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»

Выполнил             Е. А. Киселёва

Проверил                          Н. Ю. Гриценко

Минск 2024

**СОДЕРЖАНИЕ**

[1 Постановка задачи 3](#_Toc157722973)

[2 Краткие теоретические сведения 4](#_Toc157722974)

[3 Результаты выполнения лабораторной работы 6](#_Toc157722975)

[Выводы 7](#_Toc157722976)

[Список использованных источников 8](#_Toc157722977)

[Приложение А (обязательное) Листинг исходного кода 9](#_Toc157722978)

# **1 ПОСТАНОВКА ЗАДАЧИ**

Целью выполнения данной лабораторной работы является изучение элементов и конструкций shell, а также принципов интеграции Unix-программ скриптами shell. Кроме того, необходимо реализовать скрипт для оболочки bash, который реализует небольшие текстовые часы, меняющие   
свое расположение на ограниченной области с периодичностью в пять секунд.

# **2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Командная оболочка обеспечивает взаимодействие между пользователем и средой операционной системы. Она является специализированным программным продуктом, которые обеспечивает выполнение команд и получение результатов их выполнения. Примером оболочки может быть, например, оболочка bash операционных систем Unix/Linux.[1]

Все оболочки имеют схожие функции и свойства, в соответствии с тем, что они должны выполнять команды пользователя и отображать результаты их выполнения. К функциям оболочки относятся:

– интерпретация командной строки;

– доступ к командам и результатам их выполнения;

– поддержка переменных;

– поддержка специальных символов;

– поддержка зарезервированных слов;

– обработка файлов;

– обработка операций ввода и вывода.

Скрипты shell – это текстовые файлы, содержащие команды Unix, которые интерпретируются оболочкой операционной системы.   
Они используются для автоматизации задач, управления системными ресурсами и выполнения различных операций. Вместо того, чтобы вручную вводить команды, используются shell-скрипты. Достоинство shell-скриптов   
в том, что в одном скрипте можно объединить большое количество стандартных задач, что обеспечивает экономию времени и сокращение количества возможных ошибок при многократном выполнении однотипных задач.[2]

Shell-скрипт начинается с комбинации символов, которая сообщает,   
что это shell-скрипт. Это символы «#» и «!», после которых указывается оболочка shell, в которой должен быть выполнен скрипт, например bash.

Чтобы shell-скрипт можно было запустить, он должен быть исполняемым файлом. При помощи инструмента chmod, можно указать,   
что файл является исполняемым, то есть его содержимое может быть запущено как shell-скрипт.

Для выполнения данной лабораторной работы были использованы следующие сведения и концепции:

1 Скрипты bash: скрипты bash являются подмножеством скриптов shell, предоставляющим большие возможности для автоматизации задач.

2 Циклы: в shell-скрипте определены две переменные «ROWS»   
и «COLUMNS», которые задают размеры области, в которой появляются текстовые часы.

3 Функции: в shell-скрипте определена функция «main»,   
которая содержит основную логику программы.

4 Циклы: в shell-скрипте используется бесконечный цикл «while true», который выполняет основную программу в течение неопределенного времени.

5 Арифметические операции: в shell-скрипте используются арифметические операции для генерации случайных координат «x» и «y»   
в пределах заданных размеров области, в которой появляются текстовые часы.

6 Встроенные команды: в shell-скрипте используются встроенные команды Unix, такие как «date», «clear», «tput», «sleep», «echo» для работы   
с датой и временем, очистки экрана, управления курсором терминала   
и задержки выполнения, а также для управления выводом соответственно.

7 Случайные числа: для генерации случайных координат используется встроенная команд «RANDOM».

8 Форматирование вывода: для форматирования строки времени используется команда «date» с опцией «%T», чтобы получить текущее время в необходимом формате.

