package cmdserv2;

/\*\*

\*

\* @author n01299554

\*/

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.File;

import java.io.OutputStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class CmdServ2 {

//Part II - c

private int clientCount = 1;

public String help(String[] commands) {

return "Available commands: help, count <text>, time, file <filename>, dict <word>";

}

//Part III - 1

public String count(String[] args) {

if (args.length == 0) {

return "Usage: count <text>";

}

String text = String.join(" ",args);

String[] words = text.split("\\s+");

return "Word count: " + words.length;

}

//Part III - 2

public String time(String[] args) {

try {

//Part III - 2c Connect to the daytime service

Socket socket = new Socket("localhost", 13);

BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

//Part III - 2c Read the result from the service

String result = reader.readLine();

//Part III - 2c Close the connection

socket.close();

return result;

} catch (IOException e) {

e.printStackTrace();

return "Error connecting to the time service.";

}

}

//Part III - 4

public String file(String[] args) {

if (args.length == 0) {

return "Usage: file <filename>";

}

String filename = args[0];

// Go to home directory

String homeDirectory = System.getProperty("user.home");

// Make the file path using the home directory

File file = new File(homeDirectory, filename);

if (!file.exists()) {

return "File does not exist.";

}

//Part III - a Check file properties

StringBuilder result = new StringBuilder();

result.append("File exists.\n");

result.append("Readable: ").append(file.canRead()).append("\n");

result.append("Writable: ").append(file.canWrite()).append("\n");

result.append("Executable: ").append(file.canExecute()).append("\n");

//Part III - b

if (file.isFile()) {

result.append("Type: Regular File\n");

//Part III - c

result.append("Size: ").append(file.length()).append(" bytes\n");

} else if (file.isDirectory()) {

result.append("Type: Directory\n");

} else {

result.append("Type: Unknown\n");

}

return result.toString();

}

//Part III - 3

public String dict(String[] args) {

if (args.length == 0) {

return "Usage: dict <word>";

}

String word = args[0];

try {

Process dictProcess = Runtime.getRuntime().exec(new String[]{"dict", word});

BufferedReader reader = new BufferedReader(new InputStreamReader(dictProcess.getInputStream()));

StringBuilder result = new StringBuilder();

String line;

while ((line = reader.readLine()) != null) {

result.append(line).append("\n");

}

return result.toString();

} catch (IOException e) {

e.printStackTrace();

return "An error occurred while executing the 'dict' command.";

}

}

public void handleClient(Socket clientSocket) {

try {

BufferedReader reader = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

OutputStream outputStream = clientSocket.getOutputStream();

//Part II - 3c Announce the thread name and the port being used by the client

String threadName = Thread.currentThread().getName();

int clientPort = clientSocket.getPort();

System.out.println("Thread " + threadName + " handling client on port " + clientPort);

outputStream.write(("Thread " + threadName + " connected to the server on port " + clientPort + "\n").getBytes());

outputStream.flush();

while (true) {

String commandLine = reader.readLine();

if (commandLine.equalsIgnoreCase("Quit")) {

System.out.println("Client disconnected.");

break;

}

//Part II - 1 Split the command line into tokens

String[] tokens = commandLine.split("\\s+");

String command = tokens[0];

String[] commandArgs = new String[tokens.length - 1];

System.arraycopy(tokens, 1, commandArgs, 0, tokens.length - 1);

//Part II - 1 Call the appropriate method

String result;

switch (command) {

case "help":

result = help(commandArgs);

break;

case "count":

result = count(commandArgs);

break;

case "time":

result = time(commandArgs);

break;

case "file":

result = file(commandArgs);

break;

case "dict":

result = dict(commandArgs);

break;

default:

result = "no such command";

}

// Send the result to the client

outputStream.write(result.getBytes());

outputStream.write('\n');

outputStream.flush();

}

clientSocket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

//Part II - 3a

public synchronized int getNextClientCount(){

return clientCount++;

