МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

І. П. Голубєва, В. А. Казміренко, Ю. В. Прокопенко

Інформатика-1

Персональний комп’ютер

Лабораторний практикум

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра за спеціальністю
G5 Електроніка, електронні комунікації, приладобудування та радіотехніка

Електронне мережеве навчальне видання

Київ
КПІ ім. Ігоря Сікорського
2025

УДК 004.021 (045)
Г60

Автори: *Голубєва Ірина Петрівна*, канд. техн. наук, доц.
Казміренко Віктор Анатолійович, канд. техн. наук, доц.
Прокопенко Юрій Васильович, д-р техн. наук, проф.

Рецензент *Діденко Ю. В.*, канд. техн. наук, доц.,
доцент кафедри мікроелектроніки, КПІ ім. Ігоря Сікорського

Відповідальний редактор *Тимофєєв В. І.*, д-р техн. наук, проф.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол №9 від 26.06.2025 р.)
за поданням Вченої ради факультету електроніки
(протокол № 05/2025 від19.05.2025 р.)

Голубєва І. П.

Г60 Інформатика-1. Персональний комп’ютер [Електронний ресурс]: лаб. практикум: навч. посіб. для здобувачів ступеня бакалавра за освіт. програмою «Електронні мікро- і наносистеми та технології» спец. 176 Мікро- та наносистемна техніка / І. П. Голубєва, В. А. Казміренко, Ю. В. Прокопенко; КПІ ім. Ігоря Сікорського. – Електрон. текст. дані (1 файл: 2,00 МБ). – Київ : КПІ ім. Ігоря Сікорського, 2025. – 110 с.

Викладено відомості про особливості роботи з віддаленим сервером під керуванням UNIX-подібної операційної системи за допомогою емуляторів терміналу для персонального комп’ютера. Наведено інформацію та завдання для виконання, які мають сприяти закріпленню навичок роботи з UNIX-подібними операційними системами. Може стати у нагоді студентам інженерних спеціальностей, які користуються у роботі віддаленими та розподіленими обчислювальними системами.

УДК 004.021 (045)

Реєстр. № 24/25-622. Обсяг 5,00 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Перемоги, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів
і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.

Зміст

ВСТУП

У сучасному світі інженерна діяльність нерозривно пов’язана з активним використанням обчислювальної техніки. Серед широкого спектра програмного забезпечення, необхідного для ефективної роботи інженера, особливу нішу займають потужні програмні комплекси, призначені для розв’язання складних і комплексних задач. До таких задач належать, зокрема, розробка та моделювання електронних схем різної складності, проведення чисельних розрахунків у галузі прикладної електродинаміки, виконання ресурсомістких інженерних розрахунків та багато інших.

Дуже часто трапляється так, що апаратні вимоги подібного програмного забезпечення є надзвичайно високими, а складність процедури налаштування оптимального робочого середовища робить практично неможливою його локальну установку на персональні комп’ютери інженерів-розробників. У таких випадках оптимальним рішенням стає розгортання подібних систем на високопродуктивних серверних платформах, доступ до яких користувачі здійснюють за допомогою терміналів віддаленого доступу.

Хоча такий підхід до організації роботи з обчислювальною технікою був поширений з самого початку ери комп’ютеризації, з масовим поширенням персональних комп’ютерів він дещо відійшов на другий план. Внаслідок цього значна частина навіть досвідчених користувачів персональних комп’ютерів може не мати достатнього практичного досвіду роботи з віддаленими терміналами та специфікою взаємодії з серверними операційними системами.

На сьогоднішній день переважна більшість парку настільних комп’ютерів, а саме приблизно три з чотирьох машин, функціонують під управлінням операційної системи Windows різних версій. Водночас, на серверних платформах, які спеціалізуються на виконанні складних інженерних розрахунків та підтримці відповідного програмного забезпечення, домінує використання UNIX-подібних операційних систем, серед яких особливе поширення отримала

Linux. Це твердження є справедливим і для пакетів програмного забезпечення від лідера галузі – компанії Cadence Design Systems, які є де-факто стандартом у сфері проектування інтегральних мікросхем.

Для користувача, який не має належного досвіду роботи з UNIX-подібними системами, їх робоче середовище та принципи організації роботи можуть викликати певні труднощі. Ці труднощі, зокрема, можуть бути пов’язані з істотними відмінностями в організації файлової системи, специфікою запуску програмного забезпечення через командний рядок, питаннями керування правами доступу до файлів та каталогів на віддаленій машині, а також особливостями конфігурації мережевого оточення.

Цей навчальний посібник містить необхідні теоретичні відомості та практичні завдання для виконання лабораторних робіт, які стануть незамінним інструментом у процесі опанування навичок ефективної роботи з програмним забезпеченням, що функціонує на віддаленому сервері, використовуючи термінали віддаленого доступу. У посібнику детально описано процес налаштування необхідного робочого середовища на локальній машині користувача, розглянуто найбільш уживані та важливі команди для навігації та роботи в терміналі UNIX-подібної системи. Також розглянуто процес написання, компіляції (транслювання) та налагодження програм, написаних мовою програмування C, безпосередньо на віддаленому сервері за допомогою інструментів командного рядка.

Запропонований підхід до організації навчального процесу дозволяє ефективно організувати повноцінне інженерне робоче місце навіть на персональному комп’ютері з мінімальними апаратними характеристиками, оскільки вимагає встановлення лише кількох невеликих за розміром програмних засобів для забезпечення віддаленого доступу. Це робить навчання доступним та гнучким, дозволяючи студентам та інженерам зосередитися на опануванні професійних інструментів без зайвих витрат на потужне локальне обладнання.

Підготовка робочого середовища:

установка необхідного програмного забезпечення

У курсі інформатики основну увагу буде приділено вивченню основ програмування мовою *C*. Охочі можуть встановити на свій персональний комп'ютер окремі засоби розробки, такі як текстовий редактор, компілятор, засоби налагодження, або інтегроване середовище розробки (Integrated Development Environment – IDE). Процес встановлення та детального налаштування кожного з цих окремих інструментів або повноцінного IDE може виявитися досить складним і потребувати від студента певного рівня технічних знань та розуміння особливостей операційної системи. Покрокові інструкції з установки та налаштування інтегрованого середовища розробки буде наведено у пізніших лабораторних роботах. У випадку виникнення будь-яких труднощів на цьому етапі, студенти завжди можуть звернутися за кваліфікованою консультацією до викладача курсу, який надасть необхідну підтримку та допоможе у вирішенні проблем. Зверніть увагу, описані далі програмні засоби є рекомендованими – їх працездатність ретельно перевірена викладачами, проте студенти можуть користуватися і іншими програмними засобами за своїми уподобаннями.

Водночас, для забезпечення максимальної зручності та оперативності початку навчання, усім студентам надається безперешкодний доступ до спеціалізованої серверної машини. На цій потужній обчислювальній платформі вже попередньо встановлено та ретельно налаштовано все необхідне програмне забезпечення, включаючи компілятори, налагоджувачі та інші важливі інструменти, що дозволяє працювати як через звичний командний рядок, так і в більш інтуїтивно зрозумілому графічному режимі.

Для роботи у віддаленому режимі студентам слід виконати кілька простих кроків з налаштування свого робочого середовища шляхом установки певного програмного забезпечення (ПЗ). Студентам, які користуються пристроями під

керуванням ОС Windows, потрібно встановити ПЗ, як це описано в пп. 1.1 та 1.2. Студентам, які користуються пристроями під керуванням ОС Linux та MacOS, слід перейти до п. 1.4.

1.1. Клієнт доступу до серверів PuTTY

PuTTY – це безплатна програма з відкритим кодом, яка реалізує емулятор терміналу з використанням різних протоколів віддаленого доступу до сервера, включаючи *SSH*, *Telnet*, *rlogin*.

У ***PuTTY*** реалізована тільки клієнтська сторона з'єднання – сторона на якій щось відображається, в той час як сама робота виконується на стороні сервера. До пакета також входять утиліти командного рядка для копіювання файлів між локальною та віддаленою машинами протоколами SCP та SFTP (pscp та psftp відповідно). Зручнішу утиліту копіювання із графічним інтерфейсом буде розглянуто у п. 1.3. Для доступу до віддаленої машини використовуєть захищений протокол SSH, у якому шифрується увесь трафік. Крім того, цей протокол має потужні можливості для тунелювання інших з'єднань, у тому числі і для дисплею графічних програм X Windows System.

Клієнт ***PuTTY*** спочатку розроблявся для *Microsoft Windows*, проте пізніше був перенесений і на *Unix*. ***PuTTY*** входить в репозиторії практично всіх основних систем *Linux* (у т. ч. *Ubuntu*, *Debian*, *ALT Linux*). Код самої програми ***PuTTY*** повністю розроблений мовою *C*. ***PuTTY*** не залежить від динамічних бібліотек, інших додатків, пакетів оновлень ОС. Пакет складається тільки з файлів що виконуються, які можуть бути встановлені в будь-якому місці. Насправді, навіть процедура встановлення як така не є обов'язковою, достатньо розпакувати архів з файлами програми на локальній машині. Програма випускається під *Open Source* ліцензією MIT.

Скачати останню версію ***PuTTY*** клієнта або його установочного пакета можна на офіційній сторінці:

https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

Потрібні для роботи із *Microsoft Windows* чи *Linux* установочні пакети обведені червоним на рис. 0.1.

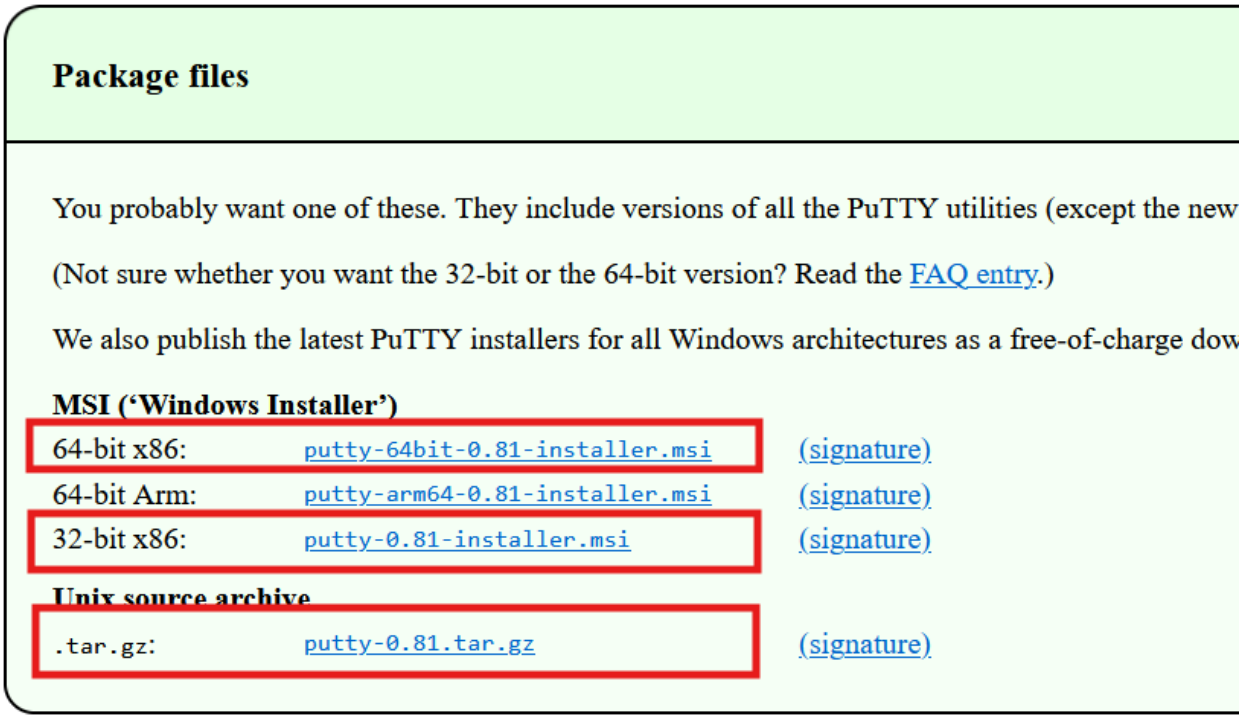


Рис. 0.1. Вигляд сторінки із інсталерамі клієнта ***PuTTY***

Лабораторні роботи, наведені в цьому навчальному посібнику, можна виконувати на сервері, до якого можна підключитись за допомогою клієнта ***PuTTY***.

На початку семестру на сервері ***sandbox***, який використовується для навчання студентів, для кожного студента створюється персональний обліковий запис виду:

dmXXYY (чи dpXXYY)

де XX – номер (шифр) групи,

YY – порядковий номер за списком (див. журнал успішності).

На початку навчання номер XXYY повинен співпадати із номером залікової книжки.

Наприклад, перший студент по списку в групі ДМ-11 буде мати обліковий запис (логін) dm1101.

Після установки та запуску клієнта, для доступу до сервера ***sandbox***, необхідно виконати наступні налаштування:

1. Вказати ім’я сервера, протокол *SSH* та порт, відповідно до вказівок адміністратора. У табл. 0.1 наведено типові налаштування:

Таблиця 0.1. Основні налаштування, необхідні для підключення до навчального сервера ***sandbox***

Налаштування	Ми працюємо	
	З Інтернету	В мережі КПІ
Host Name:	*77.47.131.146*	*sandbox.ee.kpi.ua*
Port:	*6022*	*22*
Connection type:	*SSH*	*SSH*

ПРИМІТКА: налаштування «В мережі КПІ» слід використовувати, якщо клієнтський комп’ютер підключено до мережі засобами КПІ-Телеком у гуртожитку чи у навчальному корпусі. Якщо використовується інший провайдер Інтернет, наприклад, мобільний Інтернет зі смартфона, слід використовувати налаштування «З Інтернету»

Наведені в табл. 0.1 налаштування потрібно зазначити у клієнті ***PuTTY***, як це показано на рис. 0.2.

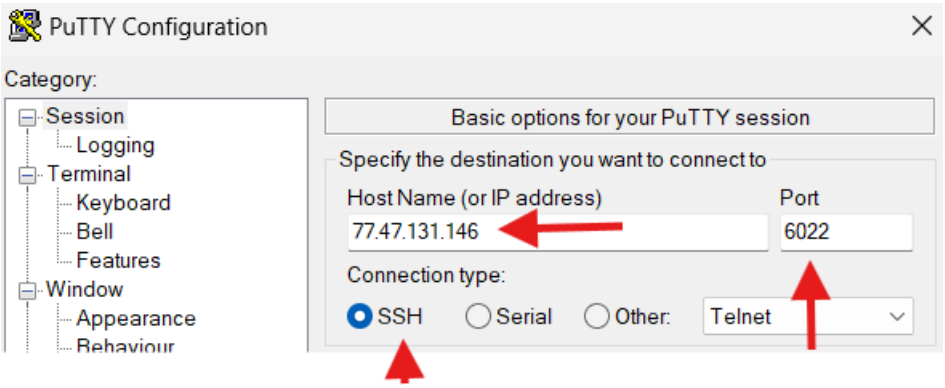


Рис. 0.2. Налаштування підключення до серверу ***sandbox*** у вікні *PuTTY Configuration*

2. Встановити трансляцію кодів клавіатури у підменю *Category: Terminal: Keyboard: The Backspace key → Contral-H* відповідно до рис. 0.3:

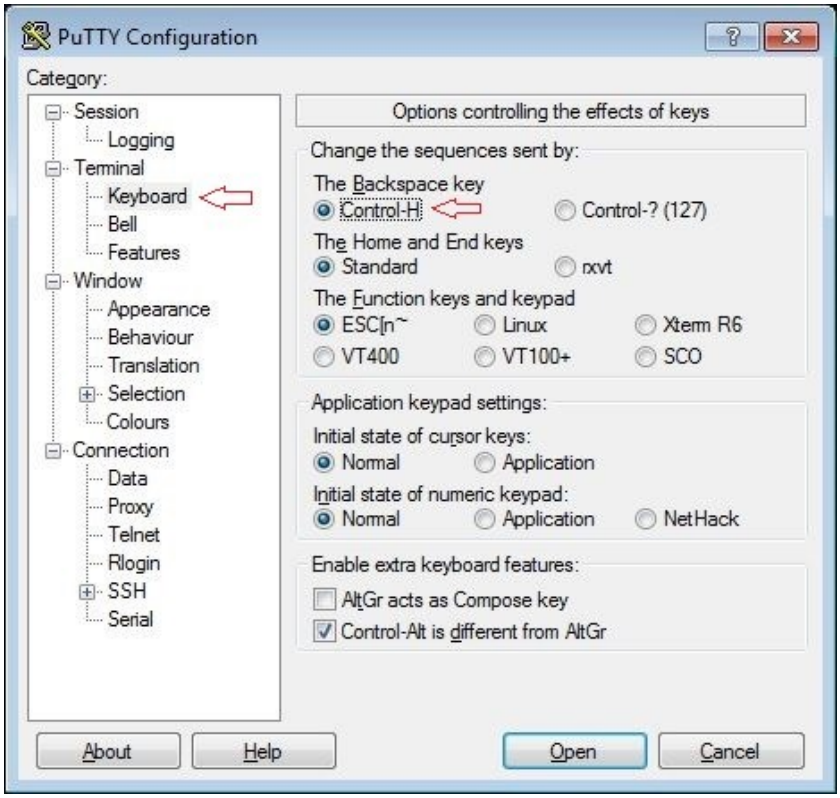
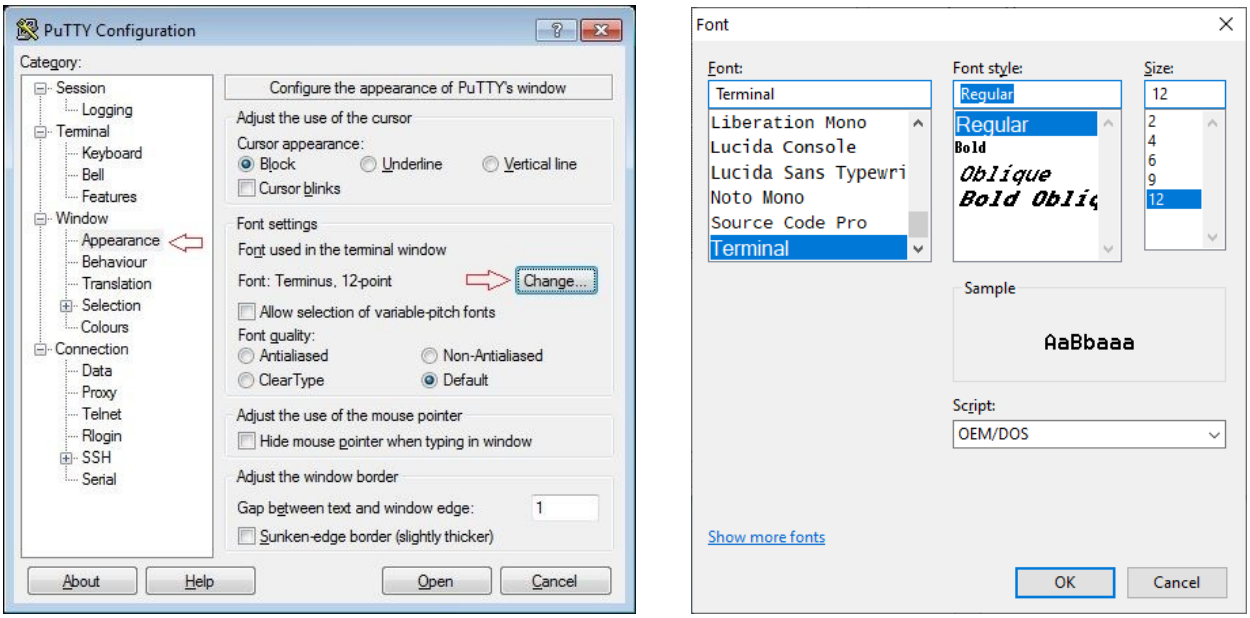


Рис. 0.3. Налаштування трансляцію кодів клавіатури

3. Встановити бажаний тип та розмір шрифту для консолі у підменю *Category: Window: Appearance:Change…* відповідно до рис. 0.4:



а б

Рис. 0.4. Налаштування типу та розміру шрифту для консолі:

а) кнопка налаштування шрифту; б) форма налаштування шрифту

4. Встановити трансляцію кодування у підменю *Category: Window: Translation:Remote Character set → UTF-8* відповідно до рис. 0.5:

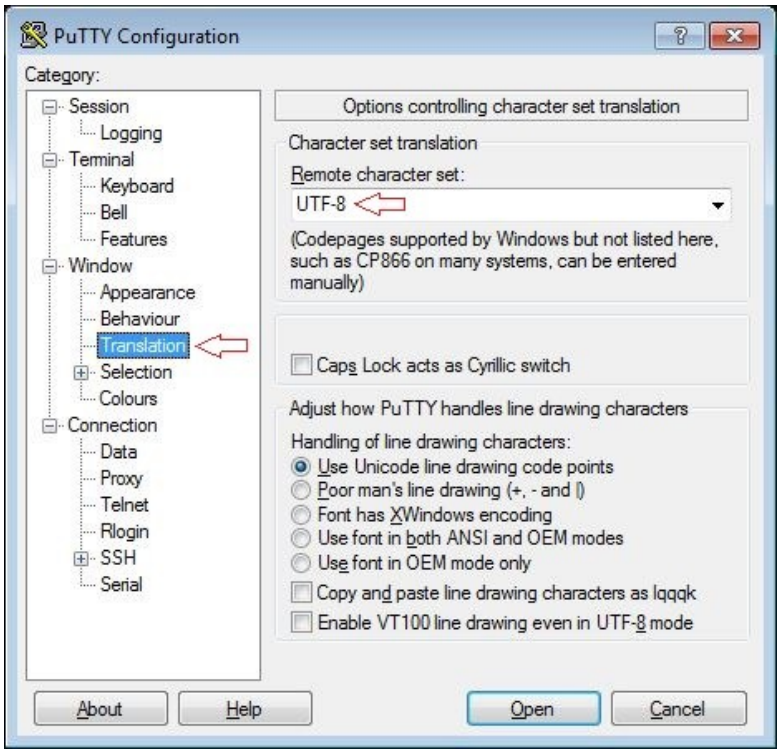


Рис. 0.5. Налаштування трансляції кодування

5. При повільному інтернет-з’єднанні комфорт роботи можна поліпшити вмиканням стиснення даних у підменю *Category: Connection: SSH:Protocol options* → √ *Enable compression* відповідно до рис. 0.6:

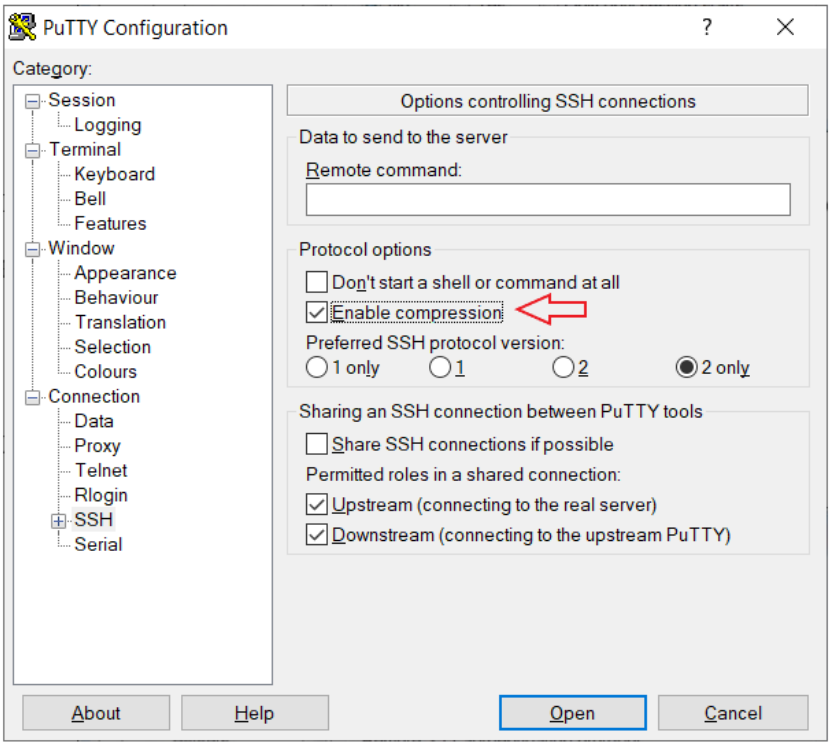


Рис. 0.6. Налаштування стиснення даних

6. Налаштувати пересилку графічного зображення програм, що виконуються. Для того, щоб графіка програми, що виконується віддаленій машині, відображалась на локальній машині, слід виконати додаткові налаштування термінального клієнта ***PuTTY***, а саме у підменю *Connection: SSH: X11* дозволити пересилку (*Enable X11 forwarding*) та вказати розташування програми отримувача (**localhost:0.0**), як це показано на рис. 0.7.

