МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни
«Технології розподілених систем та паралельних обчислень»
для студентів 4-го курсу денної форми навчання
спеціальності 122 – «Комп’ютерні науки»
(частина перша)

Одеса 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни
«Технології розподілених систем та паралельних обчислень»
для студентів 4-го курсу денної форми навчання
спеціальності 122 – «Комп'ютерні науки»
(частина перша)

ЗАТВЕРДЖЕНО
На засіданні методичної комісії факультету комп'ютерних наук, управління та адміністрування

Протокол № ___ від "____"____ 2019 р.
Голова комісії

_____ *Л.Б.Коваленко*

Одеса 2019

Методичні вказівки до виконання лабораторних робіт з дисципліни «Технології розподілених систем та паралельних обчислень» для студентів 4-го курсу денної форми навчання, які навчаються за спеціальністю 122 – «Комп'ютерні науки». Частина перша містить загальні питання, опис необхідного програмного забезпечення і лабораторну роботу № 1. Одеса: ОДЕКУ, 2019.– 60 с., укр. мова.

Укладач: ст.викл. Рольщиков В.Б.

ЗМІСТ

Вступ 4
1 Загальна частина 5
1.1 Оцінювання знань з теоретичного матеріалу 5
1.2 Оцінювання виконання практичної частини роботи 6
1.3 Техніка безпеки при виконанні робіт 7
1.4 Програмне забезпечення, необхідне для роботи 8
2 Лабораторна робота №1 Налагоджування пакету MPJ Express. Компіляция і запуск найпростіших програм 9
2.1 Теоретична частина 9
2.1.1 Характеристика парадигми Message Passing Interface 9
2.1.2 Відомості про пакет класів MPJ Express. Компіляція і запуск програм 17
2.2 Завдання до лабораторної роботи 20
2.3 Порядок виконання роботи 20
2.4 Зміст звіту про лабораторну роботу 24
2.5 Завдання для підготовки до лабораторної роботи 25
2.6 Контрольні запитання до лабораторної роботи 25
Перелік джерел посилання 28
Додаток А Форма першої сторінки протоколу роботи 29
Додаток Б Опис програми термінального SSH-клієнта Xshell 30
Додаток В Використання програми Midnight Commander 42

ВСТУП

Методичні вказівки призначені для виконання лабораторних робіт студентами четвертого курсу всіх спеціалізацій факультету комп'ютерних наук, які вивчають дисципліну «Технології розподілених систем та паралельних обчислень».

Виконання лабораторних робіт з цієї дисципліни виконується за допомогою мови програмування Java з використанням бібліотеки класів MPJ Express. Ця бібліотека реалізує підтримку технології паралельного програмування в стандарті MPI. Технологія програмування стандарту MPI є однією з найбільш поширених технологій створення паралельних застосувань як для систем з розподіленою пам'яттю, але так само вона доволі часто застосовується і в системах з поділюваною пам'яттю.

Взагалі кажучи, написання якісних паралельних програм користувача для розподілених систем є доволі складним і трудомісткім процесом без застосування тій, чи іншій технології, яка певною мірою відповідає архітектурі операційної системи та узгоджується з обладнанням паралельного комп'ютера, його програмними і інструментальними засобами.

Метою методичних вказівок є ознайомлення студентів з основами технології MPI, надання студентам як можна більш повної інформації з теоретичних питань до відповідних лабораторних робіт, ознайомлення з програмно-інструментальними засобами, які рекомендуються при виконанні робіт і допомагають при створенні паралельних програм. Інформація про ці засоби повинна допомогти студентам при створенні лабораторних робіт з технології паралельних обчислень.

Завданням методичних вказівок є навчання студентів вибирати зі всіх можливих засобів розробки додатків ті, які є найбільш придатними і ефективними у кожному конкретному випадку їх застосування.

Отримані знання та навички допоможуть студентам при виконанні курсо-

вих та дипломних робот, а також у практичній діяльності при програмуванні розподілених систем.

1 ЗАГАЛЬНА ЧАСТИНА

В цьому розділі описані загальні питання, щодо виконання робіт, а саме:

- методика оцінювання знань з теоретичного матеріалу стосовно лабораторної роботи;
- методика оцінювання виконання практичної частини завдання;
- питання додержання правил техніки безпеки;
- опис програмного забезпечення, що застосовується при виконанні робіт;

1.1 Оцінювання знань з теоретичного матеріалу

До початку лабораторного заняття студент, за рахунок часу відведеного на самостійну роботу, повинен підготуватися до виконання роботи шляхом вивчення відповідних розділів курсу використовуючи посібник [1][1)], а також за допомогою додаткової літератури [2]–[4][2)] (електронні копії всіх джерел розміщені у електронній бібліотеці ОДЕКУ). Підготовка мусить містити у собі:

- ознайомлення з метою роботи;
- вивчення теоретичних питань, що відносяться до роботи;
- відповіді на контрольні питання, котрі містяться в методичних вказівках наприкінці теоретичної частини до кожної лабораторної роботи;

[1)] [1] Рольщиков В.Б. Застосування засобів інтерфейсу передачі повідомлень при програмуванні розподілених систем мовою Java: Навчальний посібник. Одеса: ТЕС, 2018. 209 с.

[2]Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.

[3]Букатов А.А., Дацюк В.Н., Жегуло А.И. Программирование многопроцессорных вычислительных систем. Ростов-на-Дону: ООО ЦВВР, 2003. 208 с.

[2)] [4] Шпаковский Г.И., Серикова Н.В. Программирование для многопроцессорных систем в стандарте MPI: Учебное пособие. Минск: БГУ, 2002. 323 с.

– оформлення титульного аркуша й заповнення розділів 1 і 2 протоколу лабораторної роботи (див. додаток А).

Необхідні сторінки підручника, а подеколи і додаткової літератури, вказуються безпосередньо після контрольних запитань перед завданням на практичну роботу до кожної лабораторної роботи.

Перед проведенням заняття викладач зобов'язаний перевірити в студента наявність підготовленого й попередньо заповненого протоколу. *Якщо підготовлений протокол відсутній, **студент не допускається** до виконання роботи* з виставлянням нульової оцінки за практичну частину роботи. Після перевірки наявності й правильності заповнення протоколу викладач повинен провести опитування знань студента й допустити останнього до роботи з відповідною оцінкою за теоретичну частину. Опитування проводиться по контрольних питаннях до даної лабораторної роботи (3–4 питання студентові), повна, якісна відповідь на них гарантує одержання 60% балів за теоретичну частину роботи. Ця частина опитування може поводиться або в усній формі, або у письмовому вигляді. Для оцінки ступеня знайомства студента з додатковою літературою та одержання більше високої оцінки студентові необхідно відповісти на кілька додаткових питань (1–2 питання) за змістом додаткової літератури.

1.2 Оцінювання виконання практичної частини роботи

Практична частина роботи оцінюється в такий спосіб. На кожному занятті викладач оцінює відсоток виконаної роботи. Якщо лабораторна робота розрахована на чотири години й більше, то загальна оцінка за роботу прямо пропорційна відсотку виконання завдання на першому лабораторному занятті. Наприклад, якщо на першому занятті студент написав 60% програмного коду, а на другому ті 40%, що залишилися, та здав на третьому занятті повністю оформлений протокол – він одержує 100% балів за дану роботу. Навпаки, якщо на першому занятті виконано 10% завдання на другому 20%, а на заключному занятті зданий

протокол з повністю виконаною лабораторною роботою, то студент може розраховувати на одержання тільки 30% від можливих балів за практичну частину роботи. На оцінку практичної частини роботи впливає й строк здачі протоколу. Для одержання максимуму балів повністю оформлений протокол повинен бути зданий на останньому занятті, котре відведено для виконання роботи або наступного дня після виконання роботи. Затримка зі здачею протоколу знижує оцінку на 5% за кожний повний і неповний тиждень затримки. Тобто студент, котрий за ритмічність виконання роботи міг одержати, наприклад, 70% відсотків можливих балів, але без поважної причини здав протокол через 16 днів після її завершення, одержить лише 70-15=55% балів за виконання роботи.

Максимальна кількість балів за практичну і теоретичну частини кожної роботи становить 100 балів.

Допуск до підсумкового контролю знань можливий тільки при одержанні не менш ніж 50% окремо за практичну та теоретичну частину ***всіх*** запланованих лабораторних робіт. Виконання половини робіт на 100%, а другої половини на 40% оцінюється тільки в 40%, а не 70% і, таким чином, студент не одержує допуск до підсумкового контролю знань.

Наприкінці семестру виводиться середня оцінка студента за всіма заняттями. Ця оцінка є підсумковою оцінкою за виконання практичного модуля і повинна складати не менш ніж 50% від максимально можливої сумарної оцінки за всі лабораторні роботи.

1.3 Техніка безпеки при виконанні робіт

При виконанні лабораторних робіт необхідно додержуватись правил, котрі встановлені в лабораторіях кафедри за узгодженням зі службою охорони праці і техніки безпеки університету. На початку виконання курсу робіт (на початку семестру) студенти повинні шляхом проведення інструктажу ознайомитися з цими правилами і розписатись у відповідному журналі.

Особливо слід звернути увагу на те, що при роботі за комп'ютером обов'язково треба робити 10-ти хвилинну перерву на кожну годину, що проведена за комп'ютером. Не можна працювати за комп'ютером більш ніж 4 години поспіль без тривалої перерви. Не рекомендується наближатись с тильного боку до моніторів, в яких застосовані електронно-променеві трубки. В разі виявлення несправності комп'ютера, ***категорично забороняється самостійно*** вживати заходів з діагностування та спроби з її усунення, в цьому випадку слід повідомити про несправність викладача або навчально-допоміжний персонал лабораторії.

1.4 Програмне забезпечення, необхідне для роботи

В цьому розділі наводяться дані про програмне забезпечення, вміння застосовувати яке значно полегшує роботу з якісного виконання курсу лабораторних робіт.

Лабораторні роботи виконуються в середовищі операційної системи Linux на віддаленому сервері. Для підключення Windows-робочої станції до Linux-сервера застосовується термінальний SSH-клієнт XShell (стислий опис цього клієнта наведений у додатку Б).

Після підключення до Linux-сервера за допомогою клієнта XShell подальшу роботу з виконання лабораторних робіт (створення вихідного файла, набір і редагування коду, компіляцію програми тощо) можна виконувати безпосередньо у вікні терміналу, застосовуючи команди операційної системи. Але більш ефективним є використання досить зручного і потужного файлера, який працює в текстовому режимі в консолі операційної системи Linux – Midnight Commander. Цей файлер дозволяє виконувати безліч, що найчастіше зустрічаються, основних команд ОС шляхом використання меню, або за допомогою гарячих клавішів. Опис роботи в середовищі Midnight Commander і у відповідному редакторі текстів Midnight Editor наданий у додатку В.

2 ЛАБОРАТОРНА РОБОТА №1
НАЛАГОДЖУВАННЯ ПАКЕТУ MPJ EXPRESS. КОМПІЛЯЦИЯ І ЗАПУСК НАЙПРОСТІШИХ ПРОГРАМ

Мета роботи – початкове ознайомлення студентів з реалізацією технології MPI в пакеті класів MPJ Express, підготовка оточення операційної системи для роботи з цим пакетом, отримання вмінь компіляції і запуску Java-програм з використанням методів пакету.

Література до заняття: [1][1)] – стор. 12–17, 28–32, [4][2)] – стор. 21–27.

Оцінка роботи

Максимальна кількість балів за роботу становить:

- теоретична частина 100 балів;
- практична частина 100 балів.

Остаточна оцінка визначається за методикою, описаної в загальній частині даних методичних вказівок.

2.1 Теоретична частина

2.1.1 Характеристика парадигми Message Passing Interface

Система програмування MPI в першу чергу пристосована до класу MIMD комп'ютерів з індивідуальною пам'яттю, тобто до багатопроцесорних систем з обміном повідомленнями. MPI має наступні особливості.

По-перше, MPI – це бібліотека, а не мова програмування. Вона лише визначає імена, виклики процедур і результати їхньої роботи. Програми, які пишуться на FORTRAN, C, C++ або Java компілюються звичайними компіляторами й пов'язуються з MPI-бібліотекою.

По-друге, MPI – це тільки опис, а не реалізація. Всі постачальники паралельних комп'ютерних систем пропонують власні реалізації MPI для своїх ма-

---

[1)] [1] Рольщиков В.Б. Застосування засобів інтерфейсу передачі повідомлень при програмуванні розподілених систем мовою Java: Навчальний посібник. Одеса: ТЕС, 2018. 209 с.

[2)] [4] Шпаковский Г.И., Серикова Н.В. Программирование для многопроцессорных систем в стандарте MPI: Учебное пособие. Минск: БГУ, 2002. 323 с.

шин. Правильна MPI-програма повинна виконуватися на всіх реалізаціях без зміни.

І по-третє, MPI відповідає моделі багатопроцесорної ЕОМ з передачею повідомлень.

У моделі передачі повідомлень процеси, що виконуються паралельно, мають роздільні адресні простори. Зв'язок між ними відбувається тоді, коли частина адресного простору одного процесу скопійована в адресний простір іншого процесу. Ця операція спільна й можлива тільки коли перший процес виконує операцію передачі повідомлення, а другий процес – операцію його одержання.

Процеси в MPI належать групам. Якщо група містить `n` процесів, то процеси нумеруються усередині групи номерами – цілими числами від 0 до `n–1`. Є виділена початкова група, який належать всі процеси в реалізації MPI.

Поняття контексту й групи об'єднані в єдиному об'єкті, який називається комунікатором. Таким чином, відправник або одержувач, які приймають участь в операції посилки або одержання, завжди звертаються до номера процесу в групі, яка належить даному комунікатору.

