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

ОТЧЁТ

к лабораторной работе №1

на тему

**СКРИПТЫ SHELL**

Выполнил: студент гр.253505 Сенько Н.С.

Проверил: ассистент кафедры информатики Гриценко Н.Ю.

Минск 2025

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc178636784)

[2 Теоритические сведения 4](#_Toc178636785)

[3 Инструментальная языковая среда 5](#_Toc178636786)

[4 Результат выполнения программы 6](#_Toc178636787)

[Заключение 7](#_Toc178636788)

[Список использованных источников 8](#_Toc178636789)

Приложение А [(обязательное) Исходный код программного продукта 9](#_Toc178636791)

# 1 ЦЕЛЬ РАБОТЫ

Целью данной работы является детальное и всестороннее изучение принципов написания скриптов на языке *shell* для решения прикладных задач повышенной сложности, а также формирование навыков эффективной работы с инструментами *Unix/Linux*-среды. В процессе работы планируется изучить основные элементы скриптов, включая использование переменных и параметров для управления данными, конструкции ветвления и циклов для реализации логики программ, встроенные средства выполнения арифметических и логических операций, а также способы вызова внешних программ и обработки их результатов. Особое внимание уделяется применению таких инструментов, как *sed* и *awk* для обработки текстовых данных, фильтров для трансформации потоков ввода/вывода и программ, таких как *wget*, для взаимодействия с внешними источниками данных.

Кроме того, работа нацелена на освоение методов интеграции *Unix*-программ в единые сценарии с использованием средств *shell*, что позволяет автоматизировать выполнение сложных последовательностей операций. Это включает настройку и обработку входных данных, управление потоками ошибок, выполнение фоновых процессов и создание цепочек команд с использованием пайпов (|) и перенаправления (>, <).

Цель заключается не только в освоении технических аспектов написания *shell*-скриптов, но и в развитии понимания общей архитектуры *Unix*-системы и её философии "малых инструментов", где каждая программа выполняет одну задачу максимально эффективно. Практическое значение работы заключается в том, чтобы научиться создавать скрипты, способные решать реальные задачи, такие как автоматизация рутинных операций, обработка больших объемов данных, сбор информации из различных источников, обеспечение мониторинга и управления системой.

Итогом работы должно стать не только владение конкретными инструментами и технологиями, но и умение применять их для разработки гибких, масштабируемых и надежных решений, обеспечивающих взаимодействие компонентов операционной системы на высоком уровне.

# 2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

*Shell* (оболочка) является основным интерфейсом взаимодействия пользователя с операционной системой *Unix/Linux*, обеспечивая доступ к её функциональности через командную строку. Это мощный инструмент для управления процессами, файловой системой, обработки данных и автоматизации задач. *Shell* принимает команды от пользователя, интерпретирует их и передает операционной системе для выполнения, что позволяет выполнять как простые команды, так и сложные сценарии, написанные в виде скриптов.

Одной из ключевых особенностей *shell* является поддержка переменных, которые могут использоваться для хранения данных, управления их передачей между командами и динамического изменения поведения скриптов. Системные и пользовательские переменные играют важную роль в настройке среды выполнения, а механизмы параметризации позволяют передавать аргументы скриптам и упрощают их переиспользование.

*Shell* поддерживает конструкции ветвления (*if*, *case*) и циклов (*for*, *while*, *until*), что позволяет реализовывать сложную логику выполнения команд. Эти конструкции делают скрипты гибкими и адаптируемыми под различные условия выполнения. Команды и их параметры могут комбинироваться с использованием пайпов (|) и перенаправления потоков ввода, вывода и ошибок (>, <, 2>), что облегчает обработку данных и создание эффективных цепочек операций.

Важной частью работы с *shell* является использование встроенных команд и внешних программ, таких как *sed*, *awk*, фильтры (*grep, sort, uniq*), а также утилиты для работы с сетью, например, *wget* или *curl*. Эти инструменты позволяют обрабатывать текстовые и бинарные данные, выполнять фильтрацию, сортировку, поиск и преобразование информации, а также взаимодействовать с внешними источниками данных.

*Shell* предоставляет средства для выполнения арифметических и логических операций, обработки строк, регулярных выражений и выполнения фоновых задач. Скрипты могут использовать сигналы и обработчики событий для управления процессами и ресурсами. Также поддерживаются средства отладки и проверки выполнения команд, что важно для создания надежных и безопасных решений.

