Отчёт по лабораторной работе №10 (*11 в лабнике, 10 в ТУИС)

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Цыганков Александр Романович, НПМБВ-02-20

Содержание

Список таблиц

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).

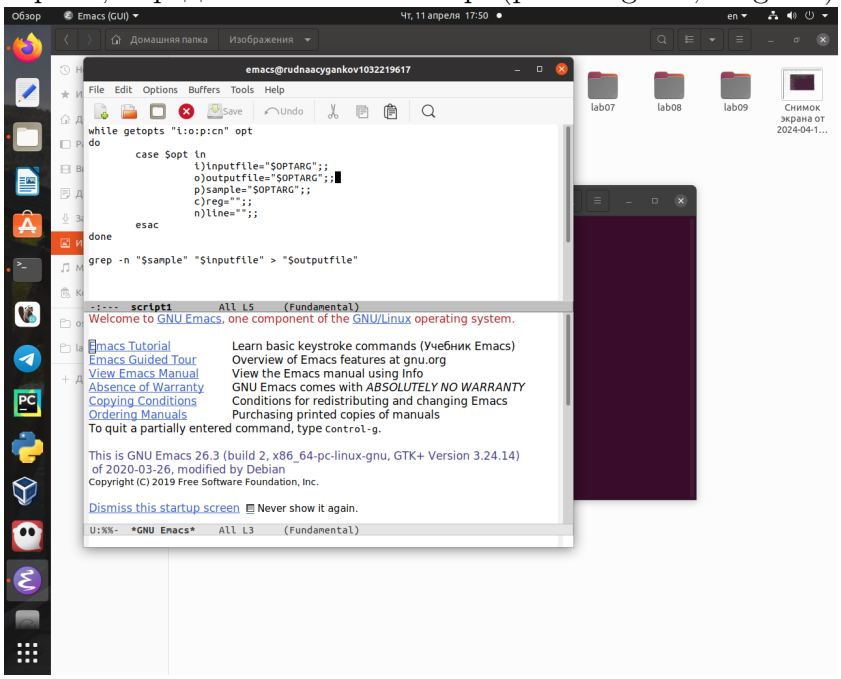
3 Теоретические сведения

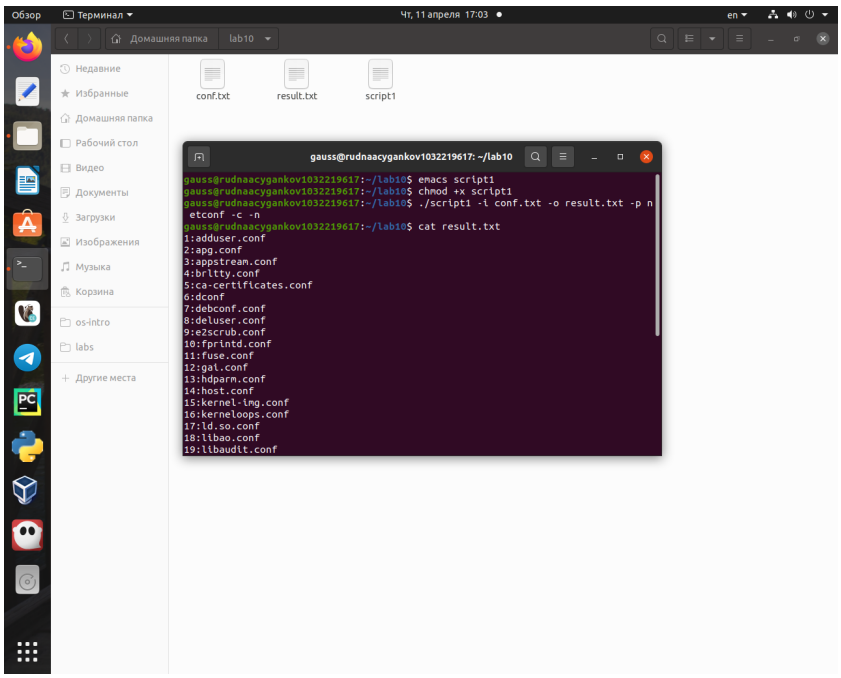
- Командный процессор (командная оболочка, интерпретатор команд shell) - это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:
  - оболочка Борна (Bourne shell или sh) - стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций;
  - C-оболочка (или csh) - надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;
  - оболочка Корна (или ksh) - напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна;
  - BASH - сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).
- POSIX (Portable Operating System Interface for Computer Environments) - набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна.

