принципами організації поштової служби в Internet та налаштування демона sendmail.

Ці питання сприятимуть поглибленому засвоєнню навчальної дисципліни „Адміністрування комп'ютерних систем і мереж".

1. ПРИЄДНАННЯ ТЕКСТОВИХ ТЕРМІНАЛІВ ДО LINUX-СИСТЕМИ

Спочатку розглянемо фізичне приєднання текстових терміналів до Linux-системи. Оскільки текстовий термінал не має своєї IP-адреси (на відміну від X-терміналів), тому не можна підключати декілька текстових терміналів до однієї лінії (декілька X-терміналів підключають до основної ЕОМ через карту Ethernet за допомогою одної лінії). Таким чином, кожний текстовий термінал (надалі просто термінал) під'єднується до одного послідовного порту головної ЕОМ.

Стандартно в PC-машині є чотири порти (від COM1 до COM4). Якщо треба підключити не більше чотирьох терміналів, то тут особливих проблем не виникає. Взаємодія між ядром та терміналами відбувається через переривання (взаємодія через переривання дозволяє економити час центрального процесора порівняно зі ситуацією, коли введення-виведення відбувається за допомогою опитування). Кожному порту, а, вочевидь, і терміналу, відповідає своє переривання.

Зазвичай виникає необхідність у використанні більше терміналів, ніж чотирьох, і тоді стандартних послідовних портів PC-машини не вистачає. У такому випадку використовують спеціальні мультипортові карти. Найбільш поширеними є 8- та 16-портові карти. Загальна структурна схема приєднання терміналів показана на рис.1.

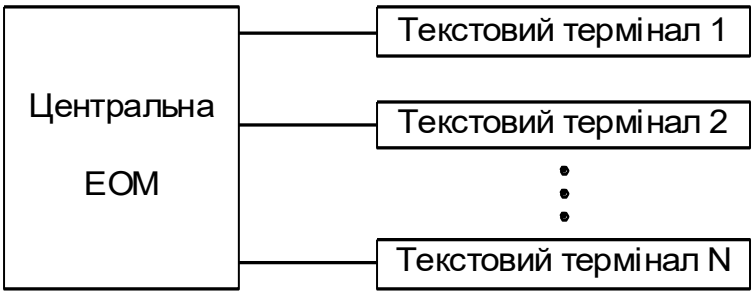

Рис.1. Схема приєднання текстових терміналів до Linux-системи.

Для обміну інформацією між ЕОМ та терміналами використовується радіальний дуплексний асинхронний послідовний інтерфейс RS-232 (струмова петля). Струмова петля забезпечує більшу завадозахищенність у порівнянні з іншими інтерфейсами. Як видно з рис.1, до кожного термінала йде окрема лінія, яка підключається до одного із портів 8-портової карти.

Мультипортовій карті відводиться тільки один номер переривання. Для того, щоб центральний процесор міг знати, з якого термінала прийшло переривання, в спеціальний регістр на карті записується номер лінії, з якої прийшло переривання (таких ліній вісім). Таким чином, ЦП, отримавши переривання, читає вміст цього регістра, а потім, знаючи номер лінії , із спеціального буфера даних читає дані, що прийшли, та обробляє їх. Кожну лінію обслуговує окремий мікроконтролер, який і має свій буфер даних. Розмір цього буфера залежить від конкретної мікросхеми. В персональних комп'ютерах інтерфейс RS-232 керується чипом 16450 UART (Universal Asynchronous Receiver/Transmitter, Унівесальний Асинхронний Приймач/Передавач УАПП), який постачається від фірми National Semiconductor, або новішою версією 16550A. Основна відмінність між УАПП 16450 та 16550 полягає в тому, що останні мають буфер FIFO на 16 байтів, в той час, як перші - тільки 1-байтовий

буфер. Це робить чипи 16450 застосовними для швидкостей до 9600 Бод, в той час, як більші швидкості вимагають 16550 - сумісних чипів. Якщо в терміналі відключений режим локального еха, тоді ЦП відповідає за відображення введених символів. Для цього символи, що прочитані з буфера центральним процесором, додатково посилаються назад на термінал.

Обмін між комп'ютером та терміналами відбувається з використанням escape-послідовностей. За інтерпретацію escape-послідовностей, що надходять в термінал, за обслуговування клавіатури та відображення символів відповідає спеціальний мікроконтролер термінала. Найбільш вживаними є мікроконтролери i8048, i8051 (аналоги К1816ВЕ48, К1816ВЕ51 відповідно). Структурна схема термінала представлена на рис.2.

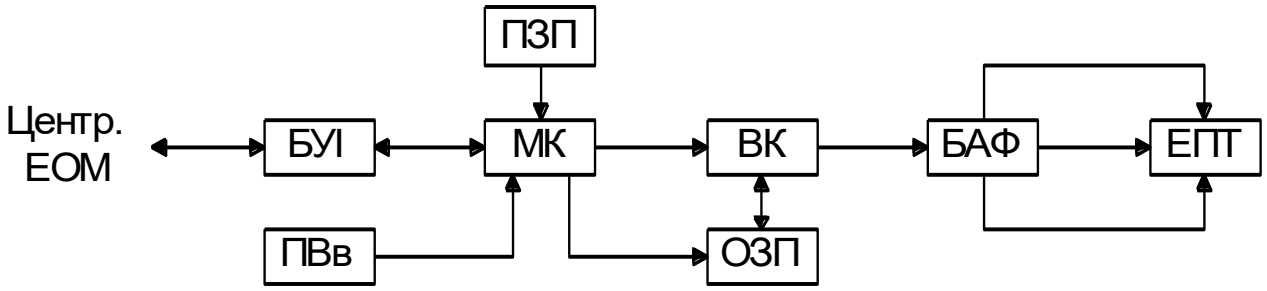

Рис.2. Структурна схема текстового термінала

На рис.2 прийняті такі скорочення:

БУІ - блок узгодження з інтерфейсом. Блок забезпечує перетворення сигналів з логічних ТТЛ-рівнів в сигнали струмової петлі.

МК - мікроконтролер, який виконує всі функції керування терміналом.

ПЗП - постійний запам'ятовуючий пристрій. Зберігає всі установки термінала.

ВК - відеоконтролер.

ОЗП - оперативний запам'ятовуючий пристрій, який вміщує інформацію, що треба відображати.

БАФ - блок аналогових формувачів для ЕПТ (електронно-променевої трубки).

ПВв - пристрій вводу (клавіатура).

Розглянемо, які дії необхідно виконати, щоб Linux міг працювати з терміналами.

Як всі пристрої в Linux-системі, послідовні порти доступні через спеціальні файли пристроїв, що розміщені в директорії /dev. Тут є два різновиди файлів-пристроїв, пов'язаних з послідовними драйверами, і для кожного порта є один файл пристрою від кожного із різновидів. Залежно від файлу пристрій буде поводитись по-різному.

Перший різновид використовується кожного разу, коли порт застосовується для вводу (dialing in); він має головний номер 4, і файли називаються ttyS0, ttyS1 і т.д. (це абревіатура від Teletype (tm), який колись був одним із основних виробників терміналів "на зорі" UNIX).

Другий різновид використовується, коли виконується вивід (dialing out) з порту; файли називаються cua0 і т.д. і мають головний номер 5.

Для того, щоб термінал запрацював, потрібна обслуговуюча програма. Вона задається в файлі inittab (директорія /etc). Коли стартує Linux, запускаються getty (так звані демони). Getty - це драйвери низького рівня, які

обслуговують струмову петлю терміналів. Файл inittab є конфігураційним файлом для процесу init.

Процес init є диспетчером процесів, він породжує всі інші процеси. Інструкції про те, які процеси треба створити, читаються процесом init з файлу /etc/inittab. Рядки цього файлу включать в себе таке:

1. Ідентифікатор стану id - який режим: однокористувацький, багатокористувацький, багатозадачний;
2. Дію, яка виконується;
3. Специфікацію програми, яка реалізує цю дію.

Приклад частини файлу inittab для ініціалізації вісьмох терміналів:

```
# Serial lines (коментар)
s1:45:respawn:/sbin/agetty -L 19200 ttyS1 vt220
s2:45:respawn:/sbin/agetty -L 19200 ttyS2 vt220
s3:45:respawn:/sbin/agetty -L 19200 ttyS3 vt220
s4:45:respawn:/sbin/agetty -L 19200 ttyS4 vt220
s5:45:respawn:/sbin/agetty -L 19200 ttyS5 vt220
s6:45:respawn:/sbin/agetty -L 19200 ttyS6 vt220
s7:45:respawn:/sbin/agetty -L 19200 ttyS7 vt220
s8:45:respawn:/sbin/agetty -L 19200 ttyS8 vt220
```

В agetty (agetty-це альтернативна версія getty) вказується швидкість передачі (19200 Бод і т.п.); далі вказується, на який пристрій (це мається на увазі з каталогу /dev), тобто tty; тип термінала (vt220). Тип термінала запам'ятовується в системній змінній оточення TERM, яка надалі використовується системою, щоб працювати з терміналом. Respawn (це опція процесу init) вказує на те, що якщо цей процес (agetty) хтось вб'є, чи може з ним щось станеться, він буде перезапущений автоматично (без участі людини), щоб зв'язок з терміналом був постійно.

У кожного термінала є багато параметрів. Головні параметри: швидкість передачі в бодах (якщо центральний комп'ютер настроєний на одну швидкість, а термінал - на іншу, то зв'язку не буде), режим локального eха (відлуння), число байтів у посиланні і т.і. Для того, щоб бути впевненим, що ядро бачить термінал, локальне exo треба вимикати. Цей режим зазвичай застосовують у період встановлення та налагодження терміналів. Щоб подивитись параметри термінала (наприклад, ttyS5), треба задати команду:

```
stty </dev/ttyS5
```

Параметри термінала можна встановити за допомогою agetty (тобто вказати параметри в файлі inittab) під час запуску системи, або за допомогою команди stty під час роботи Linux (але це дозволяється робити лише суперкористувачу).

Наступний крок - треба вказати ядру Linux, які мультипортові карти ми викоистовуємо, провести прив'язку пепреривання. Це все конфігурується в rc.serial (послідовні пристрої). Файл rc.serial міститься у директорії /etc/rc.d- це є скрипт конфігурації послідовних ліній. Тут вже є заготовлені конфігурації для всіх відомих карт, що випускаюься (вони закоментовані, коментар йде після знаку '#'). Якщо купується відома карта, то просто треба зняти коментар, що стосується карти. В іншому разі треба (за певними правилами) вказати порти, до яких приєднуються термінали, типи контролерів, номер переривання тощо.

Приклад із файлу rc.serial:

```
${SETSERIAL} /dev/cua4 uart 8250 port 0x280 irq 5 ^fourport
${SETSERIAL} /dev/cua5 uart 8250 port 0x288 irq 5 ^fourport
${SETSERIAL} /dev/cua6 uart 8250 port 0x290 irq 5 ^fourport
${SETSERIAL} /dev/cua7 uart 8250 port 0x298 irq 5 ^fourport
${SETSERIAL} /dev/cua8 uart 8250 port 0x2A0 irq 5 ^fourport
${SETSERIAL} /dev/cua9 uart 8250 port 0x2A8 irq 5 ^fourport
${SETSERIAL} /dev/cua10 uart 8250 port 0x2B0 irq 5 ^fourport
${SETSERIAL} /dev/cua11 uart 8250 port 0x2B8 irq 5 ^fourport
```

Вказані тип та номер мікросхем, що використовуються в мультипортовій карті (uart 8250), адреси портів, номер апаратного переривання (irq5), кількість портів (^fourport - карта не чотирипортова, тобто має 8 портів), який файл асоціюється з виводом на даний термінал (наприклад, /dev/cua4).

Для остаточного налаштування системи в файлі termcap треба вказати, які escape-послідовності яким діям (тобто скролінг, переміщення курсора і т.і.) відповідають. В файлі termcap є багато заготовок для різних типів терміналів, таким чином, треба зняти коментар, що відноситься до встановленого термінала. Якщо даних на термінал, шо встановлюється, в файлі termcap немає, то необхідно внести даний термінал в termcap (escape - послідовності, що відповідають даному терміналу, мають наводитись в інструкції до нього).

Файл /etc/termcap - це ASCII-файл, який містить можливості терміналів різних типів. Програми можуть читати termcap, щоб знайти індивідуальні escape-коди, які необхідні для керування відео-атрибутами використовуваного термінала. (Інші характеристики термінала керуються stty-ом.).

Елементи (точки входу, entries) termcap-а повинні бути визначені в одному логічному рядку з символом '\', який використовується для відокремлення нового рядку. Поля розділені символом ':'. Перше поле кожної точки входження починається з лівого краю та містить список назв терміналів, які розділяються символом '|'.

Перше підполе містить коротке ім'я з двох символів. Це "реліквія" системи V6, де існує база даних шириною 16 біт (це еквівалентно до двох символів). Це ім'я може містити великі і малі літери. Друге підполе містить ім'я, яке використовується змінною оточення TERM. Воно має бути написане літерами нижнього регістра. Третє підполе містить довге та описове ім'я для цієї точки входу в termcap.

Наступні поля містять набір дескрипторів (описувачів), у яких міститься інформація про функціональні можливості термінала, способах редагування інформації на екрані термінала та опис клавіатури (додаток 1). Кожний з подальших рядків можливостей має відокремлюватись (розпізнаватись) однією табуляцією (TAB) від лівої межі екрану.

Хоча немає строго визначеного порядку, але передбачається, що спочатку записані логічні (boolean), потім цифрові (numeric), і наприкінці рядкові (string) можливості, кожна просортована в алфавітному порядку без врахування нижнього чи верхнього регістра. Можливості подібних функцій можуть бути записані в один рядок. Наприклад:

```
vt200|vt220|dec-vt220|vt200-js|vt220-js|dec vt200 series:\
```

```
:im=\E[4h:ei=\E[4l:mi:dc=\E[P:dm=:ed=:al=\E[L:\
:cs=\E[%i%d;%dr:sf=\ED:sr=\EM:sb=\EM:\
:ce=\E[K:cl=\E[H\E[J:cd=\E[J:cm=\E[%i%d;%dH:nd=E[C\
:up=\E[A:\:so=\E[7m:se=\E[27m:us=\E[4m:ue=\E[24m:\
:md=\E[1m:mr=\E[7m:mb=\E[5m:me=\E[m:\
:is=\E>\E[?3l\E[?4l\E[?5l\E[?7h\E[?8h\E[1;24r\E[24;1H:\
:rs=\E>\E[?3l\E[?4l\E[?5l\E[?7h\E[?8h:\:tc=vt100:
```

Getty встановлює режим термінала, швидкість і "дисципліну лінії". Getty - це друга з трьох програм (init(1m), getty(1m), login(1m)), які використовуються системою для організації входження користувачів. Getty викликається програмою init(1m), щоб:

1. Відкрити лінії терміналів і встановити їхні режими.
2. Надрукувати запрошення для вводу і отримати ім'я користувача.
3. Розпочати процес входження користувача.

Спочатку getty розбирає свій командний рядок, якщо не було помилок, getty проглядає свій файл заготовок (defaults file), зазвичай це /etc/default/getty, щоб визначити деякі значення (etc/conf.getty компілюється з опцією FSSTND). Значення в файлі заготовок (назву якого можна змінити опцією -d defaults_file) є пріоритетнішими, ніж у командному рядку. Потім getty відкриває лінію читання та запису і відключає буферизацію стандартного вводу-виводу. Якщо була зазначена ініціалізація, вона виконується.

Після ініціалізації лінія закривається та знову відкривається, але цього разу в блокованому режимі, тому з пристроями нема зв'язку. виявлення носія сигналу (carrier signal) дозволяє лінії відкритися.

Потім getty друкує повідомлення при вводі (зберігається звичайно у файлі /etc/issue) і запрошення на ввід. Нарешті, getty зчитує вхідне ім'я користувача і викликає login (1m), якому передає це ім'я як аргумент. Під час читання імені getty намагається адаптувати систему до швидкості термінала, а також встановлює деякі параметри термінала (див. termio (7)) для узгодження з процедурою входження користувача.

Термінал, який використовує getty, визначається аргументом line. Getty використовує /dev/line як назву пристрою, до якого він підключається. Доки getty не буде викликано з прапорцем -h (або в файлі заготовок не буде зазначено HAHGUP=NO), це викличе зависання на лінії, бо швидкість встановиться рівною 0. Задання -r delay в командному рядку (або WAITCHAR=YES у файлі заготовок) примусить getty очікувати один символ з лінії, після якого він чекатиме delay секунд перш, ніж продовжити. Якщо небажана затримка, слід використати -r0. Задання -w waitfor в командному рядку (WAITFOR=waitfor в файлі заготовок) примусить getty очікувати з лінії визначений символьний рядок. Задання -t timeout (TIMEOUT=timeout) примусить getty завершити роботу, якщо через timeout секунд після друку запрошення на ввід не буде введено жодного імені користувача.

