Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЕТ

к лабораторной работе №1

на тему

**СКРИПТЫ SHELL**

Студент А. В. Скворцов

Преподаватель Н. Ю. Гриценко

Минск 2024

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 5](#_Toc146752070)

[Заключение 6](#_Toc146752071)

[Список использованных источников 7](#_Toc146752072)

[Приложение А (обязательное) Листинг кода 8](#_Toc146752073)

1. **ЦЕЛЬ РАБОТЫ**

Изучение элементов и конструкций скриптов shell: переменных, параметров, ветвлений, циклов, вычислений, команд shell и вызовов внешних программ (shell, sed, awk, wget, различные фильтры и т.д.) для решения достаточно сложной задачи, имеющей практическое значение, а также принципов интеграции Unix-программ скриптами shell.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Программирование на языке интерпретатора shell приобретает все большую популярность по мере утверждения Linux в качестве удобной в работе и отказоустойчивой операционной системы. Трудно оценить, какое количество пользователей работают с Linux. Эта операционная система распространяется бесплатно, хотя многие компании разрабатывают ее коммерческие варианты. Кроме того, несмотря на сделанные несколько лет назад неутешительные прогнозы специалистов относительно будущего UNIX, данная ОС также не теряет популярности, и число ее приверженцев продолжает расти [1].

Командный язык shell (в переводе – раковина, скорлупа) фактически есть язык программирования очень высокого уровня. На этом языке пользователь осуществляет управление компьютером [2].

В Linux автоматизация процессов в значительной степени решается исполняемыми скриптами в shell оболочке. Это подразумевает под собой создание файла, содержащего ряд команд, которые исполняются последовательно.

Bash-скрипт — это файл, содержащий последовательность команд, которые выполняются программой bash построчно. Он позволяет выполнять ряд действий, таких как переход к определенному каталогу, создание папки и запуск процесса с помощью командной строки.

Термины «оболочка» и «bash» взаимозаменяемы. Но между ними есть тонкая разница. Термин «оболочка» относится к программе, которая предоставляет интерфейс командной строки для взаимодействия с операционной системой. Bash (Bourne Again SHell) является одной из наиболее часто используемых оболочек Unix/Linux и является оболочкой по умолчанию во многих дистрибутивах Linux [3].

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате работы было создано два приложения: сервер и клиент.

Реализация серверного приложения (рисунок 1):

1. Создан серверный сокет, привязанный к определенному порту, ожидающий подключений от клиентов.
2. Сокеты клиентов сохраняются для последующей рассылки сообщений. Также сообщения всех клиентов выводятся в консоль.
3. Реализован многопоточный механизм обработки сообщений клиентов.

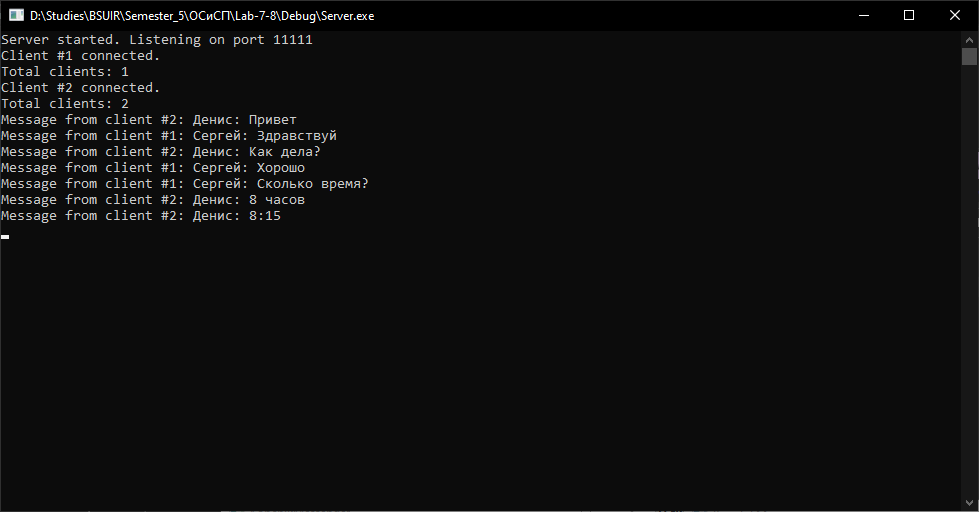


Рисунок 1 – Окно сервера

Реализация клиентского приложения (рисунок 2):

1. Создано графическое окно с элементами управления для ввода имени и сообщения, отправки сообщения, и окна для отображения сообщений.
2. Выполнено подключение к серверу по указанному *IP*-адресу и порту.
3. Реализована отправка сообщений на сервер.
4. Реализован многопоточный механизм для асинхронного приема сообщений от сервера.