Одним из ключевых принципов работы с *shell* является философия *Unix*: создание простых, но эффективных инструментов, которые могут комбинироваться друг с другом. Благодаря этому *shell* становится мощным средством интеграции различных программ в единую автоматизированную систему. Использование таких возможностей, как функции, модули и повторное использование кода, позволяет разработчику создавать масштабируемые решения.

# 3 ИНСТРУМЕНТАЛЬНАЯ ЯЗЫКОВАЯ СРЕДА

Для разработки и выполнения скриптов на языке *shell* в рамках данной работы использовалась операционная система *Arch Linux*. *Arch Linux* известен своей философией "*KISS*" (*Keep It Simple, Stupid*), что делает его идеальным выбором для опытных пользователей, желающих настроить свою систему под конкретные нужды. Arch предоставляет пользователям полный контроль над установленными пакетами и конфигурациями, что позволяет создавать легковесные и оптимизированные окружения для разработки *shell*-скриптов. Благодаря системе управления пакетами pacman и обширному репозиторию *AUR* (*Arch User Repository*), пользователи могут легко устанавливать необходимые инструменты и библиотеки.

Для редактирования команд и разработки скриптов использовался текстовый редактор ***Micro***. *Micro* — это современный текстовый редактор, который сочетает в себе простоту использования и мощные функции. Он поддерживает подсветку синтаксиса для различных языков программирования, включая *Bash*, что делает его удобным для написания и редактирования скриптов. Micro также предлагает функции автодополнения, что ускоряет процесс написания кода и снижает вероятность ошибок.

Одной из ключевых особенностей *Micro* является его легкость и возможность работы в терминале, что позволяет пользователям быстро открывать и редактировать файлы без необходимости переключаться между различными приложениями. Это создает удобный рабочий процесс, который позволяет сосредоточиться на написании и тестировании скриптов в одном интерфейсе.

Таким образом, инструментальная среда разработки, включающая *Arch Linux* и текстовый редактор *Micro*, обеспечила высокую гибкость и эффективность работы над проектом. Пользователи могут настраивать свою среду под свои нужды, а мощные функции *Micro* делают процесс разработки более удобным и продуктивным.

## 4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ

При запуске скрипта в случайном месте консоли возникают часы, как на рисунке 4.1 [1].

