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 (УГЛУБЛЕННАЯ ВЕРСИЯ)**

Выполнил: студент гр.253504 Казакевич Г.С.

Проверил: ассистент кафедры информатики Гриценко Н.Ю.

Минск 2025

**СОДЕРЖАНИЕ**

[1 Формулировка задачи 3](#_Toc183307053)

[2 Краткие теоритические сведения 4](#_Toc183307054)

[3 Описание функций программы 5](#_Toc183307055)

[Заключение 7](#_Toc183307056)

[Список использованных источников 8](#_Toc183307057)

[Приложение А (обязательное) исходный код программы 9](#_Toc183307058)

# **1 ФОРМУЛИРОВКА ЗАДАЧИ**

В данной лабораторной работе необходимо изучить и освоить основные элементы и конструкции, используемые в shell-скриптах, а также получить практические навыки работы с командным интерпретатором Bash. Это включает работу с переменными, параметрами, операторами ветвления, циклами, выполнением арифметических вычислений и использованием команд оболочки Linux.

Целью работы является не только написание конкретного скрипта, но и понимание того, как различные команды и инструменты Unix могут быть объединены в единое решение. Bash-скрипты представляют собой мощный инструмент автоматизации, позволяющий создавать гибкие и эффективные программы для решения повседневных задач.

В рамках этой лабораторной работы требуется реализовать хранитель экрана (screensaver) для консоли в виде больших текстовых часов, которые будут менять свое положение на экране. Для перемещения часов нужно разработать свой генератор случайных чисел — например, на основе мультипликативного алгоритма или другого целочисленного метода. Инициализация генератора должна производиться с использованием текущего времени, а его обновление должно происходить через фиксированные промежутки времени, такие как 10 секунд.

В процессе выполнения работы потребуется разобраться с организацией вывода информации в консоль, управлением позиционированием элементов на экране и использованием временных меток. Реализация данного проекта позволит закрепить знания по базовым элементам программирования в Bash и углубить понимание принципов создания скриптов для автоматизации задач.

# **2 КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ**

Shell-скрипты представляют собой последовательность команд, предназначенных для автоматизации различных задач в операционной системе Unix/Linux. Они позволяют пользователю управлять процессами, файлами, конфигурациями системы и выполнять сложные операции без необходимости ввода команд вручную. Основными элементами скриптов являются переменные, параметры, ветвления, циклы, встроенные команды оболочки и вызовы внешних программ [1].

Переменные в shell используются для хранения данных и могут быть как пользовательскими, так и системными. Например, переменная $USER хранит имя текущего пользователя, а $HOME содержит путь к его домашнему каталогу. Параметры передаются скрипту при его вызове и доступны в виде $1, $2 и так далее, где числа обозначают порядковый номер аргумента. Для реализации логики управления потоком выполнения применяются конструкции ветвления, такие как if-else и case, которые позволяют принимать решения на основе условий.

Важной частью скриптов являются циклы (for, while, until), которые используются для многократного выполнения команд при соблюдении заданных условий. Выражения и вычисления в shell выполняются с помощью встроенной команды expr или $(()). Это позволяет выполнять математические операции, сравнивать значения и использовать логические выражения в условиях ветвлений и циклов. Команды оболочки и вызовы внешних программ играют ключевую роль в обработке данных. Например, команда ps используется для получения информации о запущенных процессах, sort – для их сортировки, head – для выбора нужного количества строк, а awk – для форматирования вывода.

Перенаправление ввода и вывода позволяет направлять результаты выполнения команд в файлы или передавать их в другие процессы. Символ > используется для записи данных в файл, перезаписывая его содержимое, а >> – для добавления информации в конец файла. Конвейеры (|) позволяют передавать вывод одной команды в качестве ввода для другой, что значительно расширяет возможности обработки данных. Для редактирования shell-скриптов используются консольные текстовые редакторы, такие как vim, nano и micro. Они позволяют создавать, редактировать и сохранять файлы скриптов непосредственно в терминале. В данной лабораторной работе необходимо разработать shell-скрипт, который анализирует список запущенных процессов в системе и определяет топ-5 по потреблению оперативной памяти и процессорного времени. Результаты работы скрипта должны быть сохранены в файл для дальнейшего анализа.

