

Отчёт по лабораторной работе №11

Операционные системы

Ищенко Ирина Олеговна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	8
4	Выполнение лабораторной работы	9
5	Вывод	14
6	Контрольные вопросы	15

Список иллюстраций

4.1	Скрипт к заданию 1.	9
4.2	Результат выполнение скрипта 1.	10
4.3	Вспомогательная программа на C++ к заданию 2.	10
4.4	Скрипт к заданию 2.	11
4.5	Результат выполнения скрипта 2.	11
4.6	Скрипт к заданию 3.	12
4.7	Результат выполнения скрипта 3.	12
4.8	Скрипт к заданию 4.	13
4.9	Результат выполнения скрипта 4.	13

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:
 - `-I inputfile` — прочитать данные из указанного файла;
 - `-o outputfile` — вывести данные в указанный файл;
 - `-r` шаблон — указать шаблон для поиска;
 - `-C` — различать большие и малые буквы;
 - `-n` — выдавать номера строк.

а затем ищет в указанном файле нужные строки, определяемые ключом `-r`.

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. Командный 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 N (например `1.tmp`, `2.tmp`, `3.tmp`, `4.tmp` и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный 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 Теоретическое введение

Команда `getopts` осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ...]`

Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`.

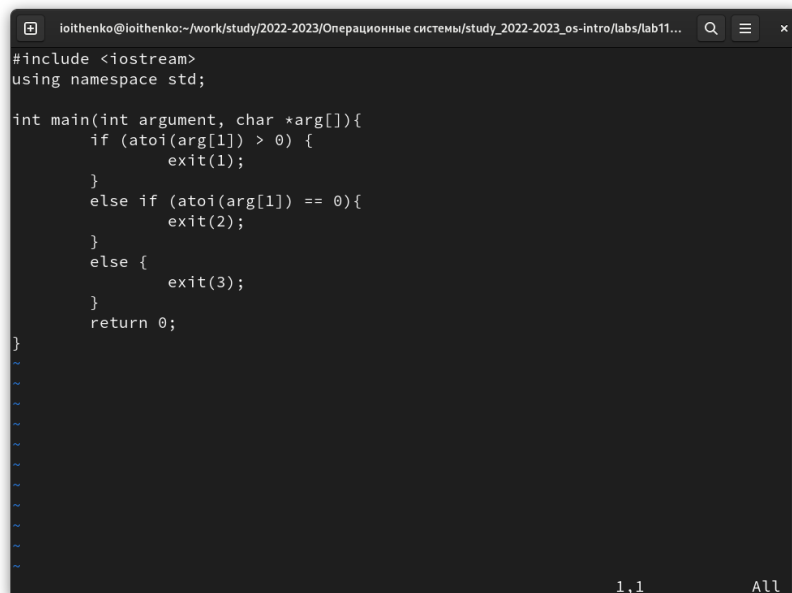

```

[ioithenko@ioithenko lab11]$ vi lab11_1
[ioithenko@ioithenko lab11]$ ./lab11_1 -i conf.txt -o output.txt -p h -c -n
[ioithenko@ioithenko lab11]$ ls
conf.txt  lab11_1  output.txt
[ioithenko@ioithenko lab11]$

```

Рис. 4.2: Результат выполнение скрипта 1.

2. На языке программирования C++ я написала вспомогательную программу, которая вводит число и определяет, является оно большим/меньшим/равным нулю. Затем программа завершается, передавая информацию о коде завершения в оболочку, с помощью функции `exit(n)`, где `n` – код.



```

#include <iostream>
using namespace std;

int main(int argument, char *arg[]){
    if (atoi(arg[1]) > 0) {
        exit(1);
    }
    else if (atoi(arg[1]) == 0){
        exit(2);
    }
    else {
        exit(3);
    }
    return 0;
}

```

Рис. 4.3: Вспомогательная программа на C++ к заданию 2.

Далее я написала командный файл, который вызывает эту программу и, проанализировав с помощью команды `$?`, выдает сообщение о том, какое число было введено(большее/меньшее/равное нулю).

```
ioithenko@ioithenko:~/lab11 — /usr/bin/mc -P /var/tmp/mc-ioithenko/mc.pwd.2452
#!/bin/bash

EXEC=compare
SRC=compare.cpp

if [ "$SRC" -nt "$EXEC" ]
then
    echo "Rebuilding $EXEC ....."
    g++ -o $EXEC $SRC
fi

./$EXEC $1
ec=$?
if [ "$ec" == "1" ]
then
    echo "argument > 0"
fi
if [ "$ec" == "2" ]
then
    echo "argument = 0"
fi
if [ "$ec" == "3" ]
then
    echo "argument < 0"
fi
~
-- INSERT -- 1,12 All
```

Рис. 4.4: Скрипт к заданию 2.