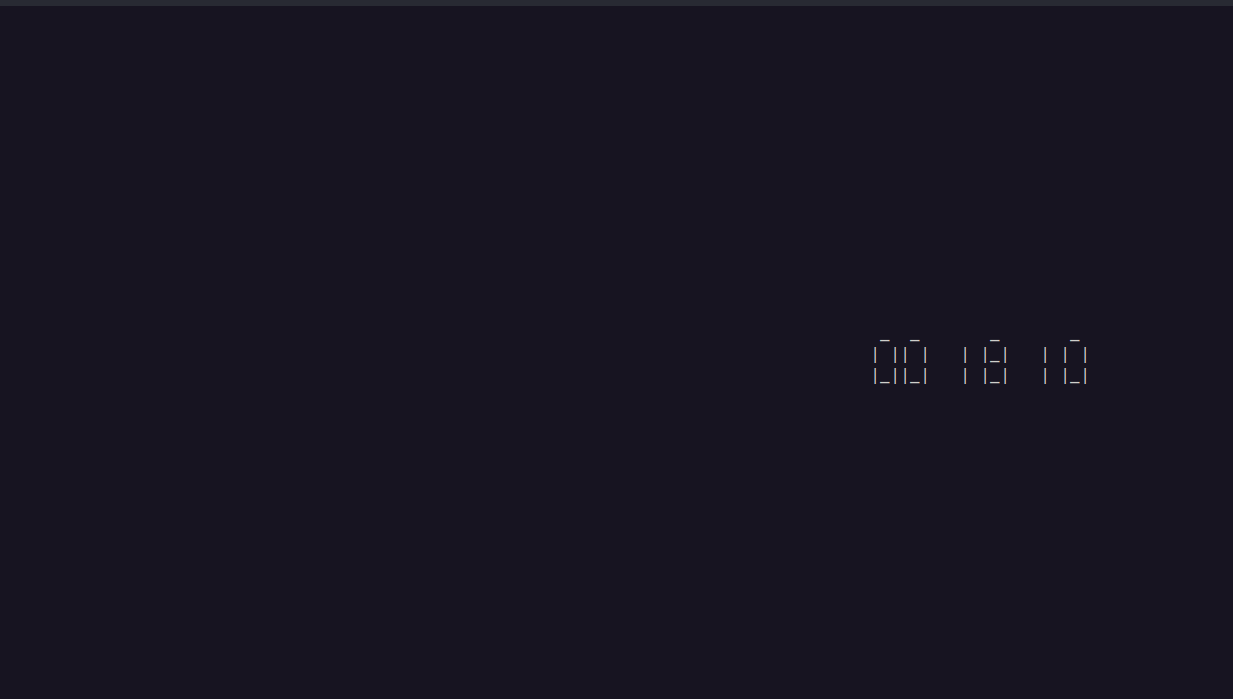


Рисунок 4.1 – Результат запуска скрипта

Каждые 5 секунд они меняют свое расположение относительно текущего окна консоли.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения лабораторной работы был разработан консольный хранитель экрана (*screensaver*), представляющий собой большие текстовые часы, которые периодически меняют своё положение на экране и цвет. Работа позволила углубить знания о программировании консольных приложений, а также освоить принципы генерации случайных чисел и работы с временными интервалами.

Основой для программы стал мультипликативный алгоритм генерации случайных чисел, инициализируемый текущим временем системы, что обеспечило уникальность начальных координат часов при каждом запуске. Алгоритм был выбран за его простоту и эффективность при целочисленных вычислениях. Периодическое обновление координат через фиксированные промежутки времени (5 секунд) продемонстрировало подходы к управлению временем в программах, а смена цвета текста каждую секунду добавила элемент динамики и визуального разнообразия.

Кроме того, разработка программы способствовала укреплению навыков работы с функциями системного времени, управления выводом на экран в текстовом режиме и использования базовых методов работы с цветами. Особое внимание было уделено правильной организации циклов и управления задержками для синхронизации обновления экрана.

Итогом работы стало создание функционального приложения, которое может быть расширено и модифицировано для выполнения дополнительных задач. Лабораторная работа продемонстрировала значимость системного подхода к решению задач и умение интегрировать математические и программные методы для реализации практических приложений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] *The Linux command line for beginners* Электронный ресурс. Режим доступа: [https://ubuntu.com/tutorials/command-line-for-beginners#1-overview](https://ubuntu.com/tutorials/command-line-for-beginners" \l "1-overview)

# ПРИЛОЖЕНИЕ А

# (обязательное)

# Исходный код программного продукта

#!/bin/bash

clear\_screen() {

echo -e "\033[2J"

echo -e "\033[H"

}

hide\_cursor() {

echo -e "\033[?25l"

}

declare -A digits

digits[0,0]=" \_ "

digits[0,1]="| |"

digits[0,2]="|\_|"

digits[1,0]=" "

digits[1,1]=" | "

digits[1,2]=" | "

digits[2,0]=" \_ "

digits[2,1]=" \_|"

digits[2,2]="|\_ "

digits[3,0]=" \_ "

digits[3,1]=" \_|"

digits[3,2]=" \_|"

digits[4,0]=" "

digits[4,1]="|\_|"

digits[4,2]=" |"

digits[5,0]=" \_ "

digits[5,1]="|\_ "

digits[5,2]=" \_|"

digits[6,0]=" \_ "

digits[6,1]="|\_ "

digits[6,2]="|\_|"

digits[7,0]=" \_ "

digits[7,1]=" |"

digits[7,2]=" |"

digits[8,0]=" \_ "

digits[8,1]="|\_|"

digits[8,2]="|\_|"

digits[9,0]=" \_ "

digits[9,1]="|\_|"

digits[9,2]=" \_|"

a=45245

m=24563425657

X=$(date +%s)

random() {

X=$(( (a \* X) % m))

}

get\_random\_position() {

local rows=$(tput lines)

local cols=$(tput cols)

local max\_row=$((rows - 4))

local max\_col=$((cols - 22))

echo $(( $X % max\_row + 1)) $(( $X % max\_col + 1))

}

display\_time() {

local time\_string=$(date +"%H%M%S")

clear\_screen

line0=""

line1=""

line2=""

for ((i=0; i<${#time\_string}; i++)); do

digit=${time\_string:i:1}

line0+="${digits[$digit,0]}"

line1+="${digits[$digit,1]}"

line2+="${digits[$digit,2]}"

if [[ $(($i % 2)) == 1 ]]; then

line0+=" "

line1+=" "

line2+=" "

fi

done

echo -e "\033[$1;$2H$line0"

echo -e "\033[$(( $1 + 1 ));$2H$line1"

echo -e "\033[$(( $1 + 2 ));$2H$line2"

}

hide\_cursor

while true; do

random

read row col < <(get\_random\_position)

for ((j=0; j<5; j++)); do

display\_time $row $col

sleep 1

done

done