Аргумент speed – це мітка запису в файлі etc/gettydefs (див.gettydefs (4)). Цей запис визначає для getty початкову швидкість і установки термінала, запрошення на ввід, кінцеву швидкість і установки та покажчик на інший запис на той випадок, якщо користувач виявить, що швидкість є некоректною. Це

виконується надсиланням символу <break> (в дійсності послідовності символів). За певних обствин повернення каретки також виконує цю функцію. Зазвичай це трапляється у випадку, коли getty налаштовується на вищу швидкість, ніж модем чи термінал. Getty послідовно переглядає файл gettydefs, шукаючи відповідний запис. Якщо аргумент speed не задавався чи не можна знайти запис, як поточне значення береться перший запис з gettydefs. Якщо ж до gettydefs немає доступу, поточним є значення, встановлене під час компіляції.

Аргумент type - це рядок, що визначає назву термінала, під'єднаного до лінії. Він має бути коректним іменем термінала, що є в базі даних termcap. Getty використовує це значення, щоб визначити,як очищувати дисплей, а також присвоює змінній оточення TERM це значення.

Аргумент lined визначає тип протокола лінії, за замовченням є LDISC0.

Як зазначалося вище, getty друкує запрошення на введення імені та читає вхідне ім'я користувача. Якщо прийнято символ NULL, припускається, що користувач натиснув <break> або Return, щоб показати, що швидкість є неправильною. Це примушує getty шукати наступний запис speed в /etc/gettydefs.

Ім'я користувача завершується символами CR або нового рядку, якщо було натиснуто CR, вони в системі відображаються у символи нового рядка. Ім'я користувача перевіряється на те, чи не містить воно лише великі літери, якщо так, то система налаштовується на відображення в майбутньому всіх верхніх літер в нижні.

Опція перевірки забезпечує тестування файлу gettydefs. Коли getty викликається з опцією -c gettydefs, він сканує зазначений файл gettydefs і видруковує на стандартний вивід всі значення, які бачить. Якщо трапляються помилки розбору (через синтаксичні помилки в gettydefs файлі), про них видаються повідомлення.

2. ОБ'ЄДНАННЯ ОБЧИСЛЮВАЛЬНИХ LINUX-СИСТЕМ В TCP/IP-МЕРЕЖІ

Для функціонування в мережі Internet кожна ЕОМ повинна мати свою IP-адресу. Є три категорії мереж: A,B,C. Об'єднання підмереж у мережі має вигляд дерева, наприклад:

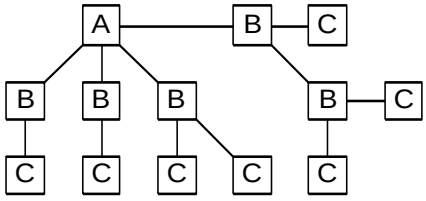

Рис. 1

Кожна машина має свою IP-адресу: XX.XX.XX.XX, де X-шістнадцяткова цифра, тобто всього - 32 розряди. Категорії A,B,C мають такі адреси в десятковому вигляді:

A - від 1.0.0.0 до 127.0.0.0,
B - від 128.0.0.0 до 197.255.0.0,
C - від 192.0.0.0 до 223.255.255.0.

Кожна підмережа має свою мережну маску. Маска вказує на ті розряди в адресі, які для даної підмережі залишаються незмінними:

A - 255.0.0.0,
B - 255.255.0.0,
C - 255.255.255.0.

Приклад організації мережі категорії B:
GW - IP-адреса машини, до якої буде підключена підмережа, SubNet - IP-адреса підмережі, IPM - маска, BC - broadcast - адреса для загальної посилки повідомлення.

GW : 140.100.0.1
SubNet : 140.100.0.0
IPM : 255.255.0.0
BC : 140.100.255.255

Приєднання Linux-системи до мережі TCP/IP виконується двома утилітами:

1. Утиліта ifconfig налаштовує унікальну IP-адресу кожному інтерфейсу системи. Інтерфейсами можуть бути eth0...ethN (Ethernet), arc0...arcN (Arcnet), ppp0...pppN (Point-to-Po int), sl0...slN (Serial Line) та ін.

2. Утиліта route для статичної маршрутизації IP-пакетів та утиліта gated (gate deamon) для динамічної маршрутизації IP-пакетів. В завдання цих утиліт входить розподіл потоку IP-пакетів за декларованими в ifconfig інтерфейсами з врахуванням IP-масок. При динамічній маршрутизації цей розподіл є більш гнучким. В даній роботі розглядається лише статична маршрутизація.

Виділена машина, на якій виконується лише програма route, має назву роутер або маршрутизатор. Запуск утиліт ifconfig та route відбувається при завантаженні системи з файлу /etc/rc.d/rc.inet1.

Наведемо приклад конфігурації підмереж для кафедри ЕОМ, що має такий вигляд (фрагмент):

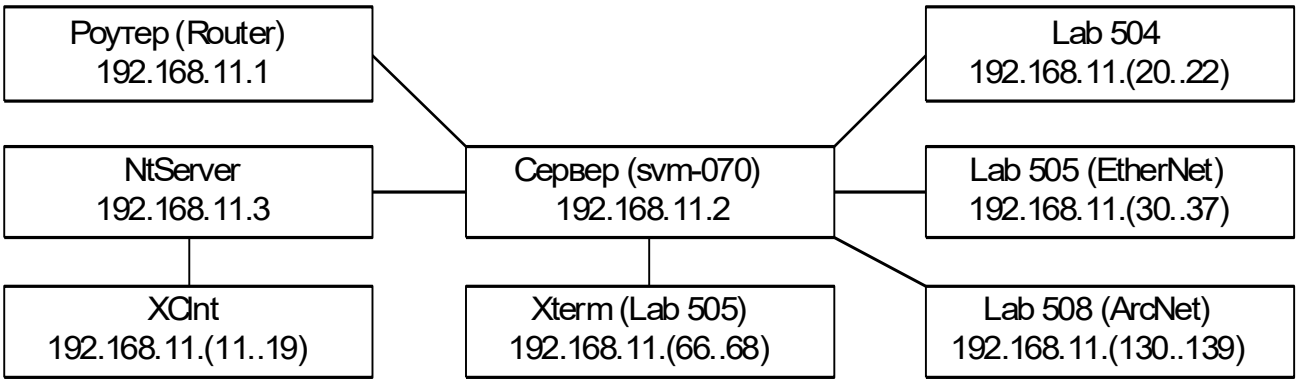

Рис.2. Конфігурація підмереж кафедри ЕОМ

Відповідний файл /etc/rc.d/rc.inet1 має вигляд:

```
# Петля на самого себе
/sbin/ifconfig lo 127.0.0.1
/sbin/route add -net 127.0.0.0 netmask 255.0.0.0 dev lo

IPADDR="192.168.11.2"       # Хост
IPADDR1="192.168.11.193"    #      Резервована      підмережа
(254...194)
IPADDR2="192.168.11.129"    # підмережа arcnet (191...130)
IPADDR3="192.168.11.65"     # Підмережа XTerm (127...66)
NETMASK="255.255.255.0"     # Хост маска
NETMASK1="255.255.255.192"  # Підмережева маска
NETWORK="192.168.11.0"      # Мережа
NETWORK1="192.168.11.192"   # Резервована підмережа
NETWORK2="192.168.11.128"   # Підмережа arcnet
NETWORK3="192.168.11.64"    # Підмережа XTerm

 BROADCAST="192.168.11.255" # broadcast

 GATEWAY="192.168.11.1"     # Gateway

/sbin/ifconfig  eth0  ${IPADDR}  broadcast  ${BROADCAST}  \
netmask ${NETMASK1}
/sbin/ifconfig  arc0  ${IPADDR2}  broadcast  ${BROADCAST}  \
netmask ${NETMASK1}
/sbin/ifconfig  arc0e  ${IPADDR2}  broadcast  ${BROADCAST}  \
netmask ${NETMASK1}
/sbin/ifconfig  eth1  ${IPADDR3}  broadcast  ${BROADCAST}  \
netmask ${NETMASK1}

/sbin/route add -net ${NETWORK} netmask ${NETMASK1} dev eth0
/sbin/route  add  -net  ${NETWORK2}  netmask  ${NETMASK1}  dev
arc0e
/sbin/route add -net ${NETWORK3} netmask ${NETMASK1} dev eth1
/sbin/route  add  -net  224.0.0.0  netmask  240.0.0.0  dev  eth0
/sbin/route add default gw ${GATEWAY} metric 1
```

1. Утиліта ifconfig (sbin/ifconfig) має такий синтаксис:
   `ifconfig [interface]`

```
ifconfig interface [aftype] options | address
```

Використовується для встановлення резидентних в ядрі мереже вих інтерфейсів. Він використовується під час завантаження для конфігурування більшості з них до завантажувального стану.

Якщо не дано аргументів, ifconfig лише відображає статус поточно визначених інтерфейсів.

Якщо заданий один аргумент, він відображає статус тільки даного інтерфейсу. Інакше він присвоює задані установки.

Якщо перший аргумент після назви інтерфейсу розпізнаний як ім'я підтримуваного сімейства адрес, то воно використовується для декодування і відображення всіх протокольних адрес. По точно підтримувані адресні сімейства включають inet (TCP/IP, за замовчуванням) і ax25 (AMPR Packet Radio).

- interface - назва NET інтерфейсу. Це звичайно ім'я типу wd0,sl3 або що-небудь подібне до такого: імя драйвера пристрою після номеру модуля.
- up - цей прапорець вказує активізований інтерфейс. Він як правило специфікований коли інтерфейсу надається нова адреса.
- down - цей прапорець ідентифікує драйвер цього інтерфейсу для виконання Shut Down, і є корисним при некоректному стартуванні.
- [-]arp - дозволяє чи забороняє використання ARP протоколу в цьому інтерфейсі. Якщо є знак мінус, прапорець встановлений в OFF.
- [-]trailers - дозволяє чи забороняє використання трейлерів в Ethernet frames. Він не використовується в поточній реалізації NET.
- [-]allmulti - дозволяє чи забороняє режим інтерфейсу promis cuous. Це є засоби отримання фреймів що надіслані до мереже вого шару системного ядра, дозволених для мережевого керування.
- metric N - цей параметер встановлює інтерфейсну міру. Це не використовується зараз, але реалізовано для майбутнього.
- mtu N - цей параметер встановлює Maximum Transfer Unit (MTU) інтерфейсу. Для Ethenet, це є число в межах від 1000-2000 (за замовчуванням є 1500). Для SLIP використовується десь між 200 і 4096. Зауважимо, що поточна реалізація не керується IP фрагментацією, так що краще зробити MTU досить великим.
- dstaddr addr - встановлює "інший кінець" IP адреси в виборі звязку від вузла до вузла, подібно PPP.
- netmask addr - встановлює IP мережеву маску для цього інтерфейсу. Це значення за замовчуванням для класів A,B або C мережевих масок (взято з інтерфейсних IP адрес), але тут може бу ти встановлене будь-яке значення для використання в підмережах.
- [-]broadcast [addr] - якщо адресний аргумент встановлює адресу протоколу повідомлення для цього інтерейсу. Інакше, він лише встанолює IFF_BROADCAST-прапорець інтерфейсу. Якщо перед ним є знак мінус, тоді прапорець очищується.
- [-]pointopoint [addr] - дозволяє інтерфейсний режим point-to-point, шляхом прямого зв'язку між двома машинами без "прослуховування" іншими. Якщо

аргумент адреси є присутній, то встановлює протокольну адресу іншої сторони зв'язку. Інакше тільки встановлює IFF_POINTTOPOINT-прапорець інтерфейсу. Якщо є мінус, прапорець очищується.

- hw - встановлює апаратну адресу цього інтерфейсу, якщо драйвер пристрою підтримує цю операцію. Це ключове слово має бути наступним за іменем апаратного класу з можливістю видрукування ASCII-еквіваленту апаратної адреси. Поточно підтримувані апаратні класи включають ether (Ethernet), ax25 (AMPR AX.25), ARCnet.
- address - ім'я хосту або IP адреси (ім'я хосту буде перетворене в IP адресу) інтерфейсу. Цей параметр є обов'язковим, хоча синтаксис не вимагає цього.

2. Утиліта route (sbin/route) має такий синтаксис:

```
route [ -vn ]
route [ -v ] add [ -net | -host ] XXXX [gw GGGG]
[metric MMMM] [netmask NNNN] [mss NNNN] [window NNNN]
[dev DDDD]
route [ -v ] del XXXX
```

Route модифікує таблицю IP перетворень ядра. Найперше використовується для встановлення статичних маршрутів до специфікованих хостів або мережевих вузлів після того, як вони були сконфігуровані програмою ifconfig.

Без опцій route друкує таблицю маршрутизації ядра, перераховуючи адреси призначення, шлюзи, мережні маски для маршрутів ("Genmask"), прапорці (U = Up, H = Host, G = Gateway, D = dynamic, M = Modified), Metric Ref, Use and Iface (тобто які вибирають карту маршрутів до них).

- -n - так само як попередня, але показуються цифрові адреси замість намагання визначити символічні імена хосту. Це є корисним якщо ви намагаєтесь визначити чому маршрут до вашого сервера зник.
- -v це прапорець для verbose.
- del XXXX - витирає маршрути асоційовані з адресою призначення XXXX.
- add [-net | -host] XXXX [gw GGGG] [metric MMMM] [netmask NNNN] [window WWWW] [dev DDDD] - додає маршрут до IP адреси XXXX. Маршрут є мережевим маршрутом, якщо: а) -net модифікатор використаний або б) XXXX є створений в /etc/networks бібліотекою функцій getnetbyname() і не використовується модифікатор -host.
- gw GGGG аргументи означають що будь-які IP пакети послані за цією адресою будуть направлені через специфікований шлюз. За уваження: специфікований шлюз повинен бути спочатку доповнений. Це є звичайний засіб який є для встановлення статичних маршрутів до шлюзів наперед.
- Модифікатор netmask NNNN специфікує мережеву маску доданого маршруту. Це має сенс тільки для мережевого маршруту, і коли адреса XXXX має сенс з специфікованою маскою. Якщо мережева маска не задана, то route намагається її вгадати, тому для нормальних установок немає потреби задавати маску.

- Модифікатор mss NNNN специфікує TCP mss для доданого маршру ту. Це як правило використовується тільки для доброї оптимі зації установок маршрутизації.
- Модифікатор window NNNN специфікує TCP вікно для доданого маршруту. Це як правило використовується для мереж AX.25 і з драйверами здатними керувати back to back фреймами.
- Модифікатор dev DDDD змушує маршрут сполучатись з визначеним пристроєм, інакше ядро саме намагається його визначити (конт ролем кожного існуючого маршруту і специфікацій пристроїв). Якщо dev DDDD є останньою опцією в командному рядку, то слово dev може бути пропущене, оскільки це значення за замовчуванням. Інакше порядок модифікаторів маршрута (метричний - Net mask - gw - dev) не має значення.

Приклади:

```
route add -net 127.0.0.0
```

Додає нормальный петельний вхід, використовуючи netmask 255.0.0.0 (Класс А мережі, визначеної з адреси призначення звязаний з Lo пристроєм (Цей пристрій був раніше встанов лений коректно ifconfig).

```
route add -net 192.56.76.0 netmask 255.255.255.0 dev eth0
```

Додає маршрут до мережі 192.56.76.x через eth0. Модифіка тор netmask класу С насправді не потрібний тут, тому що 192.* є IP адресою класу С. Слово dev може бути тут опущене.

```
route add default gw mango-gw
```

Додає заданий за замовчуванням маршрут (який може бути ви користаний, якщо нема інших варіантів маршруту). Всі паке ти, що використовують цей маршрут, будуть пропущені через "mango-gw". Пристрій який буде фактично використовуватись для цього маршрута, залежить від того як можна досягнути "mango-gw" - статичний маршрут до "mango-gw" повинен бути встановлений раніше.

```
route add ipx4 sl0
route add -net 192.57.66.0 netmask 255.255.255.0 gw ipx4
```

Ця послідовність команд додає маршрут до хосту "ipx4" через SLIP інтерфейс (приймаючи що "ipx4" є SLIP хостом). А потім додає мережу 192.57.66.0, що буде шлюзуватись через цей хост).

3. КОНФІГУРУВАННЯ ЯДРА LINUX ДЛЯ ЗАДАНОГО ПЕРИФЕРІЙНОГО ОТОЧЕННЯ

Наведемо приклад встановлення ядра операційної системи Linux з конфігуруванням периферійного обладнання:

Спочатку слід перенести новий архів ядра у директорію /usr/src. Потім за допомогою команд

```
cd /usr/src
gzip -cd linux-2.0.XX.tar.gz | tar xfv
```

розархівувати цей файл. Тут "XX" - номер нової версії ядра.