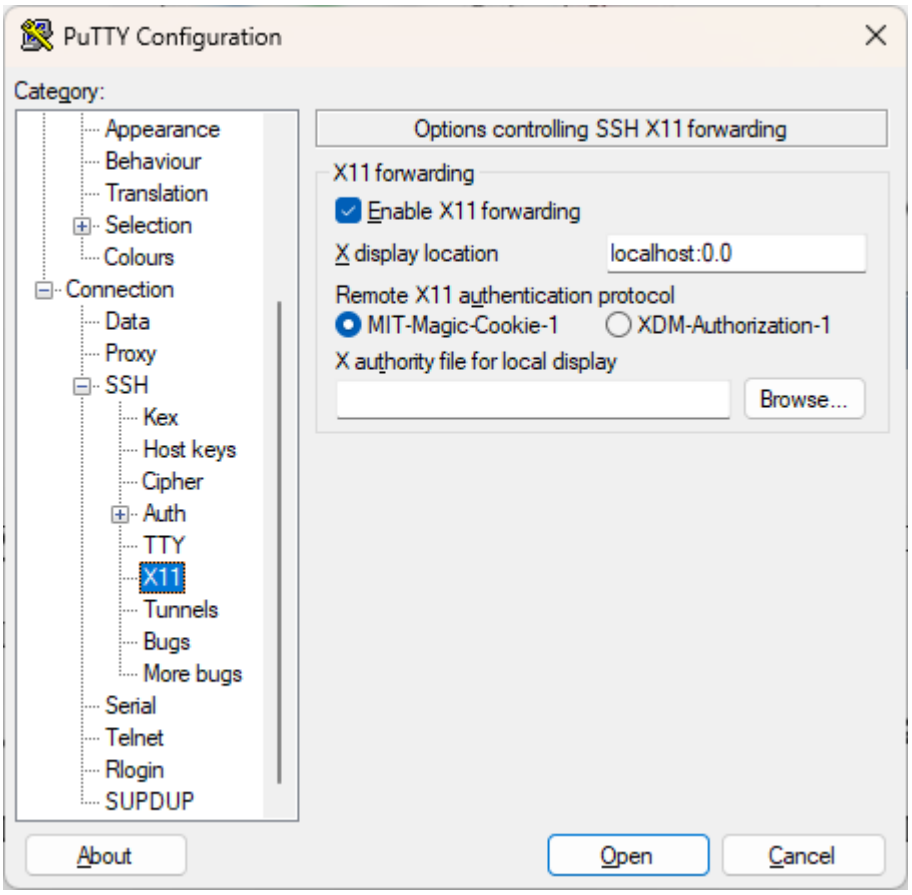


Рис. 0.7. Налаштування пересилки графічного зображення програми що виконується

7. Зберегти профіль з’єднання для подальшого використання у підменю *Session: Saved Sessions* відповідно до рис. 0.8. У полі *Saved Sessions* слід вказати змістовне ім’я профілю, яке дає розуміння про збережену інформацію. В прикладі на рис. 0.8 записано ім’я **sandbox@World** , але ви можете використовувати будь-яке, зручне для вас позначення. Після натискання кнопки *Save* у полі *Saved Sessions* має з’явитися запис про налаштоване з’єднання. На старших курсах вам доведеться працювати і з іншими серверами, процедури їх налаштування подібні щойно описаним.

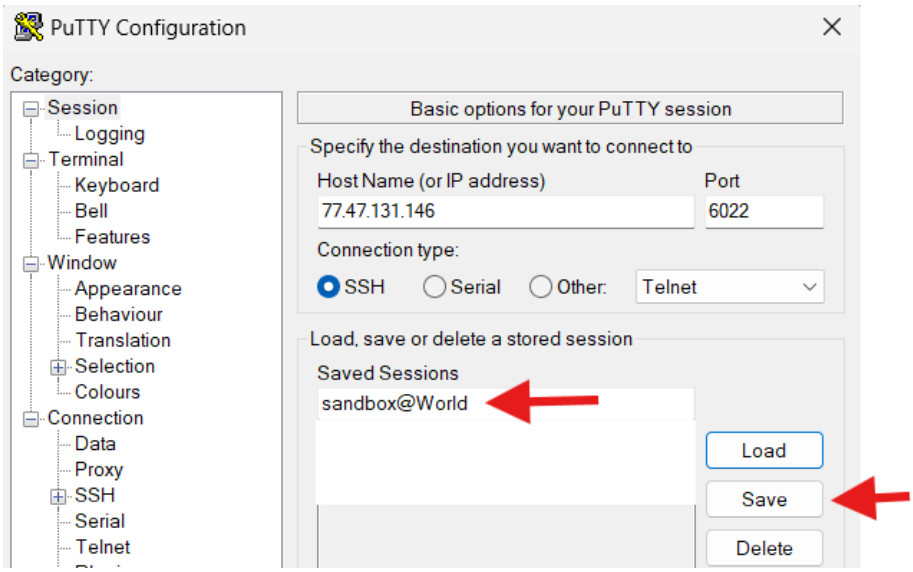


Рис. 0.8. Збереження налаштування у певному профілі

8. Запустити підключення до серверу можна натиснувши кнопку ***Open*** в правому нижньому куту віконечка, як це показано на рис. 0.9.

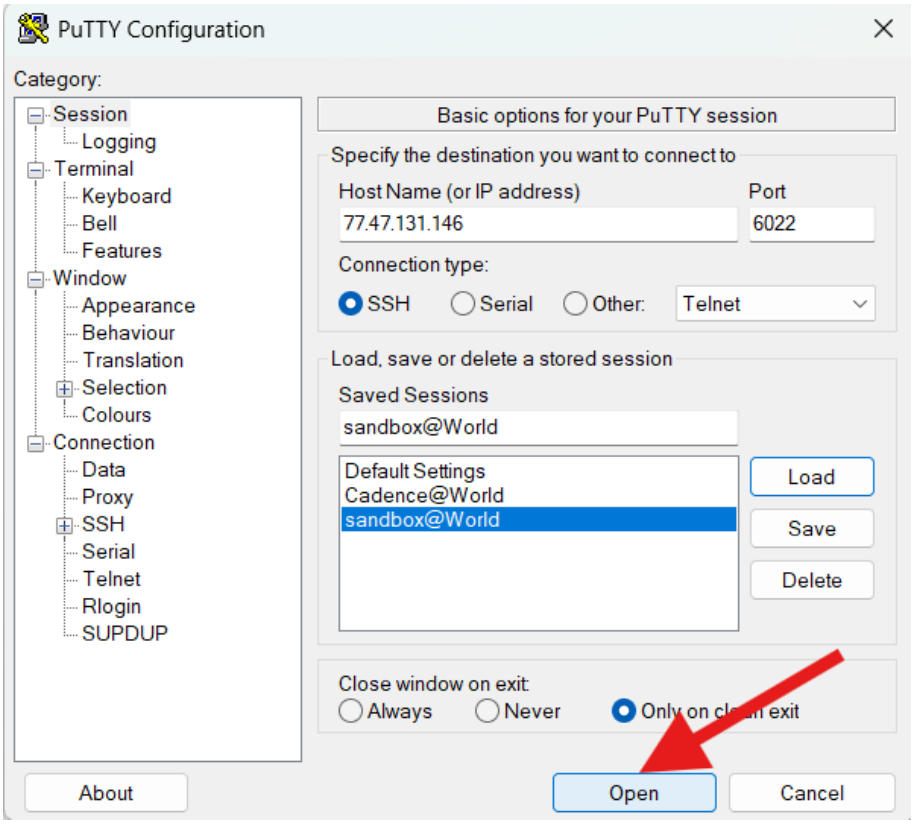


Рис. 0.9. Запуск підключення до сервера

Якщо усі налаштування виконано вірно, то вікно ***PuTTY*** буте мати вигляд, як показано на рис. 0.10.

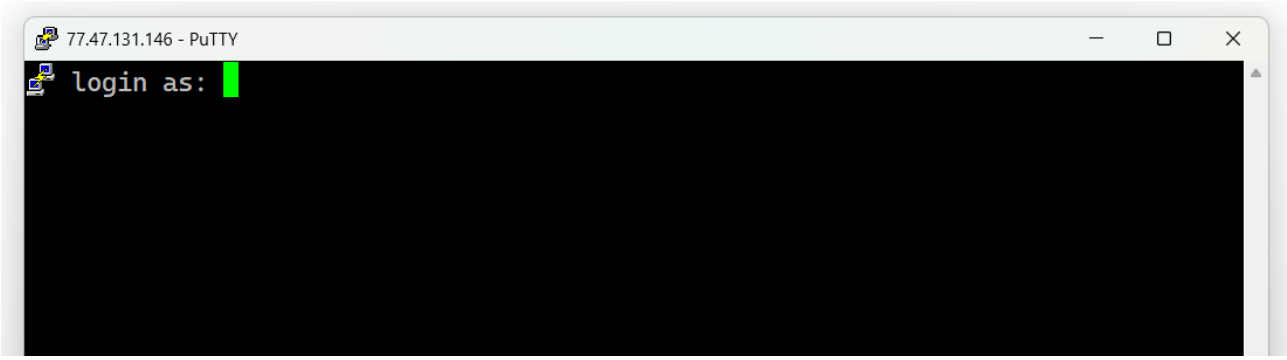


Рис. 0.10. Вигляд вікна ***PuTTY*** одразу після спроби підключитися до сервера

Доступ до серверу мають тільки люди, для яких створено обліковий запис. Ім’я вашого облікового запису (див. на початку цього параграфу) потрібно ввести в першому рядочку та натиснути *Enter*, після чого вікно ***PuTTY*** буте мати вигляд, як показано на рис. 0.11.

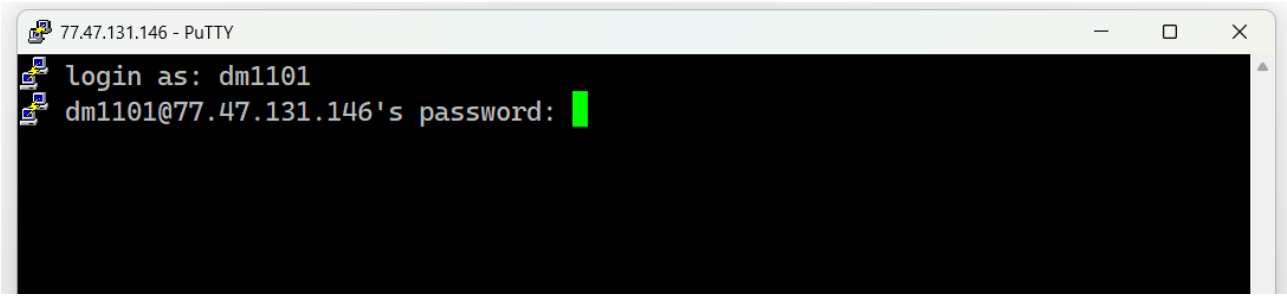


Рис. 0.11. Вигляд вікна ***PuTTY*** після введення імені облікового запису

Викладач повідомить пароль для входу до системи. За бажанням, після першого запуску програми, можна змінити пароль, набравши в командному рядку команду *passwd*.

ПРИМІТКА: при введені паролю символи НЕ ВІДОБРАЖАЮТЬСЯ, але ввід паролю відбувається. Вам потрібно ввести пароль до кінця та натиснути *Enter*, не зважаючи на те, що ніяких змін на екрані не відбувається.

Після підключення до сервера, у консолі ***PuTTY*** можна запускати програми.

При наступних підключеннях до сервера не потрібно заново виконувати усі ці налаштування – достатньо завантажити збережені налаштування за допомогою кнопки *Load*, вибравши потрібний профіль зі списку, як це показано на рис. 0.12.

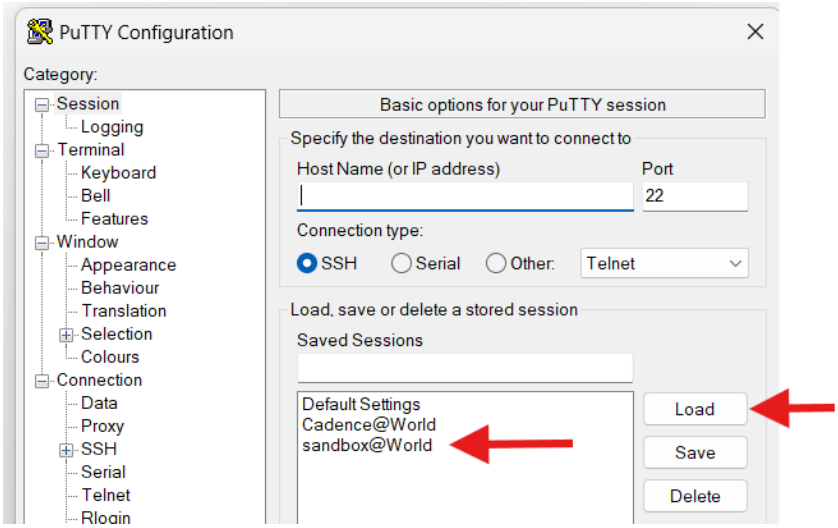


Рис. 0.12. Завантаження збереженого профіля при підключення до віддаленого сервера

Після цього вже можна підключатися натиснувши кнопку *Open*, як це показано на рис. 0.9. Також підключення відбувається подвійним кліком на імені збереженого підключення.

1.2. X-сервер (VcXsrv, XMing)

SSH клієнт ***PuTTY*** працює у режимі командного рядка, але є можливість відображати і графічну інформацію, яка передається із віддаленого сервера, для цього можна використовувати протокол для відображення та передачі зображень ***X Windows System***. Цей протокол дозволяє відображати графіку програмного забезпечення як на локальних дисплеях, так і як завгодно віддалених у мережевій доступності.

Для відображення графіки програмного забезпечення сервера на віддаленому робочому місці, необхідне налаштування спеціалізованого клієнта на робочому місці.

Найпоширенішими серверами ***X11 для Windows*** є ***VcXsrv*** та ***XMing***. Останні версії цих програм можна скачати за посиланнями:

VcXsrv: https://sourceforge.net/projects/vcxsrv/

XMing: https://sourceforge.net/projects/xming/ https://sourceforge.net/projects/xming/

ПРИМІТКА: встановлювати потрібно тільки одну із них.

Нижче наведено приклад установки та налаштування ***VcXsrv***.

Натиснувши посилання, наведене вище, переходимо на сайт, де потрібно натиснути *Download* та завантажити файл інсталятора. Далі встановити програму із наступними налаштуваннями:

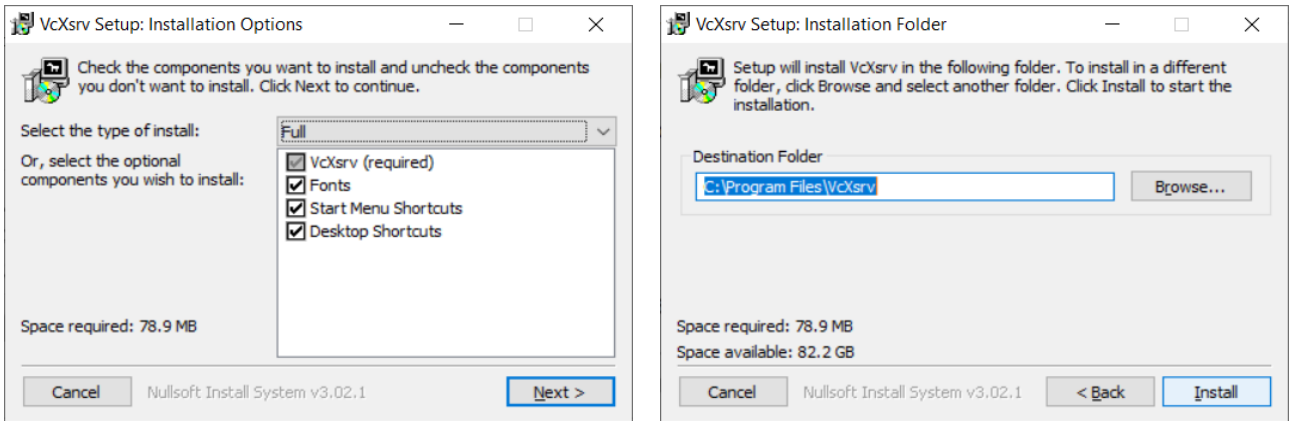


Рис. 0.13. Установка та налаштування ***VcXsrv***

Після встановлення слід запустити програму ***XLaunch***:

Рис. 0.14. Запуск ***VcXsrv***

та виконати налаштування, як показано на скріншотах нижче:

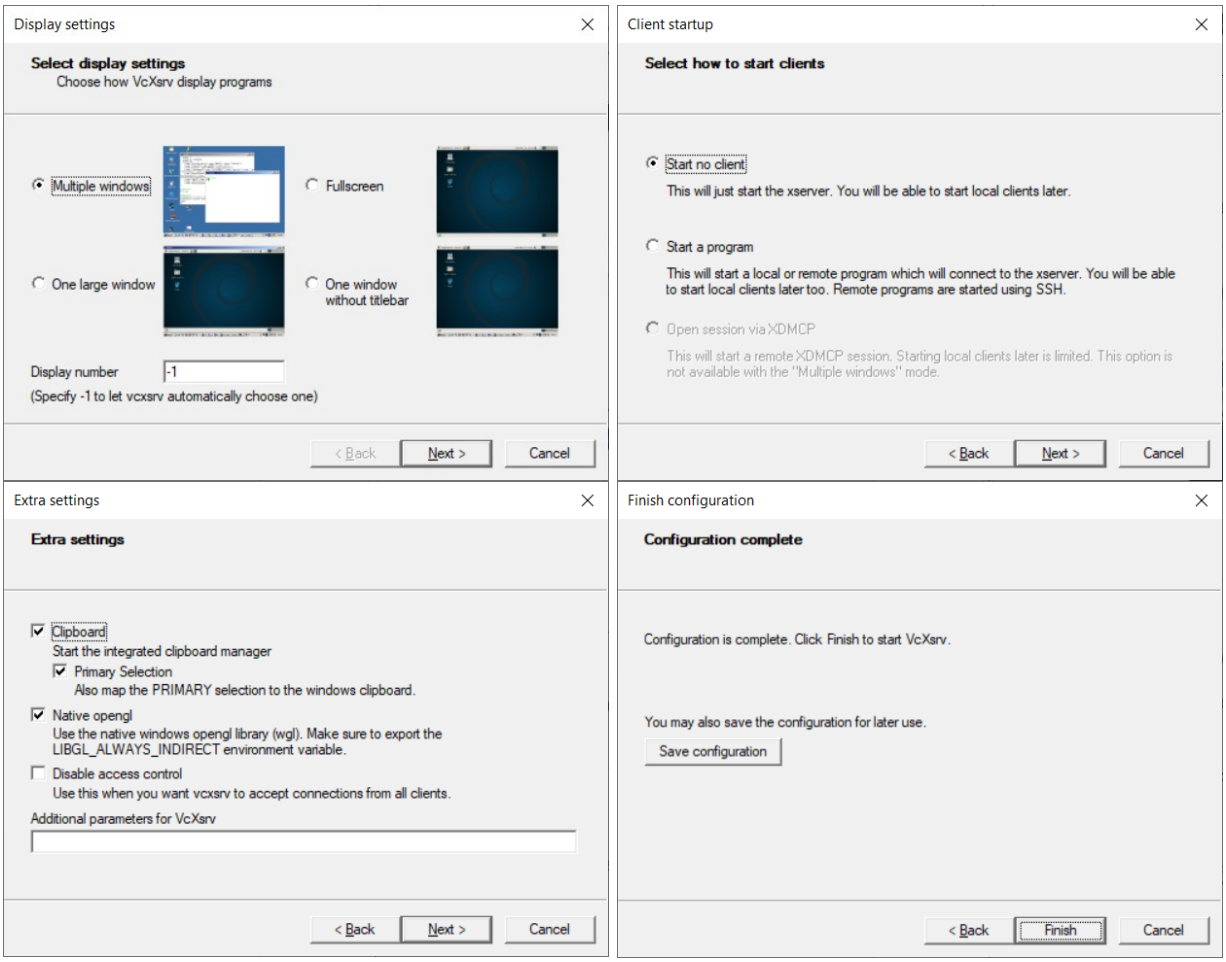


Рис. 0.15. Налаштування ***VcXsrv***

На останньому кроці можна зберегти конфігураційний файл у доступному місці, наприклад на робочому столі (після натискання кнопки *Save configuration*). Після цього натискаємо кнопку *Finish*. Це був початковий етап установки. Надалі достатньо запускати збережений конфігураційний файл на початку кожного сеансу роботи, клікнувши на іконку на робочому столі. Сервер ***VcXsrv*** продовжує працювати, навіть якщо з якось причини (збій мережі чи команда користувача) термінал було закрито. Упевнитись, що сервер працює, можна за іконкою у треї, як це показано на рис. 0.16:

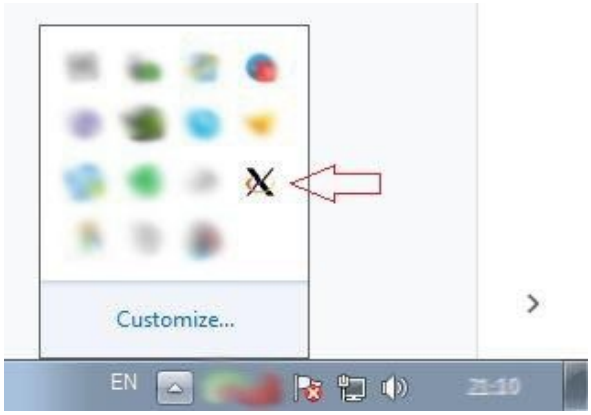


Рис. 0.16. Іконка ***VcXsrv*** в треї

Після запуску ***XMing*** або ***VcXsrv*** треба підключитися (чи перепідключитися, якщо ***PuTTY*** вже будо запущено) до сервера за допомогою термінального клієнта ***PuTTY***. Надалі зображення програм, запущений на віддаленій машині (сервері) буде відображатись на локальній машині.