У MPI базисною операцією посилки є операція[1)]:

```
MPI_Send (address, count, datatype, destination, tag, comm),
```

де (`address`, `count`, `datatype`) – кількість (`count`) об'єктів типу `datatype`, що починаються з адреси `address` у буфері посилки; `destination` – номер одержувача в комунікаторі `comm`; `tag` – ціле число, яке використовується для опису повідомлення; `comm` – ідентифікатор групи процесів і комунікаційний контекст.

Базисною операцією прийому є операція:

```
MPI_Recv (address, maxcount, datatype, source, tag, comm,
```

[1)] Опис операцій зроблено відповідно до стандарту MPI, в бібліотеці класів MPJ Express для Java є деякі відмінності від стандарту, пов'язані з особливостями мови Java. Розгляд і аналіз розбіжностей є темою другої лабораторної роботи.

```
        status),
```

де (address, count, datatype) описують буфер приймача, як у випадку MPI_Send; sourse – номер процесу-відправника повідомлення в комунікаторі comm; status – містить інформацію щодо фактичного розміру повідомлення, джерела й тега. Sourse, tag і count фактично отриманого повідомлення відновлюються на основі status.

В MPI використовуються колективні операції, які можна розділити на два види:

– операції переміщення даних між процесами. Найпростіша з них – широкомовлення (broadcasting), MPI має багато й складніших колективних операцій передачі й збору повідомлень;
– операції колективного обчислення (мінімум, максимум, сума й інші, у тому числі й операції які обумовлюються користувачем).

В обох випадках бібліотеки функції колективних операцій створюються з використанням знання про переваги архітектури комп'ютера, щоб збільшити паралелізм виконання цих операцій.

Доволі часто виникає необхідність описувати процеси в проблемно-орієнтованій топології. В MPI використовується опис процесів у топології структур графів і ґрат.

В MPI уведені засоби, що дозволяють визначати стан процесу обчислень, які дозволяють налагоджувати програми й поліпшувати їхні характеристики.

В MPI є як операції send і receive, які на деякій час блокують процес виконання програми, так і їхній варіант, який не призводить до блокування, завдяки чому закінчення цих операцій може бути визначене явно. MPI також має кілька комунікаційних режимів. Стандартний режим відповідає загальній практиці в системах передачі повідомлень. Синхронний режим вимагає блокування send на час прийому повідомлення на противагу стандартному режиму, при якому send блокується лише до моменту захоплення буфера. Режим за готовні-

стю (для `send`) – спосіб, наданий програмістові, щоб повідомити систему, що цей прийом був зафіксований, отже, нижчерозташована система може використовувати швидкісніший протокол, якщо він доступний. Є ще режим, який дозволяє користувачеві управляти буферуванням повідомлень.

Програми MPI можуть виконуватися на мережах машин, які мають різні довжини й формати зберігання того самого типу `datatype`, так що кожна комунікаційна операція визначає структуру й усі компоненти `datatype`. Отже, реалізація завжди має достатню інформацію, щоб робити перетворення формату даних, якщо вони необхідні. MPI не визначає, як ці перетворення повинні виконуватися, дозволяючи реалізації робити оптимізацію для конкретних умов.

Процес є програмна одиниця, у якої є власний адресний простір і одна або кілька ниток. Процесор – це одиниця апаратних засобів, здатний до виконання програми. Деякі реалізації MPI установлюють, що в програмі MPI завжди одному процесу відповідає один процесор; інші – дозволяють розміщати багато процесів на кожному процесорі.

Якщо в кластері використовуються SMP-вузли (симетрична багатопроцесорна система із множинними процесорами), то для організації обчислень можливі два варіанти.

Перший – це коли для кожного процесора в SMP-вузлі породжується окремий MPI-процес. MPI-процеси усередині цього вузла обмінюються повідомленнями через поділювану пам’ять (необхідно налаштувати систему MPI відповідним чином).

І другий, коли на кожному вузлі запускається тільки один MPI-процес. Усередині кожного з них виконується розпаралелювання в моделі «загальної пам’яті», наприклад за допомогою директив OpenMP.

Чим більше функцій містить бібліотека MPI, тим більше можливостей надається користувачу для написання ефективних програм. Однак для написання більшості програм принципово досить наступних шести функцій:

MPI_Init	Ініціалізація MPI
MPI_Comm_size	Визначення числа процесів
MPI_Comm_rank	Визначення процесом власного номера
MPI_Send	Посилка повідомлення
MPI_Recv	Одержання повідомлення
MPI_Finalize	Завершення програми MPI

Як приклад паралельної програми, написаної в стандарті MPI для мови C, можна розглянути програму обчислення числа π, яке можна визначити використовуючи загальновідоме співвідношення:

$$\int_0^1 \frac{1}{1+x^2} dx = arctg(1) - arctg(0) = \frac{\pi}{4}.$$

Обчислення інтеграла потім замінюють обчисленням суми:

$$\int_0^1 \frac{1}{1+x^2} dx = \frac{1}{n} \sum_{i=1}^{n} \frac{4}{1+x_i^2},$$

де: $x_i = i/n$.

Програма може мати наступний вигляд:

```
#include "mpi.h"
#include <math.h>
int main(int argc, char *argv[])
  {
    int n, myid, numprocs, i;
    double mypi, pi, h, sum, x, t1, t2,
           PI25DT = 3.14159265358979323846264 3;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    while (1)
      {
        if (myid == 0)
          {
```

```
            printf ("Enter the number of intervals: (0 quits)");
            scanf ("%d", &n);
            t1 = MPI_Wtime();
          }
        MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
        if (n == 0) break;
        else
          {
            h = 1.0/ (double) n;
            sum = 0.0;
            for (i = myid +1; i <= n; i+= numprocs)
              {
                x = h * ( (double)i - 0.5);
                sum += (4.0 / (1.0 + x*x));
              }
            mypi = h * sum;
            MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0,
                       MPI_COMM_WORLD);
            if (myid == 0)
              {
                t2 = MPI_Wtime();
                printf("Pi приблизно є %.16f. Похибка %.16f\n",
                       pi, fabs(pi - PI25DT));
                printf("час – %f секунд\n", t2-t1);
              }
          }
      }
    MPI_Finalize();
    return 0;
  }
```

У програмі після декількох рядків визначення змінних розташовані три рядки, які є в кожній MPI-програмі:

```
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&myid);
```

Звертання до `MPI_Init` повинне бути першим звертанням в MPI-програмі, воно встановлює «середовище» MPI. У кожному виконанні програми може бути присутнім лише один виклик `MPI_Init`.

Комунікатор `MPI_COMM_WORLD` описує склад процесів і можливі зв’язки між ними. Виклик `MPI_Comm_size` повертає в `numprocs` кількість процесів, які

користувач запустив у цій програмі. Значення numprocs – розмір групи процесів, яка пов'язана з комунікатором MPI_COMM_WORLD. Процеси в будь-якій групі нумеруються послідовними цілими числами, починаючи з 0. Викликаючи MPI_Comm_rank, кожний процес з'ясовує свій номер (rank) в комунікаторі. Потім головний процес (який має myid=0) передає всім іншим процесам одержане від користувача значення числа прямокутників n:

```
MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
```

Тут три перші параметри відповідно позначають адресу, кількість і тип відправлених даних. Четвертий параметр указує номер процеса-джерела даних (головний процес), п'ятий параметр – назва комунікатора. Таким чином, після звертання до MPI_Bcast всі процеси мають значення n і одержані раніш власні ідентифікатори, що є достатнім для кожного процесу, щоб обчислити mypi – свій внесок в обчислення π. Для цього кожний процес обчислює область кожного прямокутника, що починається з myid + 1.

Потім всі значення mypi, обчислені індивідуальними процесами, підсумовуються за допомогою виклику Reduce:

```
MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
```

Перші два параметри описують джерело й адресу кінцевого результату. Третій і четвертий параметр описують кількість даних (1) і їхній тип, п'ятий параметр – тип арифметико-логічної операції, шостий – номер процесу для розміщення результату.

Потім керування знов передається на початок нескінченного циклу while. Цим користувачеві надається можливість задати нове значення n і підвищити точність обчислень. Коли користувач вводить нуль у відповідь на запит про нове n, цикл завершується, і всі процеси виконують:

```
MPI_Finalize();
```

після цього будь-які операції MPI виконуватися вже не будуть.

Функція `MPI_Wtime()` використовується для вимірювання часу виконання фрагменту програми, розташованого між двома включеннями в програму цієї функції.

Раніше відмічалося, що для написання більшості програм досить лише 6 функції, серед яких основними є функції обміну повідомленнями типу «точка-точка» (або функції парного обміну). Будь-яку програму обчислення можна написати за допомогою функцій парного обміну, але, як правило, ефективнішими є функції, які відносяться до класу колективних обмінів. Колективні функції `Bcast` і `Reduce` можна виразити через парні операції `Send` і `Recv`. Наприклад, для тієї ж програми обчислення числа π операція `Bcast` для розсилання числа інтервалів виражається через цикл у такий спосіб:

```
for (i=0; i<numprocs; i++)
  MPI_Send(&n, 1, MPI_INT, i, 0, MPI_COMM_WORLD);
```

Паралельна MPI програма може містити різні виконувані файли. Цей стиль паралельного програмування часто називається MPMD (безліч програм при безлічі даних) на відміну від програм SPMD (одна програма при безлічі даних). SPMD не слід плутати з SIMD (один потік команд при безлічі даних). Вищенаведена програма обчислення числа π відноситься до класу програм SPMD. Такі програми значно легше писати й налагоджувати, чим програми MPMD. У загальному випадку системи SMP і MPI можуть імітувати одна одну:

– за допомогою організації єдиного адресного простору для фізично розділеної по різних процесорах пам’яті.
– на SMP-машинах виродженим каналом зв’язку для передачі повідомлень служить поділювана пам’ять.
– шляхом використання комп’ютерів з поділюваною віртуальною

пам’яттю. В таких системах загальна пам’ять як така відсутня. Кожний процесор крім власної локальної пам’яті може звертатися й до локальної пам’яті інших процесорів, використовуючи «глобальну адресу». Якщо «глобальна адреса» указує не на власну локальну пам’ять, то доступ до пам’яті реалізується за допомогою повідомлень, що пересилаються по комунікаційній мережі.

2.1.2 Відомості про пакет класів MPJ Express. Компіляція і запуск програм

На сьогодні для програмування мовою Java існує декілька різних пакетів класів, які реалізують інтерфейс MPI. У зв’язку з цим, ще наприкінці 1998 року для Java прив’язки стандарту передачі повідомлень була сформована робоча група «Message-Passing Working Group of the Java Grande Forum». Ця група виступила з початковим проектом для загальної специфікації API для MPI. Пізніше у проект були внесені деякі зміни і поточний API, тепер вже відомий під абревіатурою MPJ поступово стає еталонним інтерфейсом при програмуванні мовою Java.

Завдяки внесеним змінам, MPJ API дещо відрізняється від первісної реалізації, яка відповідає специфікації mpiJava API 1.2. Різниця між цими двома API, полягає в основному в схемах іменування класів і методів. Функціональні можливості, що надаються як API, для користувачів, по суті, ті ж самі.

У якості програмного забезпечення для виконання лабораторних робот з дисципліни «Технології розподілених систем та паралельних обчислень» обраний пакет класів специфікації MPJ API, який має назву MPJ Express (MPJE). Пакет поставляється за дозвільною ліцензією MIT (Massachusetts Institute of Technology) і доступний як MPJ Express на офіційному сайті розробника програмного забезпечення (ПЗ) – http://mpjexpress.org/.

До недоліків обраного пакета можна віднести підтримання здебільшого лише функцій стандарту MPI-1.1. Однак, слід відзначити, що й дотепер навіть варіант стандарту 2.0 MPI у повному обсязі не набув широкого втілення у багатьох реалізаціях MPI API. Майже повністю його підтримку здійснюють лише коштовні комерційні проекти. З вільного ПЗ на сьогодні можна відзначити хіба що пакет функцій MPICH, розробники якого анонсують підтримку стандарту 2.0. А підтримка останньої версії стандарту 3.1 поки що тільки почалася.

Але цей недолік зовсім не заважає знайомитись з принципами технології MPI і створювати цілком працездатні і доволі складні програми паралельних обчислень.

Пакет класів MPJ Express, як практично все ПЗ для Java, є вільно розповсюджуваним. Останню версію пакету можна завантажити зі сторінки розробника за адресою https://sourceforge.net/projects/mpjexpress/files/releases/. Отриманий архів не вимагає ніякого встановлення в системі, а просто розпаковується у будь-яке придатне для цього місце у файловій системі. В лабораторіях кафедри пакет вже розпакований і для його установки вибраний каталог «/usr/share/mpj» на Linux-сервері з IP-адресою 192.168.70.202.

Згідно з даними розробників пакета класів MPJ Express, програми створені з використанням методів класів можуть працювати у двох основних режимах: багатоядерному на процесорах з декількома ядрами і у багатопроцесорному на кластері робочих станцій. Можливе також використання комбінованого режиму виконання програм. Лабораторні роботи передбачають створення і запуск програм лише в багатоядерному режимі процесора.

Але для нормальної роботи з API MPJ Express недостатньо лише розпакувати архів у той чи інший архів, треба ще виконати настроювання деяких змінних оточення операційного середовища. Конкретніше, слід встановити нову змінну оточення на ім’я `MPJ_HOME`, призначення якої – вказати на місце знаходження файлів бібліотек класів і інших файлів пакета в дереві каталогів опера-

ційної системи. А також змінити змінну PATH таким чином, щоб операційна система могла легко знайти шлях до бінарних файлів пакета.

