Отчёт по лабораторной работе №11

Операционные системы

Сячинова Ксения Ивановна

Содержание

Список иллюстраций

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;
- -pшаблон — 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. 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 Выполнение лабораторной работы

1. Для начала сосздадим папку, где буду находиться все наши скринкасты к лабораторной работе. (рис. 3.1).

Рис. 3.1: Создание рабочего пространства

Создадим файл для написания скрипта и откроем его. (рис. 3.2)

Рис. 3.2: Создание файла

Напишем скрипт для выполнения первого задания. (рис. 3.3),(рис. 3.4)

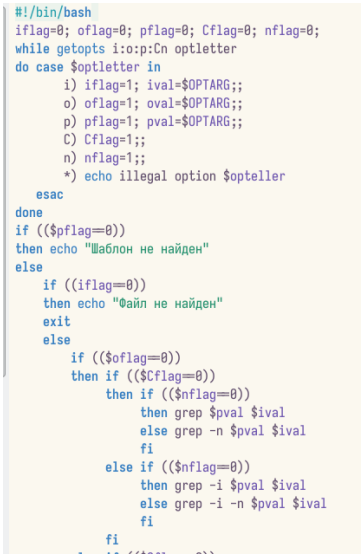

Рис. 3.3: Текст скрипта

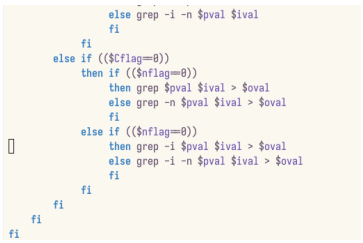

Рис. 3.4: Текст скрипта

Для проверки работы скрипта создадим два текстовых файла. В один из них запишем текст для проверка.

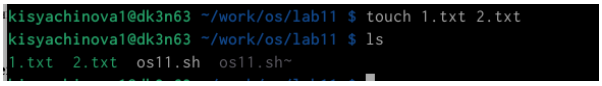

Рис. 3.5: Создание файлов

Даём право на выполнение с помощью команды “chmod” и используем команды для проверки. Скрипт работает корректно.(рис. 3.6)

Рис. 3.6: Проверка скрипта

2. Напишем скрипт для выполнения второго задания. Для этого также создадим файл. (рис. 3.7)

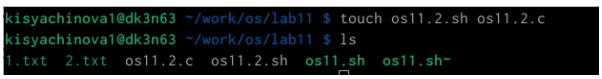

Рис. 3.7: Создание файлов

Напишем скрин на языке “C” и “bush”. (рис. 3.8),(рис. 3.9)

```c
#include <stdio.h>
#include <stdlib.h>
int main ()
{
  printf("Введите число\n");
  int a;
  scanf ("%d", &a);
  if (a<0)exit(0);
  if (a>0) exit(1);
  if (a==0) exit(2);
  return 0;
}
```

Рис. 3.8: Скрипт 1

Рис. 3.9: Скрипт 2

Затем даём право на выполнение и проверяем. Всё работает корректно. (рис. 3.10)

Рис. 3.10: Проверка

3. Далее выпоним третье задание. Создаём новый файл для скрипта и открываем его.(рис. 3.11),(рис. 3.12),

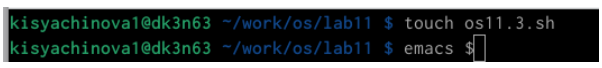

Рис. 3.11: Создание файла

```
#!/bin/bash
opt=$1;
format=$2;
number=$3;
function Files()
{
    for (( i=1; i≤$number; i++ )) do
    file=$(echo $format | tr '#' "$i")
    if [ $opt == "-r" ]
    then
        rm -f $file
    elif [ $opt == "-c" ]
    then
        touch $file
    fi
done
```