1.3. Графічний клієнт для копіювання файлів між сервером та комп'ютером для Windows

Відображення на локальній машині графічної інформації від програми, яка виконується на віддаленому сервері, дає звичне відчуття роботи, як і зі звичайними програмами Windows, які виконуються на локальній машині. Проте слід пам'ятати, що файлові системи клієнта та сервера відокремлені. Тому навіть якщо спробувати у програмі, запущеній на віддаленому сервері, зберегти файл командою, подібною до ***Save As***…, на вибір буде запропоновано шляхи у файловій системі віддаленого сервера. Через це, наприклад, немає змоги надрукувати збережений файл на принтері локальної машини, вставити його вміст до іншого документа, чи перенести на знімний носій. Так само, підготовлений заздалегідь файл на локальній машині не буде доступним на віддаленому сервері. Аби мати змогу виконувати перелічені дії, слід скопіювати потрібні файли з файлової системи сервера до локальної файлової системи чи навпаки.

Для завантаження файлів зі свого комп'ютера під керуванням *Windows* на сервер під керуванням **UNIX**-подібної системи, і навпаки зручно користуватися програмою ***WinSCP***. ***WinSCP*** – це графічний клієнт протоколів *SFTP* і *SCP*, призначений для *Windows*. Має відкритий вихідний код та поширюється за ліцензією *GNU GPL*. Забезпечує захищене копіювання файлів між комп'ютером і серверами, що підтримують ці протоколи.

Основні можливості ***WinSCP***:

- Графічний інтерфейс в стилі *Norton Commander* чи подібний до провідника *Windows Explorer* (на вибір).
- Всі основні файлові операції – копіювання, видалення і т. д.
- Автоматизація за допомогою скриптів і інтерфейсу командного рядка.
- Інтеграція з *Pageant* (***PuTTY*** *Agent*) з підтримкою авторизації по відкритих ключах.
- Інтеграція з *Windows* (підтримка *Drag & Drop*, ярликів, підтримка схем *URL*).
- Робота з ключами і версіями протоколу *SSH*.
- Вбудований текстовий редактор.
- Підтримка різних типів авторизації: по паролю, аутентифікації з відкритим ключем, *Kerberos*.
- Можливість зберігати налаштування з'єднань.
- Локалізації інтерфейсу для кількох десятків мов.

Для завантаження ***WinSCP*** можна скористатися посиланням:

https://winscp.net/eng/download.php

Після інсталяції ярлик ***WinSCP*** буде виглядати наступним чином:

Рис. 0.17. Ярлик ***WinSCP***

Після запуску ***WinSCP*** відкривається вікно налаштування з'єднання, як це показано на рис. 0.18:

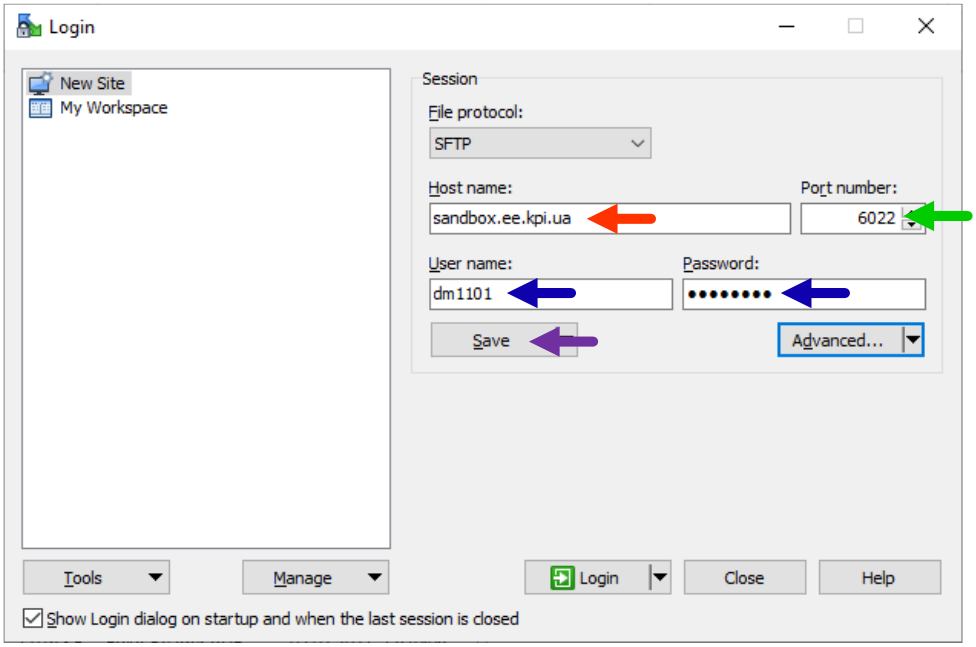


Рис. 0.18. Налаштування під'єднання до сервера у ***WinSCP***

Для під'єднання до сервера потрібно вказати ім'я сервера (червона стрілка), номер порту (зелена стрілка), ім'я та пароль користувача (синя стрілка). Ці налаштування такі самі, як було описано раніше для підключення до віддаленого сервера за допомогою клієнта ***PuTTY***. Після цього можна натиснути кнопку *Login* і під'єднатися до сервера. Якщо раніше було збережено налаштування підключень ***PuTTY***, їх також можна імпортувати.

Щоб кожного разу не вказувати усі ці параметри, можна зберегти налаштування, натиснувши кнопку *Save*, після чого з'явиться вікно збереження сесії, як це показано на рис. 0.19:

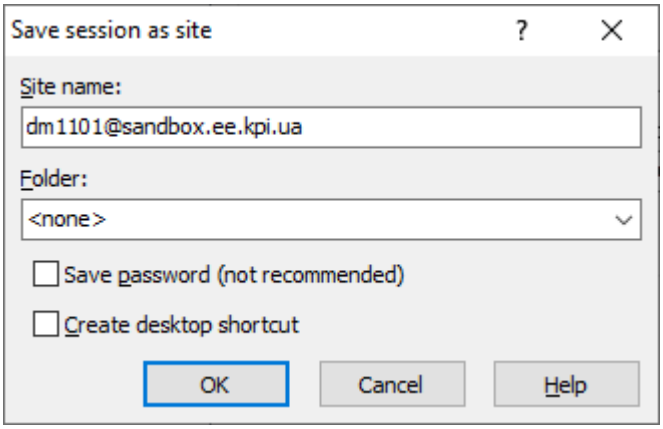


Рис. 0.19. Вікно збереження сесії підключення до віддаленого серверу у програмі ***WinSCP***

За замовчуванням, пароль не зберігається, але якщо поставити галочку напроти *Save password*, то при збереженні налаштувань в тому числі буде збережено і пароль.

Після збереження налаштувань, в панелі *Login* з'явиться запис, який відповідає збереженому профілю. На рис. 0.20 він обведений зеленим:

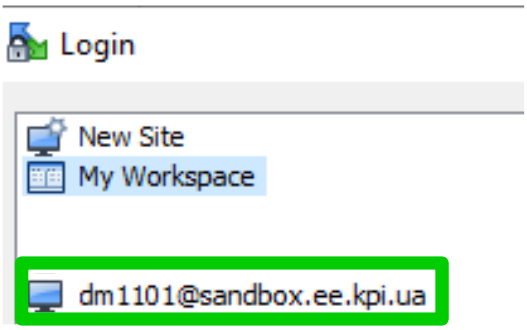


Рис. 0.20. Область із наявними збереженими сесіями під'єднання до віддалених серверів

Після з'єднання із сервером відкриється вікно, як показано на рис. 0.21:

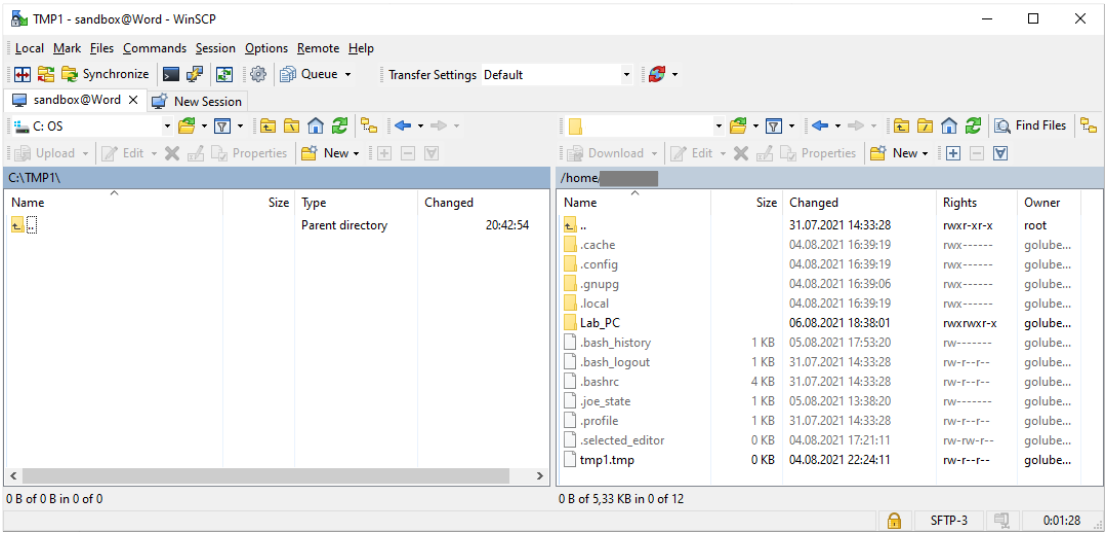


Рис. 0.21. Приблизний вигляд вікна ***WinSCP*** після під'єднання до віддаленого сервера

У цьому вікні ***WinSCP*** ліворуч знаходиться панель, яка показує файли, розташовані на локальному комп'ютері, а праворуч – панель, в якій показані файли на сервері, з яким виконано з'єднання. Перемикатися між цими панелями можна за допомогою клавіши *Tab*.

Виконувати маніпуляції із файлами можна або за допомогою гарячих клавіш, або за допомогою кнопок, які розташовані вище панелей:

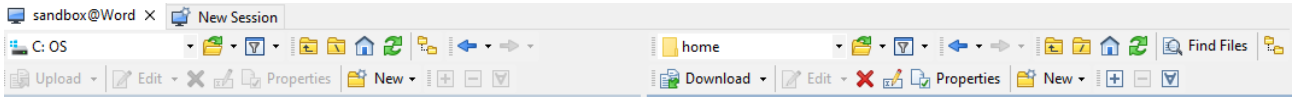


Рис. 0.22. Панель інструментів ***WinSCP***

Певні дії із конкретним файлом можна виконати також і за допомогою контекстного меню, яке відкривається натисканням правої кнопки миші:

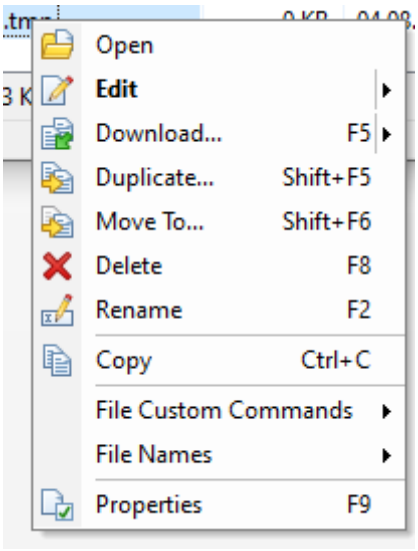
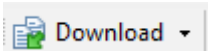


Рис. 0.23. Контекстне меню ***WinSCP***

Скопіювати файл із або на сервер можна обравши ім'я файлу та натиснувши кнопку 

відповідної панелі, або натиснувши гарячу клавішу *F5*.

1.4. Налаштування доступу до віддаленого сервера для ОС Linux та MacOS

Якщо на комп'ютері, на якому планується виконання лабораторної роботи, використовується ОС *Linux*, *MacOS* чи щось інше, що має вбудований термінал та *SSH* клієнт, то достатньо набрати в терміналі наступну команду:

```
ssh sandbox.ee.kpi.ua -p 6022 -l $LOGIN
```
,

де $LOGIN – це ваш логін на сервері як це було описано вище (тобто перший студент в списку в групі ДМ-11 повинен набрати в командному рядку наступну команду: "**ssh sandbox.ee.kpi.ua -p 6022 -l dm1101**").

ПРИМІТКА 1: зверніть увагу, що після номеру порту (6022) записується ключ із латинської букви Ель, а не одиниця.

ПРИМІТКА 2: інколи, коли не вдається підключитися із вищевказаною інструкцією, спробуйте замість символьного імені сервера скористатися його IP-адресою такою командою:

“**ssh 77.47.131.146 -p 6022 -l dm1101**”

1. Лабораторна робота №1
Основи роботи з UNIX-подібними системами

Мета роботи: вивчення основних прийомів і отримання практичних навичок роботи з ***UNIX***-подібними системами.

1.1. Теоретичні відомості

1.1.1. UNIX та UNIX-подібні системи

UNIX – родина операційних систем, які є багатозадачними, ними можуть користуватися декілька користувачів та їх легко переносити.

Перша система ***UNIX*** була розроблена в 1969 році в підрозділі *Bell Labs* компанії *AT&T*. З тих пір була створена велика кількість різних ***UNIX***-систем. Юридично лише деякі з них мають повне право називатися «***UNIX***»; інші ж, хоча і використовують подібні концепції і технології, поєднуються терміном «***UNIX***-подібні» (англ. *Unix-like*).

На теперішній час ***UNIX***-системи широко використовуються на серверах, а також як вбудовані системи для різного устаткування. Також є варіанти і для особистого використання.

Серед прикладів відомих ***UNIX***-подібних операційних систем є ***BSD***, ***Solaris, Linux, Android, MeeGo, NeXTSTEP, Mac OS X, Apple iOS***. Система ***Linux*** набула поширення та популярності також як операційна система для персональних комп'ютерів. Безплатну копію операційної системи можна отримати в Інтернет з так званих дистрибутивів. На персональному комп'ютері також можна мати більше, ніж одну операційну систему. Майте на увазі, що встановлення додаткової операційної системи поряд з існуючою вимагає певної кваліфікації, тоді як помилки у цьому процесі загрожують втратою даних.

1.1.2. Доступ до UNIX-серверів

Традиційно для доступу (роботи чи адміністрування) до роботи із сервером під керуванням ***UNIX***-подібних системам можна використовувати клієнти термінального доступу.

Такі клієнти можуть використовувати різні протоколи доступу до сервера, наприклад ***RLOGIN, TELNET*** чи ***SSH***. В даний час для віддаленого доступу до системи найчастіше застосовується мережевий протокол ***SSH*** (особливо його версія 2), оскільки він є більш захищеним. На відміну від ***RLOGIN, TELNET***, протокол ***SSH*** шифрує весь трафік, включаючи і паролі, що передаються.

SSH дозволяє безпечно передавати в незахищеному середовищі практично будь-який інший мережевий протокол. ***SSH*** може використовувати стиснення даних, що передаються, що є зручним, наприклад, при роботі через повільні канали зв'язку.

Підтримка ***SSH*** реалізована у всіх ***UNIX***-подібних системах, і на більшості з них в числі стандартних утиліт присутні клієнт і сервер ***SSH***. Існує безліч реалізацій ***SSH***-клієнтів і для не-***UNIX*** ОС. Велику популярність протокол отримав після широкого розвитку аналізаторів трафіку і способів порушення роботи локальних мереж, як альтернатива незахищеному протоколу ***Telnet*** для управління важливими вузлами.

Для роботи по ***SSH*** потрібен ***SSH***-сервер та ***SSH***-клієнт. Сервер прослуховує з'єднання від клієнтських машин та при встановленні зв'язку виконує аутентифікацію, після чого починає обслуговування клієнта. Клієнт використовується для входу на віддалену машину та виконання команд.

Існує велика кількість SSH-клієнт для різних платформ, наприклад:

- ✓ ***GNU/Linux, *BSD: kdessh, lsh-client, openssh-client, putty, ssh, Vinagre, Tectia SSH (SSH Communications Security) Client,***

- ✓ ***Windows: PuTTY\KiTTY, SecureCRT, ShellGuard, Axessh, ZOC, SSHWindows, ProSSHD, XShell, Tectia SSH (SSH Communications Security) Client,***
- ✓ ***MS Windows Mobile: PocketPuTTy, mToken, sshCE, PocketTTY, OpenSSH, PocketConsole, Tectia SSH (SSH Communications Security) Client,***
- ✓ ***Mac OS: iTerm2, NiftyTelnet SSH, vSSH, ZOC,***
- ✓ ***Symbian OS: PuTTY,***
- ✓ ***iPhone: i-SSH, ssh (в комплекті із Terminal), Termius, OpenTerm, LibTerm, Shelly,***
- ✓ ***Android: connectBot, Admin Hands, Server Auditor, JuiceSSH,***
- ✓ ***Blackberry: BBSSH,***
- ✓ ***MAEMO 5: OpenSSH,***
- ✓ ***MeeGo 1.2 Harmattan: OpenSSH.***

Встановлення та налаштування ***SSH-***клієнту описано в параграфі «Клієнт доступу до серверів PuTTY» на стор. 8. Для виконання лабораторної роботи потрібно уважно прочитати цей параграф та виконати усі налаштування, описані в ньому.

1.1.3. Файлова система

Будь який файл ОС ***UNIX*** у відповідності з його типом може бути віднесений до однієї з наступних чотирьох груп: звичайні файли, каталоги, спеціальні файли, канали.

Файлова система ОС ***UNIX*** має ієрархічну (деревоподібну) структуру. У вершинах дерева знаходяться каталоги (директорії), що містять списки файлів. Ці файли в свою чергу можуть бути або знову каталогами, або звичайними

файлами, або спеціальними файлами, що представляють різні пристрої введення-виведення.

На відміну від інших ОС, в ***UNIX***-подібних системах не використовуються логічні диски. Усі наявні носії належать до єдиного дерева, вершиною якого є кореневий каталог. Кореневий каталог має ім'я "/". Він, зазвичай, містить каталоги:

bin для програм, які найбільше використовуються;

dev для спеціальних файлів, що представляють пристрої (дисплеї, диски, ...);

etc для зберігання налаштувань системи;

lib найважливіші бібліотеки;

mnt для підключення (монтування) нових файлових систем;

sys для інформації про апаратне забезпечення та драйверів;

tmp для зберігання тимчасових файлів;

usr каталоги і звичайні файли, які залучаються при розв'язанні задач користувача.

Зверніть увагу, що на відміну від деяких інших ОС, для відокремлення імен каталогів використовується символ «прямий слеш» "/", а не "\".

Кожному зареєстрованому користувачеві відповідає деякий каталог файлової системи, який називається «домашнім» (*home*) каталогом користувача. При вході в систему користувач зазвичай одержує необмежений доступ до свого домашнього каталогу і всіх каталогів і файлів, що містяться в ньому. Потенційно можливий доступ і до всіх інших файлів, однак він може бути обмежений, якщо користувач не має достатніх привілеїв.

Доступ до домашнього каталогу можна отримати за допомогою його псевдоніма ~, наприклад, команда

```
cd ~/work
```

здійснює перехід в підкаталог *work* домашнього каталогу поточного користувача.

1.1.4. Програми

Програма, яка виконується, може бути запущена в інтерактивному режимі як команда **shell** або виконана в окремому процесі, утвореному вже запущеною програмою. Для деяких програм може запускатися графічний інтерфейс. При цьому командний рядок залишається заблокованим, допоки працює графічний інтерфейс запущеної програми. Щоб зберегти можливість вводу інших команд, такі програми можна запускати у фоновому режимі, якщо у кінці команди додати символ амперсанда (&). Наприклад, текстовий редактор з графічним інтерфейсом можна запустити командою

```
nedit &
```

Часто новачки ненавмисно натискають комбінацію ***Ctrl-Z***, коли поведінка програми, яка виконується у терміналі, не відповідає очікуванням. Однак при цьому відбувається призупинення виконання поточної програми і відсилка її у фоновий режим. Повернути таку програму до інтерактивного режиму можна командою ***fg***.

1.1.5. Командна мова ОС Unix (shell)

Традиційний спосіб взаємодії користувача з ***UNIX***-подібною системою ґрунтується на використанні командних мов.

Зазвичай в системі підтримується кілька командних інтерпретаторів зі схожими командними мовами, які можуть мати різні можливості. Загальна назва для будь-якого командного інтерпретатора ОС ***UNIX*** – це **shell** (оболонка), оскільки будь-який інтерпретатор є зовнішнім оточенням ядра системи.