Затем я добавила права на выполнение файла и выполнила его, указав необходимые опции и аргументы.

```
[ioithenko@ioithenko lab11]$ ./lab11_2 3
Rebuilding compare .....
argument > 0
[ioithenko@ioithenko lab11]$ ./lab11_2 0
argument = 0
[ioithenko@ioithenko lab11]$ ./lab11_2 -1
argument < 0
[ioithenko@ioithenko lab11]$
```

Рис. 4.5: Результат выполнения скрипта 2.

3. Я создала командный файл, который создает n файлов последовательно пронумерованных (1.tmp, 2.tmp, 3.tmp, 4.tmp и т.д. до n), где n задается как аргумент командной строки. Также этот файл умеет удалять все подобные файлы, если они имеются. Для этого нужно указать другую опцию.

4. Я создала командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории, модифицировала его так, чтобы он запаковывал только те файлы, который изменялись менее недели тому назад, используя команду find.

[illegible]

Рис. 4.8: Скрипт к заданию 4.

Затем я добавила права на выполнение файла и выполнила его, указав необходимые опции и аргументы.

```
[ioithenko@ioithenko lab11]$ ./lab11_4 -d /home
[ioithenko@ioithenko lab11]$ ls
archive_lab11_4.tar  compare      conf.txt    lab11_2    lab11_4
arch.txt             compare.cpp  lab11_1    lab11_3    output.txt
[ioithenko@ioithenko lab11]$
```

Рис. 4.9: Результат выполнения скрипта 4.

5 Вывод

В ходе выполнения лабораторной работы я изучила основы программирования в командной оболочке ОС UNIX, а также научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

6 Контрольные вопросы

1. Каково предназначение команды `getopts`?

Команда `getopts` осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ...]`

Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`.

2. Какое отношение метасимволы имеют к генерации имён файлов?

При перечислении имен файлов текущего каталога можно использовать следующие символы:

- `-` — соответствует произвольной, в том числе и пустой строке;
- `?` — соответствует любому одному символу;

- `[c1-c1]` — соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`;
- `echo *` — выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`;
- `ls .c` — выведет все файлы с последними двумя символами, равными `.c`;
- `echo prog.?` — выдаст все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog`;
- `[a-z]` — соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

3. Какие операторы управления действиями вы знаете?

- Точка с запятой (;) Вы можете разместить две и более команд в одной и той же строке, разделив эти команды с помощью символа точки с запятой ;.
- Амперсанд (&) В том случае, если строка команды оканчивается символом амперсанда `&`, командная оболочка не будет ожидать завершения исполнения этой команды. Сразу же после ввода команды будет выведено новое приглашение командной оболочки, а сама команда будет исполняться в фоновом режиме. В момент завершения исполнения команды в фоновом режиме вы получите соответствующее сообщение.
- Символ доллара со знаком вопроса (\$?) Код завершения предыдущей команды сохраняется в переменной командной оболочки с именем `$?` .
- Двойной амперсанд (&&) Командная оболочка будет интерпретировать последовательность символов `&&` как логический оператор “И”. При использовании оператора `&&` вторая команда будет исполняться только в том случае, если исполнение первой команды успешно завершится (будет возвращен нулевой код завершения).

- Двойная вертикальная черта (||) Оператор || представляет логическую операцию “ИЛИ”. Вторая команда выполняется только тогда, когда исполнение первой команды заканчивается неудачей (возвращается ненулевой код завершения).
- Комбинирование операторов && и || Вы можете использовать описанные логические операторы “И” и “ИЛИ” для создания структур условных переходов в рамках строк команд.
- Знак фунта (#) Все написанное после символа фунта (#) игнорируется командной оболочкой. Это обстоятельство оказывается полезным при возникновении необходимости в написании комментариев в сценариях командной оболочки, причем комментарии ни коим образом не будут влиять на процесс исполнения команд или процесс раскрытия команд командной оболочкой.
- Экранирование специальных символов () Символ обратного слэша позволяет использовать управляющие символы без их интерпретации командной оболочкой; процедура добавления данного символа перед управляющими символами называется экранированием символов.

4. Какие операторы используются для прерывания цикла?

Для управления ходом выполнения цикла служат команды `break` и `continue` [1] и точно соответствуют своим аналогам в других языках программирования. Команда `break` прерывает исполнение цикла, в то время как `continue` передает управление в начало цикла, минуя все последующие команды в теле цикла.

5. Для чего нужны команды `false` и `true`?

Команда `true` всегда возвращает ноль в качестве выходного статуса для индикации успеха.

Команда `false` всегда возвращает не-ноль в качестве выходного статуса для индикации неудачи.

6. Что означает строка `if test -f mans/i.$s`, встреченная в командном файле?

Веденная строка означает условие существования файла `mans/i.$s`

7. Объясните различия между конструкциями `while` и `until`.

Разница между циклом `while` (пока) и `until` (пока не) – это условие проверки. Пока ВЫПОЛНЯЕТСЯ условие проверки, цикл `while` будет продолжать работать. Однако цикл `until` будет выполняться только пока условие ЛОЖНО.