4 Выполнение лабораторной работы

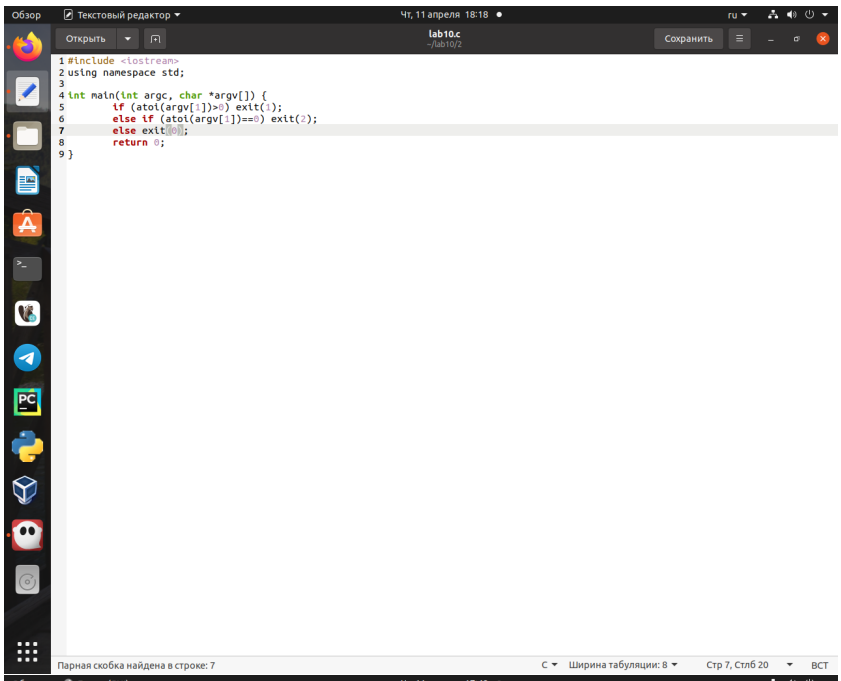
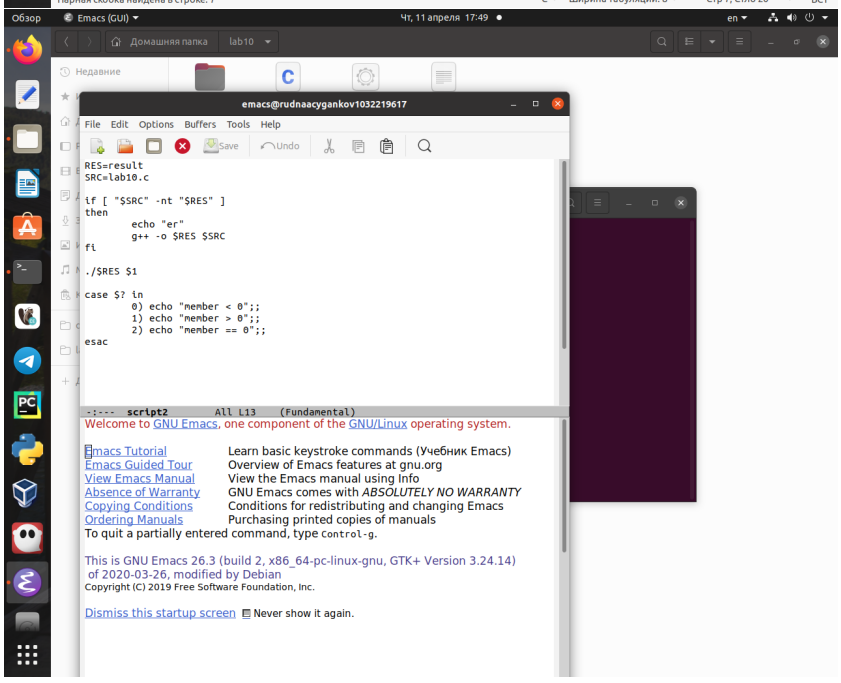
1. Используя команды getopts grep, написать командный файл, который анализирует командную строку с ключами:

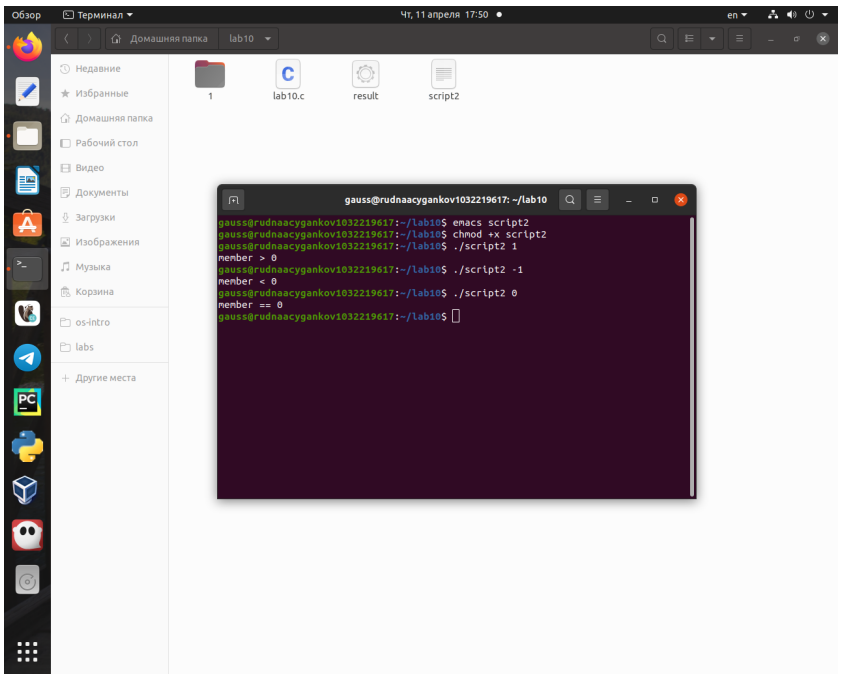
   - -iinputfile — прочитать данные из указанного файла;
   - -ooutputfile — вывести данные в указанный файл;
   - -pшаблон — указать шаблон для поиска;
   - -C — различать большие и малые буквы;
   - -n — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом -p. (рис. @fig:001, @fig:002)