Linux надає можливість замінити ядро в системі двома шляхами:

– через patch-релізи, які містять лише необхідні зміни, які треба внести в попередні версії для поновлення,

– інсталювати повністю нове ядро.

Для першого способу необхідно виконати такі дії:

```
cd /usr/src
gzip -cd patchXX.gz | patch -p0
```

Заміну можна здійснити послідовно самому, чи за допомогою спеціально створеного скрипт-файлу:

```
cd /usr/src
linux/scripts/patch-kernel
```

За замовчуванням директорія для вихідних (source) кодів – /usr/src/linux. Вона задається як перший аргумент, а при потребі другим аргументом можна задати альтернативну директорію.

Далі необхідно виконати символьне зв'язування вихідних кодів ядра:

```
cd /usr/include
rm -rf linux
rm -rf asm
ln -s /usr/src/linux/include/linux linux
ln -s /usr/src/linux/include/asm-i386 asm
```

Після цього перевіряємо, чи маємо не старі .old-файли і продовжуємо:

```
cd /usr/src/linux make mrproper
```

Тепер ми маємо коректно інстальовані вихідні коди ядра.

Linux має три альтернативні команди конфігурування:

- "make config" - основна програма конфігурування ядра, в якій реалізовано інтерфейс з користувачем типу "питання-відповідь" (Q&A);
- "make menuconfig" - через текстове кольорове меню;
- "make xconfig" - через спеціальну програму під X windows.

Всі способи дозволяють досягти однакової мети, тому для прикладу доцільно розглянути один з них – текстове меню.

Вся система меню побудована у вигляді дерева, яке динамічно змінює свою конфігурацію згідно з вибраними раніше опціями. В меню організована система допомоги (клавіша H), де можна докладніше дізнатись про ту чи іншу опцію. Нижче наведені короткі пояснення щодо більшості з них:

- Code maturity level options: Користувач має можливість вибрати, чи заносити в ядро драйвери чи інші коди, які ще не пройшли повного тестування і їх коректність роботи не гарантується. Це стосується підтримки деяких файлових систем, мережних протоколів, драйверів (особливо до нових пристроїв) тощо.

- Loadable module support: Дозвіл підтримки модулів, що підвантажуються у ядро динамічно (чи вивантажуються з ядра) – команди insmod та rmmod. Це стосується драйверів пристроїв, файлових систем тощо.

Чи запам’ятовувати інформацію про версії модулів. Це необхідно для можливого їх поновлення чи перекомпіляції ядра.

Чи має ядро підтримувати демони. Таким чином ядро подбає про те, щоб завантажувані модулі автоматично вивантажувались, якщо вони не використовуються. До того ж відпадає необхідність використовувати команду rmmod.

- General Setup:

Емуляція математичного процесора.
Підтримка мереж.
Обмеження пам'яті до 16 М. (Якщо є сумніви в коректній роботі "материнської плати" при більшому об'ємі ОЗП).
Підтримка шини PCI.
Підтримка стандарту System V, IPC, який надає змогу процесам синхронізуватись та обмінюватись інформацією.
Підтримка ядром виконуваних і об'єктних файлів типу a.out, що використовувався в ранніх версіях UNIX.
Підтримка ядром виконуваних і об'єктних файлів у форматі ELF, що використовується в різних архітектурах і ОС.
Вибір типу процесора.
Вибір типу компіляції ядра (a.out чи ELF).
- Floppy, IDE and other block devices: Надається можливість вибрати підтримку ядром таких пристроїв:
Звичайний гнучкий диск.
EIDE/MFM/RLL - диски, стрічки тощо.
IDE/ATAPI CD-ROM і стримери.
PCMCIA.
Підтримка чипсетів CMD640, 430FX (Triton), RZ1000, та ін. (Необхідно для виправлення помилок у деяких з них).
Loopback device support - доступність цієї опції надає мож ливість монтувати файли як файлові системи.
Multiple devices driver support - цей драйвер дозволяє комбінувати декілька розділів твердого диска в один логічний блок.
RAM disk support - надається можливість згодом частину ОЗП організувати як блок-пристрій, тобто створити файлову систему на ньому, читати/писати на нього, наприклад, як на звичайний твердий диск.
XT harddisk support - прідтримка 8-ми бітових контролерів твердих дисків.

- Networking options:

Networking firewalls - організовується на спеціальному комп'ютері для перевірки всіх пересилань між певною локальною мережею і "зовнішнім світом".
Networking aliasing - встановлення різних IP адрес для то тожніх низькорівневих мережних драйверів пристроїв.
TCP/IP networking - підтримка протокола, який використовується в Internet та більшості локальних мереж.
The IPX protocol - підтримка протоколу мереж Nowell, який зазвичай використовується для локальних мереж Windows-машин.
Appletalk DDP - протокол обміну між Apple-машинами.
Amateur Radio AX.25 Level 2 - протокол зв'язку між комп'ютерами через аматорське радіо.
Kernel/User Network link driver - цей драйвер надає можливість двома шляхами зв'язуватись деяким частинам ядра з модулями і процесами користувача.

- SCSI support: Пропонується користувачу вибрати драйвери певних пристроїв як звичайних, так і для паралельного порту.
SCSI-ЖД, стримери, CD-ROM, сканери тощо.
Можливість ядра звітувати при помилках у SCSI-пристроях.
Probe all LUNs on each SCSI device - всі SCSI-пристрої, в системі повинні мати унікальний номер (наприклад: HD=1, CD-ROM=5). Ці номери виставляються за допомогою ключів на пристроях.
SCSI low-level drivers - опція необхідна у разі використан ня спеціальних драйверів SCSI-пристроїв, які поставляються виробником.

- Network device support:
Dummy net driver support - чутливий bit-bucket пристрій.
EQL (serial line load balancing) support – використовується якщо комп'ютер під'єднаний через два послідовні порти до інших комп'ютерів і використовується SLIP - протокол (передача через телефонні лінії) чи PPP (покращений SLIP). При цьому вони розглядаються як один зв'язок тільки з вдвічі більшою швидкістю.
PLIP (pallel port) support - (Parallel Line Internet Protocol) - використовується для зв'язку двох (рідко більше) локальних машин у "маленьку мережу".
PPP (point-to-point) support - покращений SLIP.
SLIP (serial line) support - доступність цієї опції надає можливість використовувати SLIP чи CSLIP (compressed SLIP) при під'єюнанні до Internet чи Linux-машин. А також сконфігурувати дану Linux-машину як SLIP/CSLIP сервер для інших користувачів, під'єднаних через мережу.
Radio network interface (AX.25) - інтерфейс для Linux-машин, базований на радіо зв'язку.
Ethernet (10 or 100 Mbit) - протокол що використовується в локальних мережах навчальних закладів і установ.
Token Ring - мережний протокол, запропонований IBM.
ARCnet support - поширений протокол локальних мереж.

- ISDN support:
(Integrated Services Digital Network) - це спеціальний тип повної цифрової телефонної лінії (спеціально виділеної).

Часто використовується в Internet з SLIP/CSLIP-протоколами.
Support audio via ISDN - модем-емулятор підтримує EIA Class 8 Voice commands. Є лінії які підтримують аналогові сигнали.
ICN 2B & 4B support - дозволяється підтримка 2-х типів ISDN-карт німецької фірми ICN. 2B - це стандарт для одної ISDN-лінії з двома B-каналами, а 4B - це дві таких лінії.
PCBIT-D support - ці карти виготовляє португальська фірма Octal. (Це різновид ISDN-карт).
Teles/NICCY1016PC/Creatix support - підтримка Teles ISDN карт S0-16.0, S0-16.3, S0-8 і сумісних з ними.

- CD-ROM drivers (not for SCSI or IDE/ATAPI drives): На цьому кроці конфігурування надається можливість вибрати драйвери для пристроїв, які не використовують стандартні інтерфейси.

- Filesystems: Надається широкий вибір файлових систем, які так чи інакше можуть використовуватись у Linux: MS-DOS; VFAT (Windows-95); HPFS (OS2); MINIX; EXT, EXT2 (Linux); Coherent (експериментальна) т.д.

- Character devices: Користувачу надається можливість підтримувати так-звані символьні пристрої, як мишка, принтер, драйвери мультикарт послідовних та паралельних портів.
Adwanced Power Management BIOS support - підтримка засобів BIOS керування споживаною потужністю.
Watchdog Timer support - спеціальна робота з системним таймером.
Enchanced Real Time Clock support - використовується для доступу до вбудованого годинника реального часу.

- Sound: Пропонується вибрати тип звукової карти, синтезатора, інтерфейсу (наприклад MIDI) тощо.

- Kernel hacking: Доступність цієї опції дозволяє пізніше проводити налагодження ядра. Про використання такої специфічної опції докладно можна прочитати в одноіменному розділі документації.

Далі можна зберегти нову конфігурацію чи прочитати збережену раніше в іншому файлі. На закінчення коректного конфігурування ядра необхідно виконати таку команду:

```
make dep
```

При цьому створється нове дерево залежності між програмними модулями, тобто визначається та частина початкового коду, що буде компілюватись після внесення змін у конфігурацію.

Компіляція ядра:

- Команда "make zImage" - створює зархівований образ ядра. Якщо потрібно створити завантажувальну дискету (без root файлової системи чи lilo), необхідно вставити дискету в A: дисковод і задати команду "make zdisk". Потім можна задати "make zlilo", якщо є інстальоване lilo, але перед тим перевірити його установки.

- Тапер слід скопіювати нове ядро (після вдалої компіляції воно лежить в директорії /usr/src/linux/arch/i386/boot/zImage) на місце, з якого відбувається завантаження.

Це може бути дискета, тоді слід скористатися командою

```
cp /usr/src/linux/arch/i386/boot/zImage /dev/fd0
```

Або ж твердий диск, тоді ядро зазвичай зберігається як: /vmlinuz, /zImage або /etc/zImage. Якщо використовується lilo, це можна подивитись у файлі /etc/lilo.conf. Для використання нового ядра треба скопіювати його на місце старого (ОБОВ'ЯЗКОВО ЗБЕРЕГТИ СТАРЕ ЯДРО!). Потім потрібно поновити lilo.

- Для переінсталяції lilo використовується файл /sbin/lilo. Можна змінити установки lilo, відредагувавши файл /etc/lilo.conf. Старе ядро можна після цього використовувати (зазвичай його називають /vmlinuz.old), додавши в lilo.conf відповідний блок.

- Надалі, змінюючи кореневу директорію, ядро, відео-режим, та ін. слід користуватись програмою "rdev".

4. ЛОГІЧНА СТРУКТУРА ТВЕРДИХ ДИСКІВ. ЗАВАНТАЖУВАЧ ОПЕРАЦІЙНИХ СИСТЕМ LILO

LILO є програмою початкового завантаження (boot loader) операційної системи Linux. Він не залежить від типів використовуваних файлових систем і може завантажувати як ядра Linux, так і інші операційні системи з гнучких та твердих дисків. До числа інших ОС, що можуть бути завантажені за допомогою LILO можна віднести такі:

PC/MS-DOS, DR DOS, OS/2, 386BSD, SCO UNIX, Unixware, тощо.

LILO здатний забезпечити багатоваріантне завантаження до 16 різних ОС та ядер Linux.

При виборі методу завантаження Linux важливим є розуміння структури дискових підсистем цієї ОС. Найпростішу структуру мають гнучкі диски, які складаються із сектора завантаження (boot sector), блоку системної інформації (DOS - FAT , Linux - SuperBlock) та блоку даних:

Boot sector	Сектор завантаження
FAT or SuperBlock	Блок системної інформації
Data area	Блок даних

Фізичний диск в ОС Linux є певним блоковим пристроєм (наприклад /dev/fd0).

Сектор завантаження ОС MS-DOS має таку структуру:

0x000	Jump to the program code	Перехід на завантаження
0x003	Disk parameters	Параметри диска
0x02C / 0x03E	Program code	Програма завантаження
0x1FE	Magic number (0xAA55)	Тестове число (0xAA55)

LILO використовує схожу структуру сектора завантаження, яка не містить інформації про параметри диска. Це не є проблемою для файлових систем Minix, Ext2FS, XiaFS, але застосування такого сектора завантаження для FAT робить неможливим доступ ОС MS-DOS до диска.

Тверді диски мають складнішу організацію. Вони містять кілька блоків даних - так званих розділів (partitions). Твердий диск, сформатований під ОС MS-DOS, може містити до чотирьох первинних розділів. При необхідності створення більшої кількості розділів один із первинних розділів використовується для створення розширенного розділу, який може містити декілька логічних розділів.

Перший сектор кожного твердого диска, розширений розділ та кожний логічній розділ містять власні таблиці розділів.

Partition table		/dev/hda	Таблиця розділів
	Partition 1	/dev/hda1	Первинний розділ 1
	Partition 2	/dev/hda2	Первинний розділ 2

Тверді диски в ОС Linux є блоковими пристроями - /dev/hda, /dev/hdb, /dev/sda, тощо. Первинні розділи доступні як /dev/hda1, /dev/hda2, dev/hda3, /dev/hda4.

Partition table	/dev/hda	Таблиця розділів
Partition 1	/dev/hda1	Первинний розділ 1
Partition 2	/dev/hda2	Первинний розділ 2
Extended partition	/dev/hda3	Розширений розділ
Extended partition table		Розширена таблиця розділів
Partition 3	/dev/hda5	Логічний розділ 1 (диск) у розширеному розділі
Extended partition table		Розширена таблиця розділів
Partition 4	/dev/hda6	Логічний розділ 2 (диск) у розширеному розділі

В даному випадку твердий диск має два первинні розділи та розширений розділ з двома логічними розділами. Логічні розділи доступні як блокові пристрої /dev/hda5 ... Слід зауважити, що таблиці розділів логічних розділів не є доступними як блокові пристрої, на відміну від головної таблиці розділів, секторів завантаження та таблиці розділів розширеного розділу.

Таблиці розділів зберігаються у секторах завантаження розділів. Зазвичай для початкового завантаження використовується лише один сектор завантаження, який часто називають головним записом завантаження (master boot record (MBR)).

0x000	Program code	Програма завантаження
0x1BE	Partition table	Таблиця розділів
0x1FE	Magic number (0xAA55)	Тестове число (0xAA55)

Сектор завантаження, що використовується програмою LILO, може бути застосованим як сектор завантаження розділу при наявності області даних для таблиці розділів. Тому LILO може бути встановлений у таких зонах гнучких та твердих дисків:

- завантажувальному секторі гнучкого диска з ОС Linux.
- MBR першого твердого диска. (/dev/hda)
- завантажувальному секторі активного розділу ОС Linux на першому твердому диску. (/dev/hda1, ...)
- завантажувальному секторі розділу розширеного розділу на першому твердому диску. (/dev/hda1, ...)

LILO не може бути встановлений у таких зонах гнучких та твердих дисків:

- завантажувальному секторі гнучкого чи твердого диска, що не містять ОС Linux.

- розділі віртуальної пам'яті ОС Linux
- завантажувальному секторі логічного розділу розширеного розділу.

При завантаженні з гнучкого диска зчитується перший сектор диска - так званий сектор завантаження (boot-sector). Він містить невелику програму що здатна завантажити певну операційну систему. Сектор завантаження ОС MS-DOS додатково містить блок даних з параметрами диска та файлової системи.

При завантаженні з твердого диска зчитується MBR, який містить програму завантаження і таблицю розділів диска. Програма завантаження передає керування сектору завантаження одного з первинних чи розширених розділів, який і завантажує операційну систему.

Завантаження ОС MS-DOS:

Master Boot Record	Boot sector	Operating system
DOS-MBR ⟶	⟶ MS-DOS ⟶	⟶ COMMAND.COM

При завантаженні ОС MS-DOS MBR визначає активний розділ та завантажує сектор завантаження MS-DOS, який після старту системи передає керування командному процесору COMMAND.COM.

Використання програми LOADLIN для завантаження ОС Linux:

Master Boot Record	Boot sector	Operating system
DOS-MBR ⟶	⟶ MS-DOS ⟶	⟶ COMMAND.COM ⟶ LOADLIN ⟶ Linux

При використанні програми LOADLIN відбувається звичайне завантаження ОС MS-DOS з опрацюванням файлів CONFIG.SYS та AUTOEXEC.BAT. LOADLIN може бути запущена або із файлу AUTOEXEC.BAT, що реалізує багатоваріантне завантаження, або як звичайна програма MS-DOS. Такий підхід не потребує внесення змін до секторів завантаження чи структури дисків.

Використання програми LILO для завантаження ОС Linux:

Master Boot Record	Operating system
LILO ⟶	⟶ Linux ⟶ other OS

LILO може повністю контролювати процес завантаження при інсталяції його як MBR та завантажувати інші операційні системи. На жаль, необхідна повна перевірка інсталяції, оскільки LILO не завжди коректно працює на твердих дисках великої місткості.