}

public void startServer(int port) {

//Part II - 3b

ExecutorService threadPool = Executors.newFixedThreadPool(3);

try {

ServerSocket serverSocket = new ServerSocket(port);

System.out.println("Server is running and waiting for connections...");

while (true) {

Socket clientSocket = serverSocket.accept();

System.out.println("New client connected.");

final int currentClient =getNextClientCount();

threadPool.submit(() -> {

Thread.currentThread().setName("Client" + currentClient);

handleClient(clientSocket);

});

}

} catch (IOException e) {

e.printStackTrace();

}finally {

threadPool.shutdown();

}

}

public static void main(String[] args) {

int port = 40057;

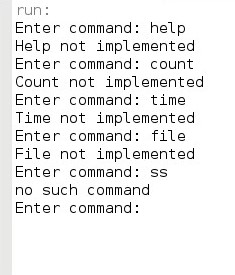
new CmdServ2().startServer(port);

}

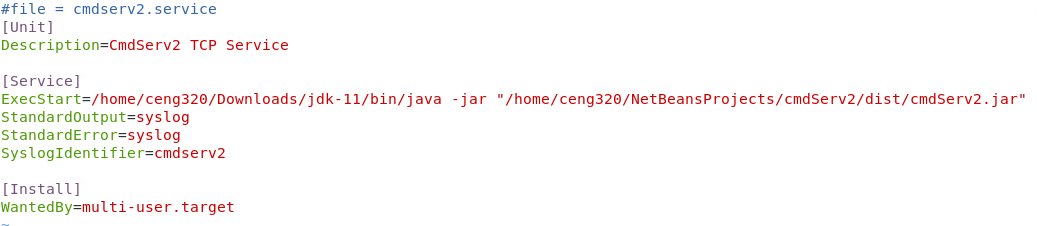
}

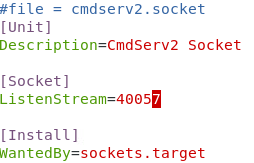
**Part II: Coding a Command Server (13 marks)**