У відкритих операційних системах такі дії виконуються за допомогою системної команди «export» до якій у якості параметра передаються ім'я змінної оточення операційного середовища і її нове значення. Наприклад, це може робитись так:

```
export MPJ_HOME=/usr/share/mpj
```

де MPJ_HOME – ім'я змінної, а

/usr/share/mpj – рядок символів, значення якого привласнюється змінній (рядок лапками не оточується).

Тому що операційні системи типу Linux критичні відносно регістра символів у команді, слід дуже ретельно слідкувати за вірним написанням, також неможна ставити пусті розділювачі (пробіли) в команді там де їх не повинно бути.

За аналогією з встановленням змінної оточення MPJ_HOME виконується й встановлення змінної PATH.

Завдання значень для цих двох змінних надає можливість компілювати й запускати на виконання відкомпільовані програми з використанням класів пакета MPJ Express.

Компіляція програми виконується стандартними засобами SDK Java, тобто компілятором байт-коду «javac». В командному рядку компіляції, крім зазначення імені файла з вихідним кодом програми, за допомогою ключа «–cp» слід вказати повний шлях «.:$MPJ_HOME/lib» до архівного бібліотечного файла «mpj.jar», в якому міститься байт-код всіх необхідних у програмі методів класів.

Запуск відкомпільованої програми виконується за допомогою спеціального файла системного сценарію (скріпта) – «mpjrun.sh». Як параметри у скріпт

передаються: ключ «-np» після якого вказується кількість потоків, які треба запустити в системі; ім'я класу (без суфіксу «.class»), з методу «main» якого повинна стартувати програма.

2.2 Завдання до лабораторної роботи

Провести настроювання змінних оточення операційного середовища для одержання можливості компілювання й запуску Java-програм, написаних із застосуванням методів класів пакету MPJ Express.

Відкомпілювати й запустити на виконання найпростіші програми вихідний код яких міститься або у вже готовому .java-файлі в робочому каталозі студента, або наведений в цих методичних вказівках нижче. Проаналізувати роботу одержаних програм і зробити відповідні висновки.

2.3 Порядок виконання роботи

1. З'єднатися з сервером за допомогою термінального SSH-клієнта Xshell (див. додаток Б).
2. Якщо з'єднання виконане успішно, для загрузки файлового менеджера Midnight Commander у командному рядку Linux ввести команду «mc» і натиснути клавішу <Enter>.
3. Перевірити, що інтерфейс файлового менеджера Midnight Commander має вигляд без спотворювань. В противному випадку слід встановити кодування символів Unicode UTF-8 так, як це описано у додатку Б.
4. У командному рядку файлера послідовно набрати таки дві команди операційної системи (при наборі ретельно слідкувати за використанням регістру символів, знаків, які не є буквами і пробілів):

```
export MPJ_HOME=/usr/share/mpj
```

```
export =$PATH:$MPJ_HOME/bin
```

після кожного рядка слід натискувати клавішу <Enter>.

5. Командою:

```
javac –cp .:$MPJ_HOME/lib/mpj.jar Hello.java
```

відкомпілювати програму Hello.java, яка знаходиться у домашньому каталозі користувача. Після вдалої компіляції у каталозі повинний з'явитись і відобразитись у панелі Midnight Commander файл на ім'я Hello.class. Відсутність файла свідчить або про невірне завдання командою `export` змінної оточення операційного середовища `MPJ_HOME` у пункті 4, або про помилку у команді компіляції програми.

Про характер помилки можна дізнатись проаналізував вивід повідомлень ОС. Цей вивід можна побачити на екрані терміналу відключив панелі Midnight Commander. Для їх термінового відключення використовується комбінація клавішів <Ctrl-O> на клавіатурі. Повторне натискання комбінації повертає відображення панелей.

6. Якщо з якої-небудь помилки компіляція не вдалась, повторити виконання пунктів 4 і 5.
7. У разі вдало виконаної компіляції, командою:

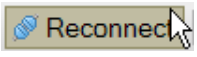
```
mpjrun.sh –np 7 Hello
```

запустити отриману програму на виконання (вказана кількість процесів 7 є довільною і лише повинна бути не менше трьох). Результати роботи програми можна побачити на екрані терміналу (пам'ятайте про комбінацію клавішів <Ctrl-O> на клавіатурі).

Наявність повідомлення ОС про неможливість запуску програми

ї, як наслідок, відсутність декількох привітань користувачеві з боку програми свідчить або про помилкове завдання змінної оточення `PATH` у команді `export` у пункті 4, або про помилку у команді запуску програми.

В цьому разі слід дуже ретельно перевірити обидві введені команди, виправити замічені помилки і повторити запуск програми (повторну компіляцію виконувати не треба).

8. При вдалій компіляції і запуску програми слід у панелі Midnight Commander вибрати файл з ім'ям .bashrc, натисканням клавіші <F4> відкрити його у текстовому редакторі для редагування, перейти у самомий кінець файла і дописати дві раніш введені команди export для MPJ_HOME і для PATH, клавішею <F2> зберегти відредагований файл, клавішею <F10> вийти з редактора. Відтепер система вважається готовою для подальшого застосування пакету класів MPJ Express.
9. Для перевірки вірності виконання пункту 8 треба: натиснути клавішу <F10> і вийти з Midnight Commander, в панелі інструментів Xshell вибрати кнопку «Disconnect (Alt+C)» і розірвати зв'язок з сервером, потім натиснувши на кнопку 

знов відновити зв'язок з сервером.
10. Завантажити Midnight Commander (див. пункт 2).
11. За допомогою сполучення клавіш <Shift-F4> відкрити на редагування пустий новий файл і в ньому набрати таку програму:

```
import mpi.*;
class TestMPJ {
    public static void main(String[] args) throws MPIException
      {
        System.out.println("До виклику MPI.Init");
        MPI.Init (args);
        System.out.println("Паралельна секція");
        MPI.Finalize ();
        System.out.println("Після MPI.Finalize");
      }
  }
```

зберегти файл з ім'ям класу `TestMPJ` (слідкувати за регістром букв) і з суфіксом `.java`.

12. Командою:

```
javac –cp .:$MPJ_HOME/lib/mpj.jar TestMPJ.java
```

виконати компіляцію щойно створеної програми TestMPJ.java. Повідомлення компілятора про помилки і відсутність в каталозі файла TestMPJ.class свідчать або про помилку завдання змінної оточення операційного середовища `MPJ_HOME` командою `export` в файлі `.bashrc`, або про помилку у команді компіляції ще можлива наявність помилок в набраному вихідному коді програми.

В разі успішної компіляції перейти до пункту 14.

13. Ретельно перевірити вихідний код і при необхідності виправити знайдені помилки і вернутись до пункту 12. В разі відсутності помилок в коді ретельно перевірити командний рядок компіляції програми, виправити замічені помилки і виконати компіляцію.

Якщо все вірно, відкрити на редагування файл `.bashrc`. Перевірити завдання змінної оточення операційного середовища `MPJ_HOME` командою `export` і виправити її. Зберегти відредагований файл і в командному рядку ввести команду:

```
source .bashrc
```

після чого повернутись до пункту 12.

14. Командою:

```
mpjrun.sh –np 3 TestMPJ
```

(кількість процесів 3 – величина умовна) виконати запуск програми.

Повідомлення ОС про помилки свідчать або про помилку завдання змінної оточення `PATH` операційного середовища командою export в файлі `.bashrc`, або про помилку у команді запуску. Слід перевірити команду запуску і повторити запуск програми.

Якщо все вірно – перейти до пункту 16.

15. Якщо команда вірна, то необхідно знов відкрити на редагування файл .bashrc. Перевірити завдання змінної оточення операційного середовища PATH командою export і виправити її. Зберегти відредагований файл і в командному рядку ввести команду:

```
source .bashrc
```

після чого повернутись до пункту 14.

16. Завершити виконання лабораторної роботи і перейти до дооформлення протоколу роботи.

2.4 Зміст звіту про лабораторну роботу

Попередньо оформлена коротка теоретична частина протоколу лабораторної роботи в якій повинна містити:

– стислі відомості про настроювання систем відкритого типу для роботи з бібліотекою класів MPJ Express у багатоядерному режимі;
– так само стислі відомості про поняття «Комунікатор», методи операцій загального призначення і деякі константи визначені у класі MPI, наприклад, `COMM_WORLD`, `ANY_TAG`, `BYTE` тощо.

Після виконання роботи у протоколі має відображатись наступне:

– всі дії виконані при настроюванні системи;
– команди компіляції програм і їхнього запуску на виконання;

– вихідний код програми `Hello.java` і результати ії роботи з аналізом і поясненням цих результатів;
– процес створення і редагування нового вихідного файла;
– вихідний код програми `TestMPJ.java` з результатами її роботи і з їхнім аналізом і поясненням;
– висновки з проведеної роботи, які повинні стосуватись предмета вивчення, тобто характеризувати основні властивості, які притаманні пакету класів MPJ Express, а не сповіщати саме про той факт, що Ви їх вивчали і навіть мабуть вивчили.

2.5 Завдання для підготовки до лабораторної роботи

1. За посібником [1][1)] ознайомитись зі складом пакету класів MPJ Express, його основними файлами і класами, які входять до нього.
2. Розібратися з встановленням і настройкою пакету MPJ Express у різних ОС.
3. Спробувати провести встановлення пакету на власному комп'ютері у операційній системі Windows.
4. Вивчити основні поняття технології програмування паралельних процесів MPI.
5. Ознайомитись з методами загального призначення бібліотеки MPI і з оголошенням і дією цих методів у пакеті класів MPJ Express.
6. Провести попереднє оформлення протоколу лабораторної роботи.

2.6 Контрольні запитання до лабораторної роботи

1. Опишіть процес отримання і встановлення пакета програмного забезпе-

[1)] [1] Рольщиков В.Б. Застосування засобів інтерфейсу передачі повідомлень при програмуванні розподілених систем мовою Java: Навчальний посібник. Одеса: ТЕС, 2018. 209 с.

чення MPJ Express.

2. Розкажіть про процес настроювання пакета для роботи у багатоядерному режимі у системах відкритого типу. Які змінні середовища необхідно встановити в системі?
3. Розкажіть про компіляцію паралельних Java-програм для роботи у різних ОС.
4. Опишіть синтаксис командного рядка системи для запуску Java-програми з використанням пакета класів MPJ Express.
5. З яких підпакетів складається пакет класів MPJ Express? Опишіть їхнє призначення.
6. До якого підпакета включена переважна більшість класів пакета MPJ Express? Скільки і які класи входять у нього?
7. Розкажіть про структуру класу MPI. Сформулюйте і поясніть основні властивості цього класу. По змозі нарисуйте UML-діаграму, яка йому відповідає.
8. Назвіть і стисло охарактеризуйте операції загальної групи стандарту MPI.
9. Що таке група процесів, навіщо таке поняття включено у стандарт MPI, які їхні основні властивості?
10. Що таке комунікатор процесів, навіщо він потрібен, чим ця структура відрізняється від групи?
11. Комунікатори якого типу Ви знаєте? Чим відрізняється спосіб застосування комунікаторів різних класів?
12. Назвіть і охарактеризуйте комунікатори, які створюються в системі при запуску MPI-програми.
13. Розкажіть, що таке ранг процесу, для чого він призначений, опишіть його основні властивості.
14. Дайте визначення повідомлення, розкажіть, що таке тег повідомлення, для чого він призначений, яких значень він може набувати.

15. У якій структурі зберігаються атрибути повідомлення? Що це за константи MPI.ANY_TAG і MPI.ANY_SOURCE, як їх застосовувати у програмі?
16. Який механізм служить для оповіщення про виникнення помилки в працюючій програмі?
17. Розкажіть про статичний метод класу MPI – метод Init. Які його особливості Ви можете назвати? Які параметри йому передаються і які повертаються?
18. Чи можна у MPI-прграмі обійтись без застосування методу MPI.Finalize? Які особливості має цей метод при використанні?
19. Для чого призначений метод MPI.Initialized, чи можна його застосовувати до виклику методу Init і після використання методу Finalize?
20. Чим відрізняється застосування методу MPI.Abort від застосування методу Finalize у стандарті MPI, як реалізований метод Abort в описуваній бібліотеці MPJ Express?
21. Як визнати розмір комунікатора і ранги процесів, які до нього належать, а також визначити ім’я вузла мережі, на якому запущений і виконується процес?
22. Чим відрізняється застосування методів MPI.Wtime і MPI.Wtick, для вимірювання яких параметрів системи вони призначені? Який (які) стандартний метод бібліотеки класів мови Java можна застосовувати замість методу Wtime?

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Рольщиков В.Б. Застосування засобів інтерфейсу передачі повідомлень при програмуванні розподілених систем мовою Java: Навчальний посібник. Одеса: ТЕС, 2018. 209 с.
2. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.
3. Букатов А.А., Дацюк В.Н., Жегуло А.И. Программирование многопроцессорных вычислительных систем. Ростов-на-Дону: ООО “ЦВВР”, 2003. 208 с.
4. Шпаковский Г.И., Серикова Н.В. Программирование для многопроцессорных систем в стандарте MPI: Учебное пособие. Минск: БГУ, 2002. 323 с.
5. Костромин В.А. Самоучитель Linux для пользователя. СПб.: БХВ-Петербург, 2003. 672 с.
6. Xshell 4 User Guide Secure Terminal Emualtor – Seoul: NetSarang Computer, Inc., 2011. – 157 p. URL: http://www.netsarang.com/docs/xshell4_manual.pdf (дата звернення 24.03.2019)

ДОДАТОК А

Форма першої сторінки протоколу роботи

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСКІЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ
Кафедра інформаційних технологій

ПРОТОКОЛ ЛАБОРАТОРНОЇ РОБОТИ № ___

З ТЕХНОЛОГІЇ РОЗПОДІЛЕНИХ СИСТЕМ ТА ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ

тема або назва лабораторної роботи: _____

_____

виконав(*ла*) студент(*ка*) групи _____ _____
прізвище та ініціали

Оцінка за підготовку до роботи (теоретична частина)	Оцінка за виконання роботи (практична частина)	Оцінка за захист роботи (остаточна, загальна кількість балів)

1. Варіант і завдання до лабораторної роботи:

2. Коротка теоретична частина роботи:

3. Текст програми:

4. Результати роботи програми:

5. Висновки:

ДОДАТОК Б

Опис програми термінального SSH-клієнта Xshell

Лабораторні роботи з використанням ОС Linux виконуються на віддаленому Linux-сервері з робочої станції під управлінням ОС Windows. При подібній роботі необхідний якийсь інструмент, котрий дозволяє емулювати Linux-термінал, призначений для роботи на платформі Microsoft Windows. Такий програмний продукт за назвою Xshell [6][1)] створений компанією NetSarang Computer – провідним розроблювачем і постачальником програм, призначених для продуктивної взаємодії у мережах.