   - 

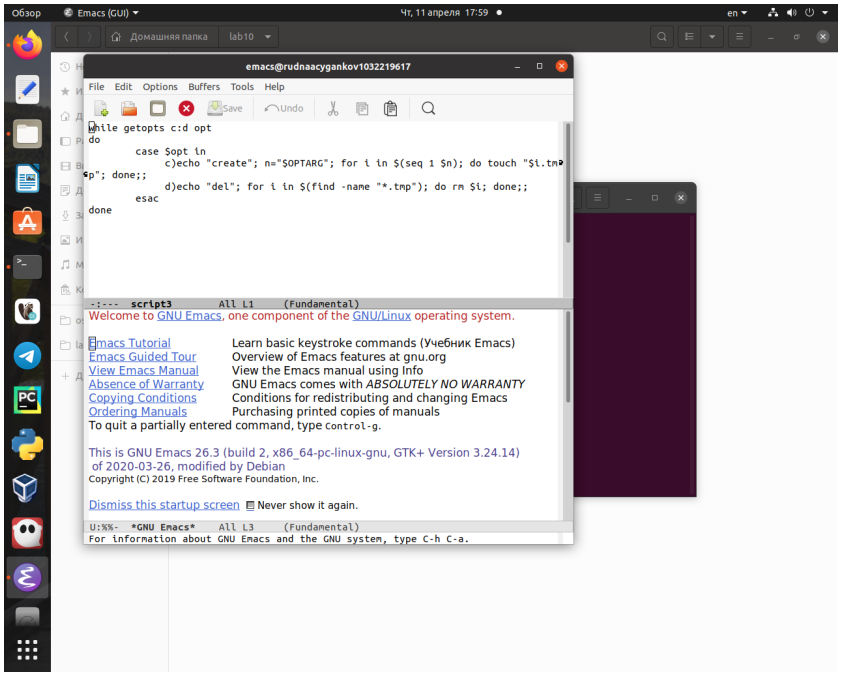
- 

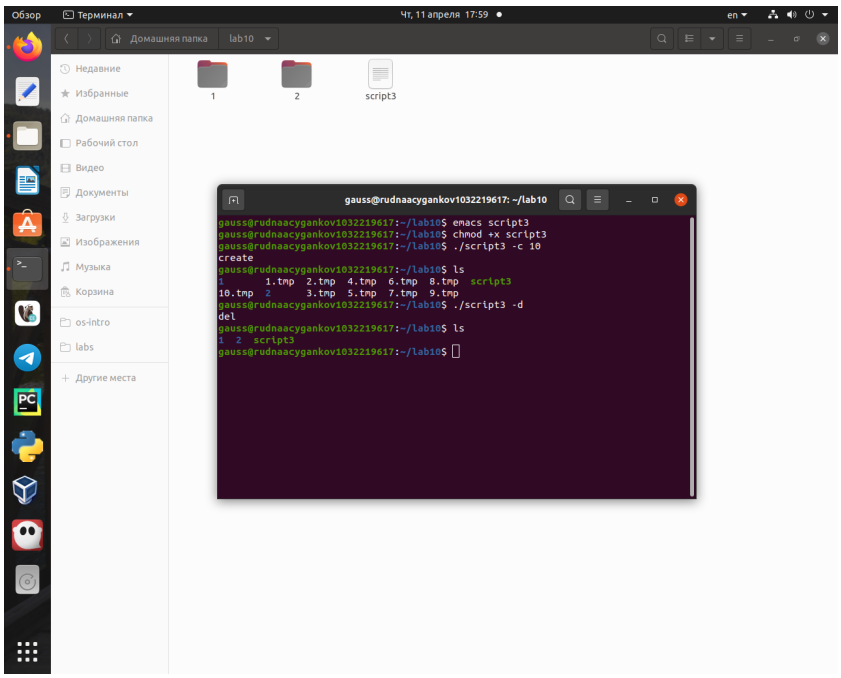
2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции exit(n), передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено (рис. @fig:003, @fig:004, @fig:005).

- 
Обзор
Текстовый редактор
Чт, 11 апреля 18:18
ru
Открыть
lab10.c
~/lab10/2
Сохранить
1 #include <iostream>
2 using namespace std;
3
4 int main(int argc, char *argv[]) {
5 if (atoi(argv[1])>0) exit(1);
6 else if (atoi(argv[1])==0) exit(2);
7 else exit(0);
8 return 0;
9 }
Парная скобка найдена в строке: 7
C
Ширина табуляции: 8
Стр 7, Стлб 20
ВСТ
- 
Обзор
Emacs (GUI)
Чт, 11 апреля 17:49
en
Домашняя папка
lab10
Недавние
emacs@rudnaacygankov1032219617
File Edit Options Buffers Tools Help
Save
Undo
RES=result
SRC=lab10.c
if ["$SRC" -nt "$RES"]
then
echo "er"
g++ -o $RES $SRC
fi
./$RES $1
case $? in
0) echo "member < 0";;
1) echo "member > 0";;
2) echo "member == 0";;
esac
-:--- script2 All L13 (Fundamental)
Welcome to GNU Emacs, one component of the GNU/Linux operating system.
Emacs Tutorial Learn basic keystroke commands (Учебник Emacs)
Emacs Guided Tour Overview of Emacs features at gnu.org
View Emacs Manual View the Emacs manual using Info
Absence of Warranty GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions Conditions for redistributing and changing Emacs
Ordering Manuals Purchasing printed copies of manuals
To quit a partially entered command, type Control-g.
This is GNU Emacs 26.3 (build 2, x86_64-pc-linux-gnu, GTK+ Version 3.24.14)
of 2020-03-26, modified by Debian
Copyright (C) 2019 Free Software Foundation, Inc.
Dismiss this startup screen
Never show it again.