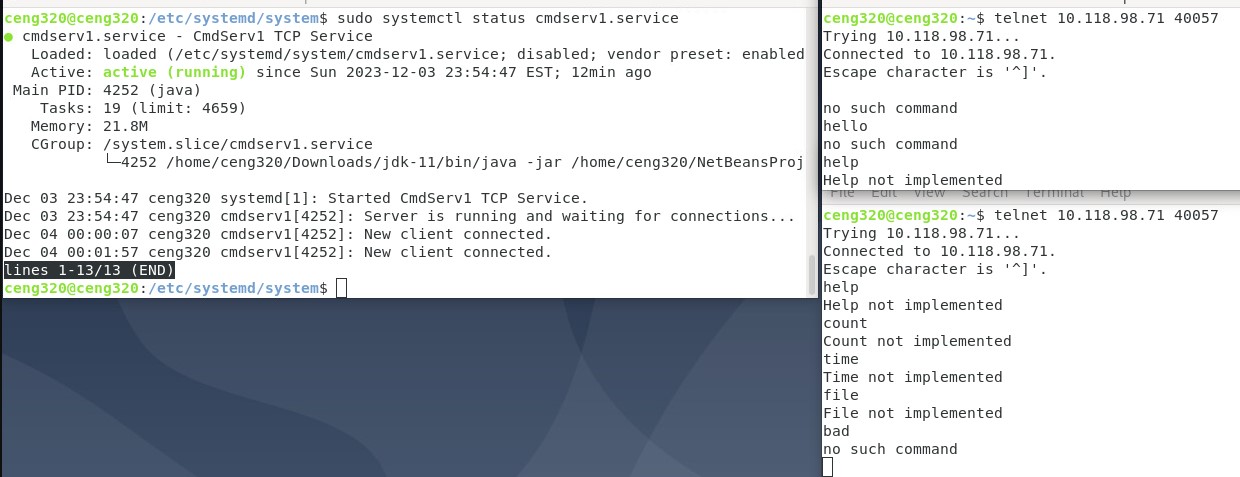
**Part II – 1:**

****

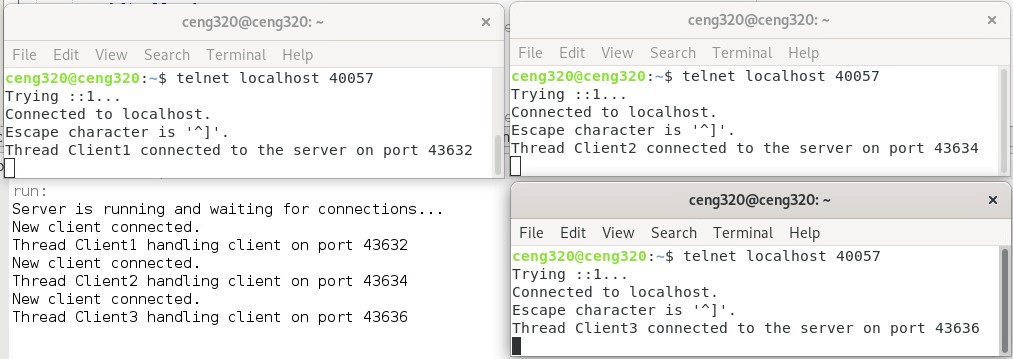
**Part II – 2**

****

****

****

**Part II – 3**



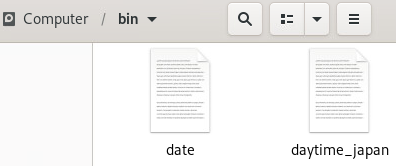
**Part III: Implementing the Command Functions (13 marks)**

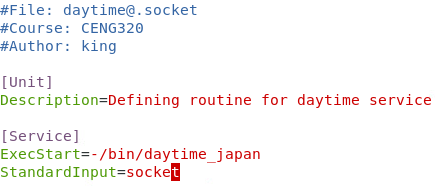
**Part III – 1**



**Part III – 2**

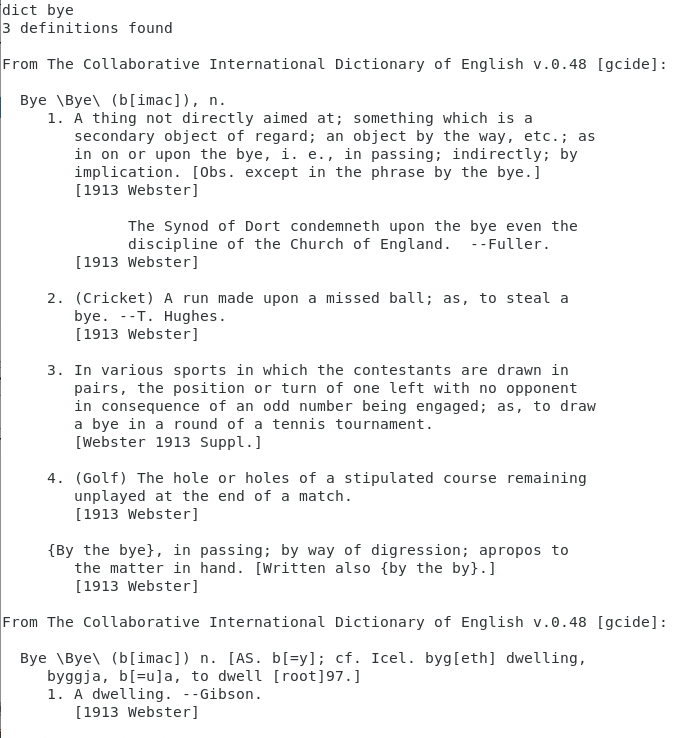




Compiled c file moved the c executable to /bin/ and changed daytime@.service to go to /bin/daytime\_japan.

**Part III – 3**



**Part III – 4**