Працюючи з командною мовою, користувач може вводити змінні, присвоювати їм значення, виконувати прості команди, управляти потоком виконання команд, об'єднувати послідовність команд в процедури (командні файли). На рівні командної мови доступні такі властивості системи як поєднання процесів через програмний канал, напрямок стандартного вводу/виводу в конкретні файли, синхронне та асинхронне виконання команд.

Командна мова **shell** фактично є мовою програмування дуже високого рівня. На цій мові користувач здійснює управління комп'ютером. Зазвичай, після входу в систему ви починаєте взаємодіяти з командною оболонкою. Ознакою того, що оболонка (**shell**) готова до прийому команд є запрошення (промптер), яке з'являється на екрані. У найпростішому випадку це символ долара ("**$**").

Shell не є необхідною і єдиною командною мовою. Наприклад, чималою популярністю користується мова **csh**, є також **ksh**, **bash, tcsh** та інші.

ЗВЕРНІТЬ УВАГУ. **shell** – це одна з багатьох команд ***UNIX***. Тобто в набір команд оболонки (інтерпретатора) "**shell**" входить команда "**sh**" – виклик інтерпретатора "**shell**". Перший "**shell**" викликається автоматично при вашому вході в систему і видає на екран промтер. Після цього ви можете викликати на виконання будь-які команди, в тому числі і знову сам "**shell**", який вам створить нову оболонку всередині колишньої.

Командні мови, що використовуються в ОС ***UNIX***, достатньо прості, щоб нові користувачі могли швидко почати працювати, і досить потужні, щоб можна було використовувати їх для написання складних програм. Остання можливість спирається на механізм командних файлів (**shell scripts**), які можуть складатися з довільної послідовності командних рядків. Якщо викликати командний файл

замість чергової команди інтерпретатор читає файл рядок за рядком і послідовно інтерпретує команди.

1.1.6. Структура команд

Будь яка командна мова сімейства **shell** фактично складається з трьох частин:

- ✓ службових конструкцій, що дозволяють маніпулювати з текстовими рядками і будувати складні команди на основі простих команд;
- ✓ вбудованих команд, які виконуються безпосередньо інтерпретатором командної мови;
- ✓ команд.

Команди в **shell** зазвичай мають наступний формат:

```
<ім'я команди> <ключ(і)> <аргумент(и)>
```

наприклад:

```
ls   -s   /usr/bin
```

В останньому прикладі елементи команди мають наступні значення:

ls ім'я команди видачі вмісту директорії,

-s ключ ("-" – ознака ключів, *s* – обсяг файлів в блоках).

/usr/bin директорія, для якої виконується команда.

1.1.7. Довідка по деяких командах

1.1.7.1. man

У більшості ***UNIX***-подібних систем є вбудована детальна довідкова інформація по більшості команд та програм. Для отримання довідки по команді використовуйте команду *man*, наприклад:

```
man ls
```

дозволяє отримати довідку на команду *ls*.

Для виходу із режиму довідки потрібно натиснути клавішу “*q*”.

1.1.7.2. apropos

Для пошуку довідки за ключовими словами, використовуйте команду *apropos*. Наприклад, команда

```
apropos “working directory”
```

дозволяє знайти команди, які працюють з поточним каталогом, наприклад знайти команду, яка служить для виведення поточного каталогу.

1.1.7.3. ls

Команда *ls* дозволяє отримати інформацію про файли або каталоги.

СИНТАКСИС

ls [-R][-a][-d][-C][-x][-m][-l][-n][-o][-g][-r][-t][-u][-c][-p][-F][-b][-q][-i][-s][-f] [ім’я ...]

ОПИС

Команда *ls* для кожного імені каталогу роздруковує список файлів які містяться у поточному каталозі; для файлів – повторюється ім'я файлу та виводиться додаткова інформація відповідно до зазначених ключів. За замовчуванням імена файлів виводяться в алфавітному порядку. Якщо імена не задані, видається вміст поточного каталогу. Якщо задані кілька аргументів, то вони сортуються за алфавітом, проте спочатку завжди йдуть файли, а потім каталоги з їх вмістом.

За замовчуванням результат видається по одному файлу в рядку; ключі -C та -x дозволяють видавати інформацію в декілька колонок, а ключ -m задає вільний формат.

Ключі, які можуть бути використані із командою *ls* наведені в табл. 1.1.

Таблиця 1.1. Ключі, які можуть бути використані із командою *ls*

Ключ	Що робить цей ключ
-R	Рекурсивно обійти підкаталоги, які зустрічаються.
-a	Вивести список всіх файлів (зазвичай не виводяться файли, імена яких починаються з крапки).
-d	Якщо аргумент є каталогом, то виводити тільки його ім'я, а не вміст. Часто використовується з ключем ***-l*** для отримання відомостей про стан каталогу.
-C	Вивід у кілька колонок з сортуванням по колонках.
-x	Вивід у кілька колонок з сортуванням по рядках.

Ключ	Що робить цей ключ
-m	Вивід у вільному форматі, імена файлів розділяються комами.
-l	Вивід в довгому форматі: перед іменами файлів видається режим доступу, кількість посилань на файл, імена власника та групи, розмір у байтах і час останньої модифікації (див. нижче). Якщо файл є спеціальним, то в поле розміру виводиться старший та молодший номери пристрою.
-n	Те ж, що і ***-l***, але ідентифікатори власника та групи виводяться у вигляді чисел, а не у вигляді імен.
-o	Те ж, що і ***-l***, але ідентифікатор групи не виводиться.
-g	Те ж, що і ***-l***, але ідентифікатор власника не виводиться.
-r	Змінити порядок сортування на зворотний алфавітний або, при наявності ключа ***-t***, спочатку виводити старіші файли.
-t	Імена файлів сортуються не за алфавітом, а за часом (спочатку йдуть найсвіжіші файли). За замовчуванням використовується час останньої зміни.
-u	Замість часу останньої зміни використовувати час останнього доступу для сортування (з ключем ***-t***) або для виводу (з ключем ***-l***).
-c	Замість часу останньої зміни використовувати час останньої модифікації описувача файлу (тобто час створення файлу, зміни режиму доступу до нього і т.п.) для сортування (з ключем ***-t***) або для виводу (з ключем ***-l***).
-p	Якщо файл є каталогом, то видавати після його імені символ /.

Ключ	Що робить цей ключ
-F	Якщо файл є каталогом, то видавати після його імені символ /; якщо файл може виконуватися, то видавати після його імені символ *****.
-b	Видавати недруковані символи, що входять до імені файла, у вісімковому вигляді (***ddd**).
-q	Видавати недруковані символи, що входять до імені файла, у вигляді символа ***?***.
-i	Видавати в першій колонці номери дескрипторів файлів.
-s	Видавати розмір файлів в блоках.
-f	Розглядати кожен аргумент як каталог і виводити його вміст. Цей ключ скасовує ключі ***-l, -t, -s, -r*** і включає ключ **-a**. Сортування імен файлів не проводиться; імена видаються в такому порядку, в якому вони перераховані в каталозі.

Режим доступу до файлу з ключем ***-l*** виводиться у вигляді 10 символів. Значення першого символа можна подивитися у табл. 1.2.

Таблиця 1.2. Значення першого символу в слові, яке кодує режим доступу до файла чи каталога

Символ	Значення цього символу
d	Файл є каталогом.
b	Файл є спеціальним блочним файлом.

Символ	Значення цього символу
c	Файл є спеціальним символьним файлом.
p	Файл є іменованим каналом.
-	Звичайний файл.

Решта 9 символів поділяються на три групи по три символи:

- ✓ 3 символи з правами доступу власника,
- ✓ 3 символи з правами доступу інших користувачів з його групи,
- ✓ 3 символи з правами доступу всіх інших користувачів.

Усередині кожної групи використовуються три символи, що позначають права на читання, запис та виконання файлу відповідно. Для каталогу під правом на виконання мається на увазі право на перегляд у пошуках потрібного файлу.

При використанні команди

```
ls -l /util/by
```

результат може виглядати наступним чином:

```
-rwxr-xr-x   1 root     sys           50 Jun 22 10:42 /util/by
```

Читаючи справа наліво можна побачити, що вміст файлу */util/by* останній раз змінювався о 10 годині 42 хвилини 22 червня. Розмір файлу 50 байт. Власник цього файлу є суперкористувачем (вхідне ім'я – *root*), також цей файл асоційовано з групою *sys*. Наступне число, в даному випадку 1, позначає кількість посилань на файл */util/by*. Нарешті, послідовність мінусів і букв на

початку рядка вказує, що власник, члени групи та інші користувачі можуть читати і виконувати файл, а власник (і тільки він) має право писати у файл.

Усі можливі символи, які можуть бути використані при позначенні прав доступу до файлу чи каталогу, наведені в табл. 1.3.

Таблиця 1.3. Відповідність прав доступу до символів, які повертає команда *ls*

Символ	Які права означає цей символ
r	Право на читання
w	Право на запис
x	Право на виконання (для каталогів – право на читання вмісту каталогу)
-	Дане право доступу відсутнє
l	Враховується блокування доступу (біт переустановки ідентифікатора групи дорівнює 1, біт права на виконання членами групи дорівнює 0). Розташовується на місці права на виконання для членів групи
s	Право встановлювати заново ідентифікатор групи або ідентифікатор власника та право виконання файлу для членів групи або власника
S	Невизначена комбінація біт: право встановлювати заново ідентифікатор власника є, а право виконання файлу для власника відсутнє
t	Встановлено «липкий» біт [дивись команду *chmod* для більш детальної інформації] у файла, який можуть виконувати інші

Символ	Які права означає цей символ
	користувачі. Розташовується на місці права на виконання для інших користувачів.
T	«Липкий» біт встановлений, а права на виконання у інших користувачів немає. Розташовується на місці права на виконання для інших користувачів.

ПРИКЛАДИ

1. Приклад 1: якщо файл має режим доступу

```
-rwxr--r--
```

він доступний власнику для читання, запису і виконання, а членам групи і іншим користувачам – тільки для читання.

2. Приклад 2: якщо файл має режим доступу

```
-rwsr-xr-x
```

це свідчить про те, що файл доступний власнику для читання, запису і виконання, а членам групи і іншим користувачам тільки для читання та виконання. Дозволена переустановка при виконанні ідентифікатора користувача на ідентифікатор власника файлу.

3. Приклад 3: після виконання команди:

```
ls -a
```

будуть виведені імена всіх файлів в поточному каталозі, включаючи і ті, які починаються з точки («приховані»), і звичайно не видаються.

4. Приклад 4: після виконання команди:

```
ls -aisn
```

буде видана різноманітна інформація:

- ✓ список всіх файлів, включаючи ті, які зазвичай не виводяться (*-a*);
- ✓ номера описувачів файлів будуть виведені в лівій колонці (*-i*);
- ✓ розміри файлів (в блоках) виводяться в другій колонці (*-s*);
- ✓ нарешті, будуть видані числові ідентифікатори власників та груп (*-n*).

1.1.7.4. mkdir

Команда *mkdir* виконує створення каталогу.

СИНТАКСИС

mkdir [-m режим_доступу] [-p] каталог ...

ОПИС

За командою *mkdir* створюється один або кілька каталогів з режимом доступу «***rwx***» (читання, запис і право на пошук доступні всім). Стандартні файли створюються автоматично; їх не можна створити по імені. Для створення каталогу необхідно мати право запису в поточному каталозі.

Ідентифікатори власника та групи нових директорій встановлюються відповідно рівними реальним ідентифікаторам власника та групи процесу.

Командою *mkdir* обробляються дві опції, опис яких наведено в табл. 1.4.

Таблиця 1.4. Опції, які можна використовувати із командою *mkdir*

m	Режим доступу. Явне завдання режиму доступу для створюваних каталогів [см. команду *chmod* для більш детальної інформації].
p	При використанні цієї опції перед створенням нового каталогу попередньо створюються всі неіснуючі проміжні каталоги.

ПРИКЛАДИ

Приклад 1. Щоб створити піддерево каталогів *tmpdir/temp/dir*, потрібно виконати наступну команду:

```
mkdir -p tmpdir/temp/dir
```

1.1.7.5. cp

Команда *cp* копіює файл чи файли.

СИНТАКСИС

cp файл1 [файл2 ...] цільовий_файл

ОПИС

Команда *cp* копіює *файл1* в *цільовий_файл*. *Файл1* не повинен співпадати з *цільовим_файлом* (будьте уважні при використанні метасимволів **shell'**а). Якщо *цільовий_файл* є каталогом, то *файл1*, *файл2*,..., копіюються в нього під своїми іменами. Тільки в цьому випадку можна вказувати кілька вихідних файлів.

Якщо *цільовий_файл* існує і не є каталогом, його попередній вміст втрачається. Режим доступу власника та групи *цільового_файла* при цьому не змінюється.

Якщо *цільовий_файл* не існує або є каталогом, нові файли створюються з тими ж режимами, що і вихідні (крім «липкого» біта, якщо ви не суперкористувач). Час останньої модифікації *цільового_файла* (і останнього доступу, якщо він не існував), а також час останнього доступу до вихідних файлів встановлюється рівним часу, коли виконується копіювання. Якщо *цільовий_файл* був посиланням на інший файл, всі посилання зберігаються, а вміст файлу змінюється.

ПРИКЛАДИ

Приклад 1. Щоб скопіювати файл *tmp1.txt* із домашнього каталогу у підкаталог *MY* поточного каталогу, потрібно виконати наступну команду:

```
cp   ~/tmp1.txt   MY
```

В цьому прикладі:

cp – це команда копіювання,

~/tmp1.txt – це *файл1 ,* де символ «~» означає домашній каталог,

MY – це *цільовий_файл*; оскільки *MY* – це каталог, то файл *tmp1.txt* копіюється в цей каталог під своїм іменем.

1.1.7.6. mv

Команда *mv* виконує переміщення (перейменування) файлів.

СИНТАКСИС

mv [-f] файл1 [файл2 ...] цільовий_файл

ОПИС

Команда *mv* переміщує (перейменовує) *файл1* в *цільовий_файл*. *Файл1* не повинен співпадати з *цільовим_файлом*. Якщо *цільовий_файл* є каталогом, то *файл1*, *файл2*, ..., переміщуються в нього під своїми іменами. Тільки в цьому випадку можна вказувати кілька вихідних файлів

Якщо *цільовий_файл* існує і не є каталогом, його старий вміст **втрачається**.

Якщо *файл1* є каталогом, то він перейменовується в *цільовий_файл*, тільки якщо у цих двох каталогів загальний надкаталог; при цьому всі файли, що знаходилися в *файлі1*, переміщуються під своїми іменами в *цільовий_файл*.

ПРИКЛАДИ

Приклад 1. Для того щоб перемістити файл *tmp1.txt* із домашнього каталогу у підкаталог *NEW* поточного каталогу під новим ім'я *my_file.txt*, потрібно виконати наступну команду:

```
mv   ~/tmp1.txt   NEW/my_file.txt
```

В цьому прикладі:

mv - це команда копіювання,

~/tmp1.txt – це *файл1 ,* де символ «~» означає домашній каталог,

NEW/my_file.txt – це *цільовий_файл*; де *NEW*– ім'я підкаталогу куди потрібно перенести файл, а *my_file.txt* – це ім'я під яким *файл1* зберігатиметься в новому підкаталозі.

1.1.7.7. rm

Команда *rm* видаляє файли або каталоги.

СИНТАКСИС

rm [-f] [-i] файл ...
rm -r [-f] [-i] каталог ... [файл ...]

ОПИС

Команда *rm* служить для видалення зазначених файлів з каталогу. Якщо задане ім'я є посиланням на файл, то файл знищується. Для видалення користувач повинен мати ***право запису в каталог***; мати ***право на читання або запис файлу не обов'язково***.

Якщо немає права на запис у файл та стандартний вивід є термінал, то видається режим доступу до файлу і запитується підтвердження; якщо воно починається з літери ***y***, то файл видаляється, інакше – ні. Якщо стандартний ввід не є термінал, команда *rm* поводиться так само, як при наявності опції ***-f***.

Наявні ключі, які можна використовувати із командою *rm,* наведені в табл. 1.5.

Таблиця 1.5. Ключі, доступні до використання із командою *rm*

Символ	Значення опції, яка відповідає цьому символу
-f	Команда не видає повідомлень, коли видаляється файл який не існує, не запитує підтвердження при видаленні файлів, на запис в які немає прав. Якщо немає права і на запис у каталог, файли залишаються.

Символ	Значення опції, яка відповідає цьому символу
	Повідомлення про помилку видається лише при спробі видалити каталог, на запис в який немає прав (див. опцію ***-r***).
-r	Відбувається рекурсивне видалення всіх каталогів і підкаталогів, перелічених у списку аргументів. Спочатку каталоги спустошуються, потім видаляються. Підтвердження при видаленні файлів, на запис в які немає прав, не запитується, якщо задана опція ***-f*** або стандартний ввід не є термінал і не задана опція ***-i***.
-i	Перед видаленням кожного файлу запитується підтвердження. Опція ***-i*** усуває дію опції ***-f***; вона діє навіть тоді, коли стандартний ввід не є термінал.

Опція ***-i*** часто використовується разом з ***-r***.

ПРИКЛАДИ

Приклад 1. Після виконання команди:

```
rm  -ir dirname
```

запитується підтвердження:

```
directory dirname: ?
```

При позитивній відповіді запитуються підтвердження на видалення всіх розміщених в каталозі файлів (для підкаталогів виконуються ті ж дії), а потім підтвердження на видалення самого каталогу.

Приклад 2. Прикладом частого ***руйнівного жарту*** над новачками є порада набрати команду:

rm -rf або rm -rf /

яка рекурсивно видаляє всі файли та каталоги від поточного, чи від кореневого каталога, що, відповідно, призводить до втрати даних та, можливо, пошкодження операційної системи. На відміну від високорівневих оболонок користувача, видалення командою ***rm*** призводить до безповоротної втрати, оскільки цією програмою не передбачено переміщення видалених об'єктів до «кошика». НЕ РОБІТЬ ТАК.

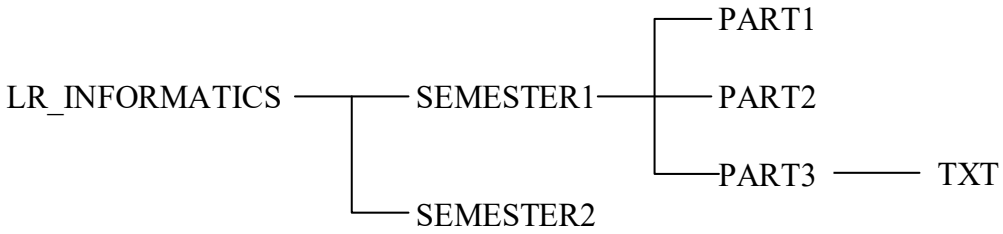
1.2. Завдання для виконання

В цій лабораторній роботі потрібно використовувати тільки засоби командного рядка.

1. Встановити клієнт ***PuTTy*** та виконати усі його налаштування, як показано в теоретичних відомостях до цієї лабораторної роботи (якщо на комп'ютері є будь-який інший SSH-клієнт, то ***PuTTy*** можна не встановлювати, а користуватися зручним для вас клієнтом).
2. Увійдіть у систему, використовуючи як свій логін та пароль за замовчуванням. Змінити пароль за замовчуванням на більш зручний та запам'ятати його.
3. Продемонструвати уміння користуватися довідкою в UNIX-подібний системах (команда *man*), вивівши на екран інформацію про команду *uname*. Визначте тип і версію операційної системи нашого сервера.
4. За допомогою команди *apropos 'working directory'* вивести на екран основні команди роботи з робочим каталогом.
5. Вивчіть довідку по команді *ls*, звертаючи увагу на вплив ключів на роботу команди.
6. Перегляньте вміст домашнього каталогу в декількох форматах (скориставшись командою *ls* із відповідними ключами) та поясніть отриманий результат:

- ✓ здійснити перегляд каталогу з виводом файлів в кілька колонок без детальної інформації про файли;
- ✓ здійснити перегляд каталогу у довгому форматі (із зазначенням режиму доступу, власника, групи та датою останньої модифікації) з переглядом розміру файлів в блоках;
- ✓ здійснити перегляд у вільному форматі – імена файлів повинні виводитися через кому.

7. Перегляньте вміст будь-якого каталогу у системі, який знаходиться вище вашого домашнього каталогу.
8. Створіть у домашньому каталозі структуру каталогів як показано на малюнку:



9. До каталогу *PART1* скопіюйте декілька будь-яких файлів (можна створити свої файли або скопіювати із будь-якого каталогу, до якого є доступ, наприклад, можна скопіювати кілька текстових файлів налаштувань з каталогу */etc*).
10. Зробити копії файлу із каталога *PART1* в каталоги *PART2* та *TXT*.
11. Виведіть на екран вміст цих каталогів.
12. Видаліть рекурсивно каталог *PART3*.
13. Видаліть рекурсивно каталог *LR_INFORMATICS*.
14. Виведіть на екран і перегляньте один з скопійованих текстових файлів.

1.3. Контрольні питання

1. У чому полягають принципові відмінності між звичайним користувачем і суперкористувачем (root) в ***UNIX***-подібних системах? Які потенційні ризики пов'язані з роботою під обліковим записом суперкористувача?
2. Яким чином можна переглянути права доступу до конкретного файлу або директорії? Поясніть значення символів у виведеному рядку прав доступу (наприклад, -rwxr-xr--). Як визначити власника та групу файлу/директорії?
3. Перелічіть основні команди для навігації та роботи з каталогами в ***UNIX***-подібних системах. Опишіть призначення та наведіть приклади використання команд **pwd**, **cd**, **ls**, **mkdir**, **rmdir**. Чим відрізняється **rmdir** від **rm -r** при видаленні директорій?
4. Назвіть найбільш вживані команди для виконання операцій з файлами в UNIX-подібних системах. Поясніть призначення та наведіть приклади використання команд **touch, cp, mv, rm, cat, less, head, tail**. Які існують способи перенаправлення стандартного виводу та стандартного вводу команд?
5. Чи впливають ключі (опції) на роботу команд в ***UNIX***-подібних системах? Наведіть декілька прикладів команд з різними ключами та поясніть, як змінюється їхня поведінка. Як отримати довідкову інформацію про доступні ключі для певної команди?
6. Що таке абсолютний та відносний шляхи в файловій системі ***UNIX***? Наведіть приклади. Коли доцільно використовувати кожен з них?
7. Яким чином можна знайти файли або директорії, що відповідають певним критеріям (наприклад, за назвою, розміром, датою модифікації)? Які команди для цього використовуються?
8. Що таке конвеєр (pipeline) в ***UNIX***? Як він створюється та для чого використовується? Наведіть приклад використання конвеєра з кількох команд.