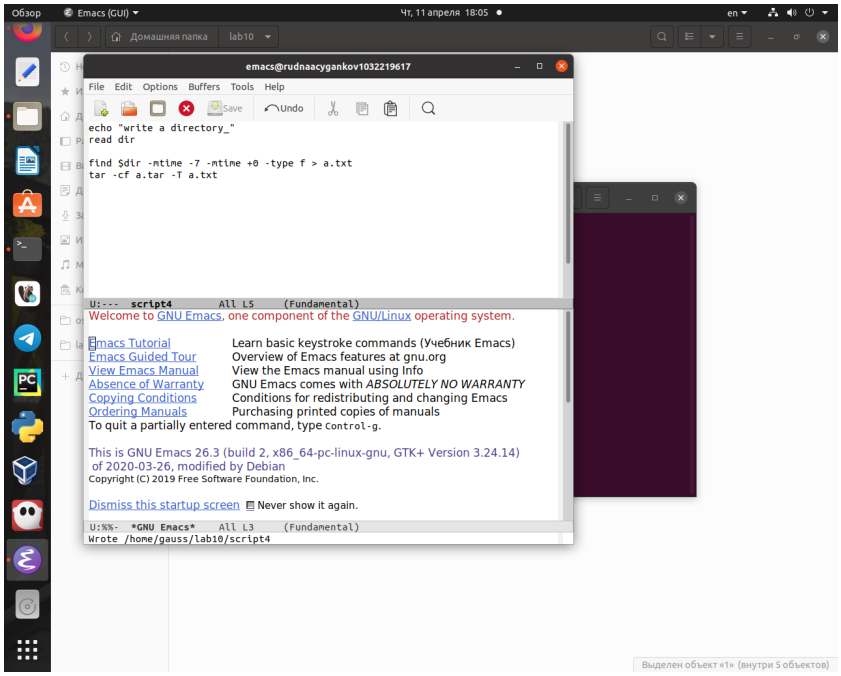
- 

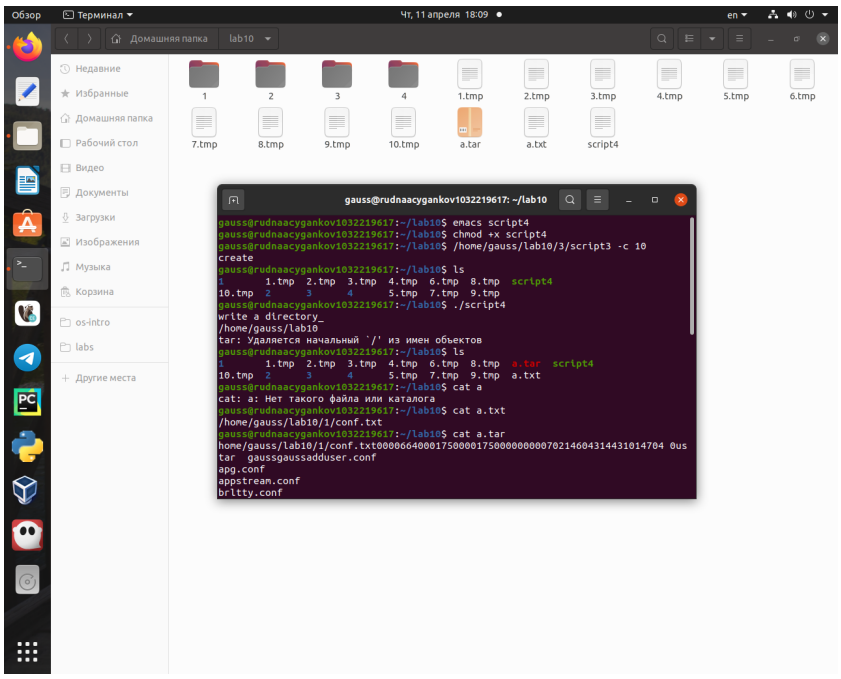
3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (например 1.tmp, 2.tmp, 3.tmp,4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют) (рис. @fig:006, @fig:007).