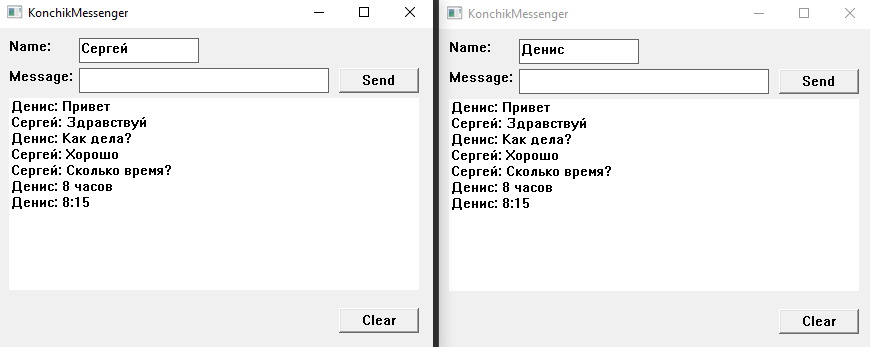


Рисунок 2 – Два запущенных клиента

# ЗАКЛЮЧЕНИЕ

Лабораторная работа была выполнена с целью изучения теоретических основ сетевого программирования, стандартных сетевых протоколов (*TCP* и *UDP*), а также основных функций и структур, предоставляемых *Windows* для работы с сокетами. В результате работы были разработаны клиентское и серверное приложения, обеспечивающие взаимодействие через сетевые сокеты. Лабораторная работа успешно достигла поставленных целей, предоставив практические навыки создания, настройки и использования сокетов в среде *Windows*. Полученные знания служат основой для дальнейшего изучения и разработки сетевых приложений, а также обеспечивают понимание принципов работы сетевых протоколов и сокетов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Тейнсли Д. Linux и UNIX: программирование в shell. Руководство разработчика: Пер.. с англ. – СПб: «БХВ-Петерберг», 2001. – 464 с.
2. ПРОГРАММИРОВАНИЕ НА shell (UNIX) [Электронный ресурс]. – Режим доступа: https://www.opennet.ru/docs/shell/shell.html. – Дата доступа: 05.02.2024.
3. Работа с командной строкой Linux & Shell [Электронный ресурс]. – Режим доступа: https://medium.com/@kkonstantinov/bash-скрипты-4c6be424edae. – Дата доступа: 06.02.2024.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – lab1.sh

#!/bin/bash

usage() {

echo "Usage: $0 [-r <regex>|-n <filename>] <dir\_path>"

exit 1

}

if [ $# -lt 2 ]; then

usage

fi

pattern=""

filename=""

path=""

while getopts ":r:n:" opt; do

case $opt in

r)

pattern="$OPTARG"

;;

n)

filename="$OPTARG"

;;

\?)

echo "Invalid option: -$OPTARG" >&2

usage

;;

esac

done

shift $((OPTIND - 1))

path="$1"

if [ -z "$pattern" ] && [ -z "$filename" ]; then

echo "Please specify either a regex pattern (-r) or a filename (-n)."

usage

fi

if [ ! -d "$path" ]; then

echo "No such file or directory."

exit 1

fi

if [ -e "output.txt" ]; then

> output.txt

fi

calculate\_checksum() {

local file="$1"

checksum=$(md5sum "$1" | awk '{print $1}')

echo "$checksum"

}

traverse\_directory() {

local dir="$1"

for file in "$dir"/\*; do

if [ -f "$file" ]; then

if [ -n "$pattern" ]; then

if [[ $(basename "$file") =~ $pattern ]]; then

checksum=$(calculate\_checksum "$file")

echo "$file founded, checksum: $checksum"

echo "$(basename "$file, checksum: $checksum")" >> output.txt

fi

elif [ -n "$filename" ]; then

if [ "$(basename "$file")" = "$filename" ]; then

checksum=$(calculate\_checksum "$file")

echo "$file founded, checksum: $checksum"

echo "$(basename "$file, checksum: $checksum")" >> output.txt

fi

fi

elif [ -d "$file" ]; then

traverse\_directory "$file"

fi

done

}

traverse\_directory "$path"

echo "Search completed! You can check results in output.txt."

echo "Enter file name, which you want to cat: "

read filename

if [ -f "$filename" ]; then

clear

lineNumber=1

while IFS= read -r line; do

printf "%-5s %s\n" "$lineNumber:" "$line"

lineNumber=$((lineNumber+1))

done < "$filename"

else

echo "$filename: no such file or directory."

fi

exit 0