Інсталяція:
- завантажити ОС Linux
- створити резервну копію MBR на гнучкому диску:

```
dd if=/dev/hda of=/fd/MBR bs=512 count=1
```

- інсталювати LILO в якості MBR
- перезавантажити комп'ютер.

Деінсталяція:
- завантажити ОС Linux
- відновити MBR з резервної копії:

```
dd if=/fd/MBR of=/dev/hda bs=446 count=1
```
- перезавантажити комп'ютер.

Під час завантаження LILO перевіряє такі події:
- чи були натиснені клавіші [Shift], [Control] або [Alt]
- чи включені тригери [CapsLock] або [ScrollLock].

В цьому випадку LILO відображає boot:-запрошення та очікує введення імені об'єкта завантаження (тобто певного ядра Linux чи іншої ОС). Інакше він завантажує об'єкт, що вибрано за замовчуванням, чи, якщо встановлено час затримки, очікує введення імені об'єкта завантаження протягом часу затримки. Список відомих об'єктів завантаження можна отримати, натиснувши клавішу [Tab]. Якщо ім'я об'єкта завантаження не було вказано і була натиснена клавіша [Enter], то завантажується об'єкт, що вибрано за замовчуванням.

LILO здатний передавати параметри командного рядка до ядра ОС Linux. Ці параметри є словами, що йдуть після імені об'єкта завантаження і розділяються пропусками. Приклад:

```
boot: linux single root=200
```

Стандартні параметри:
single - завантажити систему в режимі одного користувача.
root=<device> - вказати головний завантажувальний пристрій.
reserve=<base>,<size>,... - зарезервувати певний простір портів В/В
ro - змонтувати головний завантажувальний пристрій в режимі "лише читання".
no-hlt - не виконувати команду HLT при простоях системи.
no387 - не використовувати математичний процесор.
debug - виводити відлагоджувальну інформацію на консоль.
vga=<mode> - вибрати певний режим роботи консолі (дозволені режими: normal, extended, ask чи десятковий номер режиму).

Конфігураційна інформація зберігається у файлі /etc/lilo.conf та складається з сукупності певних змінних. Глобальні параметри конфігурації:

BACKUP=<backup_file> - зберегти оригінальний сектор завантаження у файлі чи на блоковому пристрої.

BOOT=<boot_device> - визначити ім'я пристрою, що містить сек тор завантаження.

COMPACT - оптимізувати процедуру завантаження шляхом генерації блокових запитів на читання даних з диска.

DEFAULT=<name> - визначити ім'я об'єкта завантаження за замовчуванням.

DELAY=<tsecs> - визначити час в десятих долях секунди, протягом якого LILO очікує введення імені об'єкта завантаження.

DISK=<device_name> - визначити нестандартну геометрію твердого диска.

DISKTAB=<disktab_file> - визначити ім'я таблиці параметрів дисків.

FORCE-BACKUP=<backup_file> - аналог BACKUP, що здатний переписувати стару копію сектора завантаження.

IGNORE-TABLE - ігнорувати пошкоджені таблиці розділів.

INSTALL=<boot_sector> - переписати визначений файл до сектора завантаження.

LINEAR - генерувати "лінійні" адреси для твердих дисків замість адрес формату "сектор/голова/циліндр".

LOCK - зберігати параметри командного рядка для використання в процесі подальших завантажень.

MAP=<map_file> - визначити ім'я файлу відображення (за замовчуванням /boot/map).

MESSAGE=<message_file> - визначити текст початкового повідомлення LILO (до 64K символів).

NOWARN - не виводити повідомлення про можливі загрози.

PASSWORD=<password> - встановити пароль на завантаження всієї системи.

PROMPT - визначає необхідність введення імені об'єкту завантажен ня.

TIMEOUT=<tsecs> - визначає час очікування введення імені об'єкту завантаження, після закінчення якого завантажується перший об'єкт зі списку.

Додатково в глобальній секції можуть бути визначені параметри APPEND, RAMDISK, READ-ONLY, READ-WRITE, ROOT та VGA. Вони використовуються як параметри за замовчуванням.

LILO використовує певне ім'я файлу (без шляху) для ідентифікації об'єкта завантаження. Можна застосовувати інше ім'я, використовуючи конфігураційну змінну LABEL, а також псевдонім, використовуючи конфігураційну змінну ALIAS. Конфігураційні параметри об'єкту завантаження:

APPEND=<string> - передати певні параметри до ядра системи

append = "hd=64,32,202".

LITERAL=<string> - аналог APPEND, але знищує всі інші параметри.

RAMDISK=<size> - визначає розмір віртуального диска для завантаження певного об'єкта.

READ-ONLY - визначає монтування головної файлової системи в режимі "лише читання".

READ-WRITE - визначає монтування головної файлової системи в режимі "читання-запис".

ROOT=<root_device> - визначає головний пристрій завантаження.

VGA=<mode> - визначає режим роботи консолі:

NORMAL - текстовий режим 80x25.

EXTENDED - текстовий режим 80x50.

ASK - запит на режим роботи.

<number> - код режиму роботи.

LILO підтримує такі типи об'єктів завантаження:

- ядро Linux з файлу;
- ядро Linux з блокового пристрою (гнучкого диска);
- сектор завантаження певної ОС.

Тип об'єкта завантаження визначається змінною в конфігураційному файлі LILO. Об'єкти завантаження можуть бути розташовані на довільних носіях інформації, що є доступними під час завантаження.

В секціях конфігурації усіх об'єктів завантаження використовуються такі змінні: ALIAS, LABEL, LOCK, OPTIONAL, PASSWORD та RESTRICTED. В

секціях конфігурації об'єктів завантаження ОС Linux використовуються такі змінні: ALIAS, LABEL, LOCK, OPTIONAL, PASSWORD та RESTRICTED.

LILO може завантажувати окрім ОС Linux і інші операційні системи - MS-DOS, PC-DOS, Windows тощо. Для завантаження іншої ОС необхідно визначити ім'я програми-завантажувача, пристрій чи файл, що містять сектор завантаження та пристрій з таблицею розділів. LILO використовує такі програми-завантажувачі:

- chain.b - завантажує та передає керування вказаному сектору завантаження;
- os2.b - призначений для завантаження OS/2;
- any_b.b - завантажує ОС з другого гнучкого диска;
- any_d.b - завантажує ОС з другого твердого диска.

Програма LILO при завантаженні відображає повідомлення "LILO". Кожна буква повідомлення друкується після виконання певної операції. Якщо LILO зависає на певному етапі завантаження то частина надрукованого повідомлення може бути використана для ідентифікації помилки:

(<нічого>) - Жодна частина LILO не завантажена. LILO або не встановлений, або розділ, де він розташований, не є активним.

L<код помилки>... - Перша частина LILO була завантажена і виконана, але вона не завантажила другу частину програми. Код помилки, що складається з двох цифр, визначає її тип. До такого класу помилок відносяться фізичне знищення певних даних на диску або неправильно задана геометрія диска.

LI - Перша частина LILO завантажила другу частину, але не змогла виконати її. Це переважно означає розбіжність реальної та заданої геометрії диска, або знищення чи перейменування файлу /boot/boot.b.

LIL - Друга частина LILO стартувала, але не завантажила таблицю дескрипторів розділів з файлу відображення /boot/map. Це означає помилку диска або розбіжність заданої та фактичної геометрії диска.

LIL? - Друга частина LILO завантажена за некоректною адресою. Це може бути викликано переміщенням файлу /boot/boot.b без повторного запуску інсталятора.

LIL- - Пошкоджено таблицю дескрипторів розділів.

LILO - Всі частини LILO успішно завантажені.

Приклад конфігураційного файлу LILO /etc/lilo.conf:

```
 # Глобальна секція конфігурації LILO
boot = /dev/hda
message=/boot/message
compact                 # швидке завантаження.
delay = 50              # замість prompt
#prompt                 # замість delay
vga = normal            # нормальний режим роботи консолі
ramdisk = 0             #
#! sbpcd = 0x340,SoundBlaster    # підключення звукової карти
# Кінець глобальної секції конфігурації LILO #

# Секція ядер ОС Linux
image = /vmlinuz              # файл ядра Linux
  append = "ether=11,0x300,eth1" # передача параметрів до ядра
```

```
  root = /dev/hda2      # Root - розділ
  label = Linux              # Ім'я ядра
image = /vmlinuz.old
  root = /dev/hda2
  label = LinuxOld
  read-only
# Кінець секції ядер Linux

# Секція DOS other = /dev/hda1
label = DOS
table = /dev/hda
# Кінець секції DOS
```

5. З'ЄДНАННЯ КОМП'ЮТЕРІВ ПОСЛІДОВНИМИ КАНАЛАМИ. ПРОТОКОЛИ SLIP ТА PPP

Протоколи SLIP та PPP належать до канального рівня (network interface) п’ятирівневої архітектури керування в Internet. Тобто це апаратно залежне програмне забезпечення, що реалізує розповсюдження інформації на певному відрізку середовища передавання даних.

SLIP та PPP забезпечують передавання IP-пакетів через послідовні лінії типу “точка-точка”. Їх можна розглядати як певний аналог Ethernet чи TokenRing. Як у локальній мережі IP-дейтаграми вкладаються у кадри Ethernet, так на послідовній лінії – у кадри SLIP чи PPP.

Першим стандартом канального рівня для забезпечення роботи терміналів користувачів (TCP/IP) через послідовні лінії зв’язку історично став протокол SLIP (Serial Line Internet Protocol). Він був підтриманий в ОС Linux і реалізований у програмному забезпеченні для ПК. SLIP надає можливість під’єднатись до мережі Internet через стандартний інтерфейс RS-232.

Протокол SLIP має такі недоліки:

– не забезпечується обмін адресною інформацією (не передбачене динамічне налаштування каналу);
– відсутня індикація типу протоколу, пакет якого “вкладається” у кадр SLIP, тому передається трафік лише одного мережного протоколу (IP);
– не передбачені процедури виявлення помилок та повторного передавання.

Для підвищення ефективності використання пропускної здатності послідовних ліній зв’язку використовуються алгоритми стиску даних (наприклад, за рахунок зменшення обсягу службової інформації, що міститься у заголовках пакетів IP). Таку задачу вирішує протокол CSLIP. При використанні протоколів типу TELNET для доставки одного байта даних необхідно передати 20-байтовий заголовок пакета IP і 20-байтовий заголовок пакета TCP. Протокол CSLIP забезпечує стиск 40-байтового заголовка до 3 – 5 байтів.

Протокол PPP (Point-to-Point Protocol) прийшов на зміну протоколу SLIP. Він сумісний не лише з RS-232, але й з іншими інтерфейсами (RS-422, RS-423, V.35). Протокол PPP може працювати без керуючих сигналів модема (наприклад, «Request to Send», “Clear to Send”, “Data Terminal Ready”). Єдина жорстка вимога – забезпечення лінією зв’язку дуплексного з’єднання.

Протокол PPP включає три основні компоненти:

– протокол HDLC (High level Data Link Control) для кодування даних перед відправленням у лінію – механізм обрамлення пакетів протоколів мережного рівня і формування кадрів;
– протокол LCP (Link Control Protocol) для встановлення, конфігурування і тестування з’єднання; кадри LCP пересилаються до початку передавання даних;
– протокол мережного керування NCP (Network Control Protocol) для встановлення та конфігурування процедур передавання повідомлень, які надійшли від мереж, що використовують різні мережні протоколи.

Спрощений алгоритм функціонування протоколу PPP такий (рис.). Початкова фаза починає і закінчує процес зв’язку. У випадку появи зовнішньої

події (наприклад, готовність апаратного забезпечення здійснити зв'язок) буде ініційована фаза встановлення з'єднання, впродовж якої відбувається узгодження параметрів з'єднання (обмін кадрами LCP). У випадку неможливості встановлення з'єднання процес переривається і протокол переходить у стан початкової фази. Якщо ж всі необхідні параметри узгодженні, буде ініційована фаза автентифікації, впродовж якої проводиться перевірка на справжність учасників сеансу зв'язку (якщо у цьому є потреба). У випадку невдалої автентифікації процес з'єднання перейде у фазу роз'єднання, яка готує розрив з'єднання. Якщо ж фаза автентифікації пройшла вдало, протокол переходить до фази передавання даних. В цій фазі здійснюється обмін даними. У фазі роз'єднання (використовується після закінчення передавання кадрів чи у випадку виникнення якихось помилок) передавання кадрів переривається і протокол PPP переходить у стан початкової фази.

Рис. 1. Блок-схема алгоритму функціонування протоколу PPP

Протокол PPP надає механізми стиску даних і динамічного призначення IP-адреси віддаленому вузлу за допомогою протоколу IPCP (IP Control Protocol). В IPCP сервер телефонного доступу (dial-in server) виділяє IP-адресу з наперед визначеного діапазону. Після завершення сеансу віддаленого доступу ця IP-адреса вивільняється.

В протоколі PPP передбачено два механізми захисту з'єднань:

1. З використанням протоколу автентифікації за паролем PAP (Password Authentification Protocol). Недолік – паролі передаються відкритим текстом.
2. З використанням протоколу автентифікації за запитом/відповіддю CHAP (Challenge Handshake Authentification Protocol). Сервер відсилає клієнту запит (унікальний для кожного PPP сеансу). На основі секретного значення, відомого лише клієнту і серверу, клієнт вираховує відповідь на цей запит і відправляє його серверу. Тим часом сервер проводить такі ж обчислення і порівнює отримані значення з відповіддю, прийнятою від клієнта.

Протоколи PAP і CHAP не виключають необхідності додаткового захисту, оскільки дозволяють ідентифікувати лише пристрій, але не користувача.

Додаткова функціональність PPP забезпечується за рахунок збільшення навантаження на мережу. Так, при вкладенні пакета у кадр SLIP, обсяг даних, що належиться передати у лінію, збільшується усього на 1 байт, тоді як для PPP – на цілих 8 байтів. Крім цього, перед початком передавання абонентам необхідно обмінятись декількома LCP- та NCP-пакетами. Однак це виправдовується розширенням можливостей.

Порівняно з протоколом SLIP протокол PPP є значно розвинутішим інструментом і має такі переваги:

– можливість одночасної роботи з різними мережними протоколами, а не лише з IP (наприклад, DECNet, IPX, AppleTalk);
– перевірка цілісності даних;
– підтримка динамічного обміну адресами IP;
– можливість стиску заголовків пакетів IP і TCP за алгоритмами, механізм яких схожий на реалізований у протоколі CSLIP.

Бажано, щоб організація мережі не обмежувалась до одного Ethernet. Ідеально було б використовувати мережу незалежно від того, на яких аппаратних засобах ЕОМ це виконується і на скільки підодиниць поділяється. Наприклад, у великих будівлях маємо номери окремого Ethernet, що мають бути зв'язані певним чином.

Цей зв'язок обробляється так званим gateway, що обробляє пакети, які надходять і відправляються, копіюючи їх між двома Ethernets.

Datagram-обмін керується відповідно до єдиного протоколу, який є незалежним від використовуваних аппаратних засобів ЕОМ: IP, або Internet Protocol.

IP також потребує апаратних засобів. Адреса IP описана як чотири десяткові номери, для кожної частини з 8 бітами, відділяється крапками. Наприклад, адреса IP 0x954C0C04 може бути записана як 149.76.12.4.

На послідовних лініях SLIP – це стандарт "de facto", оскільки SLIP часто використовується. Модифікація SLIP, відома як CSLIP, виконує стискання, щоб зробити кращим використання з відносно низькосмуговими послідовними каналами.

PPP має ще більші особливості ніж SLIP, включаючи стадію переговорів зв'язку. Головна перевага PPP перед SLIP, це те, що він не обмежений транспортуванням IP datagrams.

Інтерфейси вказуються іменами, які визначені внутрішньо в ядрі. Типові імена для інтерфейсів Ethernet - eth0, eth1, и т.д. Призначення інтерфейсів на пристрої залежить від замовлення, у якому пристрої конфігуровані. Наприклад, перший Ethernet буде встановлений як eth0, наступний буде eth1, і т.д. Один вийняток з цього правила – SLIP зв'язує за допомогою інтерфейса, що призначений динамічно, тобто будь-коли, коли зв'язок SLIP встановлений, интерфейс призначений до послідовного порту. Інтерфейси SLIP зв'язані з послідовними лініями в замовленні, в котрому вони разміщені для SLIP; тобто, перша послідовна лінія конфігурована для SLIP стає sl0, и т.д.

SLIP та PPP – широко використовувані протоколи для посилання IP-пакета через послідовний зв'язок. Ряд установ пропонують SLIP dialup і PPP доступ на машини, таким чином забезпечуючи IP-зв'язок.