- 

- 

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) (рис. @fig:008, @fig:009).

- 

- 
Обзор
Терминал
Чт, 11 апреля 18:09
en
Домашняя папка
lab10
Недавние
Избранные
Домашняя папка
Рабочий стол
Видео
Документы
Загрузки
Изображения
Музыка
Корзина
os-intro
labs
Другие места
1
2
3
4
1.tmp
2.tmp
3.tmp
4.tmp
5.tmp
6.tmp
7.tmp
8.tmp
9.tmp
10.tmp
a.tar
a.txt
script4
gauss@rudnaacygankov1032219617: ~/lab10
gauss@rudnaacygankov1032219617:~/lab10$ emacs script4
gauss@rudnaacygankov1032219617:~/lab10$ chmod +x script4
gauss@rudnaacygankov1032219617:~/lab10$ /home/gauss/lab10/3/script3 -c 10
create
gauss@rudnaacygankov1032219617:~/lab10$ ls
1 1.tmp 2.tmp 3.tmp 4.tmp 6.tmp 8.tmp script4
10.tmp 2 3 4 5.tmp 7.tmp 9.tmp
gauss@rudnaacygankov1032219617:~/lab10$ ./script4
write a directory_
/home/gauss/lab10
tar: Удаляется начальный `/' из имен объектов
gauss@rudnaacygankov1032219617:~/lab10$ ls
1 1.tmp 2.tmp 3.tmp 4.tmp 6.tmp 8.tmp a.tar script4
10.tmp 2 3 4 5.tmp 7.tmp 9.tmp a.txt
gauss@rudnaacygankov1032219617:~/lab10$ cat a
cat: a: Нет такого файла или каталога
gauss@rudnaacygankov1032219617:~/lab10$ cat a.txt
/home/gauss/lab10/1/conf.txt
gauss@rudnaacygankov1032219617:~/lab10$ cat a.tar
home/gauss/lab10/1/conf.txt0000664000175000017500000000007021460431431014704 0us
tar gaussgaussadduser.conf
apg.conf
appstream.conf
brltty.conf

5 Листинги

1) script1

```
while getopts "i:o:p:cn" opt
do
    case $opt in
            i)inputfile="$OPTARG";;
        o)outputfile="$OPTARG";;
        p)sample="$OPTARG";;
        c)reg="";;
        n)line="";;
    esac
done

grep -n "$sample" "$inputfile" > "$outputfile"
```

2) script2

```
RES=result
SRC=lab10.c

if [ "$SRC" -nt "$RES" ]
then
    echo "er"
    g++ -o $RES $SRC
```

```
fi

./$RES $1

case $? in
    0) echo "member < 0";;
    1) echo "member > 0";;
    2) echo "member == 0";;
esac
```

3) script3

```
while getopts c:d opt
do
    case $opt in
        c)echo "create"; n="$OPTARG"; for i in $(seq 1 $n); do touch "$i.tmp"; done;;
        d)echo "del"; for i in $(find -name "*.tmp"); do rm $i; done;;
    esac
done
```

4) script4

```
echo "write a directory_"
read dir

find $dir -mtime -7 -mtime +0 -type f > a.txt
tar -cf a.tar -T a.txt
```

6 Контрольные вопросы

1. Каково предназначение команды 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 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;
- $* \cdot$ 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. Какие операторы управления действиями вы знаете?

- Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости отрезультатов 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 управляю- щие 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. Какие операторы используются для прерывания цикла?

- 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. Для чего нужны команды false и true?

   - 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).

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

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

7. Объясните различия между конструкциями while и until.

   - 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

7 Выводы

В процессе выполнения данной лабораторной работы я изучил основы программирования в оболочке ОС UNIX. Научился применять ветвление и циклы в написании скриптов.

Список литературы

Руководство к лабораторной работе