Програмний продукт Xshell підтримує протоколи SSH1 і SSH2, способи автентифікації користувачів із застосуванням відкритого ключа DSA/RSA і дозволяє кодувати трафік за допомогою різноманітних алгоритмів шифрування. Одним з основних аргументів на користь вибору цієї програми слугує те, що вона є безкоштовною для використання дома й у закладах освіти. Правда необхідно відзначити й те, що існує і великою популярністю користується інший безкоштовний клієнт – PuTTY. Але інтерфейс користувача, що надається Xshell є більш звичним для людей, котрі працюють в Windows, ніж трохи спрощений інтерфейс PuTTY.

За твердженням авторів, програма працює в будь-якій операційній системі Windows починаючи з Windows 2000 до Windows 10 (новітня версія), можлива робота навіть в Windows 98 (більш ранні версії).

NetSarang Xshell надзвичайно потужний клієнт, котрий підтримує протоколи, що вже згадувалися – SSH1, SSH2, крім того підтримуються SFTP, TELNET, RLOGIN і SERIAL (останній для підключення тільки по COM портах), можливе здійснення роботи через proxy-сервер. Програма дозволяє одночасно підключатися до декількох серверів – у програмі убудована підтримка вкладок.

[1)] [6] Xshell 4 User Guide Secure Terminal Emualtor – Seoul: NetSarang Computer, Inc., 2011. 157 p. URL: http://www.netsarang.com/docs/xshell4_manual.pdf (дата звернення 24.03.2019)

Існує можливість швидко міняти розташування інструментів на панелях і їхній набір, для кожної вкладки, настроювати колірну схему, шрифти, перекодування й т.і. Всі схеми настроювань зберігаються, і надалі дуже просто повернутися до зручного інтерфейсу, що влаштовує користувача.

Безпосередньо робота із сервером організована так само дуже зручно – можна зберегти як окремі команди, так і їхній набір у панелі «Quick Commands». Набори зручно структуруються й у результаті можна зовсім позбутися нудотного набивання команд. Але навіть цим розроблювачі з NetSarang не обмежилися – в Xshell команди можна відправляти відразу на кілька сесій. Єдиним помітним “недоліком”, у роботі із клієнтом, можна назвати відсутність іншої мови інтерфейсу, крім англійської, але робота з ним не повинна спричиняти труднощі у студентів досить добре знайомих зі стандартним для Windows інтерфейсом вікон.

Все це різноманіття функцій має велике значення для системних адміністраторів, що віддалено працюють з Linux-серверами. Для безпосереднього виконання лабораторних робіт необхідно вміти користуватися тільки досить обмеженим набором цих функцій.

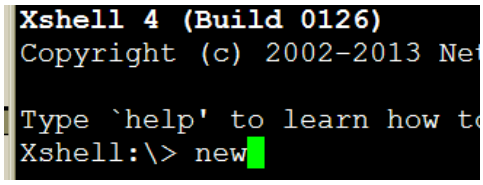
Тому нижче описані тільки запуск Xshell, створення з’єднання й здійснення деяких основних настроювань.

Запускається Xshell, стандартним для ОС Windows способом – подвійним клацанням лівої кнопки миші по іконці програми на робочому столі або, якщо такої немає, то через меню кнопки «ПУСК».

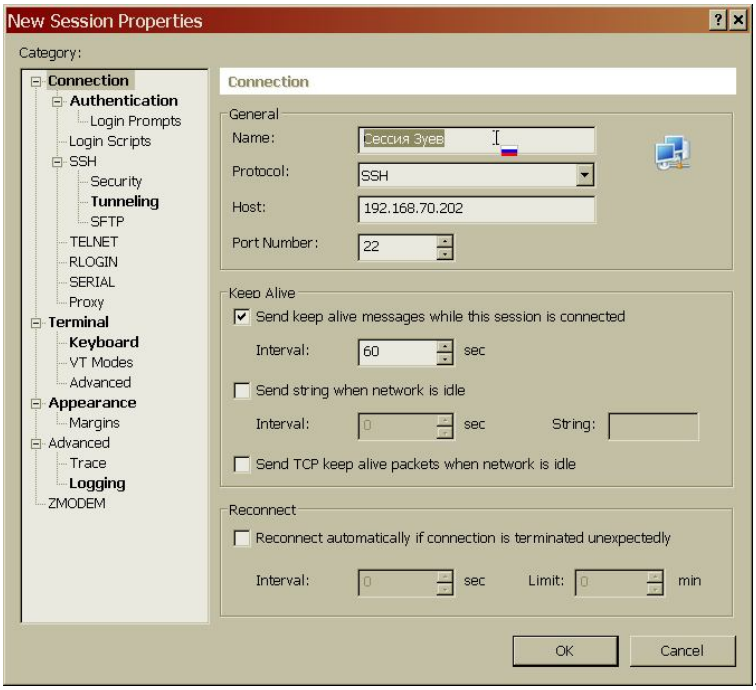
Вікно програми, котре відчиняється, хоча і являє собою емуляцію термінала, але, у той же час, є звичайним вікном Windows з рядком меню й панеллю інструментів:

Докладний опис інтерфейсу програми не має особливого сенсу, тому більш доцільним буде перейти безпосередньо до опису процесу створення з'єднання з віддаленим сервером.

Створити нову сесію можна, як звичайно, скориставшись або меню «File | New...», або кнопкою «New» на панелі інструментів, або, нарешті, гарячою клавішею «Alt+N». Є ще один шлях створення нової сесії – це просто введення у командному рядку термінала команди «`new`» і натискання клавіші «`Enter`».



При цьому для створення з'єднання із сервером відкривається діалогове вікно «New Sessions Properties»

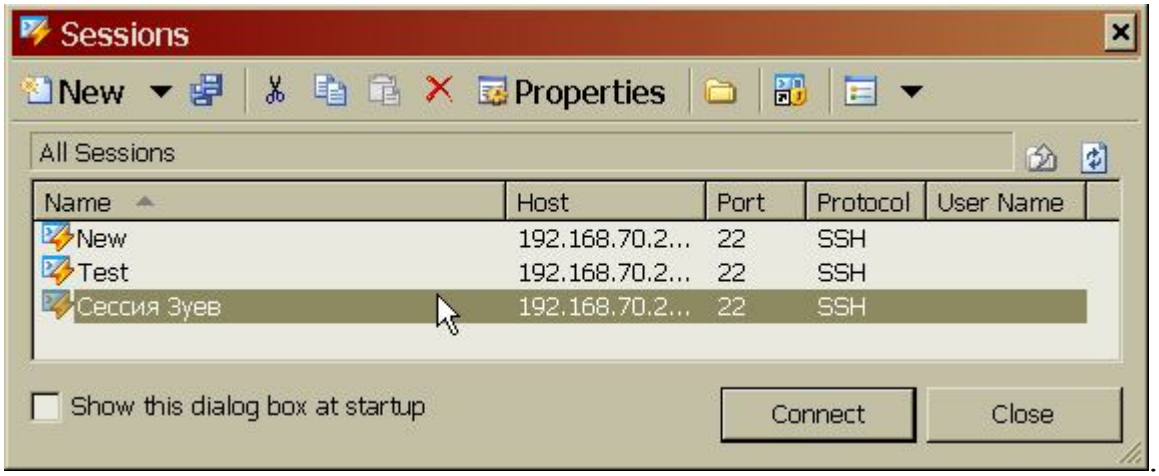


У полі вводу «Name» цього вікна необхідно ввести ім'я створюваної сесії це, наприклад, може бути логин користувача або будь-що інше. Цим ім'ям на-

далі можна буде користуватися для повторного відкриття сесії з тими ж параметрами. У полі «Host» уводиться ім’я або IP-адреса сервера, найбільш імовірна адреса – це 192.168.70.202, але можливо й інше значення, про яке можна довідатися у викладача, що проводить заняття. Інші поля вводу «Protocol», «Port Number» та інші елементи керування залишаються без зміни. У цьому ж вікні, скориставшись полем «Category:», можна задати спеціальні параметри конфігурації сесії.

Параметри можна установити й пізніше вже після з’єднання й початку роботи в сесії.

Після закінчення діалогу відкривається нове діалогове вікно – «Sessions», у якому відображається список всіх можливих сесій і пропонується приєднатися до щойно створеної, або до будь-якої іншої на вибір користувача

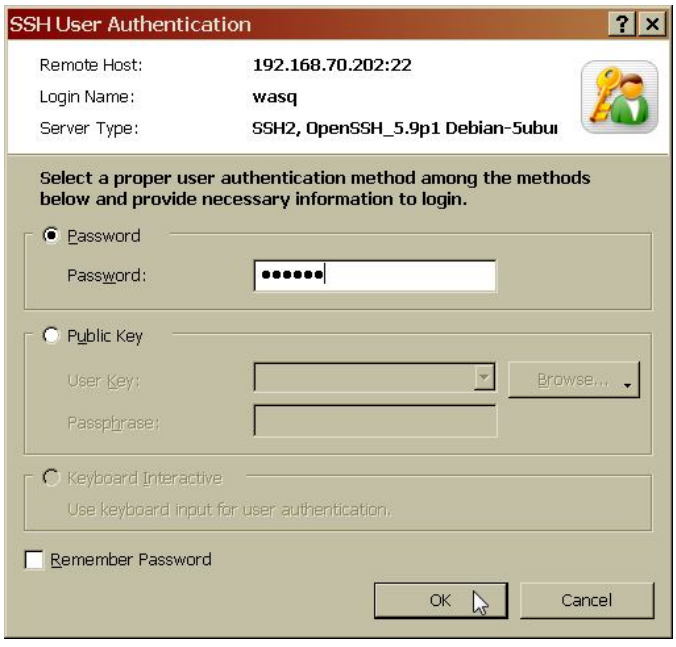


Після вибору сесії й натискання на кнопку «Connect» здійснюється з’єднання із сервером і останній переходить к процесу автентифікації (пізнавання) користувача і по-перше запитує його логин. При цьому відчиняється нове діалогове вікно «SSH User Name» у полі введення якого треба вказати робочий логин. Оскільки при введенні яких-небудь даних ОС Linux постійно слідкує за регістром символів у рядку введення і вважає помилковим логин введений не в тому регістрі, слід дуже ретельно слідкувати за правильністю використання ре-

гістра символів.

Крім рядка введення логину у віконці ще можна встановити прапорець «Remember User Name», але у лабораторії система налаштована так, що ця опція недоступна.

Після вводу логину, сервер у наступному діалоговому вікні «SSH User Authentication» запитує пароль входу в систему. Все, що було сказане вище про регістр символів вводу логину, є повністю вірним і для вводу пароля в цьому віконці.



За результатом правильного уведення логину й пароля і їхнього збігу з ба-

зою логинів і паролів операційної системи користувач допускається до його домашнього каталогу й одержує можливість працювати на віддаленому сервері.

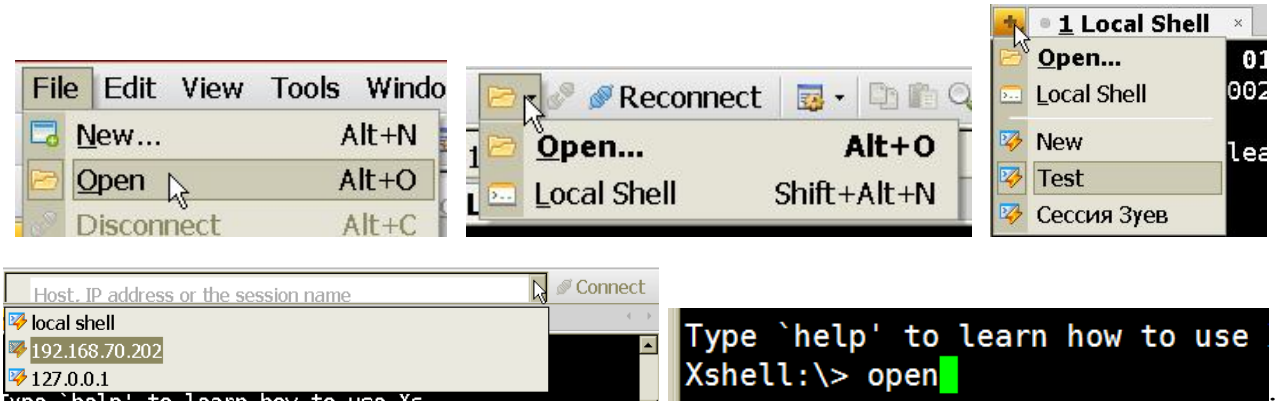
Ознакою успішного входу в систему є поява у вікні терміналу Xshell рядка запрошення подібного до наведеного на рисунку. Рядок запрошення починається з логину користувача за яким йде символ «@» і ім'я серверу. Закінчується рядок символами «:~$». Далі в рядку можна вводити будь-яку команду операційної системи. На рисунку наведена команда mc виклику файлового менеджера Midnight Commander (див. додаток В).