Таким образом, изучение элементов и конструкций shell-скриптов, включая переменные, параметры, циклы, ветвления, команды оболочки и вызовы внешних программ, позволяет автоматизировать выполнение задач и эффективно управлять процессами в Unix-подобных системах.

# **3 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ**

Программа представляет собой shell-скрипт, предназначенный для отображения текущего времени в терминале в виде крупных символов, стилизованных под цифровые часы. Для реализации функциональности используются несколько вспомогательных функций, каждая из которых выполняет определённую задачу.

Функция `generate\_random()` отвечает за генерацию случайного числа в заданном диапазоне. В качестве начального значения (seed) используется текущее время в секундах, после чего применяется линейный конгруэнтный метод для получения псевдослучайного числа. Результат вычисления ограничивается заданным диапазоном с помощью операции остатка от деления. Эта функция позволяет динамически изменять положение часов на экране, делая их расположение случайным при каждом обновлении.

Функция `get\_big\_digit()` предназначена для формирования графического представления цифр от 0 до 9 и символа двоеточия. Она использует конструкцию `case`, которая выбирает соответствующее изображение в зависимости от переданного параметра. Каждая цифра или символ представляется в виде трёхстрочной конструкции, что обеспечивает читаемость в терминале.

Основная функциональность программы реализована в бесконечном цикле `while true`. Перед каждым обновлением экран очищается с помощью функции `clear\_screen()`, а размеры терминала определяются с помощью команд `tput cols` и `tput lines`. Затем вызывается функция `generate\_random()` для выбора случайных координат для отображения часов, учитывая ограничения по размеру терминала.

Текущее время получается с помощью команды `date +"%H:%M:%S"`, после чего каждая цифра разбивается на составляющие строки с помощью функции `get\_big\_digit()`. Эти строки объединяются в массив `big\_time\_lines`, где каждая позиция соответствует одной строке для вывода. Для корректного позиционирования курсора используется функция `set\_cursor\_position()`, которая перемещает текстовое представление часов в нужное место экрана.

Обновление времени происходит каждую секунду благодаря команде `sleep 1`, после чего процесс повторяется. Кроме того, позиция часов на экране обновляется каждые 5 секунд, чтобы создать эффект динамического перемещения часов по терминалу.

Работа программы продемонстрирована на рисунке 3.1 и 3.2.