9. Які команди використовуються для перегляду та керування запущеними процесами в ***UNIX***? Поясніть призначення команд **ps**, **top** (або **htop**), **kill**. Як відправити сигнал процесу? Які найпоширеніші сигнали ви знаєте?
10. Як можна отримати інформацію про використання дискового простору в ***UNIX***? Які команди для цього існують?
11. Які команди використовуються для роботи з архівами та стисненням файлів (наприклад, **tar**, **gzip**, **zip**)? Поясніть основні операції (створення, розпакування, стиснення).
12. Як можна отримати довідку по конкретній команді в ***UNIX*** без використання зовнішніх ресурсів (наприклад, інтернет)?

2. Лабораторна робота №2
Робота з UNIX-подібними системами: маніпуляції із файлами

Мета роботи: вивчення основних прийомів роботи з файлами в UNIX-подібних системах.

2.1. Теоретичні відомості

2.1.1. Файлова система

Будь який файл ОС ***UNIX*** у відповідності з його типом може бути віднесений до однієї з наступних чотирьох груп: звичайні файли, каталоги, спеціальні файли, канали.

2.1.2. Довідка по деяких командах

2.1.2.1. cat

Команда *cat* виконує злиття та вивід файлів

СИНТАКСИС

cat [-u] [-s] [-v] [-t] [-e] файл ...

ОПИС

Команда *cat* по черзі читає вказані файли та видає їх вміст на стандартний вивід. Так, наприклад, команда

```
cat f
```

друкує вміст файлу *f*, а команда

```
cat f1 f2 > f3
```

зливає перші два файли (*f1* та *f2*) та поміщає результат в третій (*f3*).

Щоб додати файл *f1* до файлу *f2* (тобто дописати вміст файлу *f2* до вже наявної інформації у файлу *f1*), треба виконати команду

```
cat f1 >> f2
```

Якщо жоден із файлів не вказано або серед аргументів зустрівся -, команда *cat* читає дані зі стандартного вводу. Для завершення вводу слід натиснути *Ctrl-D (символ кінця передачі, який згодом також стали використовувати як символ кінця файлу, EoF).* Ключі, які можна використовувати із командою *cat* наведені в табл. 2.1.

Таблиця 2.1. Ключі, доступні до використання із командою *cat*

Символ	Значення опції, яка відповідає цьому символу
-u	Вивід не буферизується (за замовчуванням буферизується).
-s	Не повідомляється про неіснуючі файли.
-v	Візуалізація недрукованих символів (крім табуляцій, переведення рядків і переходів до нової сторінки). Керуючі символи зображуються у вигляді *^X (CTRL + X);* символ *DEL* - у вигляді *^?*. Символи, що не входять в набір *ASCII* видаються у вигляді *M-x*, де *x* - визначається молодшими семи бітами символа.

В табл. 2.2 наведені опції, які можна використовувати у комбінації із ключем ***-v***.

Таблиця 2.2. Ключі, які доступні до використання разом із ключем ***-v*** в команді *cat*

Символ	Значення опції, яка відповідає цьому символу
-t	Візуалізація символів табуляції у вигляді *^I*.
-e	Візуалізація символів переведення рядка у вигляді *$*

Якщо опція ***-v*** не вказана, то опції ***-t*** та ***-e*** ігноруються.

Це цікаво

Виконання команди cat > *file дозволяє перенаправляти у файл символи, набрані з консолі. Цей прийом використовувався, зокрема, для створення файлів вихідного коду у прикладах на Фортрані, С та інших мовах, наприклад:*

```
demo% cat greetings.f
    PROGRAM GREETINGS
    PRINT *, 'Real programmers write Fortran!'
    END
```

Текст цієї простої програми перетворюється на файл виконуваної програми шляхом компіляції:

```
demo% f95 greetings.f
```

За замовчуванням виконуваний файл мав назву a.out, що своєю чергою походить від скорочення "assembler output", тобто в результаті компіляції створюється файл a.out.

На основі цих традицій Інтернет-жартівники створили мем:

Real programmers write:
cat > a.out

який можна інтерпретувати так, що справжні програмісти безпосередньо з консолі набирають машинний код програми, виконуючи компіляцію подумки.

2.1.3. Зміна напрямку вводу/виводу

Механізм зміни напрямку вводу/виводу є одним з найбільш елегантних, потужних та водночас простих механізмів ОС ***UNIX***. Мета, яка ставилася при розробці цього механізму, полягала в наступному. Оскільки ***UNIX*** – це інтерактивна система, то зазвичай програми вводять текстові рядки з терміналу та виводять текстові рядки на екран терміналу. Для того щоб забезпечити більш гнучке використання таких програм, бажано вміти забезпечити їм введення з файлу або з виходу інших програм і направити їх результат у файл або на вхід інших програм.

Реалізація механізму ґрунтується на наступних властивостях ОС ***UNIX***.

По-перше, будь яке введення/виведення трактується як введення з деякого файлу та виведення в деякий файл. Клавіатура й екран терміналу теж інтерпретуються як файли (перший можна тільки читати, а в другий можна тільки писати).

По-друге, доступ до будь-якого файлу здійснюється через його дескриптор (додатне ціле число).

Фіксуються три значення дескрипторів файлів:

- ✓ Файл з дескриптором 1 називається файлом стандартного вводу (**stdin**),
- ✓ файл з дескриптором 2 – файлом стандартного виводу (**stdout**),

- ✓ файл з дескриптором 3 – файлом стандартного виводу діагностичних повідомлень (**stderr**).

По-третє, програма, запущена в деякому процесі, "успадковує" від батьківського процесу всі дескриптори відкритих файлів.

У головному процесі інтерпретатора командної мови файлом стандартного введення є клавіатура терміналу користувача, а файлами стандартного виводу і виводу діагностичних повідомлень – екран терміналу. Однак при запуску будь-якої команди можна повідомити інтерпретатору (засобами відповідної командної мови), який файл повинен служити файлом стандартного вводу для програми яка запускається і який файл повинен служити файлом стандартного виводу чи виведення діагностичних повідомлень для програми. Тоді інтерпретатор перед виконанням системного виклику **exec** відкриває вказані файли, підміняючи зміст дескрипторів 1, 2 та 3.

Звичайно, те ж саме може зробити і будь-яка інша програма, яка запускає третю програму в спеціально створеному процесі. Отже, все, що потрібно для нормального функціонування механізму зміни напрямку вводу/виводу – це дотримуватися при програмуванні угоди про використання дескрипторів **stdin**, **stdout** та **stderr**. Це не дуже важко, оскільки в найбільш поширених функціях бібліотеки вводу/виводу **printf**, **scanf** та **error** взагалі не потрібно вказувати дескриптор файлу. Функція **printf** неявно використовує **stdout**, функція **scanf** – **stdin**, а функція **error** – **stderr**.

2.1.4. Групування команд

Перерахуємо засоби групування команд та зміни напрямку вводу/виводу:

- ✓ **cmd1 arg ...; cmd2 arg ...; ... cmdN arg ...** – послідовне виконання команд;
- ✓ **cmd1 arg ...& cmd2 arg ...& ... cmdN arg ...** – асинхронне виконання команд;

- ✓ **cmd1 arg ... && cmd2 arg ...** – залежність наступної команди від попередньої таким чином, що подальша команда виконується, якщо попередня видала нульове значення;
- ✓ **cmd1 arg ... || cmd2 arg ...** – залежність наступної команди від попередньої таким чином, що подальша команда виконується, якщо попередня видала ненульове значення;
- ✓ **cmd > file** – стандартна зміна напрямку виводу в файл **file**;
- ✓ **cmd >> file** – стандартна зміна напрямку виводу в кінець файлу **file** (дописування);
- ✓ **cmd < file** – стандартне введення інформації із файлу **file**;
- ✓ **cmd1|cmd2** – конвеєр команд, в якому стандартний вихід команди **cmd1** спрямований на стандартний вхід команди **cmd2**.

2.1.5. Текстові редактори

2.1.5.1. vi

vi, view, vedit – екранно-орієнтований редактор (**vi***sual editor*)

СИНТАКСИС

vi [option...] [command...] [filename...]
view [option...] [command...] [filename...]
vedit [option...] [command...] [filename...]

ОПИС

Зараз в це складно повірити, але у ранніх випусках ***UNIX***-систем не було програми-редактора текстів у нашому теперішньому розумінні, тобто програми, яка б відображала результат редагування безпосередньо на екрані. Натомість, використовувався командний редактор **ed**. Користувач спочатку повинен був

переглянути файл за допомогою програм відображення, визначити необхідні правки і дати команду редактору, наприклад, «*додати символ А у третій рядок*» і т.п. Робота з такими засобами вимагала уважності та вправності. Тому пізніше був створений редактор, де процес зміни тексту відображався на екрані по мірі редагування. Пізніше цей підхід розвинувся у концепцію *WYSIWYG – What You See Is What You Get*, тобто відтворення на екрані зображення близького до того, що буде надруковано. Проте навіть попри відображення процесу редагування користування редактором *vi* вимагає певних навичок, з чим пов'язані численні жарти.

Цей редактор поставляється з переважною більшістю ***UNIX***-подібних систем. Крім того, в аварійних режимах роботи системи дуже часто *vi* – це єдиний доступний редактор, тому базові навички роботи з ним знадобляться для відновлення працездатності пошкодженої віддаленої системи.

2.1.5.1.1 Запуск та завершення роботи редактора VI

Перелік засобів запуску редактора *vi* та шляхи завершення роботи в цьому редакторі наведені в табл. 2.3.

Таблиця 2.3. Синтаксис основних команд запуску та завершення роботи в редакторі *vi*

Синтаксис	Опис
vi *filename*	Відкриття файлу *filename* для редагування
vi -r *filename*	Відкрити останню збережену версію *filename* після аварійного виходу

Синтаксис	Опис
vi *filename file2* ...	Відкрити *filename*, потім *file2* ... Перехід до наступного файлу - :n
ZZ або :wq або :x	Зберегти модифікації та завершити *vi*
:w	Зберегти модифікації в поточному файлі
:w!	Безумовно зберегти поточний файл
:w *file*	Зберегти поточний файл під ім’ям *file*
:q!	Завершити *vi* без зберігання модифікацій

Жарт із інтернету: *vi може працювати у двох режимах – пищати або все псувати. (Інтернет-фольклор)*

Редактор *vi* може працювати у двох режимах: в ***командному*** режимі та режимі ***вводу***. Відразу після запуску редактор знаходиться в командному режимі, в якому можна переміщатися по тексту, копіювати і вставляти текст з буфера. Спроби натискати символи на клавіатурі інтерпретуються як команди і якщо певний символ чи комбінація не є чинною командою, видається звуковий сигнал, проте не відбувається зміна тексту на екрані («режим пищати»).

Режим введення, тобто власне редагування тексту, включається командами вставки, доповнення, заміни. Натискання [*ESC*] повертає редактор у командний режим. Більшість команд виконується відразу після натискання клавіші. Виняток – команди рядкового редактора ***ex***, що починаються з двокрапки, які виконуються після натискання [*Return*].

Зберігання файла або завершення роботи програми після редагування вимагає переходу у командний режим [*ESC*]. Новачки часто забувають про це, тому запит «*how to exit vi*» у пошуку *Google* повертає понад триста мільйонів результатів.

Порада: Перед будь-якими діями з документом введіть таку команду

```
:set showmode
```

Після цього поточний режим роботи редактора відображається в правому нижньому кутку екрана.

2.1.5.1.2 Вставка тексту

В таблицях 2.4 та 2.5 наведені шляхи вставки тексту.

Таблиця 2.4. Шляхи вставки тексту в командному режимі

Синтаксис	Опис
i	Вставка перед курсором
I	Вставка перед рядком
a	Додавання за курсором
A	Додавання після рядка
:r *file*	Вставити вміст *file* після поточного рядка

Таблиця 2.5. Шляхи вставки тексту в режимі вводу

Синтаксис	Опис
CTRL-i або TAB	Вставити зміщення вправо на *shift width*
CTRL-d	Зменшити зміщення вправо на *shift width*
CTRL-h або Backspace	Видалити попередній символ
CTRL-w	Видалити попереднє слово
CTRL-u	Видалити попередній рядок

2.1.5.1.3 Переміщення по тексту

Керуючі символи, які дозволяють переміщатися по тексту, та їх опис наведені в табл. 2.6.

Таблиця 2.6. Керуючі символи переміщення по тексту

Синтаксис	Опис
h	Ліворуч
j	Вниз
k	Вгору
l	Праворуч
«стрілки»	Переміщення по тексту.

Синтаксис	Опис
w	До наступного слова
b	До початку слова
^	До першого не непробільного символу рядка
+ або <cr>	До першого символу наступного рядка
-	До першого непробільного символу попереднього рядка
e	До кінця слова
(	До початку речення
)	До кінця речення
{	До початку абзацу
}	До кінця абзацу
Примітка: абзаци розділяються порожнім рядком.	
0 або |	До початку рядка
$	У кінець рядка
1G	До першого рядка файлу
G	До останнього рядка файлу
H	До верхнього рядка екрана
L	До нижнього рядка екрана

Синтаксис	Опис
CTRL-d	Прокрутка вниз на пів-екрана
CTRL-f	Прокрутка вниз на екран
CTRL-u	Прокрутка вверх на пів-екрана
CTRL-b	Прокрутка вверх на екран
%	К парної дужці: ( ), { }, [ ]

2.1.5.1.4 Видалення тексту

Більшість команд видалення утворюються з символу **d** і команди переміщення. Наприклад, **dw** видаляє слово. Синтаксис інших керуючих команди та символів видалення тексту наведені в табл. 2.7:

Таблиця 2.7. Керуючі символи видалення тексту

Синтаксис	Опис
x	Видалити поточний символ
nx	Видалити n символів праворуч, починаючи з поточного
X	Видалити символ ліворуч від курсору
nX	Видалити n символів ліворуч від курсору
dd або :d	Видалити поточний рядок

Синтаксис	Опис
"(a-z)dd	Вирізати поточний рядок в іменований буфер

2.1.5.1.5 Копіювання в буфер

Подібно командам видалення, більшість команд копіювання в буфер складаються з символу **y** і команди переміщення.

Таблиця 2.8. Керуючі символи копіювання тексту в буфер

Синтаксис	Опис
yy або :y	Копіювати поточний рядок
"(a-z)yy	Копіювати поточний рядок в іменований буфер (від a до z)

2.1.5.1.6 Модифікація тексту

Команди зміни (модифікації) діють як команди видалення з подальшим переведенням в режим введення. Більшість з них складається з символу **c** і команди переміщення. Наприклад **cw** замінює слово.

2.1.5.1.7 Вставка з буферу

Команди вставки тексту із буферу наведені в табл. 2.9.

Таблиця 2.9. Керуючі символи вставки тексту із буфера

Синтаксис	Опис
p	Помістити текст з буфера після курсору
P	Помістити текст з буфера перед курсором
"(a-z)p або "(a-z)P	Помістити текст з буфера (з a до z) після/перед поточним рядком

2.1.5.1.8 Пошук/заміна підрядків

Інколи виникає потреба знайти якийсь зразковий текст в певному файлі. Шляхи пошуку в редакторі *vi* наведені в табл. 2.10.

Таблиця 2.10. Керуючі символи пошуку та\чи заміни зразкового тексту

Синтаксис	Опис
/*str*	Пошук *str* вперед від курсору
?*str*	Пошук *str* назад от курсору
n	Повторити пошук в тому же напрямку
:n,ms/*str1*/*str2*/*opt*	Шукати рядок *str1* з рядка n до m (n и m можуть бути пропущені); замінити *str1* на *str2*; з опціями *opt*. Можливі опції: c - підтверджувати заміну, p - друк модифікованого рядка
&	Повторити останню команду :s

Синтаксис	Опис
:g/*str1*/s/*str2*/*str3*/	Знайти рядок, яка містить *str1*, та замінити *str2* на *str3*
;	Повторити пошук
,	Повторити пошук в зворотному напрямку

2.1.5.1.9 Регулярні вирази

Інколи, в виразах, можуть зустрічатися символи чи їх комбінація, які мають спеціальне значення. Перелік деяких таких регулярних виразів наведено в табл. 2.11.

Таблиця 2.11. Керуючі символи роботи із регулярними виразами

Синтаксис	Опис
. (крапка)	Будь-який символ, окрім «нового рядка»
*	0 або більше входжень кожного символу
[...]	Кожен із зазначених символів
[^...]	Будь-який з символів, крім зазначених
\<	Строго на початку слова
\>	Строго в кінці слова
^	На початку рядка

Синтаксис	Опис
$	В кінці рядка
\(...\)	Група умов
\n	Значення n-ої групи
\\	Скасування особливої дії символу \

2.1.5.1.10 Лічильники

Практично перед будь-якою командою можна вказати кількість її повторень. Наприклад, **5dw** призведе до видалення 5 слів, а **3fe** - до переміщення курсору вперед на 3-е входження символу «*e*». Подібним чином можна діяти і з командами вставки, наприклад, вставити 100 однакових рядків.

2.1.5.1.11 Режими VI

Зауважте: вводиться **:set option** для включення режиму та **:set nooption** для виключення. Щоб встановити налаштування для кожного запуску *vi*, в домашньому каталозі створюється файл *.exrc*, що містить дані команди без двокрапки перед ними.

Таблиця 2.12. Керуючі символи зміни режиму роботи редактора *vi.*

Синтаксис	Опис
:set ai	Автоматичний відступ (по попередньому рядку)

Синтаксис	Опис
:set all	Вивести на екран всі режими
:set eb	Звуковий сигнал при помилках
:set ic	Ігнорувати регістр при пошуку
:set list	Показувати символи табуляції (^ l) і кінця рядка ($)
:set nu :set number	Показати номер рядка
:set showmode	Показати поточний режим в нижній частині екрану
:set wa	Виключити нормальну перевірку перед записом

Інше

Синтаксис	Опис
~	Зміна регістра символу над курсором
J	Злиття наступного рядка з поточним
nJ	Злиття n рядків
.	Повторити останню команду, яка змінює текст
u	Скасувати останню зміну
U	Відмінити всі зміни в рядку

Синтаксис	Опис
CTRL-G	Показати ім’я файлу, номер рядка, загальне число рядків і положення в файлі (%)

Редактор *vi* було розроблено під використання алфавітної клавіатури, без блоків стрілок та додаткової цифрової клавіатури. Якщо натискання клавіш стрілок приводить до пошкодження відображуваного тексту («режим все псувати»), то, найімовірніше, у програмі терміналу не налаштовано належну трансляцію кодів символів.

2.1.5.2. joe

joe – Joe’s Own Editor, консольний екранний редактор. Це потужний текстовий редактор, який можна зустріти у багатьох системах. У найпростішому випадку запускається командою

```
joe ім’я.файла
```

У нижньому рядку відображається підказка

```
Type Ctrl-K Q to exit or Ctrl-K H for help
```

Новачкам варто скористатися порадою і увімкнути відображення підказок комбінацією «*Ctrl-K H*». Вікно редактора матиме вигляд:

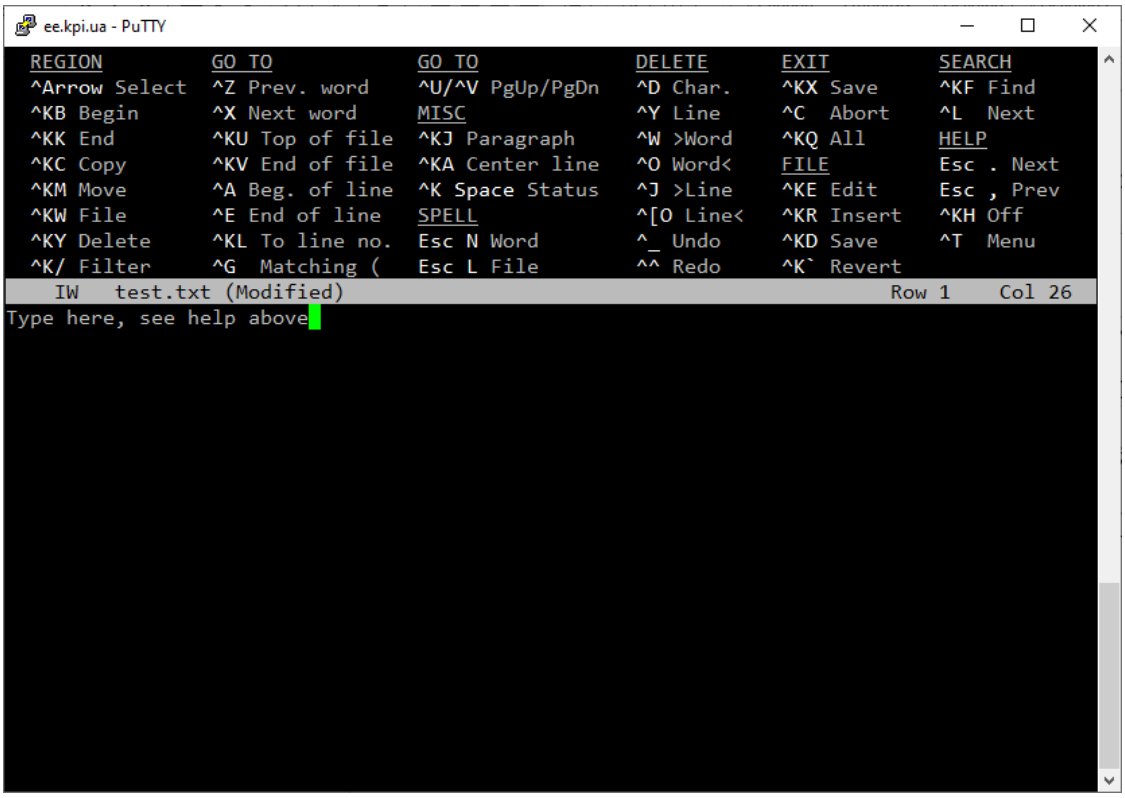


Рис. 2.1. Вікно редактора joe з підказками

Символ «^» означає клавішу *Ctrl*, а екран допомоги відображає підказки на найпоширеніші команди. Цей редактор не входить до стандартної конфігурації системи, тому на іншій ***UNIX***-машині його може не бути, якщо адміністратор не подбав про це.