У противному разі, при незбігу логину і пароля процес автентифікації повторюється.

Якщо сесія вже була створена раніше, то викликати діалогове вікно «Sessions» можна багатьма іншими способами. Для цього є такі можливості:

– через пункт меню «File | Open»;
– за допомогою кнопки «Open» на панелі інструментів;
– скористатися кнопкою «+» ліворуч в області заголовків вкладок поточних сесій;
– введенням імені сесії або IP-адреси хосту у рядку адреси, який розташований під панеллю інструментів, або вибрати їх зі списку, що випадає, і натиснути кнопку «Connect»;
– введенням термінальної команди «open»;
– насамкінець, по гарячій клавіші «Alt+O» (іноді може не спрацювати).

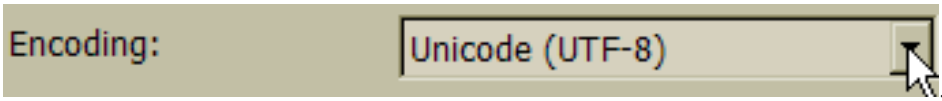
Всі ці можливості наведені на рисунку нижче.



При створенні нової сесії настійно рекомендується встановлювати деякі її параметри. Насамперед це необхідно виконувати для встановлення стандартного для ОС Linux кодування символів «Unicode (UTF-8)». Як уже було сказано вище, при створенні нової сесії, настроювання ii параметрів виконується у полі «Category» вікна «New Sessions Properties». Відкрити це діалогове вікно можна і з вікна «Sessions» скориставшись пунктом меню

. Після цього в полі вибору «Category» вибирається пункт

і потім у полі «Encoding» зі списку, що випадає, вибирається кодування символів «Unicode (UTF-8)».

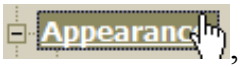


Тут же у пункті

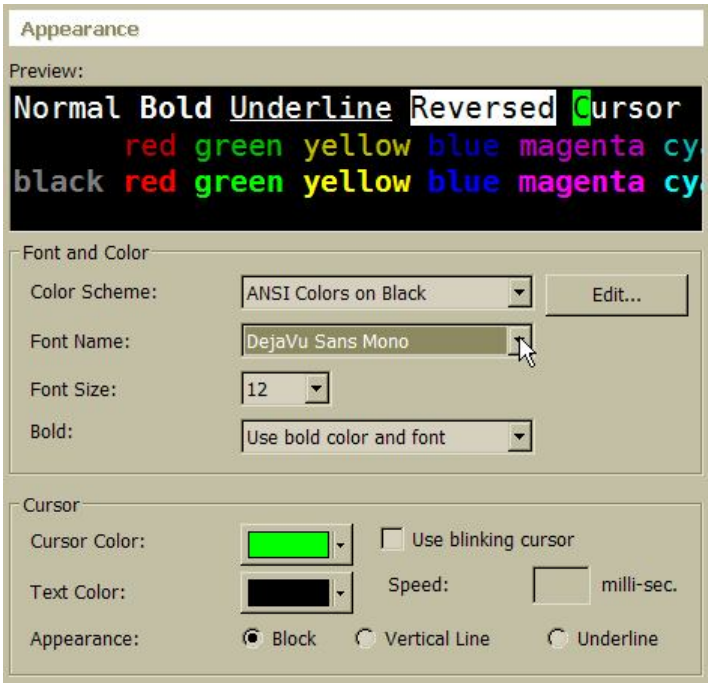
у полі «Function Key Type» зі списку, що випадає, можна вибирати «Linux»

,

хоча цей параметр можна і не змінювати, а залишити його значення за замовчуванням «<Default>».

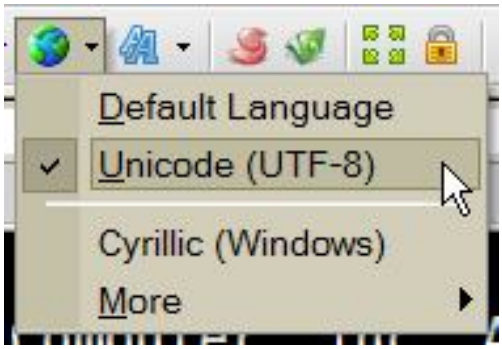
Бажаючі, скориставшись пунктом 

, можуть змінити тип і розмір шрифту термінала, кольорову схему і інше.



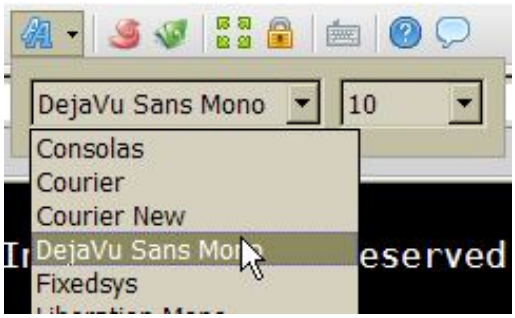
Якщо при створенні (відкритті) сесії настроювання кодування символів і шрифту не були виконані, то це можна зробити й пізніш в уже відкритій сесії.

Для кодування символів це виконується за допомогою натискання кнопки на панелі інструментів з наступним вибором кодування «Unicode UTF-8»:



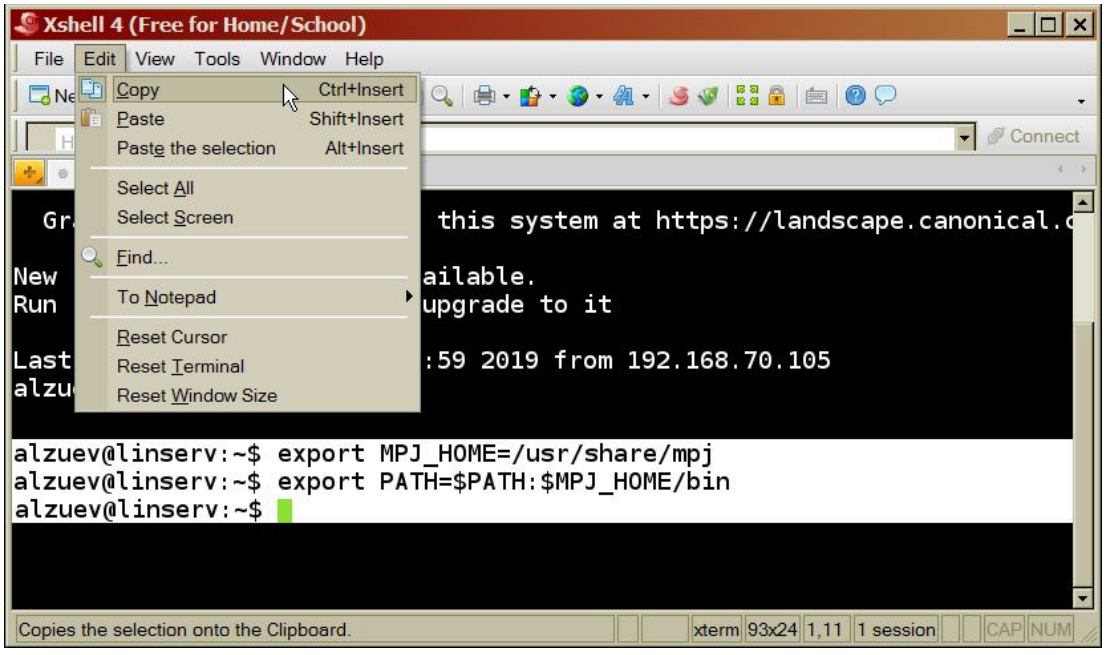
.

Для зміни шрифту застосовується кнопка на панелі інструментів :



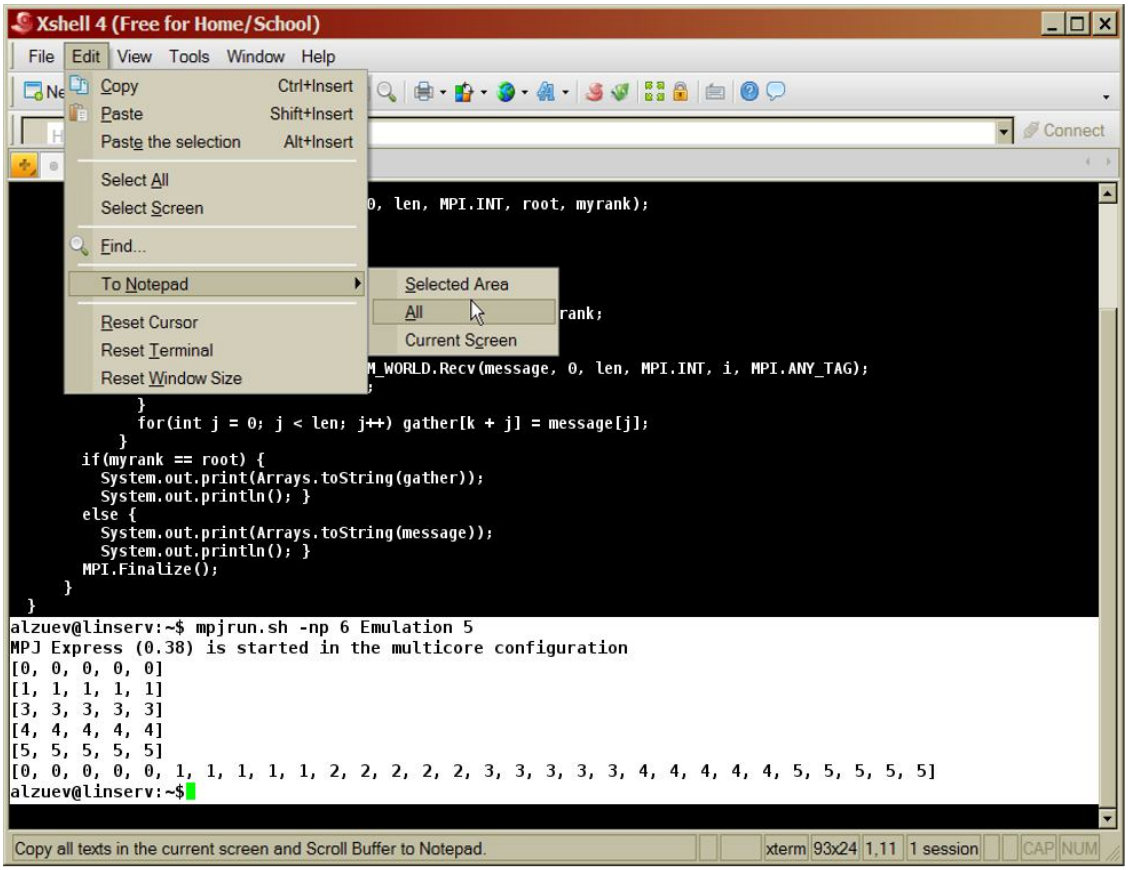
При документуванні результатів роботи в системі за допомогою термінального клієнта Xshell і при роботі з текстовими документами дуже корисним може виявитись застосування пункту меню клієнта «Edit».

Так на рисунку наведено застосування функції меню «Edit | Copy» для копіювання введених командних рядків в системний буфер обміну для подальшого занесення у протокол лабораторної роботи.

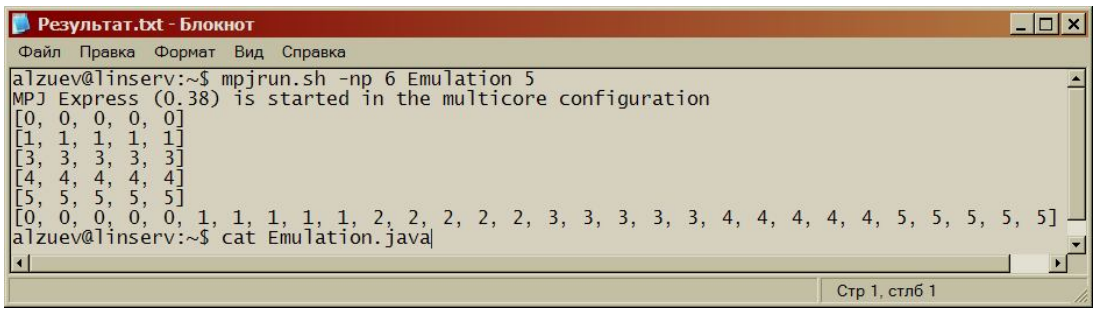


Виділення необхідної частини терміналу виконується шляхом пересування миші при натиснутій лівій клавіші.