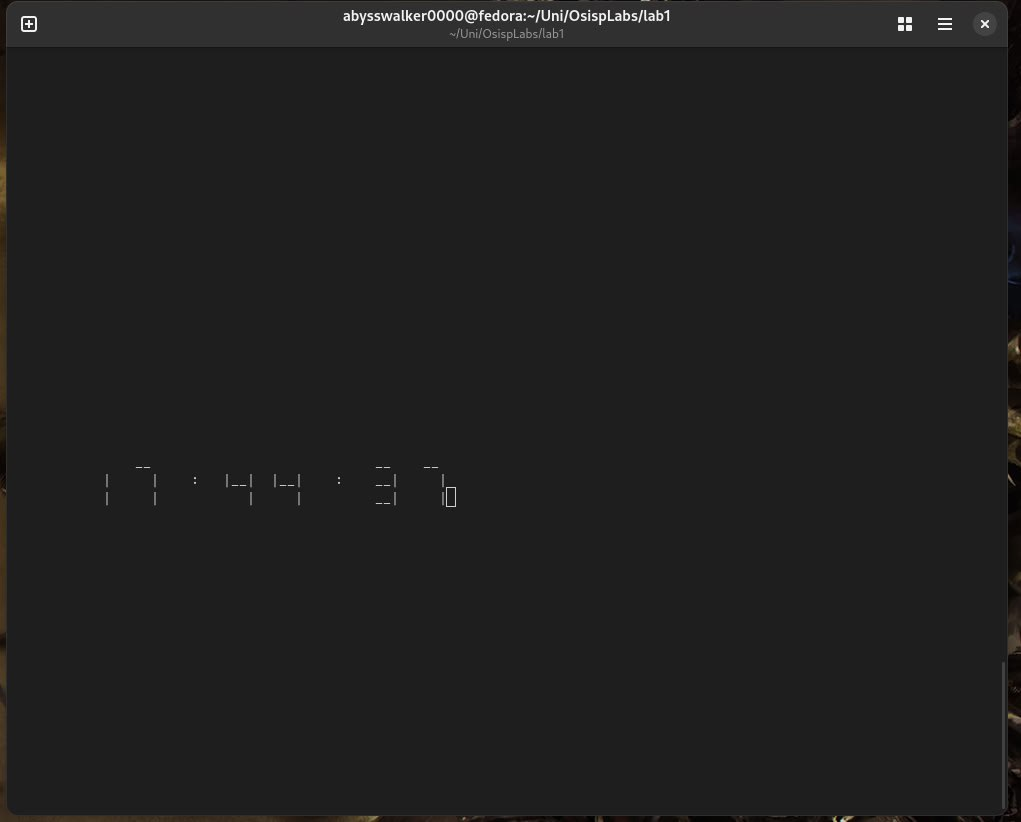


Рисунок 3.1 – Демонстрация работы программы

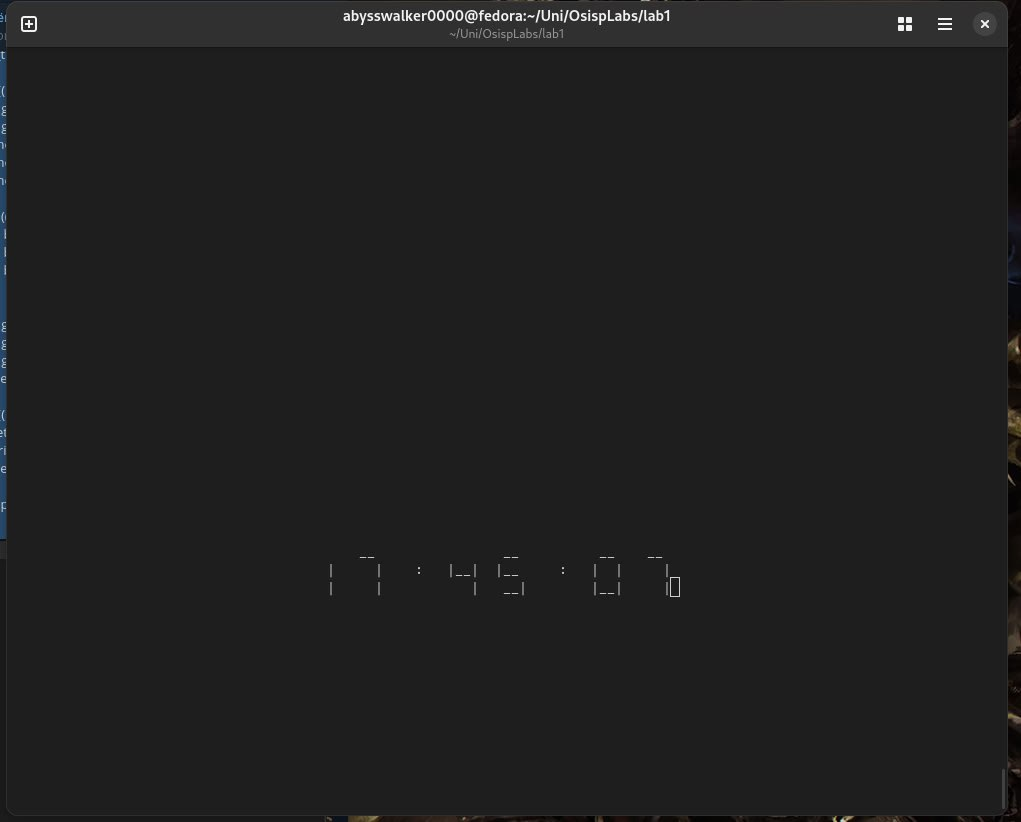


Рисунок 3.2 – Демонстрация работы программы через промежуток времени

Таким образом, программа демонстрирует использование базовых элементов программирования в Bash, таких как функции, циклы, обработка строк и работа с терминалом, а также показывает пример интеграции этих элементов для создания интерактивного консольного приложения.

# **ЗАКЛЮЧЕНИЕ**

Программа представляет собой shell-скрипт, реализующий хранитель экрана (screensaver) для консоли в виде крупных текстовых часов, отображающих текущее время. Часы периодически меняют своё положение на экране, создавая динамический эффект.

Для генерации случайных координат используется функция `generate\_random()`, основанная на линейном конгруэнтном методе. Эта функция принимает максимальное значение диапазона и возвращает случайное число в пределах этого диапазона. В качестве начального значения (seed) используется текущее время в секундах, что обеспечивает уникальность последовательности псевдослучайных чисел при каждом запуске программы. Результат вычисления ограничивается заданным диапазоном с помощью операции взятия остатка от деления.

