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 Ющук И.А.

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 *Linux* *Ubuntu*. *Ubuntu* предоставляет удобную среду для работы с командной строкой, включая встроенный интерпретатор *shell* (обычно *Bash*), а также богатый набор инструментов, необходимых для автоматизации задач и обработки данных. Открытая архитектура системы и широкая поддержка программного обеспечения делают *Ubuntu* идеальным выбором для разработки *shell*-скриптов.

Для редактирования команд и разработки скриптов использовалась современная интегрированная среда разработки *Visual* *Studio* *Code* (*VS* *Code*). Этот редактор кода предоставляет множество полезных функций, таких как подсветка синтаксиса, автодополнение команд, встроенный терминал и поддержка расширений. Расширения для работы с *Bash* и *shell*-скриптами, такие как "*ShellCheck*" для проверки кода на наличие ошибок и рекомендаций по оптимизации, позволяют повысить качество написанных скриптов.

Встроенный терминал *VS* *Code* был использован для непосредственного выполнения команд и тестирования скриптов, что обеспечивает удобный рабочий процесс, объединяющий написание кода, выполнение и отладку в одной среде. Такой подход ускоряет процесс разработки и упрощает управление проектом, позволяя сосредоточиться на решении поставленных задач.

Таким образом, инструментальная среда разработки, включающая *Linux* *Ubuntu* и *Visual* *Studio* *Code*, обеспечила удобство, гибкость и высокую эффективность работы над проектом.

## 4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ

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

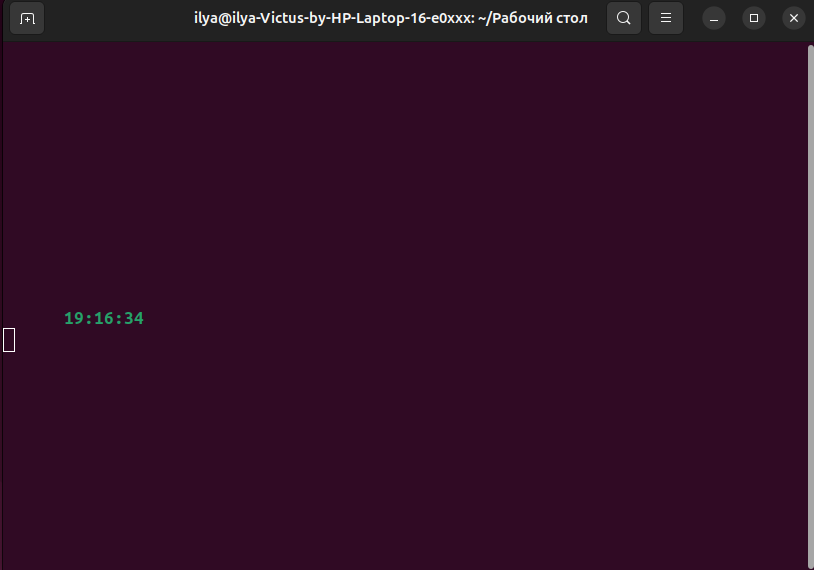


Рисунок 4.1 – Результат запуска скрипта

Каждую секунду цифры меняют цвет, также каждые 10 секунд опи меняют свое расположение относительно текущего окна консоли, рисунок 4.2.

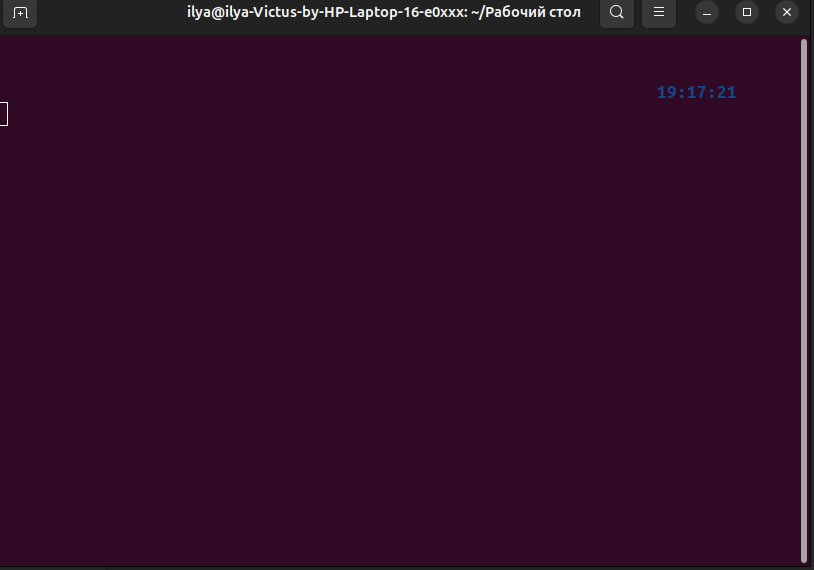


Рисунок 4.2 – Изменение цвета и расположение часов

**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 (10 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

generate\_random() {

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

echo $seed

}

color\_code=30

seed=$(date +%s)

multiplier=1664525

increment=1013904223

modulus=4294967296

refresh\_interval=5

update\_count=0

draw\_clock() {

clear

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

local term\_width=$(tput cols)

local term\_height=$(tput lines)

local rand\_x=$(($(generate\_random) % term\_width))

local rand\_y=$(($(generate\_random) % term\_height))

local pos\_x=$((rand\_x % (term\_width - 10)))

local pos\_y=$((rand\_y % term\_height))

tput cup $pos\_y $pos\_x

printf "\e[${color\_code}m%s\e[1m\n" "$time\_now"

}

while :; do

if ((update\_count >= refresh\_interval)); then

seed=$(date +%s)

update\_count=0

fi

draw\_clock

((update\_count++))

(( color\_code=$((30 + (color\_code + 1) % 8))))

sleep 1

done