Рис. 3.12: Скрипт

Првоерим работу скрипта. Даём право на выполнение файла. Создадим три файла, а затем удалим их с помощью программы.(рис. 3.13)

```
kisyachinova1@dk3n63 ~/work/os/lab11 $ chmod +x os11.3.sh
kisyachinova1@dk3n63 ~/work/os/lab11 $ ls
1.txt  abc1.txt  abc3.txt  os11.2.c   os11.2.sh   os11.3.sh   os11.sh
2.txt  abc2.txt  os11.2    os11.2.c~  os11.2.sh~  os11.3.sh~  os11.sh~
kisyachinova1@dk3n63 ~/work/os/lab11 $ ./os11.3.sh -c abc#.txt 3
kisyachinova1@dk3n63 ~/work/os/lab11 $ ls
1.txt  abc1.txt  abc3.txt  os11.2.c   os11.2.sh   os11.3.sh   os11.sh
2.txt  abc2.txt  os11.2    os11.2.c~  os11.2.sh~  os11.3.sh~  os11.sh~
kisyachinova1@dk3n63 ~/work/os/lab11 $ ./os11.3.sh -r abc#.txt 3
kisyachinova1@dk3n63 ~/work/os/lab11 $ ls
1.txt  os11.2    os11.2.c~  os11.2.sh~  os11.3.sh~  os11.sh~
2.txt  os11.2.c  os11.2.sh  os11.3.sh   os11.sh
```

Рис. 3.13: Проверка

4. Создаём файл и пишем скрипт. (рис. 3.14)

```
#!/bin/bash
files=$(find ./ -maxdepth 1 -mtime -7)
listing=""
for file in "$files" ; do
    file=$(echo "$file" | cut -c 3-)
    listing="$listing $file"
done
dir=$(basename $(pwd))
tar -cvf $dir.tar $listing
```

Рис. 3.14: Скрипт

Даём право на выполнение и првоеряем. Всё работвет верно.(рис. 3.15), (рис. 3.16)

```
kisyachinova1@dk3n63 ~/work/os/lab11 $ chmod +x os11.4.sh
kisyachinova1@dk3n63 ~/work/os/lab11 $ ./os11.4.sh
os11.sh
1.txt
2.txt
os11.sh~
os11.2.sh
os11.2.c
os11.2
os11.3.sh
os11.2.c~
os11.2.sh~
os11.4.sh
os11.3.sh~
os11.4.sh~
kisyachinova1@dk3n63 ~/work/os/lab11 $ tar -tf lab11.tar
os11.sh
1.txt
2.txt
os11.sh~
os11.2.sh
os11.2.c
os11.2
os11.3.sh
```

Рис. 3.15: Проверка

Рис. 3.16: Проверка

4 Выводы

В процессе выполнения данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

5 Ответы на контрольные вопросы

1. 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, для команды ls флагом может являться -F. Строка опций option-string – это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда getopts может распознать аргумент, то она возвращает истину. Принято включать getopts в цикл while и анализировать введённые данные с помощью оператора case. Функция getopts включает две специальные переменные среды – OPTARG и OPTIND. Если ожидается дополнительное значение, то OPTARG устанавливается в значение этого аргумента. Функция getopts также понимает переменные типа массив, следовательно, можно использовать её в функции не только для синтаксического анализа аргументов функций, но и для анализа введённых пользователем данных.

2. При перечислении имён файлов текущего каталога можно использовать следующие символы:

- – – соответствует произвольной, в том числе и пустой строке;
- ? – соответствует любому одинарному символу;

- [c1-c2] – соответствует любому символу, лексикографически находящемуся между символами c1 и c2. Например,
- echo * – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды ls;
- ls *.c – выведет все файлы с последними двумя символами, совпадающими с .c.
- echo prog.? – выведет 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. Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости от результатов проверки некоторого условия. Для решения подобных задач язык программирования bash предоставляет возможность использовать такие управляющие конструкции, как for, case, if и while. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути, являются операторами языка программирования bash. Поэтому при описании языка программирования bash термин оператор будет использоваться наравне с термином команда. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда test, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.
4. Два несложных способа позволяют вам прерывать циклы в оболочке bash. Команда break завершает выполнение цикла, а команда continue завершает

данную итерацию блока операторов. Команда break полезна для завершения цикла while в ситуациях, когда условие перестаёт быть правильным. Команда continue используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.

5. Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования bash: это команда true, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда false, которая всегда возвращает код завершения, не равный нулю (т. е. ложь). Примеры бесконечных циклов: while true do echo hello andy done until false do echo hello mike done

6. Строка if test -f man$s$/i.s, man$s$/i.s и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).

7. Выполнение оператора цикла while сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово while, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово do, после чего осуществляется безусловный переход на начало оператора цикла while. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово while, возвратит ненулевой код завершения (ложь). При замене в операторе цикла while служебного слова while на until условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла while и оператор цикла until идентичны.