2.1.5.3. nano

nano – Nano's ANOther editor, компактний редактор із дружнім інтерфейсом. За замовчуванням, у нижніх двох рядках відображається підказка щодо найуживаніших команд, як це показано на рис. 2.2:

Рис. 2.2. Вигляд вікна редактора *nano* із двома рядками підказок внизу вікна

Цей редактор не входить до стандартної конфігурації системи, тому на іншій UNIX-машині його може не бути, якщо адміністратор не подбав про це.

2.1.5.4. nedit

NEdit (Nirvana editor) – це текстовий редактор із редактором вихідного коду для системи *X Window*. Він має інтерфейс, подібний до текстових редакторів у *Microsoft Windows* та *Macintosh*. Спочатку він був розроблений Марком Еделем для *Fermilab* та випущений під дуже ліцензією яка накладала багато обмежень, але сьогодні він поширюється під ліцензією із меншими обмеженнями GPL-2.0 або пізнішою версією і розроблений як незалежний відкритий код.

Перед використанням текстового редактору *NEdit* потрібно переконатися, що у вас запущено сервер *X11.* Упевнитись, що клієнт працює, можна за іконкою у треї, як це показано на рис. 2.3:

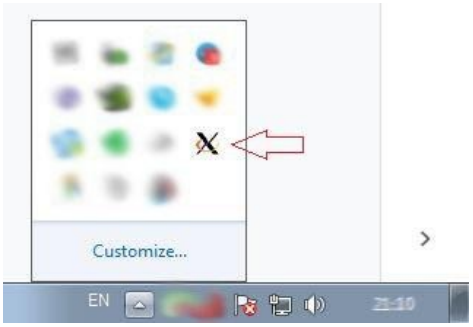


Рис. 2.3. Значок запущеного сервер *X11* у треї

Якщо ця іконка відсутня, то потрібно спочатку запустити *VcXsrv* , а потім треба підключитися до сервера за допомогою термінального клієнта ***PuTTY***.

Відкрити текстовий редактор із графічним інтерфейсом *nedit* можна за допомогою команди *nedit*, яка набирається в командному рядку вашого ***SSH*** клієнта, після чого відкривається графічне вікно редагування файлів із інтуітивно зрозумілим інтерфейсом, як це показано на рис. 2.4:

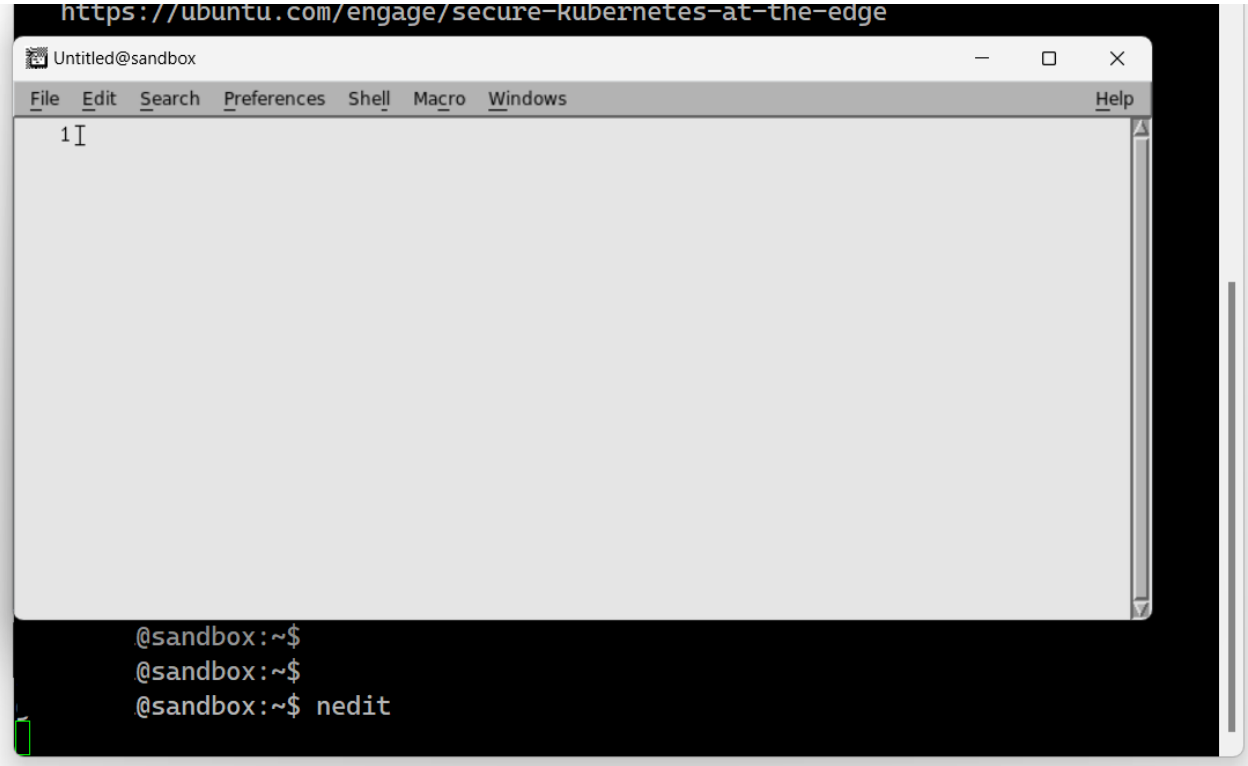


Рис. 2.4. Запуск редактора *nedit* за допомогою команди *nedit* та вигляд вікна цього редактора одразу після відкриття

Запуск редактора призведе до того, що консоль буде зайнята його виконанням і ніякі інші команди запустити буде неможливо, допоки редактор не буде закритий. Аби запобігти цьому, редактор можна запустити асинхронно за допомогою команди *nedit &*.

2.2. Завдання для виконання

1. Скопіюйте з консолі (наберіть з клавіатури) текстовий файл з розширенням *inf*. У нього запишіть номер групи, прізвище, ім'я. Збережіть файл у домашньому каталозі. (*Підказка*: можна використовувати команду ***cat*** та символи перенаправлення).
2. Скопіюйте цей файл під іншим ім'ям.
3. Перейменуйте копію так, щоб вона мала розширення *txt*.
4. За допомогою команд перенаправлення вводу/виводу кілька разів скопіюйте інформацію з будь якого файлу в новий файл так, щоб кількість рядків склало не менше 20.
5. Скористайтеся текстовим редактором *nedit* для редагування останнього файлу. В кінці файлу допишіть поточну дату та день тижня. Збережіть цей файл.
6. За допомогою клієнта *WinSCP* скопіюйте останній файл із сервера на свій робочий комп'ютер.
7. Продемонструйте скопійований файл на своєму комп'ютер. Допишіть в нього останнім рядком назву факультету та зберегти під новим ім'ям.
8. За допомогою клієнта *WinSCP* скопіюйте останній файл із свого робочого комп'ютера на сервер та продемонструйте розташування копії в домашньому каталозі та її вміст.

2.3. Контрольні питання

1. Які можливості надають команди перенаправлення стандартного вводу (<), стандартного виводу (>), та додавання до файлу (>>) в ***UNIX***-подібних системах? Наведіть приклади їх використання для створення файлів, запису в них даних та об'єднання вмісту файлів.
2. Які програмні продукти (команди) вам відомі для безпечного та небезпечного скачування файлів з віддаленого сервера на локальну машину та завантаження файлів з локальної машини на віддалений сервер через термінал? Опишіть основні принципи їхньої роботи та наведіть приклади використання.
3. Які текстові редактори для роботи з текстовими файлами безпосередньо в терміналі ***UNIX***-подібних систем вам відомі? Назвіть хоча б три.
4. Порівняйте переваги та недоліки щонайменше двох відомих вам текстових редакторів для роботи з файлами в ***UNIX***-подібних системах (наприклад, **nano, vi/vim, emacs** у текстовому режимі) з точки зору зручності використання для початківців, функціональності, можливостей налаштування та вимог до ресурсів.
5. Як створити порожній файл у ***UNIX***-подібній системі, не використовуючи текстовий редактор? Наведіть декілька способів.
6. Яка команда використовується для перейменування файлів та/або переміщення файлів між каталогами? Поясніть її синтаксис та наведіть приклади обох операцій.
7. Як скопіювати один або декілька файлів до іншого каталогу? Які існують опції для рекурсивного копіювання директорій?
8. Яка команда використовується для видалення файлів та директорій? Поясніть різницю між видаленням файлів та порожніх/непорожніх директорій. Які існують опції для примусового видалення?

9. Як переглянути вміст текстового файлу без відкриття текстового редактора? Назвіть декілька команд для цього та поясніть їхні особливості (наприклад, посторінковий перегляд, виведення перших/останніх рядків).
10. Чи можна одночасно виконати декілька команд перенаправлення для однієї команди? Наведіть приклади.
11. Як можна використати перенаправлення виводу для збереження результату виконання команди у файл? Як додати вивід команди до вже існуючого файлу, не стираючи його вміст?
12. Які особливості роботи з двійковими файлами при скачуванні/завантаженні порівняно з текстовими файлами? Чи існують спеціальні опції для цього у відомих вам програмах?

3. Лабораторна робота №3
Основи роботи з UNIX-подібними системами: файловий менеджер Midnight Commander

Мета роботи: вивчення основних прийомів і отримання практичних навичок маніпуляції із файлами в ***UNIX***-подібних системах.

3.1. Теоретичні відомості

3.1.1. Midnight Commander

Інколи, при роботі із термінальними клієнтами, зручно користуватися файловими менеджерами, які наочно відображають вміст однієї чи кількох локацій у файловій системі та пропонують засоби для типових дій над файлами, групами файлів, каталогами, як от копіювання, переміщення, перейменування, видалення, перегляд вмісту, редагування тощо. Свого часу для був популярний файловий менеджер *Norton Commander* для операційної системи MS-DOS, впізнаваний за характерним інтерфейсом з панелями синього кольору. Він працював у текстовому режимі, а для створення таблиць використовувалися символи псевдографіки. Відтоді було створено велику кількість файлових менеджерів з подібним інтерфейсом, у тому числі і для ***UNIX***-подібних систем Прикладом такого менеджера може слугувати *GNU Midnight Commander* (*MC*) – це двохпанельний файловий менеджер з текстовим інтерфейсом типу *Norton Commander* для ***UNIX***-подібних операційних систем, який було створено в 1994 році Мігелем де Ікас.

Midnight Commander є частиною проекту *GNU*, а це означає що всі розробники передали свої права на програму на користь *Free Software Foundation*. Зараз *Midnight Commander* є складовою частиною більшості дистрибутивів *Linux*.

Midnight Commander надає цілий ряд можливостей при роботі із файлами та каталогами:

- *MC* дозволяє роботу із різними архівами та образами файлових систем, таким самим чином, якщо б вони були звичайними каталогами,
- Є можливість роботи з мережевими файловими системами *Samba*, *FTP*, *SFTP*,
- Можна виконувати копіювання та переміщення файлів у фоновому режимі,
- *MC* в своєму складі моє вбудований текстовий редактор з підсвічуванням синтаксису,
- Є текстовий режим, який заснований на бібліотеках *ncurses* та *S-Lang*. Це дає можливість програмі працювати як в консолі, так і в різних емуляторах терміналу і через віддалені з'єднання, наприклад, за протоколом *SSH*,
- Підтримується багатомовний інтерфейс,
- Підтримується кодування *UTF-8*.

Для запуску *Midnight Commander (MC)* в термінальному клієнті ***PuTTY***, достатньо набрати команду *mc* в командному рядку. Приклад вигляду вікна *Midnight Commander* одразу після відкриття наведено на рис. 3.1.

Для виходу із цього файлового менеджера потрібно натиснути ***F10***.

Екран *MC* ділиться на чотири частини (див. рис. 3.1). Майже весь екран зайнятий двома панелями. За замовчуванням, другий рядок знизу є командним, а в самому нижньому рядку відображається призначення функціональних клавіш. Самий верхній рядок – рядок меню. Він може бути невидимим, але при натисканні клавіші ***F9*** або при натисканні миші цей рядок відображається у верхній частині екрану.

MC дозволяє одночасно спостерігати вміст двох каталогів за допомогою двох панелей. Одна з панелей є поточною (в цій панелі знаходиться виділений рядок). Практично всі команди виконують дії над поточною панеллю. Хоча деякі файлові операції, такі як *Rename* або *Copy*, за замовчуванням використовують каталог неактивної панелі в якості каталогу призначення (при виконанні подібних команд завжди з'являється запит на підтвердження).

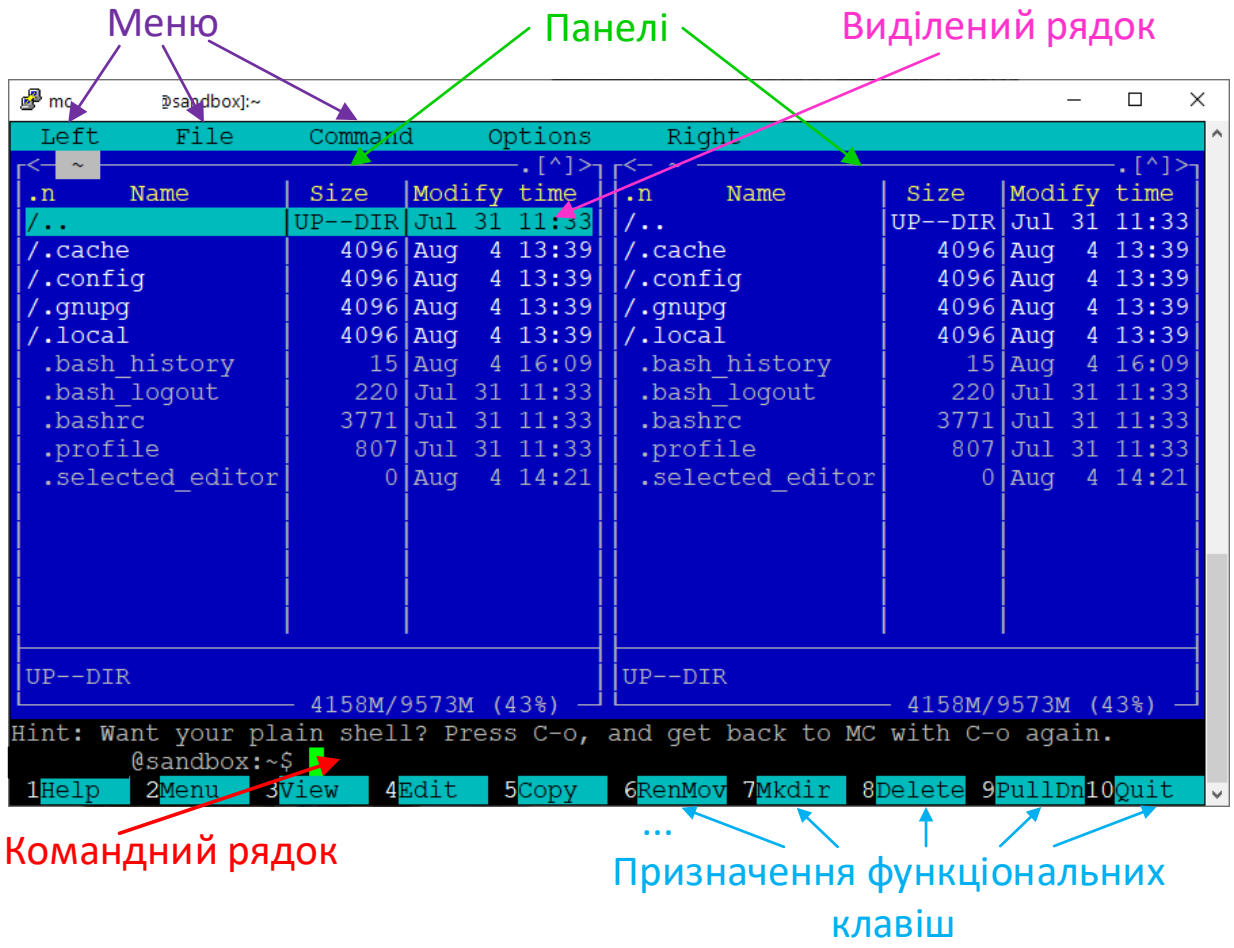


Рис. 3.1. Приклад вигляду вікна *Midnight Commander* одразу після відкриття

Можна виконувати системні команди прямо з *MC* просто шляхом їх набору на клавіатурі. Все, що набирається на клавіатурі, з'являється в командному рядку, а при натисканні клавіші *Enter*, *MC* виконає команду, яка була набрана в командному рядку.

MC поставляється з підтримкою миші. Миша підключається, при запуску *MC* з *xterm(1)* терміналу або з *Linux* консолі при наявності *gpm mouse* сервера.

При натисканні *лівою кнопкою миші* на файлі, цей файл стає поточним. При натисканні *правою кнопкою*, файл позначається (або позначка знімається, в залежності від попереднього стану).

При *подвійному натисканні на файлі*, буде зроблено спробу його запуску, якщо це файл програми. Якщо розширення файлу пов'язано з якою-небудь програмою, то ця програма буде запущена.

Якщо *MC* налаштований з підтримкою миші, то ви можете вирізати і склеювати фрагменти тексту, використовуючи клавішу *Shift*.

В таблицях 3.1 - 3.8 наведені гарячі клавіші (та їх комбінації), які дозволяють виконувати маніпуляції із файлами.

Таблиця 3.1. Робота з файлами в MC

Клавіши	Дія
F3	Переглянути файл
F4	Редагувати файл
Shift + F4	Створити новий файл
+	Викликати вікно для виділення групи
\	Зняти позначку з групи

Таблиця 3.2. Робота з командним рядком в MC

Клавіші	Дія
Alt + Enter	Вставити файл або каталог, на якому встановлений курсор, в командний рядок
Alt + a	Вставити в командний рядок повний шлях в активній панелі
Alt + p	Повернути попередню виконану команду в командний рядок
Ctrl + x, t	Вставка всіх виділених об’єктів в командний рядок
Ctrl + x, Ctrl + p	Вставити в командний рядок повний шлях, встановлений на неактивній панелі
Alt + t	Зміна режиму відображення панелі
Alt +.	Приховати або показати приховані файли і каталоги, які починаються на точку
Alt + a	Вставити в командний рядок повний шлях, встановлений на активній панелі
Alt + c	Зміна поточного каталогу
Alt + h	Перегляд останніх виконаних команд
Alt + i	Зміна каталогу на неактивній панелі на каталог, в якому знаходиться користувач
Alt + o	Неактивна панель переходить в каталог, на який вказує активна панель
Ctrl + x, q	Перегляд вмісту файлу на другий панелі
Ctrl + x, i	Перегляд інформації про поточний об’єкті на другий панелі

Клавіші	Дія
Ctrl + o	Приховати або показати панелі
Ctrl + r	Передивитися вміст каталогу
Ctrl + Space	Показати розмір поточного каталогу

Таблиця 3.3. Робота з пошуком

Клавіші	Дія
Alt + Shift +?	Розширений пошук
Ctrl + s	Пошук файлу або каталогу

Таблиця 3.4. Робота з правками

Клавіші	Дія
Ctrl + x, o	Перегляд або зміна власника, групи
Ctrl + x, c	Перегляд або зміна прав доступу

Таблиця 3.5. Робота з посиланнями

Клавіші	Дія
Ctrl + x, l	Створити жорстке посилання
Ctrl + x, s	Створити символьне посилання *symlink*
Ctrl + x, Ctrl + s	Редагувати символічне посилання

У *Midnight Commander* є вбудований текстовий редактор *mcedit* (виклик за допомогою клавіші ***F4***). Серед можливостей *mcedit* є підсвітка синтаксису для різних мов програмування, здатність працювати у режимах *ASCII* та *HEX*.

Для використання цього редактора за замовчуванням, потрібно виконати певні налаштування: як показано на рис. 3.2 та рис. 3.3. Спочатку в меню обрати *Options: Configuration…* (рис. 3.2).

