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 В. М. Вергасов

Преподаватель Н. Ю. Гриценко

Минск 2024

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc146752068)

[2 Теоретические сведения 4](#_Toc146752069)

[3 Результат выполнения 6](#_Toc146752070)

[Заключение 6](#_Toc146752071)

[Список использованных источников 7](#_Toc146752072)

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

1. **ЦЕЛЬ РАБОТЫ**

Целью данной работы является 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*. В ходе работы будет написан bash-скрипт для системы *MacOS*. Написанный скрипт позволит выбрать алгоритм хеширования и посчитать хеш-сумму для указанного файла.

1. **ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Вот небольшое теоретическое введение в *Bash*, использованное для написания работы.

В скриптах *shell* переменные используются для хранения значений, которые могут быть использованы в различных частях скрипта. Объявление переменных происходит с помощью оператора =, например: *variable*="*value*". Значение переменной можно получить с помощью префикса $, например: *echo* $*variable*. Переменные могут быть глобальными или локальными в зависимости от их области видимости [1].

Параметры скрипта представляют собой значения, переданные скрипту в качестве аргументов при его запуске. Параметры скрипта доступны через специальные переменные $1, $2, $3, и так далее, где $1 представляет первый параметр, $2 – второй параметр и т.д. Можно использовать оператор *shift*, чтобы сдвинуть параметры и получить доступ к следующим значениям.

Ветвления используются для выполнения различных действий в зависимости от условий. Конструкция *if*-*then*-*else* позволяет выполнить одно действие, если условие истинно, и другое действие, если условие ложно. Конструкция *case* позволяет проверить значение переменной на соответствие нескольким паттернам и выполнить соответствующий блок кода.

Циклы позволяют выполнять повторяющиеся действия. Конструкция *for* используется для итерации по списку значений или файлам. Конструкция *while* выполняет цикл, пока условие истинно. Конструкция *until* выполняет цикл, пока условие ложно [2].

В скриптах *shell* можно выполнять простые арифметические операции с помощью встроенных математических операций, таких как +, -, \*, /. Для выполнения более сложных вычислений можно использовать внешние программы, такие как *expr* или *bc*.

В скриптах *shell* можно использовать различные команды и вызывать внешние программы для выполнения различных задач. Встроенные команды, такие как *echo*, *read*, *cd*, доступны непосредственно в оболочке *shell*. Вы можете вызывать внешние программы, такие как *sed*, *awk*, *wget* и другие, для обработки текстовых данных, работы с сетью и других задач.

Одно из главных преимуществ скриптов *shell* – их способность объединять воедино различные Unix-программы и использовать их результаты в дальнейшей обработке.

Различные программы могут быть вызваны в скрипте, а их вывод может быть перенаправлен, фильтрован или обработан с помощью конвейеров (|).

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате работы был создан скрипт, пример вычисления зеш суммы по алгоритму md5 представлен на рисунке 1:

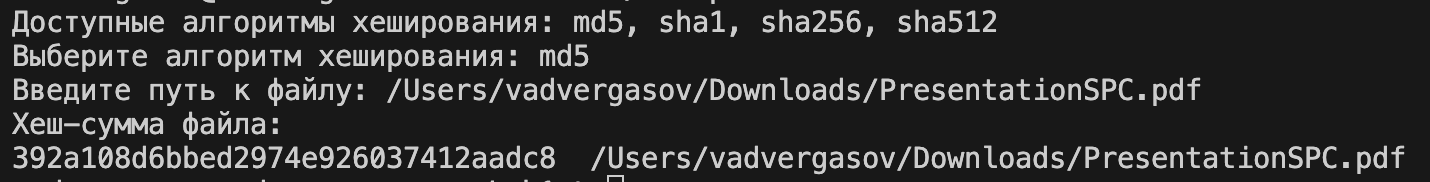


Рисунок 1 – Вывод программы при использовании алгоритма md5

При использовании другого алгоритма хеширования, вывод, что логично, отличается (рисунок 2):

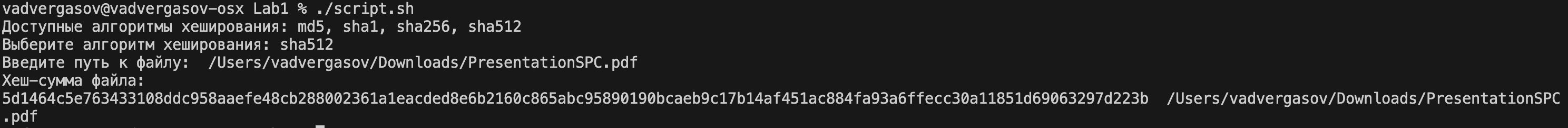


Рисунок 2 – Подсчёт хеш-суммы с использованием алгоритма sha512

# ЗАКЛЮЧЕНИЕ

В ходе данной лабораторной работы были изучены элементы и конструкции скриптов *shell*: переменные, параметры, ветвления, циклы, вычисления, команды *shell* и вызовы 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*. В ходе работы был написан bash-скрипт для *MacOS* системы. Написанный скрипт позволяет считать хеш сумму для указанного файла с помощью разных алгоритмов.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Bash* скрипты [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/ruvds/articles/325522/. – Дата доступа: 04.02.2024.
2. *Introduction to Bash Scripting* [Электронный ресурс]. – Режим доступа: https://bjpcjp.github.io/pdfs/bash/bash-scripting.pdf. – Дата доступа: 04.02.2024.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Lab1.sh

#!/bin/bash

calculate\_hash() {

local algorithm="$1"

local file="$2"

case "$algorithm" in

md5)

md5sum "$file" ;;

sha1)

sha1sum "$file" ;;

sha256)

shasum "$file" -a 256 ;;

sha512)

shasum "$file" -a 512;;

\*)

echo "Неподдерживаемый алгоритм хеширования: $algorithm"

exit 1 ;;

esac

}

echo "Доступные алгоритмы хеширования: md5, sha1, sha256, sha512"

read -p "Выберите алгоритм хеширования: " algorithm

if [[ "$algorithm" != "md5" && "$algorithm" != "sha1" && "$algorithm" != "sha256" && "$algorithm" != "sha512" ]]; then

echo "Неподдерживаемый алгоритм хеширования: $algorithm"

exit 1

fi

read -p "Введите путь к файлу: " file

if [[ ! -f "$file" ]]; then

echo "Файл не найден: $file"

exit 1

fi

echo "Хеш-сумма файла:"

calculate\_hash "$algorithm" "$file"