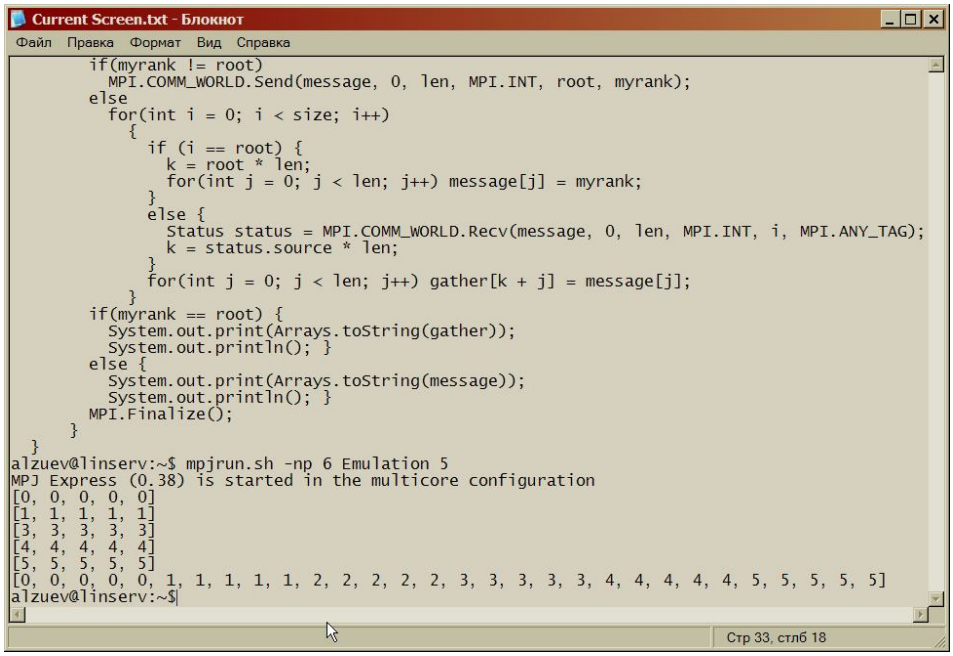
Ще більше можливостей роботи терміналам надає застосування пункту меню «Edit | To Notepad» у якому міститься ще три підпункти, а власне «Selected Area», «All» і «Current Screen». Ці пункти меню наведені на наступному рисунку.



При звертанні к цим пунктам автоматично викликається програма «Блокнот» і в її вікні відображається вміст фрагменту виділеного мишей якщо користувач застосував пункт меню «Edit | To Notepad | Selected Area». Результати застосування цієї функції термінального клієнта наведені на рисунку.



На наступному рисунку наведений результат використання функції термінального клієнта «Edit | To Notepad | Current Screen». В цьому разі в «Блокнот» переноситься вміст повідомлень віддаленого серверу, які потрапили в видну частину екрану термінала.



Іноді користувачеві не вистачає навіть і цих можливостей термінального клієнта. Наприклад, така ситуація виникає в разі коли необхідно перенести у «Блокнот» результати тривалої роботи користувача в системі, або вміст текстового файла (вихідної програми) який містить понад 150–200 рядків тексту. Хоча і є можливість системною командою `cat <ім'я_файла>` вивести на екран весь вміст файла, а потім переглядати його застосовуючи скролінг вікна, але все-таки у вікні одразу можна спостерігати лише незначну частину тексту.

В цьому разі можна скористуватися наступним пунктом меню «Edit | To Notepad | All». При цьому в «Блокнот» переноситься весь вміст досить великого текстового буфера термінального клієнта. Цілком зрозуміло, що потім все зайве, що потрапило в «Блокнот», може бути видалено, а вся потрібна інформація збережена у файлі на постійному носієві.

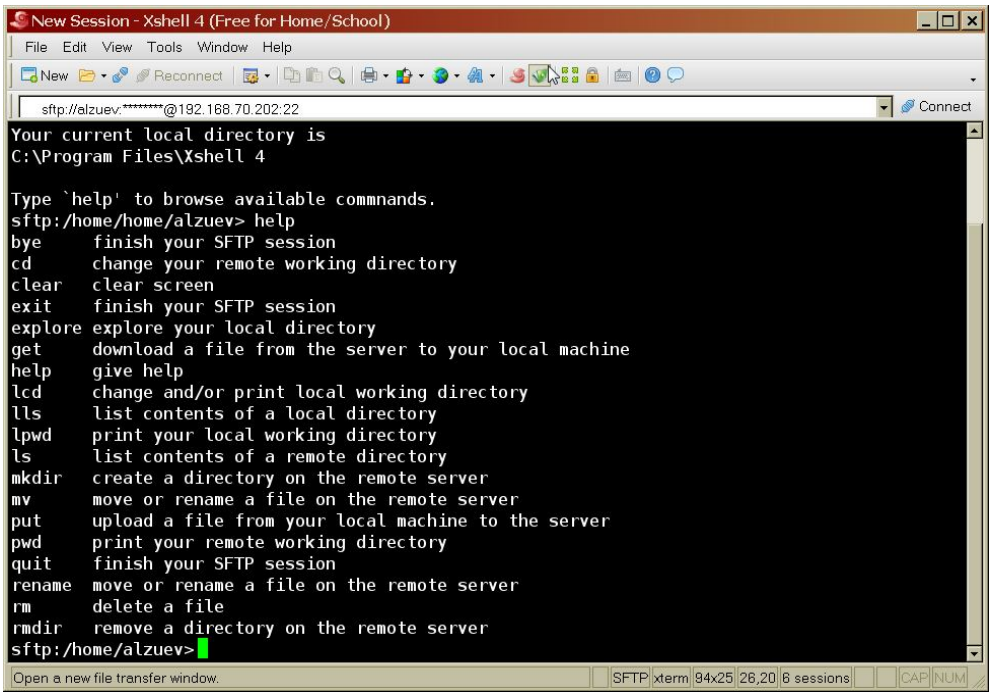
Не дивлячись навіть на досить великий текстовий буфер термінального клієнта, він все ж не безмежний і іноді може виникнути потреба збереження на локальній машині великого (або не текстового, а бінарного) файла, який розташовується на віддаленому сервері.

На цей випадок в Xshell передбачена можливість пересилання таких файлів з серверу на локальну робочу станцію і в зворотному напрямку за допомо-

гою протоколу ftp.

Така можливість забезпечується просто відкриттям мережної ftp-консолі за допомогою кнопки «New File Transfer (Ctrl+Alt+F)» на панелі інструментів термінального клієнта. Знаходячись в ftp-консолі і виконуючи доступні в ній команди, можна керувати файлами і файловими системами на обох машинах – віддаленому сервері і локальній робочій станції.

На рисунку наведений вигляд ftp-консолі з переліком всіх доступних команд з керування файлами в ній.



У даних методичних вказівках наведені тільки наймінімальні відомості необхідні для початку роботи в програмі Xshell. Більш докладно з роботою в програмі можна ознайомитися в керівництві користувача [6][1)] (англійською).

[1)] [6] Xshell 4 User Guide Secure Terminal Emualtor – Seoul: NetSarang Computer, Inc., 2011. – 157 p. UML: http://www.netsarang.com/docs/xshell4_manual.pdf (дата звернення 24.03.2019)

ДОДАТОК В

Використання програми Midnight Commander

Для управління файловою системою й для роботи з файлами в консольному режимі UNIX-сумісних операційних систем використовуються команди інтерпретатора системних команд ОС, наприклад такі, як `pwd`, `ls`, `cd`, `mv`, `mkdir`, `rmdir`, `cp`, `rm`, `cat` і т.ін. Але набагато зручніше виконувати більшу частину роботи з обслуговування файлової системи за допомогою спеціальних командних оболонок. В ОС UNIX і сумісних з нею при роботі в текстовому режимі найбільш часто використовується програма Midnight Commander [5][1)].

Midnight Commander – це програма, яка призначена для перегляду вмісту каталогів і виконання основних функцій управління файлами в UNIX-подібних операційних системах. Програма Midnight Commander (або просто `mc`) надає можливість наочно представити всі виконувані дії, полегшуючи тим самим виконання операцій з файловою системою. Програма дозволяє переглянути структуру каталогів і виконати основні операції по управлінню файловою системою. Інакше кажучи, це файловий менеджер.

В ОС типу Windows також існують аналоги цієї програми, наприклад FAR. Більш за те, Midnight Commander, як програма, розповсюджується під ліцензією GPL і перенесена в ОС Windows у вигляді файла `mc.exe`. Студенти, що мають досвід роботи в Windows з програмами FAR або Total Commander, легко зможуть працювати й з `mc`, оскільки навіть основні комбінації «гарячих клавіш» у цих програмах збігаються. Для тих же хто не знаком з цими програмами у даному розділі методичних вказівок наведений короткий опис роботи з файловим менеджером `mc`.

Необхідно сказати, що оскільки робота з програмою під час лабораторних робіт виконується в емуляторі термінала, а не в самому терміналі ОС Linux, мо-

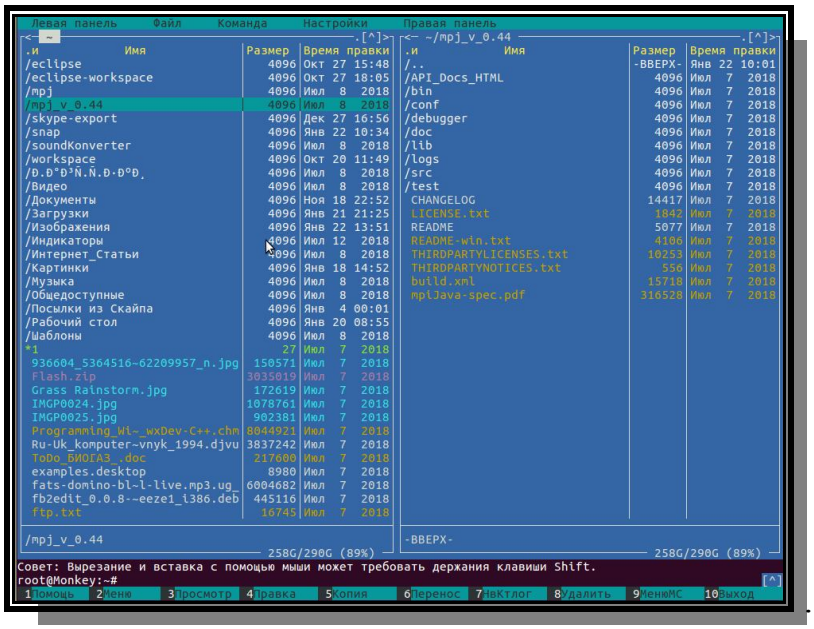
---

[1)] [5] Костромин В.А. Самоучитель Linux для пользователя. СПб.: БХВ-Петербург, 2003. 672 с.

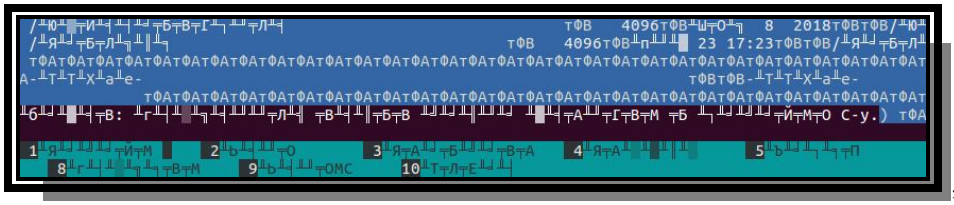
жлива деяка розбіжність у реакції програми на натискання гарячих клавіш на відміну від звичайної реакції. Це пов’язане з тим фактом, що обробка деяких гарячих клавіш перехоплюється терміналом і виконується за його правилами, наприклад, натискання клавіші <F10>, може викликати запит на закриття термінала, а не програми mc. Але таке поводження характерно тільки при використанні гарячих клавіш, робота ж по управлінню файловою системою через меню програми до таких несподіванок не веде.

Запуск програми виконується шляхом набору команди «mc» в командному рядку запрошення термінала і натискання клавіші <Enter> (див. вище у додатку Б).

У результаті виконання команди, з’являється робоче вікно програми, котре виглядає так:



Якщо ж на терміналі з’являється щось подібне до такого, то це означає,


,

що при запуску термінальної програми Xshell в її настойках було не вірно вибране кодування символів.

Для виправлення становища треба змінити кодування символів з «Default Language» на «Unicode (UTF-8)» так, як це описано у додатку Б. Після чого на клавіатурі слід двічі натиснути клавіші <Ctrl-O>.

У вірно відображеному вікні програми mc можна виділити кілька окремих структурних елементів.

Насамперед, це дві «панелі», які при роботі з Midnight Commander займають майже весь простір екрана й, найчастіше за все, відображають списки файлів двох каталогів.

Така частина, як рядок вводу команди

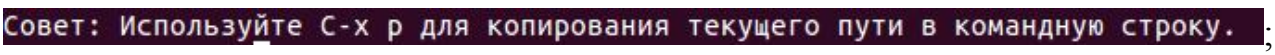
(точніше – це командний рядок поточної оболонки ОС), у якому можна ввести й запустити на виконання будь-яку команду системи, є основною й обов’язковою частиною в інтерфейсі програми.

Всі інші структурні елементи, тобто

– головне меню

;

– рядок порад програми



;

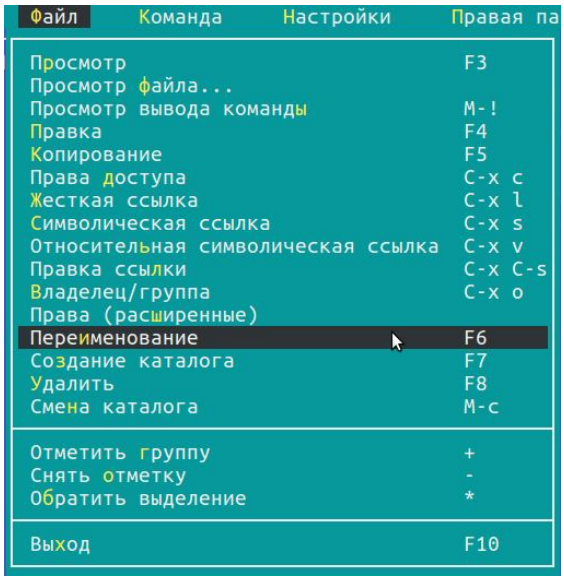
– рядок функціональних клавіш

;

за допомогою якого можна безпосередньо запускати відповідні команди, натискуючи клавіші <F1>–<F10> на клавіатурі, або клацаючи мишкою по відповідній екранній кнопці 1–10 (цей рядок можна просто вважати підказкою по використанню функціональних клавіш) можуть бути відключені для збільшення корисного простору файлових панелей при настроюванні зовнішнього вигляду програми.