Таким образом, при использовании всех вышеперечисленных концепций был разработан shell-скрипт, который реализует небольшие текстовые часы, меняющие свое расположение на ограниченной области терминала периодичностью в одну секунду.[3]

# **3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ             РАБОТЫ**

В ходе выполнения лабораторной работы был разработан bash-скрипт, который реализует текстовые часы, меняющие свое расположение на области командной строки с периодичностью в пять секунд. Результат работы bash-скрипта представлен на рисунке 3.1.

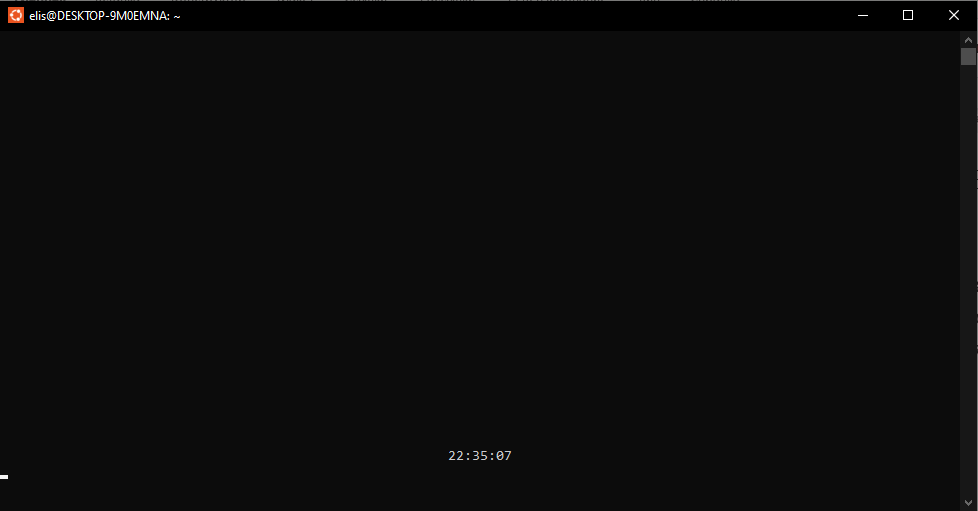


Рисунок 3.1 – Результат работы bash-скрипта

Промежуток времени, через который часы меняют расположение, можно изменить только внося изменения в сам bash-скрипт.

В результате получился рабочий скрипт, при запуске которого в командной строке отображаются текстовые часы, которые меняют свое расположение каждые пять секунд.

# **ВЫВОДЫ**

В ходе лабораторной работы были изучены элементы и конструкции shell, принципы интеграции Unix-программ скриптами shell, также был разработан скрипт для оболочки bash, который реализует текстовые часы, меняющие свое расположение на ограниченной области периодичностью в одну секунду.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Оболочка пользователя в Linux [Электронный ресурс]. – Режим доступа: https://white55.ru/shells.html. – Дата доступа: 03.02.2024

[2] Shell скрипты [Электронный ресурс]. – Режим доступа: <https://support.apple.com/ru-ru/guide/terminal/>. – Дата доступа: 03.02.2024

[3] Bash tutorials [Электронный ресурс]. – Режим доступа:   
https://linuxconfig.org/bash-scripting-tutorial. – Дата доступа: 03.02.2024.

## **ПРИЛОЖЕНИЕ А**

## **(обязательное)**

## **Листинг исходного кода**

Листинг 1 – Программный код разработанного приложения

#!/bin/bash

print\_time() {

local time=$(date +%T)

tput cup $1 $2

echo "$time"

}

random\_number() {

seed=$(date +%s)

multiplier=1103515245

increment=12345

modulus=$((2\*\*31))

seed=$((($multiplier \* $seed + $increment) % $modulus))

echo $seed

}

max\_x=$(tput cols)

max\_y=$(tput lines)

while true; do

x=$(($(random\_number) % max\_x))

y=$(($(random\_number) % max\_y))

clear

print\_time $y $x

sleep 5

done