

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №11

дисциплина: Операционные системы

Студент: Кармацкий Никита Сергеевич

Группа: НФИбд-01-21

Москва

2022 г.

Цель работы:

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

Основные этапы выполнения работы

1. Используя команды `getopts` `grep`, написали 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

Напишем программу

Листинг программы:

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

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

Работа программы:

```
[nskarmackiyj@fedora tmp]$ ./lab11 -i lab11 -o outp.txt -p n etconf -c -n
[nskarmackiyj@fedora tmp]$ cat outp.txt
1:while getopts "i:o:p:cn" opt
3:case $opt in
4:i)inputfile="$OPTARG";;
6:p)shablon="$OPTARG";;
8:n)number="";;
10:done
12:grep -n "$shablon" "$inputfile" > "$outputfile"
```

Рис.1 Работа программы

2. Написали 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.

Листинг программы:

```
cho "Insert number:"

read n

if [ $n -gt 0 ]

then echo ">0"

elif [ $n -eq 0 ]

then echo "=0"

else echo "<0"

fi
```

Работа программы:

```
[nskarmackiyj@fedora tmp]$ ./lab11_2
Insert number:
3
>0
[nskarmackiyj@fedora tmp]$ ./lab11_2
Insert number:
0
=0
[nskarmackiyj@fedora tmp]$ ./lab11_2
Insert number:
-1
<0
```

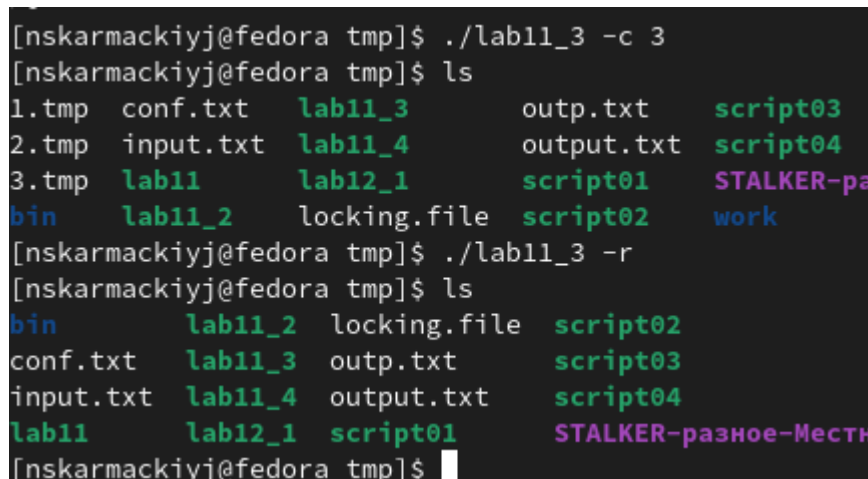
Рис.2 Работа командного файла файла

3. Написали 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).

Листинг программы:

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

Работа программы:



```
[nskarmackiyj@fedora tmp]$ ./lab11_3 -c 3
[nskarmackiyj@fedora tmp]$ ls
1.tmp  conf.txt  lab11_3  outp.txt  script03
2.tmp  input.txt lab11_4  output.txt script04
3.tmp  lab11     lab12_1  script01  STALKER-pa
bin    lab11_2  locking.file script02  work
[nskarmackiyj@fedora tmp]$ ./lab11_3 -r
[nskarmackiyj@fedora tmp]$ ls
bin    lab11_2  locking.file script02
conf.txt lab11_3  outp.txt  script03
input.txt lab11_4  output.txt script04
lab11    lab12_1  script01  STALKER-разное-Местн
[nskarmackiyj@fedora tmp]$
```

Рис.3 Работа командного файла файла

4. Написали командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировали его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовали команду find).

Листинг программы:

```

echo "Insert directory: "

read directory

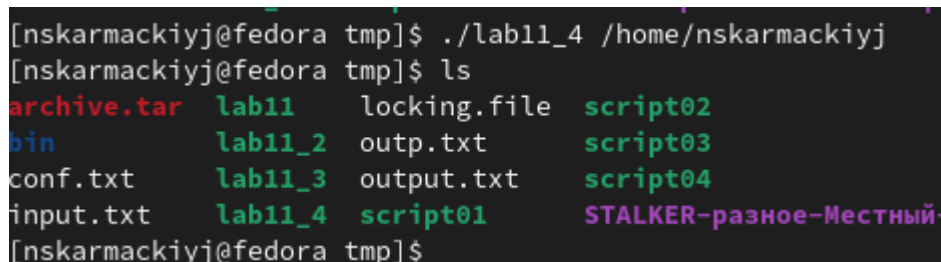
echo "Insert file format: "

read format

find $direcrory -name "$format" -type f | wc -l

```

Работа программы:



```

[nskarmackiyj@fedora tmp]$ ./lab11_4 /home/nskarmackiyj
[nskarmackiyj@fedora tmp]$ ls
archive.tar  lab11      locking.file  script02
bin          lab11_2    outp.txt      script03
conf.txt     lab11_3    output.txt    script04
input.txt    lab11_4    script01      STALKER-разное-Местный
[nskarmackiyj@fedora tmp]$

```

Рис.4 Работа командного файла файла

Вывод

Мы изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Ответы на контрольные вопросы

1. Весьма необходимой при программировании является команда `getopts`, которая 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`. Предположим, необходимо распознать командную строку следующего формата: `testprog -ifile_in.txt -ofile_out.doc -L -t -r` Вот как выглядит использование оператора `getopts` в этом случае: `while getopts o:i:Ltr optletter do case $optletter in o) oflag=1; oval=$OPTARG;; i) iflag=1; ival=$OPTARG;; L) Lflag=1;; t) tflag=1;; r) rflag=1;; *) echo Illegal option $optletter esac done` Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента (будет равна `file_in.txt` для опции `i` и `file_out.doc` для опции `o`). `OPTIND` является числовым индексом на упомянутый аргумент. Функция

getopts также понимает переменные типа массив, следовательно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

2. 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. 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 Вам 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 дела являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда.
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 перестает быть правильным. Пример бесконечного цикла `while`, с прерыванием в момент, когда файл перестает существовать: `while true do if [! -f $file] then break fi sleep 10 done`
5. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.
6. Введенная строка означает условие существования файла `man$s/$i.$s`
7. Если речь идет о 2-х параллельных действиях, то это `while`. когда мы показываем, что сначала делается 1-е действие. потом оно заканчивается при наступлении 2-го действия, применяем `until`.