Щоб керувати SLIP чи PPP, ніяке модифікування апаратних засобів ЕОМ не потрібне, можна використовувати будь-який послідовний порт. Так як послідовна конфігурація порта не призначена до TCP/IP организації мережі, SLIP є дещо посередній, дозволяючи і діалогове, і недіалогове використання. SLIP використовують, щоб набрати номер мережі университетських містечок або деякий інший вид SLIP-з'єднання, щоб керувати FTP сесіями, и т.д. SLIP

може також використовуватись як постійний чи напівпостійний зв'язок для LAN-to-LAN зчеплення, хоча це цікаво тільки з ISDN.

На деяких операційних системах, драйвер SLIP – програма користувача; В Linux це частина ядра, яка робить все швидше. Драйвер SLIP безпосередньо розуміє ряд змін на протоколі SLIP. Крім звичайного SLIP, також розуміє CSLIP, який покращує продуктивність для діалогових сесій повідомлень. Додатково існують версії з шістьма бітами для кажного з цих протоколів.

Простий шлях перетворювати послідовну лінію для SLIP способу – використовуючи slattach інструмент. Припустимо, що наш модем на /dev/cua3, і маємо доступ в SLIP server. Тоді будемо виконувати:

```
slattach /dev/cua3 &
```

Це буде перемикати дисципліну лінії cua3 до SLIPDISC, і перекладати на один з интерфейсів мережі SLIP. Якщо це перший активний св'язок SLIP, лінія буде прикладена до sl0; другий був би прикладений до sl1, і т.д.. Фактично, slattach не тільки дозволяє SLIP, але й інші протоколи, що використовують послідовну лінію, наприклад PPP чи KISS.

Після передачі лінії SLIP драйверу, потрібно конфігурувати інтерфейс мережі. Припустимо, що від vlager маємо сервер, що називається cowslip. Тоді виконуємо:

```
ifconfig sl0 vlager pointopoint cowslip
route add cowslip
route add default gw cowslip
```

Це ж саме можна зробити за допомогою утиліти dip, написавши такий файл конфігурації:

```
# Set local and remote name and address
get $local vlager
get $remote cowslip
port cua3     # choose a serial port
speed 38400        # set speed to max
modem HAYES        # set modem type
reset              # reset modem and tty
flush              # flush out modem response
# Prepare for dialing.
send ATQ0V1E1X1\r
wait OK 2
if $errlvl != 0 goto error
dial 41988
if $errlvl != 0 goto error
wait CONNECT 60
if $errlvl != 0 goto error
# Okay, we're connected now
sleep 3
send \r\n\r\n
wait ogin: 10
if $errlvl != 0 goto error
send Svlager\n
wait ssword: 5
if $errlvl != 0 goto error
```

```
send hey-jude\n
wait running 30
if $errlvl != 0 goto error
# We have logged in, and the remote side is firing up SLIP.
print Connected to $remote with address $rmtip
default            # Make this link our default route
mode SLIP     # We go to SLIP mode, too
# fall through in case of error
error: print SLIP to $remote failed.
```

6. ОРГАНІЗАЦІЯ СЛУЖБИ INTERNET НА LINUX-СИСТЕМІ

Стандартні протоколи та послуги в TCP/IP перераховуються у файлах /etc/protocol і /etc/services.

Protocol - конфігураційний файл який містить імена всіх протоколів відомих локальному хосту.

Services - конфігураційний файл в якому асоціюються кожне ім'я сервісу і псевдонім з номером порту і протоколом, який кожен із сервісів використовує.

Приклад файлу protocol:

```
#
# Internet (IP) протокол
#
ip 0 IP          # internet протокол
icmp 1 ICMP      # internet протокол повідомлень про помилки
igmp 2 IGMP      #  internet  протокол  багатокористувацьких
груп
tcp 6 TCP        # протокол керування для IP
udp 17 UDP       # протокол користувацьких дейтаграм
raw 255 RAW      # RAW IP-інтерфейс
```

Файл services описує номери портів, що визначені у "RFC", для стандартних служб Internet. Докладно кожний запис робиться за певним шаблоном service port/protocol [aliases], де:

service - специфікує ім'я сервісу;

port - визначає сервісний порт для даного типу сервісу;

protocol - визначає який транспортний протокол використовується для даного типу сервісу; У загальному випадку це може бути UDP чи TCP. Для сервісу можливо запропонувати більш ніж один протокол, так само як можливо запропонувати різні сервіси для одного і того ж порту;

aliases - (псевдоніми) дозволяють визначити альтернативні імена для того ж самого сервісу. Зазвичай, не потрібно змінювати сервісні файли, які є разом з мережним програмним забезпеченням у системі.

Приклад файлу services:

```
echo 7/tcp
echo 7/udp
discard 9/tcp sink null
discard 9/udp sink null
daytime 13/tcp
daytime 13/udp
chargen 19/tcp ttytst source
chargen 19/udp ttytst source
ftp-data 20/tcp
ftp 21/tcp
telnet 23/tcp
smtp 25/tcp
nntp 119/tcp readnews
#
```

```
# UNIX services
exec 512/tcp          # BSD rexecd
biff 512/udp comsat       # mail notification
login 513/tcp             # remote login
who 513/udp whod      # remote who and uptime
shell 514/tcp cmd         # remote shell
syslog 514/udp        # remote sys logging
printer 515/tcp spooler        # remote printer
route 520/udp router routed    # remote route
```

Перелічимо деякі головні служби Internet:

telnet - сервіс віртуального терміналу використовується для роботи з віддаленим терміналом.

smtp - поштовий сервіс використовується для прийому/передачі пошти.

ftp-data / ftp - File transfer protocol. Слід відмітити, що при використанні даного сервісу, протокол обміну однаковий для даних і для команд, а порти різні.

nntp - сервіс новин, використовується для читання новин у системі.

exec - сервіс запуску процесів з віддаленого терміналу з відображенням на цьому терміналі результатів роботи і введення даних для даного процесу.

printer - сервіс використовується для приймання даних від віддаленого терміналу для місцевого принтера.

syslog - сервіс використовується для віддаленого входу до системи через системний login.

login - сервіс віддаленого login для віддаленого терміналу.

Слід зазначити, що, наприклад, echo-сервіс дозволений для порту 7 як для TCP, так і для UDP і що порт 512 використовується для двох різних сервісів.

Часто послуги надаються за допомогою так званих 'демонів'. Демон - це програма, яка відкриває певний порт і очікує на спроби з'єднань, що надходять. Якщо це трапилось, демон створює дочірній процес, який сприймає це з'єднання, тоді як батьківський процес продовжує очікувати на наступні запити.

Ця концепція має таке пояснення: для кожної послуги, що пропонується, демон має виконати те, що очікує на порту з'єднання, що в загальному призводить до витрат системних ресурсів, наприклад swap-простору. Таким чином, майже всі реалізації Linux запускають супер-сервер, який створює комунікаційні гнізда (сокети) для багатьох послуг і слухає на всіх них одночасно, використовуючи системний виклик select(2). Коли віддалений хост запитує одну з послуг, супер-сервер відмічає це і викликає сервер, призначений для цього порту.

Загалом застосовується суперсервер inetd (Internet Daemon). Він запускається під час завантаження системи і бере список послуг, які він повинен надавати, з конфігураційного файлу /etc/inetd.conf. Додатково до цих серверів, які викликаються ззовні, inetd має багато тривіальних послуг, які він виконує сам, і які називаються внутрішніми послугами. Вони включають chargen (character generator), який просто генерує рядок символів, і daytime, який повертає системне значення часу доби. Цей файл складається з рядків, які у свою чергу містять і поля:

service type protocol wait user server cmdline, де:

service - назва послуги, передається на номер порту, який вказано у файлі /etc/services.

type - визначає тип сокету, stream - для протоколів, орієнтова них на з'єднання, або dgram - для протоколів дейтаграм. Отже, послуги на основі TCP, повинні завжди використовувати тип stream, а на основі UDP - тип dgram.

protocol - назва транспортного протоколу, який використовує послуга. Це має бути коректна назва протоколу, яка міститься у файлі protocols.

wait - ця опція застосовується тільки до сокетів типу dgram. Може приймати два значення: wait і nowait. Якщо зазначено wait, inetd виконуватиме лише одне обслуговування (сервер) для вказаного порту у будь-який час. В іншому випадку він негайно після завершення обслуговування продовжить слухати порт. Це корисно для серверів, що містять одну нитку (thread) і читають усі вхідні дейтаграми, потім цей сервер завершується. Більшість RPC-серверів саме такого типу, тому для них має зазначатися wait. Інший тип, 'багатониткові' сервери, дозволяють одночасне виконання необмеженої кількості примірників; це дуже рідко застосовується. Для таких серверів повинно зазначатися nowait. Для сокетів типу stream треба завжди використовувати nowait.

user - це вхідне ім'я користувача, для якого виконується процес. Часто це буває користувач root, але деякі послуги можуть використати інші варіанти. Дуже вдалою є ідея застосувати тут принцип найменших привілеїв, який полягає у тому, що не треба запускати програму як привілейовану, якщо сама програма не потребує цього для своєї коректної роботи.

server - це назва повного шляху до програми-сервера, що виконує обслуговування. Внутрішні послуги позначаються ключовим словом internal.

cmdline - це командний рядок, який передається серверу. Рядок включає аргумент 0, або назву програми. Зазвичай це має бути назва програми-сервера, якщо програма не виконує інших функцій, коли викликається під іншим ім'ям. Для внутрішніх послуг це поле є чистим.

Нижче наведено приклад файлу inetd.conf. Послуга finger є закоментованою, тому до неї немає доступу. Це часто робиться з міркувань безпеки, тому що finger може використовуватись під час атак на систему, щоб отримати імена користувачів. Так само закоментовано послугу tftp, яка реалізує примітивний протокол передачі файлів, що дозволяє передавати будь-які файли зі системи без перевірки паролів і т.д. Це особливо небезпечно для файлу /etc/passwd, тим більше якщо не використовуються тіньові паролі. TFTP в основному застосовується клієнтами без власних твердих дисків та X-терміналами, щоб переписати свій код з сервера. Якщо необхідно запустити tftpd з цієї причини, переконайтесь, що його область бачення (scope) обмежена лише тими директоріями, з яких клієнти отримують свої файли. Для цього назви цих директорій вказуються у командному рядку для tftpd (другий рядок з tftp у прикладі).

```
# inetd services
ftp stream tcp nowait root /usr/sbin/ftpd in.ftpd
telnet stream tcp nowait root /usr/sbin/telnetd in.telnet
#finger stream tcp nowait bin /usr/sbin/fingerd in.finger
#tftp dgram udp wait nobody /usr/sbin/tftpd in.tftpd
```

```
#tftp dgram udp wait nobody /usr/sbin/tftpd in.tftpd
login stream tcp nowait root /usr/sbin/rlogind in.rlogin
shell stream tcp nowait root /usr/sbin/rshd in.rshd
exec stream tcp nowait root /usr/sbin/rexecd in.rexecd
# inetd internal services
daytime stream tcp nowait root internal
daytime dgram udp nowait root internal
time stream tcp nowait root internal
time dgram udp nowait root internal
echo stream tcp nowait root internal
echo dgram udp nowait root internal
discard stream tcp nowait root internal
discard dgram udp nowait root internal
chargen stream tcp nowait root internal
chargen dgram udp nowait root internal
```

Inetd треба запускати під час завантаження (файл /etc/rc.local). Потім він прослуховує з'єднання на визначених сокетах для Internet. Коли з'єднання знайдено на одному з цих сокетів, inetd вирішує, з якою послугою пов'язаний даний сокет, і викликає програму для обслуговування запиту. Після завершення програми він продовжує слухати на сокеті. Важливим є те, що inetd дозволяє запуск одного демона для виклику декількох інших, зменшуючи цим навантаження на систему.

Синтаксис запуску: inetd [-d] [configuration file], де -d - перейти у режим відлагодження.

Під час виконання inetd зчитує інформацію про свою конфігурацію з конфігураційного файлу, яким за замовчуванням є /etc/inetd.conf. Повинно бути входження для кожного поля конфігураційного файлу, які розділяються пропусками або символами табуляції. Коментарі позначаються знаком "#" на початку рядка. Конфігураційний файл містить такі поля:

service name
socket type
protocol
wait/nowait[.max]
user[.group]
server program
server program arguments
Для послуг, що базуються на Sun-RPC, мають бути такі поля:
service name/version
socket type
rpc/protocol
wait/nowait[.max]
user[.group]
server program
server program arguments

Розглянемо деякі з цих полів:

service name - назва доступної послуги, вказаної у файлі /etc/services. Для внутрішніх послуг назва послуги повинна бути офіційною назвою послуги (перше поле в /etc/services). Доступна RPC-послуга вказується у файлі /etc/rpc. Справа від "/" в такому випадку вказується номер версії RPC або діапазон версій.

socket-type - приймає одне зі значень `stream', `dgram', `raw', `rdm', або `seqpacket', залежно від того, чи є сокет потоком, дейтаграмою, raw, повідомленням з надійною доставкою (re liably delivered message), чи сокетом з пакетом послідовностей (sequenced packet socket).

Протокол має збігатися з протоколом, наведеним у файлі /etc/protocols. Прикладами можуть бути `tcp' чи `udp'. Для послуг на основі RPC вказується тип `rpc/tcp' або `rpc/udp'.

Inetd внутрішньо підтримує декілька тривіальних послуг, використовуючи власні підпрограми. Цими послугами є `echo', `discard', `chargen' (генератор символів), `daytime' (час для сприйняття людиною), `time' (час для сприйняття машиною - кількість секунд від півночі 1 січня 1980 року). Всі ці послуги базуються на tcp.

7. РОЗПОДІЛЕНА ФАЙЛОВА СИСТЕМА NFS ДЛЯ ОБЧИСЛЮВАЛЬНИХ КОМПЛЕКСІВ

Мережна файлова система NFS (Network File System) є найкращим засобом для використання мережі з допомогою RPC. Вона дозволяє мати доступ до файлів віддалених машин в такий же спосіб, як до локальних файлів. Це стало можливим шляхом змішування ядра на клієнтській стороні і на NFS-сервері, що забезпечує доступ до даних. Цей доступ до файлів є повністю прозорим для клієнта і працює з серверами різної архітектури.

NFS забезпечує такі переваги. Дані, що ними користуються, можуть міститись на центральному хості, а клієнти будуть монтувати ці директорії під час завантаження. Наприклад, можна тримати всіх користувачів під'єднаними до одного хоста і мати примонтованими у всі хости мережі директорію /home з цього хоста. Також, використовуючи NIS користувачі можуть увійти в систему і працювати зі своїм набором файлів.

Дані, що займають багато дискового простору можуть зберігатись на одному хості. Наприклад, всі файли і програми, що стосуються LATEX і METAFONT можуть зберігатись в одному місці.

Дані для керування і адміністрування можуть зберігатись на одному хості. Тоді не потрібно використовувати rpc для інсталювання однакових файлів на 20 різних машинах.

Розглянемо, як працює NFS. Клієнт може запросити монтування директорії з віддаленого хоста у локальну директорію так само, як він може монтувати фізичний пристрій.

Синтаксис, що використовується для вказування віддалених директорій є різним. Наприклад, для монтування /home з хоста vlager в /users на vale, адміністатор має написати такий командний рядок на vale:

```
mount -t nfs vlager:/home /users
```

Mount спробує з'єднатись з демоном mountd на vlager через RPC. Сервер перевірить, чи vale дозволено монтувати директорію і, у випадку присутності такого дозволу, поверне покажчик файлу. Цей покажчик буде використовуватись при всіх звертаннях до файлів у директорії /users.

Коли хтось хоче мати доступ до файлів через NFS ядро поміщає виклик RPC у nfsd (демон NFS) на комп’ютері-сервері. Цей виклик бере покажчик файлу, назви файлів, до яких буде проводитись звертання, користувацькі userid, groupid як параметри. Вони використовуються для визначення прав доступу до вказаних файлів. Для забезпечення неможливості несанкціонованого доступу до файлів вимагається, щоб при звертанні userid і groupid користувача і файлів були однаковими на обох хостах.

У більшості Linux реалізаціях NFS-функціональність для клієнта і сервера є вбудована як демони рівня ядра що запускаються з простору користувача при системному завантаженні. На сервері працює демон nfsd, а на клієнтській машині – Block I/O Daemon (biod). Для поліпшення передачі biod забезпечує

асинхронне введення/виведення, використовуючи читання з випередженням і запис із затримкою (read-ahead and write-behind).

Реалізація NFS в Linux трохи відрізняється – код клієнта вже безпосередньо інтегрований у рівень віртуальної файлової системи VFS (Virtual File System) ядра і не потребує додаткового контролю від biod. З іншого боку код серверу працює в області користувача, тому використання кількох копій серверу одночасно є неможливим, тому, що порушується синхронізація між копіями.

