

Шаблон отчёта по лабораторной работе N.11

Кристина Эспиноса

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	10
5	Контрольные вопросы	11

Список иллюстраций

3.1 7

Список таблиц

1 Цель работы

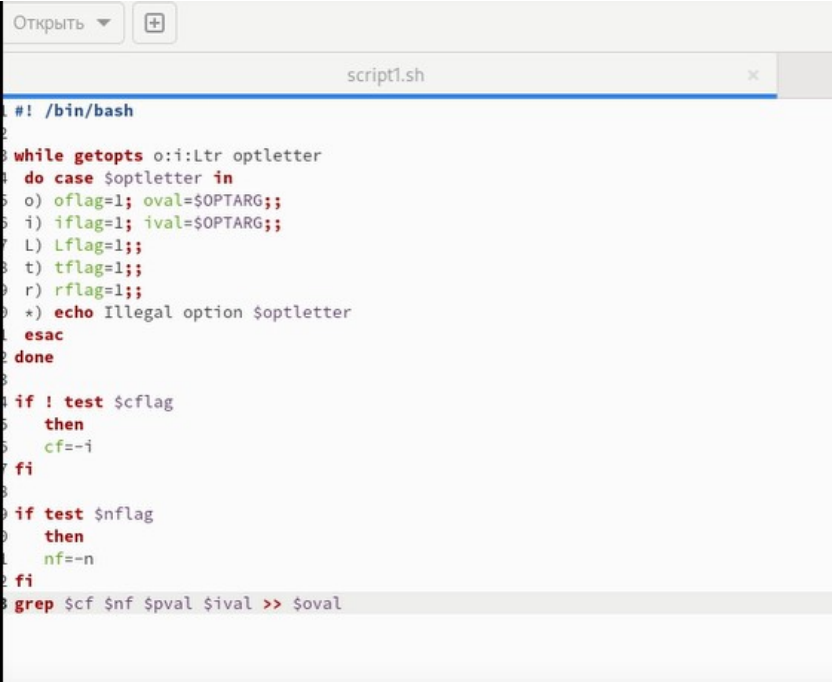
Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов

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; `-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 ∞ (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 `tag` 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. Script 1: Сначала создала два файла «input.txt», где вы набрали 4 слова, и «output.txt», пустой файл. В одном файлеб котовый я назвала Script1.sh необходимо написать командный файл, который анализирует командную строку с ключами и ищет в указанном файле строки с ключом p.



```
#!/bin/bash
1
2
3 while getopts o:i:Ltr optletter
4 do case $optletter in
5 o) oflag=1; oval=$OPTARG;;
6 i) iflag=1; ival=$OPTARG;;
7 L) Lflag=1;;
8 t) tflag=1;;
9 r) rflag=1;;
10 *) echo Illegal option $optletter
11 esac
12 done
13
14 if ! test $cflag
15 then
16 cf=-i
17 fi
18
19 if test $nflag
20 then
21 nf=-n
22 fi
23
24 grep $cf $nf $pval $ival >> $oval
```

Рис. 3.1:

Затем с помощью следующей команды скопируйте все слова, содержащие «os»,
и переместите их в файл output.txt.

```
[cristinaespino@vmehspino lab1]$ bash script1.sh -p os -i input.txt -o output.txt -c -n
[cristinaespino@vmehspino lab1]$
```

```
script.sh  x  input.txt
1 Julia
2 Maria
3 Carlos
4 Cristina
```

```
script.sh  x  input.txt
1 Carlos
```

2. Script 2: В этой задаче у меня не получилось. Но я начал с написания 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, проана- лизировав с помощью команды `$?`, выдать сообщение о том, какое чис-

```
Открыть  +
script
1 #! /bin/bash
2
3 gcc -o cscript script2.c
4 ./cscript
5 case $? in
6 0) echo "Number = 0";;
7 2) echo "Number less than 0";;
8 3) echo echo "Number bigger than 0";;
9 esac
```

ло было введено.


```

1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int main() {
5     int n
6     printf ("Enter a number: ");
7     scanf ("%d" , &n);
8     if (n>0) {
9         exit(1);
10    }else if (n==0){
11        exit(0);
12    } else {
13        exit(2);
14    }
15
16 }

```

3. Script 3:

Написала 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 командный файл должен уметь удалять все созданные им файлы (если они существуют).

```

1 #! /bin/bash
2 for ((i=1; i<=$*; i++))
3 do
4     if test -f "$i".tmp
5     then rm "$i".tmp
6     else touch "$i".tmp"
7     fi
8 done

```

```

[cristinaespino@vmehspino lab11]$ bash script3.sh 3
[cristinaespino@vmehspino lab11]$ ls
0-1.tmp  1.tmp  2.tmp  3.tmp  input.txt  output.txt  script1.sh  scr
[cristinaespino@vmehspino lab11]$ rm -f $i.tmp

```

```

[cristinaespino@vmehspino lab11]$ bash script3.sh 3
[cristinaespino@vmehspino lab11]$ ls
input.txt  output.txt  script1.sh  script2.c  script2.sh  script3.c  script3.sh
[cristinaespino@vmehspino lab11]$

```

4. Script 4: В последнем шаге, нужно 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

```

1 #! /bin/bash
2
3 find . -mtime -7 -mtime +0 -type f > FILES.txt
4 tar -cf archive.tar -T FILES.txt

```

find).

4 Выводы

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

5 Контрольные вопросы

1. Каково предназначение команды getopt?

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 -infile_in.txt -outfile_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. 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?

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

`true` : всегда возвращает 0 в качестве кода выхода.

`false` : всегда возвращает 1 в качестве кода выхода.

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

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

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

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`.