При цьому доступ до рядка головного меню вибору команд можна одержати шляхом натискання клавіші <F9> або, якщо з якої-небудь причини вона не працює, шляхом послідовного натискання клавіш <Esc> і <9>, або за допомогою миші – клацанням її лівої кнопки по верхньому рядку файлера.

Використовуючи пункти основного меню Midnight Commander можна виконати будь-які доступні файлові команди. Навіть коли рядок функціональних клавіш (котрий можна розглядати просто як підказку) відключений, можливість виконання будь-якої операції з числа асоційованих з функціональними клавішами, крім <F1>, <F2> і <F9>, забезпечується через пункт меню «Файл»

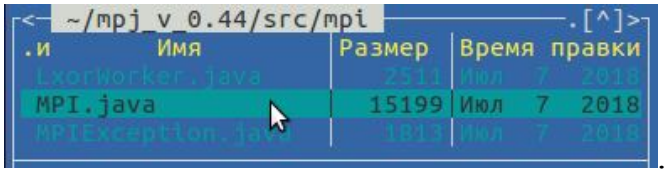


Кожна з двох панелей складається із заголовка, списку файлів якого-небудь каталогу й рядків «міні-статусу» (останній може бути відсутнім, це задається настроюваннями програми). У заголовку панелі вказується повний шлях

до каталогу, уміст якого відображається в панелі, наприклад,

а також чотири екранні кнопки – <, ., [^] і >, які використовуються для управління програмою за допомогою миші. У рядках міні-статусу відображаються деякі дані про файлову систему, наприклад, ім’я файла або каталогу, на який у цей момент указує підсвічування, розмір каталогу, вміст якого відображається в панелі, загальний розмір файлової системи й вільне місце в системі у відсотках (уміст міні-статусу задається в настроюваннях програми)

Одна з панелей є поточною (активною), про що свідчить підсвічування ім’я каталогу в заголовку панелі й підсвічування одного з її рядків

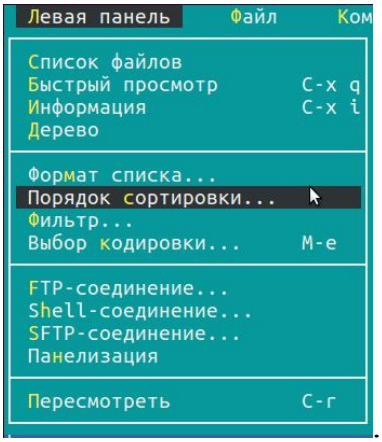


Відповідно, каталог, що відображується в активній панелі, є поточним для ОС у цей момент. Тому майже всі доступні файлові операції виконуються в цьому каталозі. При цьому каталог, який в цей час відображується у другій панелі, слугує цільовим каталогом при операціях копіювання <F5> і перенесення <F6> файлів. Файлові операції, крім команди створення нового каталогу – <F7>, виконуються над підсвіченим файлом або каталогом. Підсвічування в активній панелі можна переміщати за допомогою клавіш управління курсором.

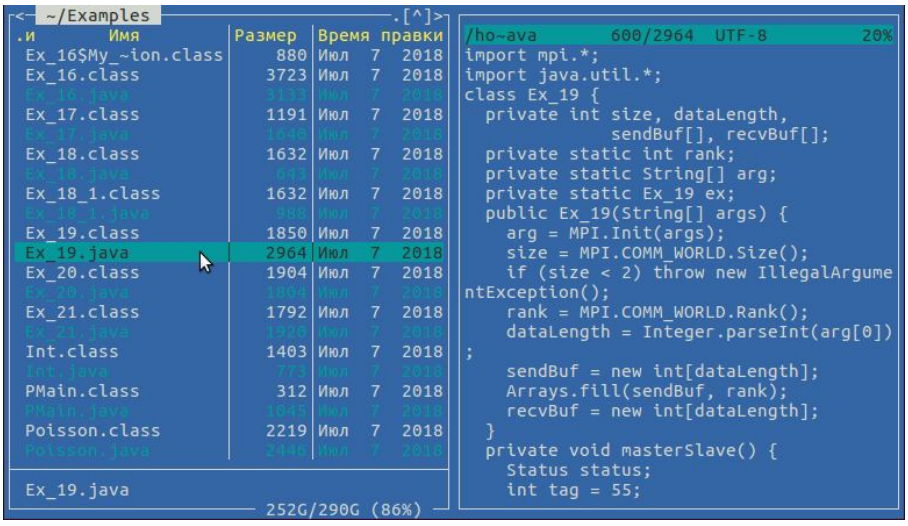
Панелі програми Midnight Commander найчастіше відображають уміст ка-

талогів файлової системи (тому іноді вони називаються панелями каталогів). Однак на панель може бути виведена й деяка інша інформація.

Вигляд панелей, у яких відображаються списки файлів і підкаталогів, може бути змінений через команди меню, що випадає, лівої й правої панелі «Левая панель» і «Правая панель» відповідно



Наприклад, переведення панелі в режим «Быстрый просмотр» дає можливість при переміщенні підсвічування в іншій панелі частково переглядати вміст файлів, на які вказує підсвічування, як це видно на рисунку

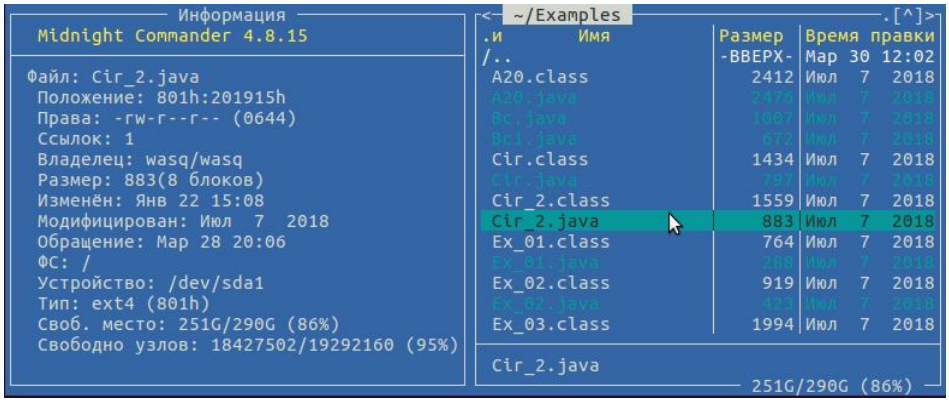


,

тим самим завжди легко можна знайти файл, котрій цікавить користувача.

У режимі «Информация» на панель виводиться докладна інформація про

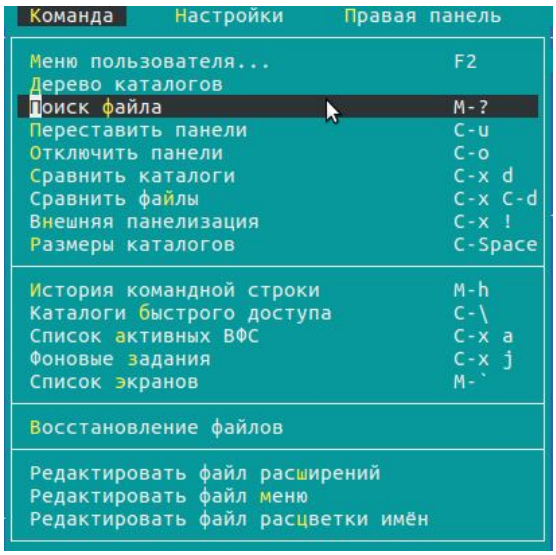
підсвічений в іншій панелі файл й про поточну файлову систему



Використовуючи пункти меню панелей, можна змінити вигляд відображення списку файлів, на наведених вище рисунках використаним є «стандартний» формат, при якому відображаються ім’я файла, його розмір і час останньої модифікації.

У меню кожної з панелей є команда «Перечитать». За цією командою оновлюється список файлів, котрий відображується у панелі, це буває корисно тоді, коли інші процеси незалежно від mc створюють або видаляють файли.

Використання пункту головного меню – «Команда» і підпунктів з його списку дозволяє виконати низку операцій з управління файловою системою, а також виконати деякі команди, що змінюють вигляд панелей Midnight Commander і відображувану в панелі інформацію



Команда «Писк файла» (гарячі клавіші <Meta-?> або <Esc><?>) викликає появу діалогового вікна, у якому необхідно вказати параметри файла для розшуку (можна використовувати стандартні символи ОС для підставлення). Завершення діалогу дозволяє знайти на диску файл із заданими параметрами.

По команді «Отключить панели» (<Ctrl-O>) показується вивід команди оболонки shell, що виконувалась останньою.

По команді «Сравнить каталоги» (<Ctrl-X><D>) порівнюється вміст каталогів, відображуваних на лівій і правій панелях.

Особливий інтерес може викликати команда «Сравнить файлы» (<Ctrl-X>< Ctrl-D>) яка дозволяє порівняти і наочно представити відмінності вмісту двох файлів, які були вибрані на двох панелях. На рисунку наведений результат такого порівняння файлів «Cir.java» на лівій панелі і «Cir_2.java» на правій.

Докладний розгляд всіх можливих команд програми виходить далеко за межі даних методичних вказівок і може служити об’єктом самостійної роботи студентів. Тим більше що при роботі із програмою Midnight Commander практично в будь-який момент можна звернутися до інтерактивної підказки, виклик якої здійснюється натисканням клавіші <F1> (<Esc><1>). Підказка організована як гіпертекст, тобто в неї зустрічаються посилання на інші розділи. Такі поси-

лання виділені блакитним фоном. Для переміщення у вікні перегляду підказки можна використовувати клавіші переміщення курсору або мишу. Крім того завжди можна скористатися системними командами «`man mc`» або «`info mc`» для одержання докладного посібника з користування програмою.

Для виконання ж лабораторних робіт цілком достатньо знати невеликий набір команд та пов'язаних з ними гарячих клавіш. Нижче наведені ці гарячі клавіші й дії, виконувані програмою при їхньому натисканні.

Клавіша або комбінація клавіш	Виконувана дія
<F1>	Виклик контекстно-залежної підказки
<F3>	Перегляд підсвіченого в активній панелі файла
<F4>	Виклик текстового редактора для підсвіченого файла
<F5>	Копіювання файла або групи файлів з поточного каталогу в каталог на другій панелі. При копіюванні одного файла можна поміняти його ім'я або вказати інший каталог
<F6>	Перенесення файла або групи відзначених файлів. Як і при копіюванні, можна поміняти ім'я файла або цільового каталогу
<F7>	Створення підкаталогу в поточному каталозі
<F8>	Видалення файла/каталогу або групи файлів
<F9>	Виклик головного меню програми
<Tab>	Зміна активної панелі
<Ctrl-O>	Показ виводу останньої команди, що виконувалась
<Alt-?>, <Esc><?>	Пошук файла
<Ctrl-X><D>	Порівняння каталогів
<Ctrl-X><Q>	Переведення неактивної панелі в режим швидкого перегляду файлів у активній

<Ctrl-X><I>	Переведення неактивної панелі в режим інформації про файли у активній
<Ctrl-X><C>	Перегляд і зміна прав доступу до файла
<Alt-P>	Переміщення на одну команду назад у переліку команд, що запускалися раніше
<Alt-N>	Переміщення на одну команду вперед в історії команд
<Ctrl-X><T>	Перенесення ім’я підсвіченого файла до командного рядка
<Ctrl-S>, <Alt-S>	Ініціалізація пошуку файла в поточному каталозі за першими символами ім’я
<Shift-F4>	Створення й редагування нового файла
<Insert>	Помітка файла для групових операцій
<Gray +>	Виділення групи файлів для наступної групової операції
<Gray –>	Зняття виділення групи файлів
<Enter>	Запуск виконуваного файла на виконання
<F10>, <Esc><Esc>	Вихід з Midnight Commander, відмова від пункту меню або діалогового вікна

При використанні комбінацій клавіш, встановленим має бути латинський регістр клавіатури, регістр великих або маленьких літер значення не має.

Коли в ОС запущений драйвер миші, програма Midnight Commander забезпечує й підтримку миші. Клацанням лівої кнопки можна перемістити підсвічування на будь-який файл кожної з панелей. Для позначки або зняття позначки будь-якого файла досить клацнути правою кнопкою миші на імені файла.

Подвійне клацання лівою кнопкою миші на імені файла означає спробу запустити файл на виконання, якщо це виконувана програма (виконувані файли позначаються символом зірочки «*» перед ім’ям файла).

Клацанням миші по функціональній кнопці можна також викликати програму, асоційовану з нею. Клацання по пункту верхнього меню викликає випадаюче підменю.

Однією з основних дій при виконанні лабораторних робіт є створення й редагування файлів вихідного коду програм і тому необхідно відзначити, що, в порівнянні з роботою через командній рядок, в Midnight Commander особливо зручно здійснювати перегляд, створення й редагування файлів, оскільки для цього він володіє відповідним інструментарієм.

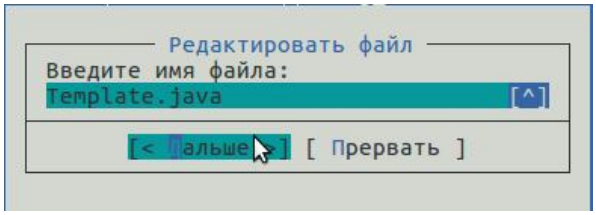
При роботі в командному рядку доводиться користуватися редактором vi або його трохи доробленими нащадками. Однак для тих користувачів, які звикли до текстових редакторів середовища Windows, ці редактори навряд-чи здадуться зручними. Причина в тім, що в них немає звичних меню й підказок і треба пам'ятати безліч клавіатурних комбінацій для уведення команд. Крім того робота в редакторі організована у вигляді двох роздільних режимів – уведення тексту й уведення команд, і новачок часто просто плутає режими. Серед користувачів навіть існує жарт: "Усе, що треба знати про редактор vi – це команда «:q»", тобто закінчити роботу й вийти з редактора.