Найбільшою проблемою NFS Linux 1.0 є те, що ядро системи не може виділити память шматками більшими ніж 4К, і як наслідок, мережний код не може керувати дейтаграмами більшими ніж 3500 байтів після виділення розміру заголовка. Це означає, що пересилання до і з NFS демонів, що працюють у системах, які використовують більші UDP-дейтаграми за замовчуванням (8K для SunOS), повинні штучно зменшувати розміри. Це зменшує продуктивність системи. Ця проблема ліквідована в пізніших реалізаціях Linux-1.1. Також модифікована клієнтська частина.

Перед використанням NFS треба впевнитись чи ядро має вбудовану підтримку NFS. Новіші ядра мають для цього простий інтерфейс в proc-файловій системі, це файл /proc/filesystems, який можна подивитись використовуючи команду cat:

```
$ cat /proc/filesystems
unix
ext2
msdos
nodev proc
nodev nfs
```

Якщо nfs немає у списку, то треба перекомпілювати ядро, включивши підтримку NFS.

Для старших версій Linux найлегшим шляхом перевірки підтримки NFS ядром є спроба монтування NFS локальної файлової системи. Наприклад:

```
mkdir /tmp/test
mount localhost:/etc /tmp/test
```

Якщо монтування перерветься повідомленням "fs type nfs no supported by kernel" , то потрібно переробити ядро з дозволом NFS. Якщо зявиться інше повідомлення, то це означає, що NFS демони ще не сконфігуровані.

NFS-томи монтуються так само, як і файлові системи:

```
mount -t nfs nfs_volume local_dir options
```

nfs_volume подається як remout_host:remout_dir. Оскільки, такий запис є унікальним для файлової системи NFS, то параметр -t nfs можна опускати.

Є кілька додаткових опцій, які можна передати mount при монтуванні тому NFS. Вони можуть вводитись через -o в командному рядку або через задання опцій в /etc/fstab. В обох випадках, кілька опцій розділяються комами. Опції, що задаються у командному рядку завжди передують тим, що задані у файлі fstab.

Приклад з файлу fstab:

```
# volume mount point type options
```

```
news:/usr/spool/news /usr/spool/news nfs timeo=14,intr
```

Цей том можна примонтувати, використовуючи:

```
mount news:/usr/spool/news
```

Наприклад, хочемо примонтувати домашній каталог з машини moonshot, що використовує за замовчуванням блоки в 4К для операцій запису/читання. Ми повинні зменшити розмір блоку до 2К для роботи Linux дейтаграм шляхом посилання команди:

```
mount moonshot:/home /home -o rsize=2048,wsize=2048
```

Перелік дозволених опцій є описаний в nfs(5) manual page. Наведемо неповний список опцій, що найчастіше використовуються.

rsize=n and wsize=n - задається розмір дейтаграми, що використовується клієнтами для запитів читання/запису. За замовчуванням 1024 байтів.

tineo=n - задається час (в десятих секунди) очікування клієнта на завершення запиту. За замовчуванням 0.7 сек.

hard - маркування тому як жорстко-примонтованого. За замовчуванням soft - протилежне до hard.

intr - дозволяє сигнали для переривання викликів NFS. Корисно для завершення програм, коли сервер довго не відповідає.

Виключно для rsize і wsize всі ці опції впливають на поведінку клієнта у випадку, коли сервер тимчасово є недоступним. Вони впливають разом у такий спосіб: будь-коли, коли клієнт виконує запит до сервера, він очікує завершення операції протягом заданого інтервалу timeo. Якщо від сервера не надходить підтвердження протягом цього часу, то виникає minor-timeout, і операція повторюється зі збільшеним удвічі інтервалом timeout. Після досягнення максимуму 60 сек виникає major-timeout.

За замовчуванням виникнення majotr-timeout викликає друк повідомлення на консолі клієнта та ініціює повтор виконання запиту. Потенційно це може продовжуватись до безмежності. Том, що весь час намагається з'єднатись із сервером називається жорстко-примонтованим (hard-mounted). Альтернативний випадок – мяко-примонтований (soft-mounted) том генерує помилку введення/виведення для процесу, що викликає, у випадках виникнення major-timeout. Оскільки для запису використовується технологія write-behind через кеш, то процес, що викликає функцію write, не може бути впевненим, що операція запису загалом пройшла нормально.

Вибір жорстокого чи мякого монтування не є просто справою смаку, а визначається характером даних, що будуть зберігатись у примонтованому томі. Наприклад, якщо примонтувати X-програми за допомогою NFS то можна зробити неможливим свою роботу через те, що хтось запустить декілька xv чи випадково відімкне Ethernet-з'єднувач. При жорсткому монтуванні треба впевнитись, що компютер готовий до відновлень звязку з сервером. З іншого боку, не критичні дані, такі як розділи новин чи FTP-архіви можуть бути мяко примонтованими і це не зірве роботу сеансу у випадку, коли віддалений комп’ютер є тимчасово недоступним чи вимкненим. Якщо звязок через мережу є повільним чи використовується маршрутизатор, то треба збільшити час таймауту, використовуючи опцію timeo чи жорстко примонтувати том, але

дозволити сигнали переривання запитів NFS, що дозволить вивести систему із зациклювання при порушенні звязку чи доступу.

Зазвичай, демон mountd відслідковує звязки між приєднаними директоріями і хостами. Цю інформацію можна викликати, використовуючи програму showmount, яка входить у комплект NFS-сервера.

Якщо хочемо забезпечити NFS-сервіс для інших хостів, то необхідно на своєму комп'ютері запустити демони nfsd і mountd. Це є RPC-програми, тому вони не підлягають керуванню від inet, але стартують при завантаженні і реєструються через встановлення портів (portmapper). Впевнитись у правильній їх роботі можна лише після запуску rpc.portmapper. Звичайно, додаємо такі два рядки у скрипт rc.inet2:

```
if [ -x /usr/sbin/rpc.mountd ]; then
/usr/sbin/rpc.mountd; echo -n " mountd"
fi
if [ -x /usr/sbin/rpc.nfsd ]; then
/usr/sbin/rpc.nfsd; echo -n " nfsd"
fi
```

Інформацію про власника файлів демони NFS надають клієнтам у вигляді цифрових ідентифікаторів користувача і групи. Якщо клієнт і сервер асоційовані з однаковими цими кодами, то вони розділяють один простір ідентифікаторів користувача і групи. Наприклад, в цьому випадку можна використати NFS для розповсюдження інформації passwd на всі хости з нашої мережі.

В деяких випадках ці ідентифікатори не збігаються. Для їх приведення у відповідність один одному можна використати демон ugidd, що буде мапувати (mapping) ці ідентифікатори. Використовуючи опцію map_daemon можна задати nfsd мапувати uid/gid сервера до клієтнських uid/gid з aid ugidd клієнта.

ugidd є RPC-сервером і запускається з rc.inet2 так, як nfsd та mountd:

```
if [ -x /usr/sbin/rpc.ugidd ]; then
/usr/sbin/rpc.ugidd; echo -n " ugidd"
fi
```

До цього було розглянуте конфігурування клієнтської сторони. Тепер слід розглянути конфігурування серверу. Тут також використовуються різні опції, що задаються у файлі /etc/export.

За замовчуванням mountd не дозволяє монтувати директорії з локального хосту, хоча в деяких випадках це є необхідним. Для дозволу одному чи декільком хостам монтувати директорію, вона повинна бути експортованою, тобто бути вказаною у файлі exports. Наприклад, цей файл може виглядати так:

```
# expotrs file fot vlager
/home vale(rw) vstont(rw) vlight(rw)
/usr/X386 vale(ro) vstont(ro) vlight(ro)
/usr/TeX vale(ro) vstont(ro) vlight(ro)
/ vale(rw,no_root_squash)
/home/ftp (ro)
```

Тут кожен рядок визначає директорію і хости, яким дозволено монтувати їх. Звичайно назви хостів задаються повною назвою домену, але може також додатково містити символи шаблонів * ?. Наприклад, lab*.foo.com визначає як

lab01.foo.com, так і laber.foo.com. Якщо не задано ніяких імен хостів, то монтування цих директорій дозволено будь-яким хостам.

Коли перевіряється назва клієнта у файлі exports, то mountd переглядає назви клієнтів, використовуючи виклик gethostbyaddr(). З DNS цей виклик повертає канонічну назву клієнта, тому треба впевнитись, що у файлі exports не використовуються синоніми. Без використання DNS повернуте імя є першим, що співпало зі знайденими у файлі hosts і збігається з клієнтською адресою.

Назви хостів можуть бути продовжені опціями-прапорцями, що розділяються комами і поміщені у дужки. Прапорці можуть приймати такі значення:

insecure - дозволяє неавтентифікований доступ з цього комп'ютера.

unix-rpc - вимагає автентифікацію RPC Linux-домену з цього комп'ютера. Це означає, що вимагається доступ зі зарезервованих портів (номери портів менші 1024).

secure-rpc - вимагає RPC автентифікацію з цього комп'ютера.

kerberos - вимагає Kerberos-автентифікацію.

root_squash - цей прапорець забороняє суперкористувачу на вказаному хості довільні права доступу через мапування запитів від uid 0 сервера у uid 65534 (-2) на сервері. Цей uid може бути асоційований з користувачем nobody.

no_root_squash - не мапувати запити від uid 0.

ro - монтувати том лише для читання.

rw - монтувати том для читання-запису.

link_relative - конвертувати обсолютний символьний зв'язок (коли вміст звязку починається зі слеш) у відносний зв'язок через додавання певної кількості слешів для вказання на кореневу директорію серверу. Ця опція має сенс лише у тих випадках, коли монтується файлова система, що відрізняється від використовуваної.

link_absolute - залишає всі символьні зв'язки такими, якими вони є.

map_identity - повідомляє серверу про мапування клієнтських uid, gid у відповідні id серверу.

map_daemon - повідомляє NFS-сервер про те, що клієнт і сервер не розділяють одного uid/gid простору. Тоді nfsd будує список мапування id між клієнтом та сервером шляхом запитів через rpc ugidd клієнта.

Під час завантаження при прегляді демоном syslogd файлу exports виводяться відповідні повідомлення.

Слід зауважити, що назви хостів беруться з клієнтських IP-адрес шляхом зворотнього мапування, тому треба мати правильно сконфігурований resolver.

Інколи необхідно монтувати NFS-томи дуже часто. Вручну це робити недоцільно, тому треба мати засіб для монтування томів при завантаженні. Для цього є автомонтувальник (automounter). Це демон, що автоматично і прозоро для користувача монтує потрібні директорії і автоматично відмонтовує їх після деякого часу, коли ці томи не використовувались. Одною з корисних властивостей автомонтувальника є монтування певних томів з використанням альтернативних місць. Наприклад, можна тримати копію своїх X-програм і файлів підтримки на двох чи трьох хостах і мати підмонтованими ці хости через NFS. Використовуючи автомонтувальник, можна вказати всі три хости для

монтування в /usr/X386 і автомонтувальник буде монтувати ці томи доти, поки монтування не завершиться успішно.

В системі Linux автомонтувальник зазвичай називається amd. Ключі для використання можна знайти в супроводжувальних файлах до цієї програми.

8. ПОШТОВА СЛУЖБА В INTERNET

Sendmail - дуже потужна програма. Використання Sendmail+IDA знімає необхідність редагувати файл sendmail.cf і дозволяє адміністратору визначати конфігурацію адрес та шляхи трасування для кожного вузла через відносно прості файли підтримки, що називаються таблицями.

Жодним іншим транспортним агентом пересилання пошти не може бути виконана швидше і простіше, ніж за допомогою sendmail+IDA. Конфігурування, яке, зазвичай, є дуже складною задачею, значно спрощується.

Існуюча версія sendmail5.67b+IDA1.5 є доступна через анонімний FTP за адресою vixen.cso.uiuc.edu. Всі потрібні додатки для забезпечення її роботи під Linux вже внесені в вихідні тексти і ніяка їх модифікація не потрібна.

Всі файли конфігурації, що потрібні для отримання цих вихідних текстів містяться у файлі newspak-2.2.Tar.gz, який є доступний через анонімний FTP на sunsite.unc.edu в каталозі /pub/Linux/system/Mail.

Традиційно sendmail встановлюється через файл конфігурування системи (у загальному випадку /etc/sendmail.cf або /usr/lib/sendmail.cf), що не має ніяких спільних рис з якою-небудь мовою, що існує. Редагування файлу sendmail.cf для змін якихось параметрів потребує великого досвіду.

Sendmail+IDA забезпечує зберігання всіх опцій конфігурації у табличному вигляді з досить простим синтаксисом. Ці опції починають діяти після запуску m4 (макропроцесора) або dbm (процесора баз даних) над численною кількістю вихідних файлів через використання makefile.

Файл sendmail.cf визначає лише поведінку системи за замовчуванням, а все спеціальне налаштування здійснюється через низку додаткових таблиць, без безпосереднього редагування файлу sendmail.cf.

Файл sendmail.cf для системи sendmail+IDA не редагується безпосередньо системним адміністратором, але генерується з .m4 файлу конфігурації sendmail.m4. Він містить невелику кількість визначень, але вказує на таблиці, де виконується реальна робота. Загалом у цьому файлі визначаються лише такі параметри:

+ Імена шляхів та імен файлів, що використовуються на локальній системі.
+ Адреса(и) вузла для потреб електронної пошти.
+ Який mailer діє за замовчуванням.

Є велика кількість різних параметрів які можуть визначати поведінку нашого вузла або змінити опції компіляції вихідних текстів. Ці опції конфігурації містяться у файлі IDA/CF/OPTIONS в початковому каталозі.

Файл sendmail.m4 для мінімальної конфігурації (UUCP чи SMTP зі всією не локальною поштою, що передається до безпосереднього підлюченого хосту) може бути коротким: 10 чи 15 рядків, виключаючи коментарі.

"MailerTable" визначає спеціальну поведінку для віддалених головних EOM.

"Uucpxtable" примушує UUCP виконувати передавання пошти на головні вузли, що містяться у форматі DNS.

"Pathtable" зберігає UUCP-маршрути до віддалених головних вузлів чи області.

"Genericfrom" перетворює внутрішні адреси у видимі зовні.

"Xaliases" перетворює зовнішні адреси у/з внутрішні адреси.

"Decnetxtable" перетворює адреси формату "RFC-822" у формат DECnet.

Невелика кількість параметрів у файлі sendmail.m4 потрібна постійно, інші можуть бути проігноровані, якщо задовільняє варіант, що є прийнятий за замовчуванням. Приклад файлу sendmail.m4:

```
  dnl #------------ SAMPLE SENDMAIL.M4 FILE ------------- dnl
  # ( 'dnl' це m4 еквівалент закоментованого рядка)
  define(LOCAL MAILER DEF, mailers.linux)dnl # mailer
  define(POSTMASTERBOUNCE)dnl # почтмейстер
  define(PSEUDODOMAINS, BITNET UUCP)dnl # вузли без DNS
  dnl #------------------------------------------------ dnl #
  define(PSEUDONYMS, vstout.vbrew.com vstout.UUCP vbrew.com)
  dnl #
  define(DEFAULT HOST, vstout.vbrew.com)dnl # наше ім'я
  define(UUCPNAME, vstout)dnl # наше uucp ім'я
  dnl #
  dnl #------------------------------------------------ dnl #
  define(UUCPNODES, |uuname|sort|uniq)dnl # наші uucp сусіди
  define(BANGONLYUUCP)dnl # пошта оброблена коректно
  define(RELAY HOST, moria)dnl # хост
  define(RELAY MAILER, UUCP-A)dnl # ми досягли це через uucp
  dnl #
  dnl #------------------------------------------------ dnl #
  define(ALIASES, LIBDIR/aliases)dnl # системні назви
  define(DOMAINTABLE, LIBDIR/domaintable)dnl # домени
  define(PATHTABLE, LIBDIR/pathtable)dnl # база даних шляхів
  define(GENERICFROM, LIBDIR/generics)dnl #
  define(MAILERTABLE,  LIBDIR/mailertable)dnl  #  mailers  на
хості
  define(UUCPXTABLE, LIBDIR/uucpxtable)dnl #
  define(UUCPRELAYS, LIBDIR/uucprelays)dnl #
  dnl #
  dnl #------------------------------------------------ dnl #
  dnl # вставка реального коду, що виконує всю роботу
  dnl # (поставляється з вихідним текстом)
  dnl #
  include(Sendmail.mc)dnl # Це обов'язково
  dnl #
  dnl #------- кінець SENDMAIL.M4 ------
```

LIBDIR визначає каталог де sendmail+IDA буде шукати файли конфігурації, різні dbm-таблиці та спеціальні локальні налаштування. У стандартній поставці все це міститься в sendmail і не повинно явно встановлюватись у файлі sendmail.m4.

Слово dnl, що записано на початку рядка означає, що даний рядок є коментарем.

