Лабораторная работа номер 13

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Титков Ярослав Максимович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов

2 Задание

1. Используя команды getopts grep, написать командный файл, который анализирует командную строку с ключами: – -iinputfile — прочитать данные из указанного файла; – -ooutputfile — вывести данные в указанный файл; – -ршаблон — указать шаблон для поиска; – -C — различать большие и малые буквы; – -n — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом -p.
2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции exit(n), передавая информацию в о коде завершения в оболочку. Команд- ный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено.
3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до (например 1.tmp, 2.tmp, 3.tmp,4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же ко- мандный файл должен уметь удалять все созданные им файлы (если они существуют).
4. Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).

3 Выполнение лабораторной работы

```
#!/bin/bash

while getopts "i:o:p:Cn" opt; do
  case $opt in
    i) inputfile="$OPTARG" ;;
    o) outputfile="$OPTARG" ;;
    p) pattern="$OPTARG" ;;
    C) case_sensitive=1 ;;
    n) with_line_numbers=1 ;;
    *) echo "Usage: $0 -i inputfile -o outputfile -p pattern [-C] [-n]" ; exit 1 ;;
  esac
done

if [ -z "$inputfile" ] || [ -z "$outputfile" ] || [ -z "$pattern" ]; then
  echo "Missing required options"
  exit 1
fi

grep_opts=""
[ "$with_line_numbers" = 1 ] && grep_opts="$grep_opts -n"
[ "$case_sensitive" != 1 ] && grep_opts="$grep_opts -i"

grep $grep_opts "$pattern" "$inputfile" > "$outputfile"
```

Рис. 3.1: Используя команды getopts grep, написал командный файл, который анализирует командную строку с ключами

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int number;
    printf("Введите число: ");
    scanf("%d", &number);

    if (number > 0)
        exit(1);
    else if (number < 0)
        exit(2);
    else
        exit(0);
}
```

Рис. 3.2: Написал на языке Си программу,

Рис. 3.3: Написал командный файл, создающий указанное число файлов

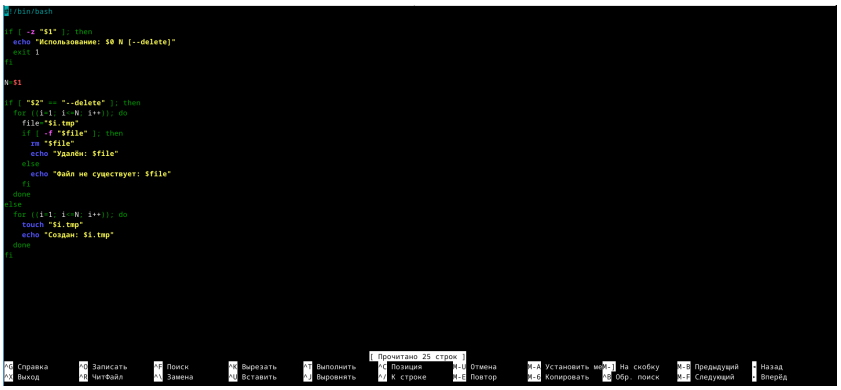

Рис. 3.4: Написал командный файл, который работает с архивом

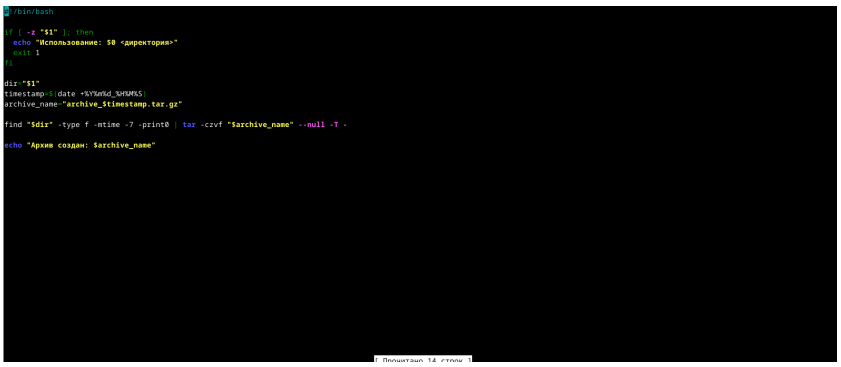

Рис. 3.5: Командный файл

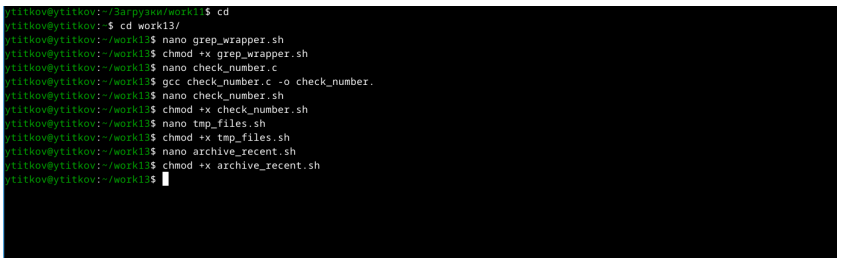

Рис. 3.6: Компиляция всего

4 Выводы

Изучил основы программирования в оболочке ОС UNIX. Научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов

5 Контрольные вопросы

1. getopts обрабатывает аргументы командной строки в скриптах, разбирая опции и их параметры.

2. Метасимволы (*, ?, []) используются в шаблонах имен файлов для их генерации по заданному паттерну.

3. Операторы управления действиями: if, case, for, while, until, select.

4. Для прерывания цикла: break (выход из цикла), continue (переход к следующей итерации).

5. false возвращает код ошибки (1), true — успешный код (0); используются для управления потоком выполнения.

6. if test -f man*s*/i.*s*, *man*s/*i*.s и является ли он обычным файлом.

7. while выполняет цикл, пока условие истинно, until — пока условие ложно.