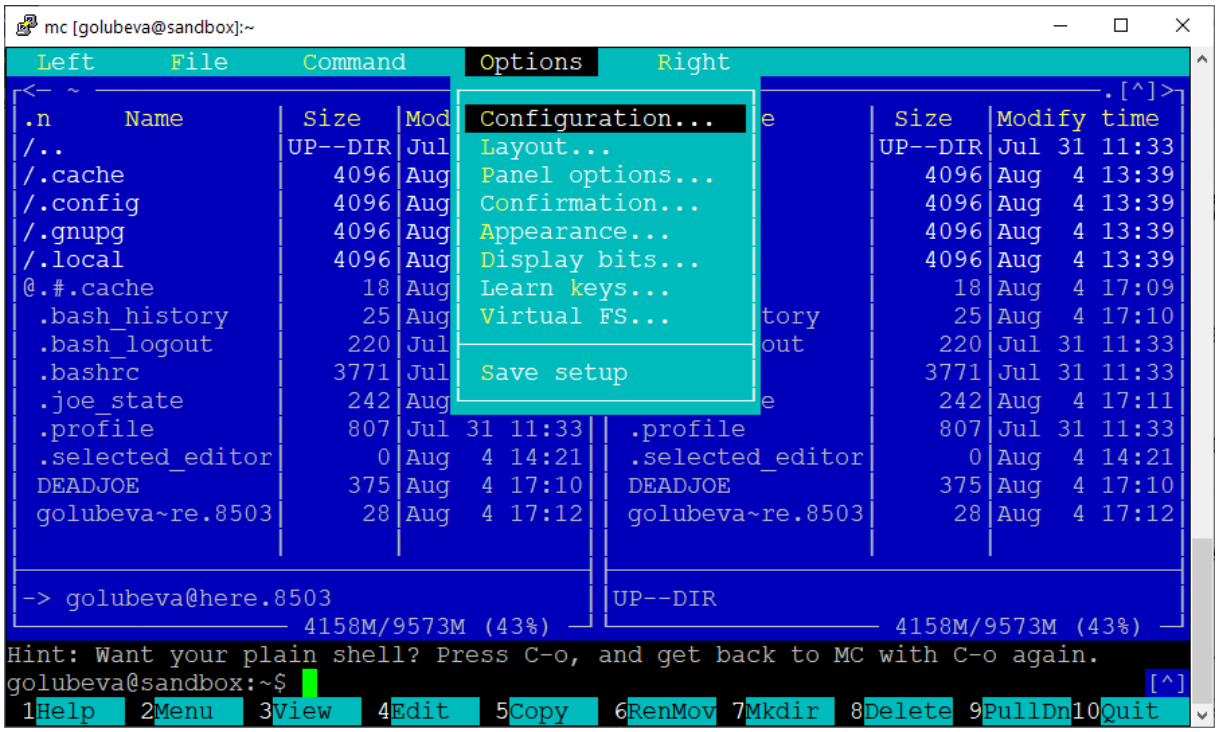


Рис. 3.2. Налаштування використання внутрішнього редактора *mcedit*

А далі в меню що випало поставити галочку навпроти напису *Use internal edit*, як це показано на рис. 3.3 нижче:

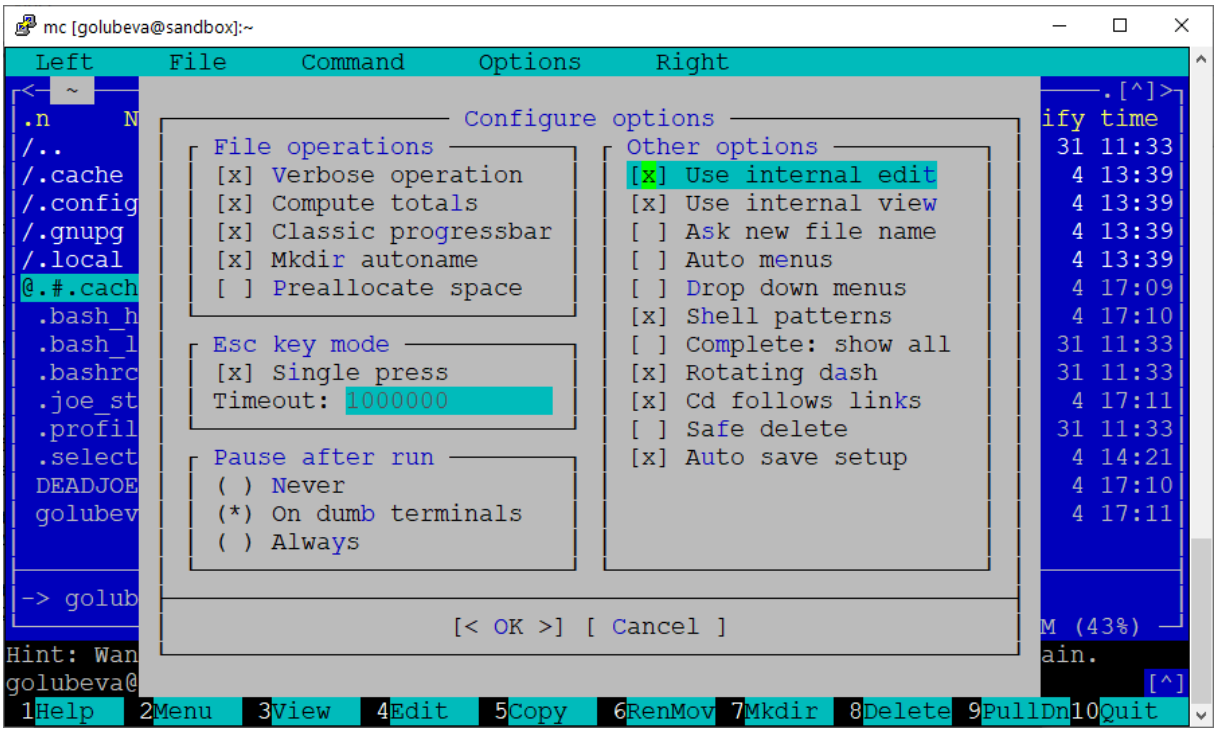


Рис. 3.3. Налаштування використання внутрішнього редактора *mcedit.* (продовження)

Таблиця 3.6. Зміна виду редактора в *mcedit*

Клавіші	Дія
Alt + *n*	Показати або приховати нумерацію рядків
Ctrl + *s*	Включити або вимкнути підсвічування синтаксису
Alt + *Shift* + -	Включити або вимкнути відображення табуляцій та пробілів
Alt + *Shift* ++	Включити або вимкнути автовирівнювання за допомогою повернення каретки

Таблиця 3.7. Робота з текстом в *mcedit*

Клавіши	Дія
F3	Почати виділення тексту, повторне натискання *F3* закінчить виділення
Shift + F3	Почати виділення блоку тексту, повторне натискання *F3* закінчить виділення
F5	Копіювати виділений текст
F6	Перемістити виділений текст
F8	Видалити виділений текст
Ctrl + k	Видалити частину рядка до кінця рядка
Ctrl + y	Видалити рядок
Ctrl + x	Перейти в кінець слова
Ctrl + z	Перейти на початок попереднього слова
Alt + l	Перехід на рядок по номеру рядка
Alt + u	Виконати команду і вставити друк результату команди в позицію під курсором
Alt + t	Сортування рядків виділеного тексту
Tab	Зсунути праворуч виділений текст
Ctrl + u	Скасувати дії

Таблиця 3.8. Робота з файлами в *mcedit*

Клавіші	Дія
Ctrl + n	Створити новий файл
Ctrl + f	Копіювати виділений текст в файл
Shift + F5	Вставка тексту з файлу

Більш детальну інформацію про *Midnight Commander* можна отримати на сайті проекту

https://midnight-commander.org/

у розділі документації. Зазвичай програма постачається із вбудованою системою довідки. Щоб отримати контекстну допомогу щодо поточного вікна, вмісту меню чи команди, скористайтесь гарячою клавішею *F1*. Детальна інформація про програму постачається у вигляді man-сторінок, які можна переглянути командою

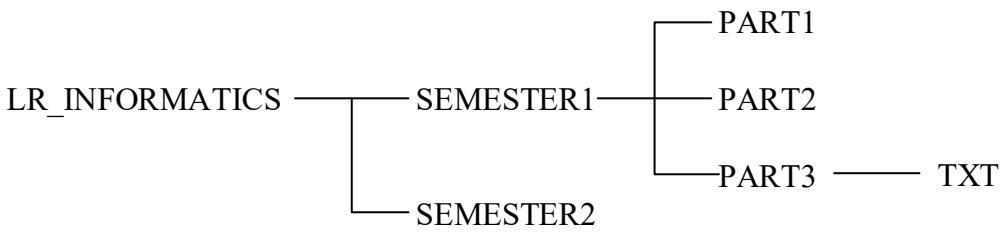
```
man mc
```

3.2. Завдання для виконання

В цій лабораторній роботі, для виконання завдань, потрібно користуватися файловим менеджером *Midnight Commander.*

1. Зайдіть на сервер та запустіть файловий менеджер *Midnight Commander.*
2. Продемонструйте уміння користуватися довідкою в файловому менеджері *Midnight Commander*
3. Перейти в будь-який підкаталог домашнього каталогу.

4. Створіть у домашньому каталозі структуру каталогів як показано на малюнку:



5. До каталогу *PART2* скопіюйте будь-який файл із збереженням імені оригінального файлу.
6. Зробити копії цього ж файлу в каталогу TXT , змінивши розширення файлу на *ttt* при копіюванні.
7. Перейменуйте останній файл таким чином, щоб його ім’я стало *aaa.bbb*.
8. Видаліть рекурсивно каталог *SEMESTER1*.

3.3. Контрольні питання

1. Які файлові менеджери з текстовим інтерфейсом (TUI) в ***UNIX***-подібних системах вам відомі, окрім Midnight Commander? Назвіть хоча б два. Які переваги та недоліки використання файлових менеджерів з текстовим інтерфейсом порівняно з графічними файловими менеджерами?

2. Які основні можливості файлового менеджера Midnight Commander ви можете перелічити? Опишіть його інтерфейс та основні функціональні блоки. Які операції з файлами та каталогами можна виконувати за його допомогою?

3. Чи надає файловий менеджер Midnight Commander вбудовану функціональність для копіювання файлів з одночасною зміною їх імені або розширення? Якщо так, опишіть, як це зробити.

4. Чи підтримує файловий менеджер Midnight Commander можливість рекурсивного видалення підкаталогів разом з усім їхнім вмістом? Якщо так, як активувати цю функцію? Які запобіжні заходи слід вживати при рекурсивному видаленні?

5. Як запустити файловий менеджер Midnight Commander з терміналу? Які існують ключі командного рядка для запуску MC з певними параметрами?

6. Опишіть призначення двох основних панелей Midnight Commander. Як перемикатися між панелями? Як змінити поточний каталог на кожній з панелей?

7. Які клавіші або комбінації клавіш використовуються в Midnight Commander для виконання основних операцій, таких як вибір файлів/каталогів, копіювання, переміщення, видалення, створення каталогів, редагування файлів, перегляд файлів?

8. Чи підтримує Midnight Commander роботу з архівами (наприклад, tar, zip, gzip)? Якщо так, які операції з архівами можна виконувати безпосередньо з його інтерфейсу?

9. Яким чином можна змінити права доступу (permissions) до файлів та каталогів за допомогою Midnight Commander? Опишіть інтерфейс керування правами.

10. Чи дозволяє Midnight Commander виконувати команди терміналу безпосередньо зі свого інтерфейсу? Якщо так, як це зробити?

11. Які можливості навігації надає Midnight Commander (наприклад, швидкий перехід, історія каталогів)?

12. Чи можна налаштувати зовнішній вигляд та поведінку Midnight Commander? Які елементи інтерфейсу можна налаштувати?

13. Чи підтримує Midnight Commander роботу з віддаленими файловими системами (наприклад, через FTP, SFTP)? Якщо так, як підключитися до віддаленого сервера?

14. Які вбудовані інструменти перегляду файлів різних форматів є в Midnight Commander? Які формати файлів він може відображати?

15. Чи існує в Midnight Commander функція пошуку файлів та каталогів за різними критеріями? Опишіть її можливості.

4. Лабораторна робота №4
Засоби командного рядка для набору та збірки програм мовою С у UNIX-подібних системах

Мета роботи: вивчення основних прийомів та отримання практичних навичок виконання набору та збірки програми, написаної мовою *C* на ***UNIX***-подібних системах.

4.1. Теоретичні відомості

4.1.1. Загальні поняття

Програма – це детальний та закінчений опис алгоритму засобами мови програмування. Виконавцем програми є комп'ютер. Для виконання комп'ютером програма повинна бути подана у машинному коді – послідовності чисел, які зрозумілі процесору, як відповідні інструкції та/або дані. Написати програму в машинних кодах вручну досить складно. Тому сьогодні практично всі програми створюються за допомогою мов програмування вищого рівня, які за своїми синтаксисом та семантикою наближені до природної людської мови. Це знижує трудомісткість програмування. Однак, текст програми, записаний за допомогою мови програмування, повинен бути перетворений в машинний код. Ця операція виконується автоматично за допомогою спеціальної службової програми, яка називається транслятором.

Транслятори можна поділити на інтерпретатори та компілятори.

Інтерпретатори перетворюють в машинний код та виконують почергово оператори (команди) програми безпосередньо після запуску програми. Якщо команда повторюється, то інтерпретатор розглядає її так само, як і ту що зустрів вперше. Перевага інтерпретаторів – їх компактність, можливість зупинити в будь-який момент виконання програми, виконати різні перетворення даних та продовжити роботу програми. Якщо деякі оператори програми не знадобилися

під час виконання, інтерпретатор не звертатиметься до них і не перетворюватиме їх.

Компілятор перетворює увесь вихідний текст програми повністю у машинний код. Ця операція виконується заздалегідь, і після отримання виконуваного модуля компілятор більше не потрібен. Виконуваний модуль також можна переміщувати на інші машини з подібною архітектурою. Тому перевага компіляторів – швидкодія та автономність отриманих програм.

Мова програмування С – відмінний вибір для тих, хто тільки починає знайомитися із програмуванням, оскільки *С* – це відносно проста мова, але вона є потужною та широко використовується. Також мова *С* є основою для багатьох інших мов програмування, і, отже, досвід, отриманий з *С*, можна застосувати і до інших мов. Крім того, досвід роботи з *С* корисний для глибокого розуміння ***Linux*** та ***UNIX*** -подібних операційних систем, оскільки вони значною мірою написані на мові *С*.

Linux є досить сприятливим середовищем для написання програм. Це пояснюється тим, що, на відміну від деяких популярних комерційних операційних систем, немає необхідності купувати будь-яке дороге програмне забезпечення для програмування, а в багатьох випадках відповідне програмне забезпечення вже встановлено на комп'ютері. Більше того, більшість основних дистрибутивів *Linux* включають засоби програмування; такі інструменти можна дуже легко встановити під час встановлення системи або окремо пізніше.

4.1.2. Створення, збірка та налагодження програм у ***UNIX***-подібних системах

Найчастіше для створення програм на *С* потрібні три речі:

1. текстовий редактор,
2. компілятор,

3. стандартна бібліотека C.

Текстовий редактор – це програма, в якій можна створити файл, записати та зберегти текст програми (*source code*) на *C* та виконати редагування тексту програми за потреби. Програми на *C* можна писати за допомогою будь -якого із багатьох текстових редакторів, доступних для ***Linux***, таких як *nedit*, *vi*, *gedit*, *gedit, kedit* або *emacs*.

Принаймні один текстовий редактор є вбудованим у кожну ***UNIX*** - подібну операційну систему, і більшість таких систем містять кілька текстових редакторів. Щоб побачити, чи існує певний текстовий редактор у системі, все, що потрібно, це ввести його ім'я у командний, а потім натиснути клавішу ***ENTER***. Якщо він існує, редактор з'явиться у існуючому вікні, якщо це редактор командного рядка, наприклад *vi*. Якщо це редактор із графічним інтерфейсом користувача, наприклад *nedit*, то він відкриється у новому вікні.

Наприклад, якщо потрібно побачити, чи є *vi* у системі (редактор *vi* майже завжди є у системі), потрібно лише ввести таку команду та натиснути клавішу ***ENTER***:

```
vi
```

Компілятор – це спеціалізована програма, яка перетворює вихідний код в машинний код (його також називають об'єктним кодом), який зрозумілий центральному процесору. Дуже добрий компілятор *C* включений до *GNU Compiler Collection (GCC)* – одного з компонентів більшості сучасних дистрибутивів ***Linux***. *GNU* – це поточний проект *Free Software Foundation (FSF)* для створення повного, сумісного з ***UNIX***, високопродуктивного середовища, яке вільно поширюється.

Аби переконатися, що *GCC* вже встановлено у системі, слід ввести таку команду та натиснути клавішу ***ENTER***:

```
gcc
```

повідомлення про помилку виду *«gcc: no input files»*, вказує на те, що *GCC* встановлено та готовий до використання. Якщо повідомлення про помилку виглядає наступним чином: *«command not found»*, то це вказує на те, що *GCC* не було встановлено. У разі відсутності *GCC* можна встановити так само, як і інше програмне забезпечення.

glibc – це реалізовані проектом *GNU* стандартні бібліотеки *C*. Як і у випадку з *GCC*, це також один із найважливіших компонентів більшості сучасних дистрибутивів ***Linux***, які використовують його як свою офіційну стандартну бібліотеку *C*.

Бібліотека – це сукупність підпрограм, які може використати будь-який програміст, щоб зменшити кількість складного коду який часто повторюється у певних програмах. У стандартній бібліотеці зазвичай містяться модулі для роботи з файловою системою, дисплеєм та клавіатурою, набір підпрограм для обчислення математичних функцій і таке інше. Кожній ***UNIX***-подібній операційній системі потрібна бібліотека *C*.

Командою *locate* можна скористатися для підтвердження того, що бібліотеки *glibc* вже встановлено в системі. Перевірка наявності бібілотек можна виконати наступним чином:

```
locate glibc
```

Якщо файл із такою назвою присутній, то ця команда виведе на екран шлях до нього.

Інформацію про будь-які команді чи програми, включаючи *GCC*, *vi* та *nedit*, можна отримати за допомогою команди *man*. Для виходу із режиму перегляду інформації про програму, потрібно натиснути клавішу *q*.

4.1.3. Компіляція С-програми за допомогою GCC

Після того, як текст програми було написано та перевірено на наявність очевидних помилок, вона готова до компіляції. Компіляція складається з двох основних кроків: трансляції та компоновки (лінковки); *gcc* у своєму складі містить як транслятор, так і компоновщик.

Програму *program.c* можна зібрати за допомогою *GCC* наступним чином:

```
gcc  -o  program  program.c
```

Ключ *-o* інформує компілятора про бажане ім'я для файла що виконується, який буде створено після компіляції. Якщо параметр *-o* опущено, компілятор використає ім'я ***a.out*** за замовчуванням. Зверніть увагу, на відміну від ОС *DOS, Windows*, де ознакою виконуваної програми є розширення імені файла exe або com, у ***UNIX***-подібних системах виконуваний файл повинен мати атрибут «x» у режимі доступу. Його можна встановити командою **chmod**. Якесь специфічне розширення імені файла не потребується і дуже часто взагалі відсутнє.

Час компіляції залежить від потужності комп'ютера, який виконує компіляцію, об'єму та складності програми, яка компілюється.

Після компіляції краще за все перевірити вміст каталогу, для того щоб переконатися, чи було створено виконуваний програмний файл. Для цього можна скористатися або командою *ls*, або файловим менеджером *Midnight Commander*.

4.1.4. Помилки, які можуть виникнути під час компіляції

При написанні програм можуть виникати помилки різного типу – синтаксичні, логічні та інші. Зазвичай під час компіляції виявляються

синтаксичні помилки. Сучасні компілятори зазвичай надають досить докладні повідомлення про помилки, які полегшують знаходження та виправлення помилок у коді програми.

Компілятор *GCC*, при виявленні помилки при компіляції, видає повідомлення `error:` , після якого йде детальний опис помилки. Приклад такого повідомлення наведено нижче:

```
.c:6:13: error: expected ';' before numeric constant
```

При появі помилок при компіляції, файл що виконується не буде створено.

Також при компіляції можуть з'явитися повідомлення-попередження: `warning:` із їх описом.

Деякі конструкції можуть бути цілком припустимі за одних обставин та небезпечними за інших. Компілятор може видавати попередження при використанні такого коду, для цього скористайтеся ключем *-Wall*.

Наявність попереджень може не містити загрози для нормальної роботи програми. Однак, програмісти намагаються писати код, який збирається без помилок та попереджень. Хоч це і не гарантує відсутності логічних помилок у програмі, загалом якість коду стає вищою. Іноді, щоб примусити програміста переглядати потенційно небезпечний код, використовується ключ *-Werror*, завдяки якому попередження компілятора інтерпретуються як помилки, і, таким чином, запобігають успішній збірці програми.

4.1.5. Виконання скомпільованої програми

Після того, як компілятор створить файл, що виконується, він готовий до роботи. Для запуску файлу програми на виконання треба набрати його повний

шлях у запрошенні, вказати необхідні ключі та аргументи і натиснути ***ENTER***. Така процедура є громіздкою, тому зазвичай команди запускаються на виконання за їхніми іменами, а пошук файлів що виконуються відбувається у наперед заданих каталогах, перелічених у змінній оточення *$PATH*. Наприклад, на нашому сервері ***sandbox*** за замовчуванням встановлено шлях пошуку:

sandbox:~$ echo $PATH

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

Зверніть увагу, що до цього переліку за звичайних умов не входить домашній каталог користувача. Тому спроба запустити файл на виконання призведе до помилки пошуку. Щоб не вводити повний шлях до щойно створеної програми від кореневого каталога, можна скористатися спеціальними псевдонімами. Так, символ «.» (точка) означає «поточний каталог». Тому для запуску програми з поточного каталогу можна скористатися командою

```
./program
```

Результати виконання можна побачити у тому ж вікні починаючи із наступного рядка. Після закінчення виконання програм знову з’явиться запрошення командного рідка.

4.1.6. Приклад створення та налагодження програми у командному рядку

Скористаємося редактором *vi* або *nedit* для редагування файла *demo4.c*:

```
vi demo4.c
```

або

```
nedit demo4.c
```

Наберемо текст програми, як показано на малюнку:

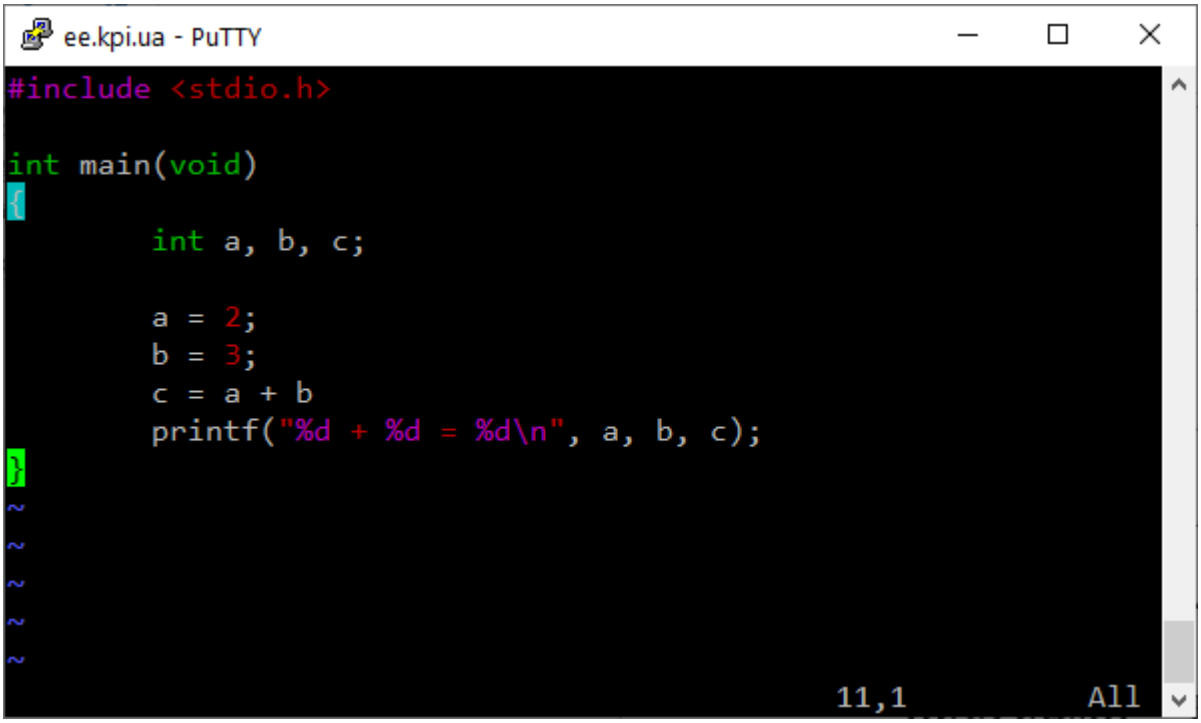


Рис. 4.1. Вікно редактора з текстом програми

Зверніть увагу, за розширенням *.c* редактор розпізнав текст програми мовою *C* і виконує підсвічування синтаксису: вбудовані типи даних (int, void), числа (2, 3) і т.п. Також, коли курсор стоїть на правій фігурній дужці в останньому рядку, то підсвічується відповідна їй ліва фігурна дужка у четвертому рядку.