Щоб змінити розташування файлів підтримки потрібно забрати dnl з написаного вище рядка, поставити маршрут і повторно перевстановити sendmail.cf.

Більшість операційних систем мають програму для запезпечення обробки пошти. Під Linux потрібно явно визначити відповідний локальний mailer, бо він не входить до стандартної поставки цієї ОС, на відміну від інших варіантів Linux. Це зроблено рядком LOCAL MAILER DEF в файлі sendmail.m4.

Для забезпечення часто використовуваної програми deliver конфігураційною інформацією LOCAL MAILER DEF потрібно встановити як mailers.linux в каталозі вказаному в LIBDIR.

При дослідженні системних файлів протокола syslogd(8), є корисним локальний системний адміністратор пошти, що повинен по заголовку отриманого файлу з поштою визначити - чи була пошта відкинута із-за помилки користувача, чи помилки конфігурації на одній із систем. Визначення змінної POSTMASTERBOUNCE приводить до посилки копії кожного поверненого повідомлення особі, що є почтмейстером системи.

Визначення PSEUDODOMAINS запобігає безрезультатним DNS-пошукам.

PSEUDONYMS визначає список всіх хостів для яких локальна система буде приймати пошту. DEFAULT HOST визначає назву хоста, що буде появлятися у повідомленнях, адресованих з локального вузла. Важливо, щоб цей параметр був вказаний правильно, бо вся пошта, що повертається буде втрачена.

Часто використовуються системи що мають одне ім'я для DNS-цілей і інше для UUCP-цілей. UUCPNAME дозволяє визначити різні hostname, що будуть в заголовках пошти UUCP. UUCPNODES визначає команди, які повертають список імен хостів для систем, з якими зв'язуємося безпосередньо через UUCP-з'єднання. BANGIMPLIESUUCP і BANGONLYUUCP гарантують сервісну поведінку, яку використовують сьогодні на Internet.

Значна кількість адміністраторів не хочуть займатися забезпеченням зв'язку своєї мережі з будь-якою іншою у світі, а просто забезпечують відсилання пошти на інший хост. RELAY HOST визначає UUCP hostname. RELAY MAILER визначає mailer, який використовується там, щоб передати повідомлення.

Файл Sendmail.mc вміщує шаблон того, що стає файлом sendmail.cf.

У разі невикористання будь-якої із додаткових dbm-таблиць, sendmail+IDA поставляє пошту через DEFAULT MAILER (і можливо RELAY HOST і RELAY MAILER), що визначений в файлі sendmail.m4, який використовують для генерування mail.cf. Це легко відмінити через зміну domaintable або uucpxtable.

Фактично всі системи мають встановлювати DEFAULT HOST і PSEUDONYMS, які визначають канонічну назву вузла і псевдоніми. Тобто не потрібно встановлювати значення relay mailer за замовчуванням, бо це працює автоматично.

UUCP хост буде ймовірно також встановлювати UUCPNAME до імені UUCP. Транспорт пошти, який потрібно використовувати, визначається в RELAY MAILER і має бути для UUCP.

Sendmail+IDA забезпечує ряд таблиць, які дозволяють змінювати поведінку sendmail за замовчуванням (визначених в файлі sendmail.m4) і визначати спеціальну поведінку для унікальних ситуацій, віддалених систем і мереж.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Призначення та вміст конфігураційних файлів, що відповідають за термінальні з'єднання (/etc/inittab, /etc/rc.d/rc.serial, /etc/termcap).
2. Особливості застосування утиліти agetty і команди talk.
3. Призначення та вміст конфігураційних файлів, що відповідають за TCP/IP з'єднання (/etc/rc.d/rc.inet1, /etc/rc.d/rc.inet2).
4. Особливості застосування утиліт ifconfig, route та команди netstat.
5. Який порядок конфігурування ядра системи?
6. Особливості застосування команди make.
7. Структура Partition Table твердого диска.
8. Призначення та вміст конфігураційних файлів, що відповідають за завантаження операційної системи.
9. Особливості застосування команд fdisk, lilo, df, du.
10. Призначення та вміст конфігураційних файлів, що відповідають за SLIP- та PPP-з'єднання (/etc/diphosts, /etc/ppp/options).
11. Особливості застосування утиліт dip та pppd.
12. Призначення та вміст конфігураційних файлів, що відповідають за підтримку Internet (/etc/inetd.conf, /etc/services, /etc/protocols, /etc/rpc, /etc/rc.local).
13. Особливості застосування програми-суперсервера inetd і команд telnet та ftp.
14. Призначення та вміст конфігураційних файлів, що відповідають за підтримку NFS (/etc/exports, /etc/fstab).
15. Особливості застосування команди mount та демонів /usr/sbin/rpc.mountd і /usr/sbin/rpc.nfsd.
16. Призначення та вміст конфігураційного файлу, що відповідає за підтримку поштової служби в Internet (/etc/sendmail.cf).
17. Особливості застосування команди mail при організації обміну електронними повідомленнями.

СПИСОК ЛІТЕРАТУРИ

1. Робачевский А.М. Операционная система UNIX. – СПб.: BHV – Санкт-Петербург, 2005. – 528 с.
2. Немет Э., Снайдер Г., Хейн Т. Руководство администратора Linux. – М.: Вильямс, 2003. – 880 с.
3. Скловская С. Л. Команды Linux. Справочник. – СПб.: ДиаСофт ЮП, 2004. – 848 с.
4. Немнюгин С. и др. Эффективная работа UNIX. – СПб.: Питер, 2001. – 688 с.

ДОДАТОК 1

Список дескрипторів (описувачів) файлу termcap

Ім'я	Тип	Опис
BT	str	Клавіша зворотної табуляції (для red).
C1	bool	Тільки основний колір (немає фонового).
C2	bool	Основний і фоновий кольори встановлюються одночасно.
CS	num	Тип перемикання регістрів РУС/ЛАТ.
CY	bool	Термінал має кирилицю.
Cb	str	Встановлення кольору тла за номером.
Cf	str	Встановлення основного кольору за номером.
DC	str	Код клавіші "виключити символ" (для red).
DL	str	Код клавіші "виключити рядок" (для red).
ER	str	Код клавіші "очищення екрана" (для red).
IC	str	Код клавіші "вставити символ" (для red).
IL	str	Код клавіші "вставити рядок" (для red).
LC	bool	Термінал має прописні й малі літери.
Mb	str	Карта кольорів тла.
Mf	str	Карта основних кольорів.
Nb	num	Число кольорів тла.
Nf	num	Число основних кольорів.
PH	num	Тип відеопам'яті, що прямо адресується .
UC	bool	Термінал має лише прописні букви.
ae	str p	Кінець роботи з альтернативним шрифтовим набором (курсивом).
al	str p*	Вставити чистий рядок.
am	bool	При досягненні кінця поточного рядка курсор автоматично переміщається на початок наступного рядка.
as	str p	Початок роботи з альтернативним шрифтовим набором (курсивом).
bc	str	Переклад курсору на одну позицію назад (якщо не \b);
bl	str	Звуковий сигнал (якщо не \7).
bs	bool	Виведення символу \b приводить до переміщення курсору на одну позицію назад.
bt	str p	Зворотна табуляція.
bw	bool	Переміщення курсору на одну позицію назад з першої позиції рядка викликає перехід у кінець попереднього рядка.
cd	str p*	Стирання інформації від поточної позиції до кінця екрана.
ce	str p*	Стирання інформації від поточної позиції до кінця рядка.
ch	str p	Адресація курсору в межах поточного рядка (по горизонталі).
cl	str p	Очищення екрана.
cm	str p	Адресація курсору.
co	num	Число позицій у рядку на екрані.
cr	str p*	Повернення в початок рядка (якщо не \r);
cs	str p	Зміна області роллерування на екрані (параметри аналогічні cm);
ct	str	Стирання всіх маркерів табуляції.
cu	str	Код символу - альтернативного курсору (яркою оцінки);

cv	str p	Адресація курсору по вертикалі.
cw	str	Встановлення вікна для виведення.
dB	num	Затримка у мілісекундах для символу \b (BS).
dC	num	Затримка у мілісекундах для символу \r (CR).
dF	num	Затримка у мілісекундах для символу \f (FF).
dN	num	Затримка у мілісекундах для символу \n (NL).
dT	num	Затримка у мілісекундах для символу \t (TAB).
da	bool	Термінал зберігає інформацію про рядки, зсунуті нагору за край екрана.
db	bool	Термінал зберігає інформацію про рядки, зсунуті униз за край екрана.
dc	str p*	Виключити символ у поточній позиції.
dl	str p*	Видалити поточний рядок.
dm	str	Перехід у режим видалення символів.
do	str p	Переклад курсору вниз на один рядок.
ds	str	Скасувати інформаційний рядок (рядок стану дисплея).
ec	str	Очистити символ.
ed	str	Кінець режиму видалення символів.
ei	str	Кінець режиму вставляння символів.
eo	bool	Пробіл стирає символ у поточній позиції.
eo	bool	Накладення стираються пробілом.
es	bool	В інформаційному рядку можна користуватися символом escape (\33).
f,	str	"," у режимі альтернативної клавіатури.
f-	str	"-" у режимі альтернативної клавіатури.
f.	str	"." у режимі альтернативної клавіатури.
f0	str	"0" у режимі альтернативної клавіатури.
f1	str	"1" у режимі альтернативної клавіатури.
f2	str	"2" у режимі альтернативної клавіатури.
f3	str	"3" у режимі альтернативної клавіатури.
f4	str	"4" у режимі альтернативної клавіатури.
f5	str	"5" у режимі альтернативної клавіатури.
f6	str	"6" у режимі альтернативної клавіатури.
f7	str	"7" у режимі альтернативної клавіатури.
f8	str	"8" у режимі альтернативної клавіатури.
f9	str	"9" у режимі альтернативної клавіатури.
ff	str *p	Якщо термінал є друкувальним пристроєм, команда прогону аркуша (за замовчуванням \f).
fs	str	Повернутися з інформаційного рядка (після ts).
g1	str	Карта псевдографічних символів тонких рамок.
g2	str	Карта псевдографічних символів подвійних рамок.
g3	str	Карта псевдографічних символів подвійних вертикальних і тонких горизонтальних рамок.
g4	str	Карта псевдографічних символів тонких вертикальних і подвійних горизонтальних рамок.
g5	str	Карта псевдографічних стрілок.

g6	str	Карта псевдографічних блоків.
ge	str	Кінець режиму псевдографіки.
gn	bool	Невизначений тип лінії (модем).
gs	str	Включити режим псевдографіки.
gt	str	Карта графічних символів (для red).
hR	str	Ім'я help-файлу для red.
hc	bool	Термінал є друкувальним пристроєм.
hd	str	Переведення на 1/2 рядка вниз.
ho	str	Переведення курсору у першу позицію першого рядка.
hs	bool	Термінал має інформаційний рядок (зазвичай 25-й рядок).
hu	str	Перехід на 1/2 рядка нагору.
hz	bool	Термінал Hazeltine не друкує ~, використати ~ не можна.
ic	str p	Вставити символ у поточній позиції.
if	str	Для ініціалізації термінала видати на нього вміст зазначеного файлу.
im	str p	Перехід у режиму вставляння символів.
in	bool	У режимі вставляння символи пробіл та порожньо розрізняються.
ip	str p*	Символ, за яким іде зазначена послідовність кодів вставляється в поточній позиції.
is	str	Послідовність кодів для ініціалізації термінала.
it	str	Початкові табуляції кожні n позицій.
k0	str	Функціональна клавіша 0.
k1	str	Функціональна клавіша 1.
k2	str	Функціональна клавіша 2.
k3	str	Функціональна клавіша 3.
k4	str	Функціональна клавіша 4.
k5	str	Функціональна клавіша 5.
k6	str	Функціональна клавіша 6.
k7	str	Функціональна клавіша 7.
k8	str	Функціональна клавіша 8.
k9	str	Функціональна клавіша 9.
kA	str	Клавіша вставити рядок.
kB	str	Клавіша зворотної табуляції.
kC	str	Клавіша очищення екрана.
kD	str	Клавіша видалення символу.
kE	str	Клавіша очищення до кінця рядка.
kF	str	Клавіша прокручування вперед.
kI	str	Клавіша вставляння символу (INSERT).
kL	str	Клавіша знищення рядка.
kN	str	Клавіша наступна сторінка.
kP	str	Клавіша попередня сторінка.
kR	str	Клавіша прокручування назад.
kS	str	Клавіша очищення до кінця екрана.
kT	str	Клавіша встановлення табуляції.
ka	str	Клавіша очищення всіх табуляцій.
kb	str	Код клавіші повернення на крок.
kd	str	Код клавіші курсор униз.

ke	str	Вимкнення режиму додаткової клавіатури.
kh	str	Код клавіші home;
kl	str	Код клавіші курсор уліво;
km	bool	Термінал має клавішу meta-shift.
kn	num	Кількість клавіш функціональної клавіатури, відмінних від "0" - "9", ".", "-".
ko	str	Опис клавіш функціональної клавіатури, відмінних від "0" - "9", ".", "-".
kr	str	Код клавіші курсор вправо.
ks	str	Увімкнення режиму додаткової клавіатури.
kt	str	Клавіша очищення маркера табуляції.
ku	str	Клавіша стрілка нагору.
l0	str	Назва функціональної клавіші 0.
l1	str	Назва функціональної клавіші 1.
l2	str	Назва функціональної клавіші 2.
l3	str	Назва функціональної клавіші 3.
l4	str	Назва функціональної клавіші 4.
l5	str	Назва функціональної клавіші 5.
l6	str	Назва функціональної клавіші 6.
l7	str	Назва функціональної клавіші 7.
l8	str	Назва функціональної клавіші 8.
l9	str	Назва функціональної клавіші 9.
le	str	Переклад курсору на одну позицію вліво.
li	num	Число рядків на екрані термінала.
ll	str	Переведення курсору у першу позицію останнього рядка.
lm	num	Число рядків пам'яті дисплея.
m1	str	Атрибути для рамки (для red).
m2	str	Атрибути для діагностик (для red).
m3	str	Атрибути для запрошення (для red).
m4	str	Атрибути для инф.повідомлень (для red).
mb	str	Включення режиму миготливих символів.
md	str	Включення режиму яскравих символів.
me	str	Скасування всіх режимів виділення тексту ("mb", "md", "ml", "mr", "us", "so").
mh	str	Включення режиму тьмяних символів.
mi	bool	Можливість переміщення курсору у режимі вставляння тексту.
mk	str	Увімкнення режиму невидимих символів.
ml	str	Захист пам'яті над курсором.
mm	str	Увімкнення режиму метасимволов.
mo	str	Вимикання режиму метасимволов.
mp	str	Увімкнення режиму захисту.
mr	str	Увімкнення режиму інверсних символів.
ms	bool	Можливість переміщення курсору у режимі виділення тексту.
mu	str	Розблокування пам'яті (після ml).
nb	bool	Заборона використання символу bell (код \7). Необхідний, якщо даний символ відображається на екрані.

nc	bool	Заборона використання символу \r, неправильно працює повернення каретки.
nd	str	Переведення курсору на одну позицію вправо.
nl	str p*	Перехід у початок наступного рядка (якщо не \n).
ns	bool	Термінал не має прокручування .
nw	str	Команда CR-LF.
os	bool	Термінал допускає накладення символів.
pb	num	Мін.швидкість, що вимагає заповнювачів.
pc	str	Код символу-заповнювача (якщо не \0).
pf	str	Вимикання друкувального пристрою.
po	str	Увімкнення друкувального пристрою.
ps	str	Друк вмісту екрана.
pt	bool	Термінал має програмувальний механізм встановлення табуляційної сітки.
rc	str	Відновити положення курсору (після sc).
rf	str	Для скасування ініціалізації видати вміст зазначеного файлу.
rs	str	Для скасування ініціалізації видати зазначену послідовність символів.
sc	str	Запам'ятати поточне положення курсору.
se	str	Кінець стандартного режиму виділення тексту.
sf	str p	Прокрутити текст уперед.
sg	num	Число символів-заповнювачів для команд so і se.
so	str	Установити стандартний режим виділення тексту.
sr	str p	Прокрутити текст назад (знизу нагору).
st	str	Установити табуляцію в поточній позиції.
sw	str	Установити вікно (область виведення на термінал).
ta	str p	Команда табуляції (якщо не \t).
tc	str	Доповнити опис властивостей термінала з опису пристрою із зазначеним ім'ям. Даний опис має бути останнім у записі.
te	str	Скасувати режим адресації курсору (cm).
ti	str	Увійти в режим адресації курсору.
ts	str	Перейти в інформаційний рядок.
uc	str	Підкреслити один символ.
ue	str	Кінець режиму підкреслення.
ug	num	Число символів-заповнювачів для команд us і uc.
ul	bool	Термінал має можливість підкреслення символів.
up	str	Переведення курсору на рядок нагору.
us	str	Увімкнути режим підкреслення виведених символів.
vb	str	Видимий аналог звукового сигналу. (Переведення термінала у режим телетайпа.)
ve	str	Вимикання екранного режиму.
vi	str	Увімкнення екранного режиму.
vs	str	Переведення термінала в екранний режим.
vt	num	Кількість віртуальних терміналів.
ws	num	Довжина рядка стану.
xb	bool	Термінал Beehive (f1=ESC, f2=CTRL/C).

xn bool Коли надрукований останній символ у рядку, перехід на наступний рядок здійснюється не відразу, а лише після друку ще одного символу. Цей прапорець уживається лише разом з am.
xo bool Термінал використовує (CTRL/Q, CTRL/S).
xr bool Код \r діє як \n\r.
xs bool Виділення зберігається при накладенні.
xt bool Заборона використання табуляції (\t) (виведення коду табуляції змінює інформацію на екрані).