Функция `get\_big\_digit()` отвечает за формирование графического представления цифр от 0 до 9 и символа двоеточия. Каждая цифра или символ представляется в виде трёхстрочной конструкции, которая создаёт читаемое изображение в терминале. Для выбора соответствующего изображения используется конструкция `case`, которая обрабатывает переданный параметр и возвращает нужную строку.

Основной логикой работы программы является бесконечный цикл `while true`. На каждой итерации цикла очищается экран с помощью команды `clear\_screen()`, а размеры терминала определяются с помощью утилит `tput cols` и `tput lines`. Затем проверяется, прошло ли не менее 5 секунд с момента последнего изменения позиции часов. Если условие выполнено, вызывается функция `generate\_random()` для расчёта новых случайных координат для отображения времени.

Текущее время получается с помощью команды `date +"%H:%M:%S"`. Каждый символ времени (цифры и двоеточие) преобразуется в его крупное графическое представление с помощью функции `get\_big\_digit()`. Эти графические элементы объединяются построчно в массив `big\_time\_lines`, где каждая строка содержит соответствующую часть изображения времени.

Для вывода часов на экран используется функция `set\_cursor\_position()`, которая перемещает курсор в нужные координаты, после чего строки массива `big\_time\_lines` выводятся на экран. После этого программа засыпает на одну секунду с помощью команды `sleep 1`, чтобы обеспечить обновление времени каждую секунду.

Таким образом, данный скрипт демонстрирует использование основных элементов программирования в Bash, таких как функции, циклы, работа со строками и управление терминалом. Программа также показывает, как можно интегрировать эти элементы для создания интерактивного консольного приложения, такого как динамический screensaver с текстовыми часами.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Introduction to Linux Shell and Shell Scripting [Электронный ресурс]. – Режим доступа: https://www.geeksforgeeks.org/introduction-linux-shell-shell-scripting/

[2] Bash скрипты [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/726316/

# **ПРИЛОЖЕНИЕ А (обязательное) Исходный код программного продукта**

#!/bin/bash

generate\_random() {

local max=$1

seed=${seed:-$(date +%s)}

seed=$(( (seed \* 1103515245 + 12345) % 2\*\*31 ))

echo $(( seed % max ))

}

clear\_screen() {

printf "\033[H\033[2J"

}

set\_cursor\_position() {

local x=$1

local y=$2

printf "\033[${y};${x}H"

}

get\_big\_digit() {

case $1 in

0) echo -e " \n| |\n||" ;;

1) echo -e " \n |\n |" ;;

2) echo -e " \n |\n| " ;;

3) echo -e " \n |\n |" ;;

4) echo -e " \n||\n |" ;;

5) echo -e " \n| \n |" ;;

6) echo -e " \n| \n||" ;;

7) echo -e " \n |\n |" ;;

8) echo -e " \n||\n||" ;;

9) echo -e " \n||\n |" ;;

:) echo -e " \n : \n " ;;

esac

}

x=0

y=0

last\_position\_change\_time=0

while true; do

clear\_screen

width=$(tput cols)

height=$(tput lines)

current\_time=$(date +%s)

if (( current\_time - last\_position\_change\_time >= 5 )); then

x=$(generate\_random $((width - 40)))

y=$(generate\_random $((height - 5)))

last\_position\_change\_time=$current\_time

fi

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

big\_time\_lines=("" "" "")

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

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

digit\_art=$(get\_big\_digit "$digit")

line1=$(echo "$digit\_art" | sed -n '1p')

line2=$(echo "$digit\_art" | sed -n '2p')

line3=$(echo "$digit\_art" | sed -n '3p')

if (( i > 0 )); then

big\_time\_lines[0]+=" "

big\_time\_lines[1]+=" "

big\_time\_lines[2]+=" "

fi

big\_time\_lines[0]+="$line1"

big\_time\_lines[1]+="$line2"

big\_time\_lines[2]+="$line3"

done

for (( i=0; i<3; i++ )); do

set\_cursor\_position $((x + 1)) $((y + 1 + i))

printf "%s" "${big\_time\_lines[i]}"

done

sleep 1

done