Збережемо файл з текстом, закриємо редактор та перейдемо до збірки програми командою:

```
gcc -Wall -ggdb -o demo4 demo4.c
```

-Wall – ключ, який дозволяє виводити на екран усі попередження, які виникають при компіляції програми,

-ggdb – ключ, який створює службову інформацію для програми налагодження *GDB*.

У вікні терміналу, починаючи із наступного рядочку, ми побачимо ось таку інформацію:

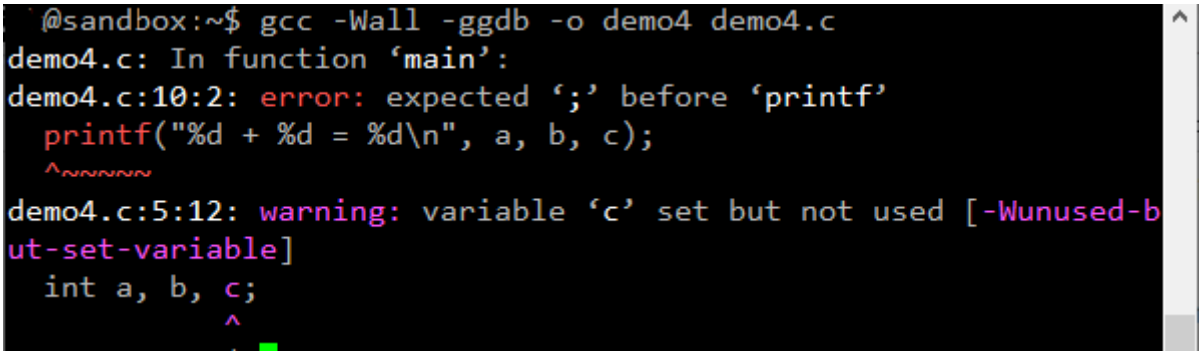


Рис. 4.2. Вікно терміналу з виводом роботи компілятора

Зрозуміло що збірка завершилася невдало: у десятому рядку виявлено помилку, а у п’ятому – попередження. Пояснення до помилки повідомляє, що у десятому рядку, другому стовпчику очікується символ «;» (крапка з комою) перед словом «*printf*». Якщо придивитися до тексту програми на попередньому скрншоті, то можна виявити, що насправді крапка з комою була втрачена у дев’ятому рядку, а в її пошуках компілятор дійшов до десятого, де і зустрів слово «*printf*». За допомогою текстового редактора виправимо помилку і повторимо спробу. Цього разу компілятор не вивів ніяких повідомлень. За допомогою команди *ls* переконаємося, що у поточному каталозі справді створено файл *demo4*. Якщо цей файл дійсно було створено, то ми можемо запустити його на виконання командою

```
./demo4
```

В результаті виконання цього файлу що виконується, у терміналі виводиться

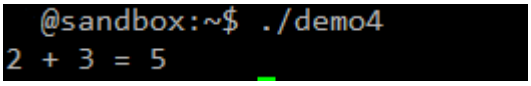


Рис. 4.3. Вікно терміналу з результатом роботи програми

У такій простій програмі логічні помилки малоймовірні, проте складніші програми, у переважній більшості, з першого разу реалізуються невдало і для пошуку помилок вдаються до налагодження програми. Цей процес передбачає зупинку виконання програми на певних рядках, перевірку контрольних значень або встановлення змінних, покрокове виконання і т. п. Ці сервіси у тому чи іншому вигляді надаються усіма системами програмування. Розглянемо приклад налагодження простої програми у текстовій консолі за допомогою програми налагодження *gdb*. Запустимо програму налагодження *gdb* для файлу що виконується *demo4 ,* наступною командою:

```
gdb   ./demo4
```

У консолі буде виведено інформацію про програму, підказка щодо пошуку документації на командний рядок налагодження (останній рядок на скріншоті нижче):

```
GNU gdb (Ubuntu 8.1.1-0ubuntu1) 8.1.1
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./demo4...done.
(gdb)
```

Рис. 4.4. Консоль з інтерфейсом gdb

Виконаємо пошук точки входу в програму, для програми мовою *C* це функція *main()*. Для цього у командному рядку наберемо *l* (скорочення від *list*), в консоль буде виведено наступні 10 рядків тексту програми:

```
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./demo4...done.
(gdb) l
1       #include <stdio.h>
2
3       int main(void)
4       {
5               int a, b, c;
6
7               a = 2;
8               b = 3;
9               c = a + b;
10              printf("%d + %d = %d\n", a, b, c);
(gdb)
```

Рис. 4.5. Результат роботи команди *list*

Щоб почати покрокове налагодження програми, зупинимо її виконання одразу на точці входу за допомогою точки зупинки (англійською – *breakpoint*). Для цього у командному рядку наберемо

```
b main
```

Символ ‘*b*’ – це скорочення від команди ‘*break*’, а параметр вказує, що потрібно зупинитися на функції *main*. У консолі з’явиться таке повідомлення:

```
Reading symbols from ./demo4...done.
(gdb) l
1       #include <stdio.h>
2
3       int main(void)
4       {
5               int a, b, c;
6
7               a = 2;
8               b = 3;
9               c = a + b;
10              printf("%d + %d = %d\n", a, b, c);
(gdb) b main
Breakpoint 1 at 0x652: file demo4.c, line 7.
(gdb)
```

Рис. 4.6. Виконання точки зупинки

Зверніть увагу, що реально точка зупинки встановлена аж у 7 рядку, позаяк попередні рядки не містять коду, який можна налагодити. Запустимо програму на виконання командою *r* (від *run*), у консолі буде виведено:

```
Reading symbols from ./demo4...done.
(gdb) l
1       #include <stdio.h>
2
3       int main(void)
4       {
5               int a, b, c;
6
7               a = 2;
8               b = 3;
9               c = a + b;
10              printf("%d + %d = %d\n", a, b, c);
(gdb) b main
Breakpoint 1 at 0x652: file demo4.c, line 7.
(gdb) r
Starting program: /home/  /demo4

Breakpoint 1, main () at demo4.c:7
7               a = 2;
(gdb)
```

Рис. 4.7. Результат виконання програми у консолі

Як бачимо, виконання програми зупинилося у 7 рядку. Переглянемо значення змінної *a* командою «*print a*»:

```
(gdb) l
1       #include <stdio.h>
2
3       int main(void)
4       {
5               int a, b, c;
6
7               a = 2;
8               b = 3;
9               c = a + b;
10              printf("%d + %d = %d\n", a, b, c);
(gdb) b main
Breakpoint 1 at 0x652: file demo4.c, line 7.
(gdb) r
Starting program: /home/  /demo4

Breakpoint 1, main () at demo4.c:7
7               a = 2;
(gdb) print a
$1 = 32767
```

Рис. 4.8. Результат перегляду значення змінної

Змінна *a* містить «дивне» значення, що можна пояснити тим, що оператор присвоювання у сьомому рядку ще не було виконано. Виконаємо 7 рядок командою «*n*» (від *next*, мається на увазі наступний рядок) і знову надрукуємо значення змінної *a*:

```
Breakpoint 1, main () at demo4.c:7
7               a = 2;
(gdb) print a
$1 = 32767
(gdb) n
8               b = 3;
(gdb) p a
$2 = 2
(gdb)
```

Рис. 4.9. Результат перегляду значення змінної після її ініціалізації

Цього разу програма зупинилася на 8 рядку, де буде оператор «*b = 3;*», а значення змінної *a* вже дорівнює 2. Для зупинки програми на подію оновлення значення змінної скористаємось командами

```
watch b
watch c
```

```
(gdb) p a
$2 = 2
(gdb) watch b
Hardware watchpoint 2: b
(gdb) watch c
Hardware watchpoint 3: c
(gdb)
```

Рис. 4.10. Встановлення зупинки програми на подію оновлення значення змінної

Продовжимо виконання програми командою «*c*» (*continue*):

```
(gdb) c
Continuing.

Hardware watchpoint 2: b

Old value = 0
New value = 3
main () at demo4.c:9
9               c = a + b;
(gdb)
```

Рис. 4.11. Вигляд консолі на подію оновлення значення змінної

Система друкує колишнє та нове значення змінної, яка переглядається і зупиняє виконання програми. Спробуємо змінити значення змінної *a* командою

```
(gdb) set variable a = 5
(gdb)
```

Рис. 4.12. Примусове встановлення значення змінної

Переглянемо значення усіх локальних змінних:

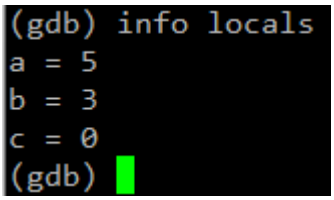


Рис. 4.13. Результат перегляду значення усіх локальних змінних

Продовжимо виконання програми командою «*c*» (*continue*):

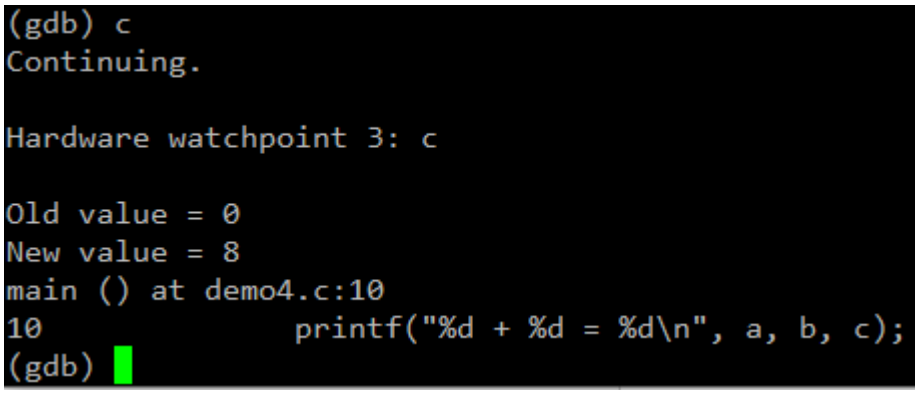


Рис. 4.14. Спрацювання точки перегляду оновлення значення змінної

Відбулося оновлення значення змінної *c*, яка була поставлена «під нагляд» командою *watch*, тому виконання програми зупинилося і в консоль виведено інформацію про оновлення значення змінної *c*. Продовжимо виконання програми командою «*c*» (*continue*), у консолі буде надруковано:

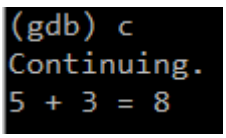


Рис. 4.15. Вивід результату обчислень до консолі

Решта повідомлень перевершують наші поточні знання, залишимо їх без розгляду. Закінчимо сеанс налагодження командою «*q*» (*quit*). У будь-який

момент можна скористатися командою *help*, щоб отримати підказку про команди та їх формати.

4.1.7. Засоби автоматизації збірки make

У процесі розробки цієї простої програми нам довелося скористатися кількома іншими програмами та викликати їх з різними ключами, які, звісно, треба пам'ятати. Якщо програма складається більше, ніж з одного файлу вихідного коду, завдання стає ще складнішим. Тому для автоматизації збірки програм розроблено програмне забезпечення *make* із командними файлами. Детальний виклад синтаксису командних файлів та можливостей *make* перевершує наші поточні знання, тому обмежимося простим прикладом. Для щойно розглянутого прикладу можна створити такий командний файл з іменем *Makefile* (з великої літери *M*):

```
1  # Default target
2  .PHONY: all
3  all: demo4
4
5  # Rule to build executable
6  demo4: demo4.c
7          gcc -Wall -ggdb -o demo4 demo4.c
8
9  .PHONY: clean
10 clean:
11         rm -f demo4
12
13 .PHONY: debug
14 debug: demo4
15         gdb demo4
16
```

Рис. 4.16. Вікно редактора із вмістом *Makefile*

Команди у *Makefile* починаються із символа табуляції. Маючи такий командний файл збірку програми можна виконати командою

```
make або make all
```

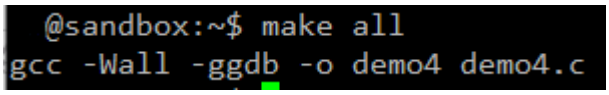


Рис. 4.17. Консоль із результатом виконання збірки програми

Якщо відредагувати файл вихідного коду *demo4.c*, то система *make* помітить це і виконає повторну компіляцію. Якщо вихідні файли не мінялися, буде надруковане відповідне повідомлення.

Для видалення згенерованих файлів можна скористатися командою очищення:

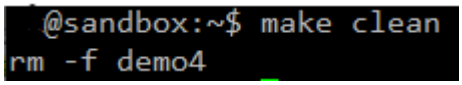


Рис. 4.18. Консоль з результатом очищення програми

Для запуску програми на налагодження можна скористатися командою:

```
make debug
```

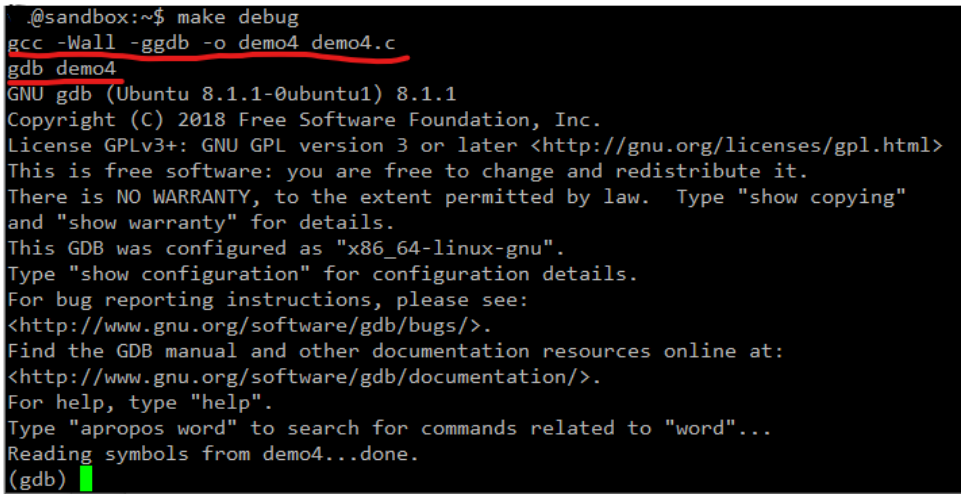


Рис. 4.19. Вигляд консолі після запуску make debug

Зверніть увагу, що оскільки для правила *debug* встановлено залежність від *demo4* (виконуваний файл програми), то спочатку було скомпільовано виконуваний файл, а лише потім запущено налагодження. Для цієї роботи охочі можуть скористатися наданим *Makefile* та переписати його на сервер до свого робочого каталога, як описано у наступному розділі.

4.1.8. Копіювання файлів між сервером і комп'ютером

SCP (Secure Copy Protocol) – команда в ***Linux*** для захищеного копіювання файлів або папок на віддалений комп'ютер (сервер) або з нього, використовуючи для цього протокол *SSH* (*Secure Shell*). *SCP* є складовою частиною пакету *OpenSSH*. Завдяки використанню *ssh*, *SCP* є не тільки зручним, але і безпечним способом копіювання файлів.

Команда копіювання локального *SourceFile* на віддалений хост:

```
scp SourceFile user@host:/directory/TargetFile
```

Команда копіювання *SourceFile* з віддаленого хоста:

```
scp user@host:/directory/SourceFile TargetFile
```

Наведені команди стануть у нагоді, якщо сервер та на локальна машина користувача працюють під керування UNIX-подібної операційної системи з відповідними налаштуваннями, однак це не спрацює, якщо, наприклад, машина користувача працює під управлінням Windows. У цьому випадку для копіювання файлів між сервером та комп'ютером також можна скористатися графічним клієнтом *WinSCP*, як це описано у вступній частині цього методичного посібника.

4.2. Завдання для виконання

Частина 1

В пунктах завдання 1-10 цієї лабораторної роботи ми НЕ використовуємо файловий менеджер *WinSCP*.

1. Зайти на сервер *sandbox.ee.kpi.ua*, скориставшись інформацією із попередніх лабораторних робіт.
2. Перейти в каталог *PROG_SEM1*, а якщо не має такого каталогу, то створити його.
3. Відкрити будь-який текстовий редактор та створити текстовий файл із ім'ям *lab4.c*, в якому набрати наступний код програми:

```c
#include <stdio.h>

int main()
{
    printf("Hello, world!\n");
    return 0;
}
```

4. Виконати компіляцію файлу *lab4.c* та створити файл що виконується із ім'ям *lab4_1*.
5. Переглянути вміст поточного каталогу (будь-яким зручним для вас способом) та переконатися, що файл *lab4_1* з'явився.
6. Запустити на виконання файл *lab4_1*.
7. Зайти в файл *lab4.c* в режимі редагування та змінити напис `return 0;` на `returnnn 0;`, додавши букви *n* в кінці слова *return*.
8. Виконати компіляцію файлу *lab4.c* та створити файл що виконується із ім'ям *lab4_2*.
9. Переглянути повідомлення про помилки при спробі скомпілювати цей файл.
10. Переконатися, що після виконання компіляції файл із ім'ям *lab4_2* НЕ з'явився в поточній директорії.

Частина 2

11. На персональному комп'ютері, на якому виконується лабораторна робота, створити файл із ім'ям *new_lab4.c*. В файл розмістити наступний текст програми:

```
#include <stdio.h>
#define PI 3.14

int main()
{
    foat r, p;
    r = 3;
    p = 2*PI*r;
    printf("r = %f, p = %f\n", r, p);
    return 0;
}
```

12. Запустити графічний клієнт для копіювання файлів *WinSCP* на персональному комп'ютері, на якому виконується лабораторна робота та зайти *sandbox.ee.kpi.ua* за допомогою *WinSCP*.
13. Скопіювати файл *new_lab4.c* із персонального комп'ютера, на якому виконується лабораторна робота, на сервер *sandbox.ee.kpi.ua,* в каталог із ім'ям *PROG_SEM1*.
14. Виконати компіляцію файлу *new_lab4.c* та створити файл що виконується із ім'ям *new_lab4_1*. Якщо буде виявлено помилки – виправити їх.
15. Переглянути вміст поточного каталогу (будь-яким зручним для вас способом) та переконатися, що файл *new_lab4_1* з'явився.
16. Запустити на виконання файл *new_lab4_1*. Переконатися що результати виконання програми коректно виводяться на екран.
17. Виконати програму покрокового налагодження програми за допомогою *gdb*:
    - ✓ Перекомпілювати файл *new_lab4.c* таким чином, щоб можна було скористатися програмою покрокового налагодження *gdb*.

- ✓ Запустити файл *new_lab4_1* в режимі налагодження.
- ✓ Встановити *breakpoint* на початку функції *main*.
- ✓ Запустити програму на виконання та подивитися поточне значення змінної *p*.
- ✓ Виконати покроково 4 наступні рядочки програми та знову подивитися поточне значення змінної *p*.
- ✓ Закінчити виконання програми *new_lab4_1*.

4.3. Контрольні питання

1. Які основні типи трансляторів вам відомі? Опишіть принцип їхньої роботи та наведіть приклади для кожного типу (наприклад, компілятор, інтерпретатор, асемблер). У чому полягають ключові відмінності між компіляцією та інтерпретацією?
2. Для чого конкретно використовується програма GCC (GNU Compiler Collection)? Які етапи проходить вихідний код програми на мові C перед тим, як стати виконуваним файлом за допомогою GCC?
3. Як називаються стандартні бібліотеки мови C, реалізовані в рамках проєкту GNU, які використовуються GCC? Яке їхнє основне призначення? Наведіть приклади функцій з цих бібліотек, які ви вже використовували.
4. Який тип помилок у програмі на мові C найчастіше виявляє компілятор на етапі компіляції? Наведіть приклади таких помилок. Які типи помилок компілятор зазвичай не може виявити?
5. Які програми або утиліти командного рядка ви знаєте для безпечного та небезпечного копіювання файлів між локальним комп'ютером та віддаленим сервером? Опишіть їхні основні переваги та недоліки.
6. Чи надає графічна програма WinSCP можливість зберігати налаштування підключення до різних серверів у вигляді окремих профілів? Якщо так, опишіть, як це зробити та які переваги це надає користувачеві.

7. Що таке Makefile і для чого він використовується в процесі розробки програмного забезпечення? Яка основна структура простого Makefile? Поясніть призначення цілей (targets) та правил (rules) у Makefile.
8. Яка команда використовується для запуску процесу збірки програми на основі Makefile? Що відбувається при виконанні цієї команди?
9. Що таке залежності (dependencies) у Makefile? Чому важливо їх правильно визначати?
10. Які переваги використання Makefile порівняно з ручною компіляцією програми за допомогою команд GCC?
11. Що таке об'єктний файл? Як він утворюється в процесі компіляції?
12. Що таке лінкер (linker) і для чого він використовується? Яку роль він відіграє у створенні виконуваного файлу?
13. Які існують рівні оптимізації при компіляції програми за допомогою GCC? Як вони впливають на розмір та швидкодію виконуваного файлу?
14. Що таке попередня обробка (preprocessing) у процесі компіляції C-програми? Які директиви препроцесора ви знаєте та для чого вони використовуються (наприклад, #include, #define)?
15. Як передати додаткові опції компілятору GCC під час збірки програми за допомогою Makefile? Наведіть приклади використання прапорів (flags) GCC.

Список рекомендованої літератури

1. Negus, C. (2020). Linux® Bible. In *Linux® Bible*. https://doi.org/10.1002/9781119578956.
2. Barrett, D. J. (2012). Linux Pocket Guide. In *Linux*.
3. Kirkbride, P. (2020). Basic Linux Terminal Tips and Tricks. In *Basic Linux Terminal Tips and Tricks*. https://doi.org/10.1007/978-1-4842-6035-7.
4. Robbins, A., Hannab, E., & Lamb, L. (2008). Learning the Vi and Vim Editors. In *O'Reilly*.
5. Stallman, R. (2004). Using te GNU Compiler Collection. *Free Software Foundation*.
6. Операційні системи. Комп'ютерний практикум [Електронний ресурс] : навчальний посібник для студентів спеціальності 113 «Прикладна математика», 125 «Кібербезпека» / КПІ ім. Ігоря Сікорського ; уклад.: М. В. Грайворонський, В. В. Демчинський. – Електронні текстові дані (1 файл: 525 Кбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 100 с. https://ela.kpi.ua/handle/123456789/43545