Типи описувачів

bool Логічний. Вказує на те, що даний пристрій має деяку властивість.
num Числовий. Задає розмір екрана термінала, тривалість затримок при передачі інформації тощо.
str Символьний. Визначає послідовність кодів, виведення яких приводить до виконання якої-небудь специфічної для даного пристрою функції.

ДОДАТОК 2

Особливості застосування деяких команд та утиліт

1. Команда telnet

telnet *параметри система*

Дозволяє встановити з'єднання з віддаленою системою за допомогою протоколу TELNET.

Параметри:

-a Спроба автоматичної реєстрації на віддаленій системі.

2. Команда talk

talk *користувач термінал*

Ведення інтерактивного діалогу двома користувачами. При виконанні команди екран розбивається навпіл. Текст, який вводиться користувачем, буде з'являтися у верхній половині екрану, а повідомлення іншого користувача – у нижній. Для завершення роботи слід натиснути комбінацію клавіш <Ctrl+c>.

Приклад:

Спочатку командою **who** чи **w** визначити, з яким користувачем і терміналом можна зв'язатись. Потім командою **talk** задати користувача і термінал:

```
$talk ki-4 ttyp1
```

У відповідь користувач, який отримав виклик, мусить підтвердити готовність до діалогу командою **talk**, у якій вказати користувача, що був ініціатором зв'язку:

```
$talk ksm-5
```

3. Команда w

w *параметри користувачі*

Виведення інформації про систему: списку користувачів, під'єднаних в даний момент до системи, статистики використання системи, а також задач, що виконуються користувачами. Команда є комбінацією команд **who, ps –a** та **uptime**.

Інформація про систему виводиться в заголовку і включає: поточний час, час, що минув після останнього перезавантаження системи, кількість користувачів, що працюють в даний момент в системі, а також середнє завантаження системи за останні 1,5 та 15 хвилин.

Інформація, яка виводиться про користувачів, включає: системний ідентифікатор користувача, ім'я термінала, ім'я віддаленої системи, час роботи в системі, час неактивності, JCPU (час, що використовується всіма процесами даного термінала), PCPU (час, що використовується поточним процесом) та командний рядок поточного процесу.

Параметри:

-h Заборона виведення заголовка.

-u Заборона виведення інформації про PCPU та командні рядки..

-s Заборона виведення інформації про час роботи в системі JCPU і PCPU.

-f Виведення поля **from** (ім'я віддаленої системи).

4. Команда who

who *параметри файл*

Виведення списку користувачів, підключених у даний момент до системи.

Параметри:

am I Виведення інформації про системний ідентифікатор користувача.

-a Використовувати всі зазначені нижче параметри.

-b Виведення дати і часу останнього перезавантаження системи.

-d Виведення списку користувачів, відключених через тривалу неактивність.

-H Виведення на початку списку заголовків стовпчиків.

-I Виведення списку ліній, доступних для входу в систему.

-n*n* Виведення в одному рядку інформації про *n* користувачів.

-p Виведення списку процесів, запущених процесом **init** і все ще активних.

-q Короткий формат; виводяться тільки системні ідентифікатори користувачів.

-r Виведення рівня запуску системи.

-s Виведення системного ідентифікатора користувача, термінала і часу неактивності (формат, що використовується за замовчуванням).

-t Виведення часу, коли останній раз за допомогою команди **clock** налаштовувався системний годинник.

-T Виведення стану кожного термінала:

+ Термінал доступний для виведення всім користувачам.

- Термінал доступний для виведення лише системному адміністратору.

? Помилка при визначенні стану термінала.

-u Виведення часу неактивності для кожного термінала.

5. Утиліта ifconfig

ifconfig *параметри*

Конфігурування мережного інтерфейсу. Без параметрів відображає стан поточних активних мережних інтерфейсів. Запускається з каталога **sbin**. Для переривання перегляду слід натиснути клавішу **<q>**.

Приклад:

```
$ifconfig |less
```

В результаті виконання команди на екран буде виведена така інформація:

Inet addr	IP-адреса.
Bcast	Адреса для широкомовного розсилання повідомлень.
Mask	Маска підмережі.
UP	Інтерфейс активний.
MTU	Максимальний розмір блоку передавання.
Metric	Ціна посилання пакета вказаним інтерфейсом.
RX packets	Кількість отриманих пакетів.
TX packets	Кількість переданих пакетів.
Errors	Кількість помилкових пакетів.
Dropped	Кількість відкинутих пакетів.
Overruns	Кількість неопрацьованих пакетів.
Frame	Кількість кадрів (блоків даних) .
Carrier	Кількість носіїв.
Collisions	Кількість конфліктів.
Interrupt	Кількість переривань.
Base addr	Кількість базових адрес.

6. Утиліта route

route *параметри*

Програма прокладання статичних маршрутів через інтерфейси, налаштовані та активізовані програмою **ifconfig**. Без параметрів (або з параметром -n) – виведення поточної таблиці маршрутизації протоколу IP. Запускається з каталога **sbin**.

Приклад:

```
$route
```

В результаті виконання команди на екран буде виведена така інформація:

Destination	IP-адреса кінцевого вузла маршруту.
Gateway	Ім’я або IP-адреса шлюзу, що використовується маршрутом (якщо шлюз не вказаний, відображається символ *).
Genmask	Маска мережі для маршруту.
Flags	Ознаки маршруту (U –маршрут активний, H – вузол, G – шлюз, D – динамічна маршрутизація, M – змінений) .
Mettrik	Ціна шляху.
Ref	Кількість інших маршрутів, що відносяться до даного.
Use	Скільки разів використовувався даний елемент таблиці маршрутизації.
Ifase	Мережний інтерфейс, через який проходить маршрут.

7. Команда netstat

netstat *параметри*

Перевірка стану мережі. Без параметрів – список активних мережних з’єднань. Для переривання перегляду слід натиснути клавішу **<q**>.

Параметри:

-a Виведення інформації про всі з’єднання.
-i Виведення статистики про всі мережні пристрої (як утиліта **ifconfig**).
-c Виведення інформації про поточний стан мережі з періодичністю в одну секунду (до переривання).
-n Виведення інформації про віддалені і локальні адреси та порти.
-o Виведення інформації про стан таймерів та додаткової інформації.
-r Виведення таблиці маршрутизації (як утиліта **route**).
-t Виведення інформації лише про TCP-сокети.
-u Виведення інформації лише про UDP-сокети.
-v Версія програми **netstat**.
-x Виведення інформації про доменні сокети Linux.

Приклад 1:

```
$netstat -a |less
```

В результаті виконання команди на екран буде виведена така інформація:

1. Розділ “Активні під’єднання Internet”.

Proto Протокол (TCP чи UDP) .
Recv-Q Кількість отриманих сокетом байтів, що не скопійовані програмою користувача.
Send-Q Кількість байтів, відправлених, але не розпізнаних віддаленим вузлом.
Local Adress Ім’я і порт локального вузла з’єднання (якщо не вказано -n, то IP-адреса транслюється у канонічне ім’я, а ім’я порту – в назву сервісу) .
Foreign Address Ім’я віддаленого вузла з’єднання (якщо не вказано -n, то IP-адреса транслюється у канонічне ім’я, а ім’я порту – у назву сервісу).
State Поточний стан сокета:
ESTABLISHED – з’єднання встановлено;
SYN_SENT – спроба встановлення з’єднання сокета з віддаленим вузлом;
SYN_RECV – з’єднання встановлюється;
FIN_WAIT1 – сокет закрився і чекає завершення з’єднання;
FIN_WAIT2 – з’єднання закрите, очікується закриття з іншої сторони;
TIME_WAIT – сокет закрився та очікує від віддаленого сокета припинення передачі;
CLOSED – сокет не використовується;
CLOSE_WAIT – віддалений сокет закрив з’єднання, локальний сокет очікує закриття з’єднання;
LOAST_ACK – з’єднання перервалось і сокет закрився, локальний сокет очікує підтвердження;

LISTEN – сокет прослуховує мережу для спроби встановлення нового з'єднання;
UNKNOWN – невідомий стан сокета;
USER – ім'я користувача, що володіє сокетом.

2. Розділ "Активні доменні сокети Linux".

Proto Протокол, що використовується сокетом.
RefCnt Кількість процесів, підключених до сокета.
Flags Ознаки.
Type Режим доступу до сокета:
OCK_DGARM – режим дейтаграми, без з'єднання;
OCK_STREAM – режим потоку зі з'єднанням;
OCK_RAW – режим, що не обробляється;
OCK_RDM – режим надійної доставки інформації;
OCK_SEQPACKET – режим послідовних посилань;
UNKNOWN – невідомий режим.
State Поточний стан сокета:
FREE – сокет не розподілений;
LISTENING – сокет очікує запитів на встановлення з'єднання;
UNCONNECTED – сокет намагається встановити з'єднання;
CONNECTED – сокет встановив з'єднання;
DISCONNECTION – сокет намагається розірвати з'єднання;
UNKNOWN – стан невідомий.
Path Шлях, що використовується іншим процесом для під'єднання до сокета.

Приклад 2:

```
$netstat -i |less
```

В результаті виконання команди на екран буде виведена таблиця з такою інформацією:
Ifase Ім'я мережного інтерфейсу.
Відсортовані параметри мережі (порівняти з результатом виконання утиліти **ifconfig**).
Flags Ознаки:
A – інтерфейс отримує багатоадресні пакети;
B – інтерфейс отримує пакети широкомовного розсилання;
D – увімкнений режим налагодження;
L – інтерфейс зворотньої петлі;
M – інтерфейс перебуває у змішаному режимі;
N – інтерфейс не обробляє трейлери в пакеті;
O – для цього інтерфейсу відключений протокол ARP;
P – інтерфейс використовується для канального з'єднання типу "point-to-point";
R – інтерфейс запущений;
U – інтерфейс активізований.

Приклад 3:

```
$netstat -o |less
```

В кінці кожного рядка розділу “Активні під’єднання Internet” додається інформація про увімкнення/вимкнення таймера; у дужках – час, що залишився до таймауту та кількість спроб.

8. Команда df

df *параметри файлова_система*

Виведення інформації про кількість вільного місця на файловій системі чи на файловій системі, зазначеній у параметрі *файлова_система*. Інші параметри дозволяють вивести кількість вільного місця (у кілобайтах чи блоках) або загальну місткість диска.

Параметри:

-b Виведення розміру вільного місця на диску у кілобайтах.

-e Виведення числа, що показує, скільки ще файлів може бути створено. Цей параметр використовується не у всіх системах.

-F *тип* Використовується для одержання інформації про незмонтовані файлові системи зазначеного *типу.* (Список використовуваних типів файлових систем у деяких версіях Linux може бути знайдений у файлі **/etc/vfstab**).

-g Повертає всю структуру **statvfs** для всіх незмонтованих файлових систем.

-k Виведення розміру зайнятого місця у кілобайтах.

-l Виведення інформації лише про локальні файлові системи.

-n Виведення *типу* файлової системи. Цей параметр використовується не у всіх системах.

-t Виводить розмір як вільного, так і зайнятого місця. Цей параметр використовується не у всіх системах

9. Команда du

du *параметри файли каталоги*

Виведення розміру простору на диску, зайнятого каталогом (з усіма його підкаталогами) у блоках (зазвичай 1 блок складає 512 байтів чи 1024 байтів). За замовчуванням виводиться інформація про поточний каталог.

Параметри:

-a Виведення інформації не лише про каталоги, але і про файли.

-r Виведення інформації про файли і каталоги, які команда **du** не може відкрити.

-s Виведення лише загального підсумку, без відображення проміжної інформації.

10. Команда mount

mount *параметри пристрій каталог*

Команда системного адміністрування. Монтування файлової системи. Команда **mount** сповіщає систему, що файлова система, яка монтується, присутня на *пристрої*. Точкою монтування є *каталог*, що має існувати і бути порожнім. Після монтування цей каталог стає коренем змонтованої файлової системи. Команда **mount** без аргументів перелічує всі підмонтовані файлові системи, їх точки монтування, відзначає системи, доступні тільки для читання, а також повідомляє дату монтування. Команда **mount** доступна лише привілейованому користувачеві.

Параметри:

-a Монтувати усі файлові системи, перераховані у файлі */etc/fstab*. **Примітка:** це єдиний параметр, що не сполучимо з аргументами *пристрій* і *вузол*.

-f Імітація монтування. Виконати всі перевірочні дії для пристрою і каталогу, але саме монтування не робити.

-n Не реєструвати монтування в */etc/fstab*.

-o *option*

Примітка: це єдиний параметр **mount,** що вимагає наявності аргументу *пристрій* або *вузол*. Аргумент *options* може приймати такі значення:

async Дозволити асинхронний ввід/вивід для пристрою.

auto Дозволити монтування з параметром **-a**.

defaults Використовувати значення за замовчуванням для всіх режимів (**async**, **auto, dev, exec, nouser, rw, suid**).

dev Інтерпретувати всі спеціальні пристрої, що існують у системі.

exec Дозволити виконання двійкових файлів.

noauto Заборонити монтування з параметром **-a**.

nodev Не інтерпретувати спеціальні пристрої, що існують у системі.

noexec Заборонити виконання файлів.

nosuid Не брати до уваги біти **suid** і **sgid.**

nouser Дозволити доступ до файлової системи, яка монтується, тільки привілейованому користувачеві.

remount Вважати, що система вже підмонтована, і перемонтувати її.

ro Дозволити доступ тільки для читання.

rw Дозволити доступ для читання і запису.

suid Брати до уваги біти **suid i sgid**.

sync Синхронний ввід/вивід для пристрою.

user Дозволити непривілейованим користувачам монтувати файлову систему. Слід звернути увагу, що значеннями за замовчуванням для такої системи будуть **nodev**, **noexec** і **nosuid,** якщо не зазначене інше.

check=relaxed | normal | strict Дозволяє вказати, наскільки строго має регулюватися інтеграція файлової системи **MS-DOS**.

conv=binary | text | auto Визначити метод перетворення файлів на файлових системах **MS-DOS** і **ISO 9660**.

debug Включити налагодження для файлових систем **MS-DOS** і **ext2fs.**

errors=continue | remount | ro | panic Дія, що виконується при виникненні помилки. Тільки для файлових систем **ext2fs**.

-r Монтування в режимі тільки для читання.

-t *type* Вказати тип файлової системи. Можливі значення: **minix, ext, ext2, xiafs, hpfs, msdos, umsdos, vfat, proc, nfs, iso9660, smbfs, ncpfs, affs, ufs, romfs, sysv**, **xenix** і **coherent.** Слід звернути увагу, що **ext** і **xiafs** доступно тільки для ядер версій менше 2.1.21 і що замість **Xenix** і **coherent** варто використовувати **sysv**.

-v Діагностика монтування.

-w Монтувати в режимі читання/запису. Режим за замовчуванням.

Файли:

/etc/fstab Список файлових систем, що монтуються і параметрів монтування.

/etc/mtab Список підмонтованих у даний момент систем і параметрів монтування.

НАВЧАЛЬНЕ ВИДАННЯ

АДМІНІСТРУВАННЯ КОМП'ЮТЕРНИХ СИСТЕМ І МЕРЕЖ

МЕТОДИЧНІ ВКАЗІВКИ ДО САМОСТІЙНОЇ РОБОТИ

для студентів першого (бакалаврського) рівня вищої освіти
спеціальності 123 "Комп'ютерна інженерія"

Укладач: Хомуляк Мирослав Олегович

Редактор

Комп'ютерне верстання

Підписано до друку 2020.
Формат 60x84 1/16. Папір офсетний. Друк на різографі.
Умовн. друк. арк. . Обл.-вид. арк. .
Наклад прим. Зам. .

Видавництво Національного університету „Львівська політехніка"
Реєстраційне свідоцтво ДК № 751 від 27.12.2001 р.

Поліграфічний центр Видавництва
Національного університету „Львівська політехніка"

вул.Ф. Колесси, 2, Львів, 79000