У свою чергу, вбудований в Midnight Commander текстовий редактор McEdit – це проста у використанні програма, котра має майже звичний для більшості користувачів інтерфейс з меню й з більш-менш звичними комбінаціями управляючих клавіш.

В середовищі Midnight Commander виклик редактора для редагування вже існуючого і підсвіченого у панелі файла здійснюється простим натисканням клавіші <F4>.

Для створення ж нового вихідного файла можна скористатись стандартною командою операційної системи `touch <ім'я_файла.суфікс>`, яка вводиться у командному рядку, а потім, підсвітивши створений файл у панелі, знов-таки натиснути клавішу <F4> для відкриття вікна редактора тексту.

Але насправді достатньо лише натиснути комбінацію клавіш <Shift-F4>. При правильній настройці Midnight Commander, ця дія викликає появу діалогового віконця, в якому користувачеві пропонується ввести ім'я нового файла



після того, як користувач введе ім’я файла і натисне клавішу <Enter>, або скористується кнопкою [<Дальше>], відкривається порожнє вікно редактора для створення й редагування означеного файла.

Зовнішній вигляд вікна редактора з відкритим для редагування файлом мовою програмування Java показаний нижче

```
/home/was~_11.java   [----] 57 L:[  1+27  28/125] *(1023/4500b) 0010 0x00A  UTF-[*][X]
import mpi.*;
import java.util.*;
class Ex_11 {
  private int matrSize, size, nRows;
  private double a[][], b[][];
  private static int rank;
  private static String[] arg;
  private static Ex_11 ex;
  public Ex_11(String[] args) {
    arg = MPI.Init(args);
    rank = MPI.COMM_WORLD.Rank();
    size = MPI.COMM_WORLD.Size();
    matrSize = Integer.parseInt(arg[0]);
    if (matrSize % size != 0)
      throw new IllegalArgumentException("" + size);
    a = new double[matrSize][matrSize];
    b = new double[matrSize][matrSize];
    nRows = (matrSize - 1) / size + 1;
  }
  public static void main(String[] args) throws MPIException {
    boolean canDo = true;
    try { ex = new Ex_11(args); }
    catch(ArrayIndexOutOfBoundsException e) {
      canDo = false;
      if (rank == 0)
        System.err.printf(" %nНе вистачає параметра для" +
                          " методу main.%nСинтаксис команди" +
                          " запуску програми:%n %n\t\t" +
1Помощь 2Сох~ть 3Блок   4Замена 5Копия  6Пер~ть 7Поиск  8Удалить 9МенюМС 10Выход
```

Вбудований редактор Mcedit забезпечує виконання більшості функцій редагування, які властиві для повноекранних редакторів тексту. З його допомогою можна редагувати вихідні файли достатньо великого розміру, оскільки верхня границя для розміру файла становить 16 МіБ.

У редакторі підтримуються наступні можливості:

– різноколірне виділення синтаксису для багатьох мов програмування;

– копіювання, переміщення, видалення, вирізання й вставка блоків тексту у поточному файлі;
– видалення поточного рядка у якому розташований курсор (комбінація клавіш <Ctrl-Y>);
– скасування попередніх операцій (по комбінації клавіш <Ctrl-U>);
– меню, що випадають;
– вставка файлів;
– макро-визначення;
– пошук і заміна по регулярних виразах (тобто по шаблонах, що сформовані з використанням спеціальних символів), а також власний варіант операцій пошуку й заміни, заснований на функціях scanf – printf;
– виділення тексту комбінацією клавіш <Shift-стрілки> у стилі MS Windows;
– переключення між режимами вставки/заміни символу;
– тощо.

Редактор дуже простий і практично не потребує вивчення. Для того, щоб довідатися, які клавіші викликають виконання певних дій, досить переглянути меню, що випадають:

У вихідному стані цей рядок у вікні редактора не показується, на його місці вгорі вікна знаходиться рядок статусу, в якому вказується ім’я файла, котрий редагується, номер рядка, в яком знаходиться курсор, його позиція в рядку і ASCII код символу у позиції курсору.

а меню з’являється лише при натисканні клавіші <F9>.

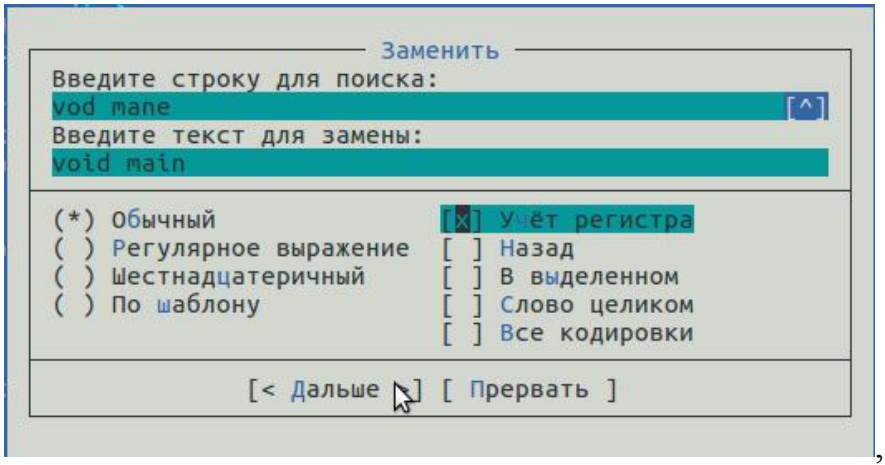
Нижче показані три основних підменю програми – «Файл», «Правка» і «Команда», котрі частіше за все використовуються при роботі в редакторі

В свою чергу пункти меню «Форматирование» і «Настройка» при написанні тексту програм практично не використовуються, тому докладно розглядати їх не має сенсу. А меню «Поиск» спеціальних пояснень не потребує оскільки є досить простим й має цілком зрозумілі пункти виконання пошуку і заміни тексту.

Скорочення управляючих комбінацій, котрі наведені в пунктах меню слід розуміти так:

– літера «M» означає клавішу <Meta> (на клавіатурах IBM це клавіша <Alt>), таким чином, позначення «M-e» означає одночасне натискання клавіш <Alt-E> (якщо не спрацьовує, то ця комбінація замінюється послідовним, а не одночасним натисканням клавіш <Esc><E>);
– літера «C» означає клавішу <Ctrl>, тобто «C-n» відповідає натисканню <Ctrl-N>;
– «S» – це клавіша <Shift>, тим самим «S-Ins» означає <Shift-Insert>;
– «F13», «F15» – це натискання <Shift> одночасно з <F3> або <F5> відповідно.

Як видно з наведених рисунків, практично кожному пункту меню відповідає або одна з функціональних клавіш <F1>–<F12>, або деяка керуюча комбінація клавіш. У редакторі, як й у Midnight Commander, функціональним клавішам <F1>–<F10> зіставлені у відповідність екранні кнопки 1–10. Натискання на функціональну клавішу або клацання мишею по екранній клавіші викликає реакцію цілком ясну з підписів на них. Так натискання клавіші <F2> викликає збереження під тим самим ім'ям файла, який у цей момент редагується. Якщо виконується редагування нового файла, то попередньо користувачеві пропонується ввести ім'я нового файла. Дія клавіші <F4> полягає у виведенні на екран діалогового вікна для здійснення пошуку й заміни тексту



,

натискання на <F7> викликає таке ж саме вікно, під назвою «Поиск» і без рядка заміни тексту.

Клавіші <F3>, <F5>, <F6> і <F8> призначені для роботи із блоками виділеного тексту, причому робота із блоками за допомогою цих клавіш трохи незвична. Так для виділення тексту необхідно встановити курсор на початок виділюваного блоку й клавішею <F3> позначити його, після чого курсор треба перемістити в кінець блоку й знов натиснути клавішу <F3>, за результатом цих дій блок виділяється і позначається блакитним кольором фона. Виділення блоку показане на наступному рисунку

```
double max = 0.0, timeStart, time, bandwidth;
do {
  timeStart = System.currentTimeMillis()/*MPI.Wtime() * 1000*/;
  for (int i = 1; i <= nTimes; i++) {
    if (rank == 0) {
      MPI.COMM_WORLD.Send(a, 0, n, MPI.DOUBLE, 1, 1);
      MPI.COMM_WORLD.Recv(a, 0, n, MPI.DOUBLE, 1, 1); }
    if (rank == 1) {
      MPI.COMM_WORLD.Recv(a, 0, n, MPI.DOUBLE, 0, 1);
      MPI.COMM_WORLD.Send(a, 0, n, MPI.DOUBLE, 0, 1); }
  }
  time = (System.currentTimeMillis()/*MPI.Wtime() * 1000*/ - timeStart) / 2 / nTime
```

Для видалення виділеного блоку достатньо натиснути клавішу <F8>.

Для копіювання або переносу блоку в інше місце тексту потрібно після виділення блоку перемістити курсор у позицію, куди буде провадитися копіювання або переміщення, і натиснути <F5> або <F6> відповідно.

Редактор так само підтримує виконання операцій з блоками в стилі редакторів MS Windows, тобто з використанням наступних комбінацій клавіш:

– <Shift-клавіші стрілок> – виділення блоку тексту;
– <Ctrl-Insert> – копіювання блоку в буфер обміну;
– <Shift-Insert> – виконує вставку останнього скопійованого в буфер блоку в позицію курсору;
– <Shift-Delete> – видаляє виділений блок тексту, запам’ятовуючи його в буфері.

При роботі з текстом слід чітко собі уявляти, що редагуємий файл знахо-

диться на віддаленому сервері, а не на локальній станції. Тому ні сам файл, ні, навіть, виділений фрагмент тексту не можна безпосередньо зберегти на локальній станції. Але в цьому разі можна скористатися засобами термінального ssh-клієнта Xshell (див. додаток Б).

Однак зараз, виділяти текст потрібно мишею і одночасно утримувати натиснутою клавішу <Shift> на клавіатурі. При цьому виділений текст має вигляд показаний на рисунку.

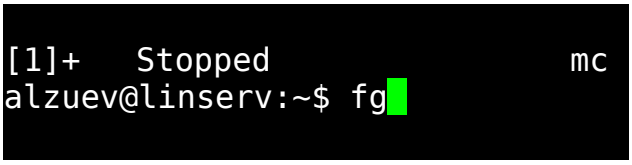
```
        int myrank = MPI.COMM_WORLD.Rank();
        int[] message = new int[len], gather = new int[len * size];
//        int gather[] = new int[len * size];
        for(int i = 0; i < len; i++) message[i] = myrank;
        if(myrank != root)
          MPI.COMM_WORLD.Send(message, 0, len, MPI.INT, root, myrank);
        else
          for(int i = 0; i < size; i++)
            {
              if (i == root) {
                k = root * len;
                for(int j = 0; j < len; j++) message[j] = myrank;
              }
              else {
                Status status = MPI.COMM_WORLD.Recv(message, 0, len, MPI.INT, i, MPI.ANY_TAG);
                k = status.source * len;
              }
              for(int j = 0; j < len; j++) gather[k + j] = message[j];
            }
        if(myrank == root) {
          System.out.print(Arrays.toString(gather));
          System.out.println(); }
        else {
          System.out.print(Arrays.toString(message));
```

В решті решт можна сказати, що робота з текстом у редакторі Mcedit достатньо проста, але іноді призначення і дія управляючих комбінацій клавіш значно відрізняється від призначення і дії комбінацій звичних при роботі в редакторах ОС Windows.

Тут особливо слід відмітити несподівану дію звичної комбінації <Ctrl-S>, її використання не веде до збереження редагуємого тексту, а лише відключає кольорове виділення синтаксису. Слід пам’ятати, що збереження файла виконується при натисканні клавіші <F2>, або екранної кнопки 2. Повторне натискання комбінації <Ctrl-S> відновлює кольорове виділення синтаксису.

Серед інших несподіваних реакцій на натискання функціональних клавіш слід відзначити комбінацію <Ctrl-Z>, яка призначена не для скасування остан-

ньої виконаної операції над текстом, наприклад, відновлення рядка тексту помилково видаленому натисканням комбінації <Ctrl-Y>**,** а лише для переміщення курсору ліворуч по тексту через одно слово. Скасування останньої здійсненої дії виконується натисканням **<**Ctrl-U**>**, а не <Ctrl-Z> як це прийнято в редакторах Windows. Використання <Ctrl-Z> іноді може призвести зовсім до несподіваної реакції – з екрана термінала зникає картинка характерна для Midnight Commander, а з'являється чорний екран приблизно з таким повідомленням:



.

В цьому разі особливо турбуватися не слід тому, що для повернення до Midnight Commander достатньо у командному рядку набрати команду «`fg`» (як на рисунку) і натиснути клавішу <Enter>, але для себе слід зробити висновок, що надалі, по змозі, треба уникати використання комбінації <Ctrl-Z>.

Для опису всіх функцій убудованого редактора, треба було б занадто багато місця. Але в цьому немає потреби, оскільки для його використання досить запам'ятати, що всі основні операції можна виконати через пункти меню, яке викликається натисканням клавіші <F9> у вікні редагування. Крім того, можна прочитати man-сторінку по команді «`man mcedit`» або «`info mcedit`».

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з дисципліни

«Технології розподілених систем та паралельних обчислень»

для студентів 4-го курсу денної форми навчання

спеціальності 122 – «Комп’ютерні науки»

(частина перша)

Укладач: Рольщиков В.Б.

Підписано до друку	Формат	Папір офсетний
Ум.друк.арк.	Тираж	Замовлення

Видавництво та друкарня

Одеський державний екологічний університет
65016, Одеса 16, вул. Львівська, 15