

Лабораторная работа №11

**Программирование в командном процессоре ОС UNIX. Ветвления и
циклы**

Лебедева Ольга Андреевна

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Вывод	12
4	Контрольные вопросы	13

Список иллюстраций

2.1	Код скрипта1	7
2.2	Исполняем файл	7
2.3	Результат работы	8
2.4	Код скрипта2	9
2.5	Результат работы	9
2.6	Код скрипта3	10
2.7	Результат работы	10
2.8	Команда удаления	10
2.9	Результат работы	10
2.10	Код скрипта4	11
2.11	Результат работы	11

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 Теоретическое введение

Цикл `for` - это оператор языка программирования `bash`, который позволяет многократно выполнять код. Цикл `for` классифицируется как оператор итерации, то есть это повторение процесса в сценарии `bash`. Например, вы можете запустить команду или задачу UNIX 5 раз или прочитать и обработать список файлов с помощью цикла `for`.

Цикл `while` используется для выполняет заданный набор команд неизвестное число раз до тех пор , как данное условие принимает значение истинно.

1. Используя команды `getopts` `grep`, написали командный файл, который анализирует командную строку с ключами: (рис. 2.1)

- `-iinputfile` — прочитать данные из указанного файла;
- `-ooutputfile` — вывести данные в указанный файл;
- `-р`шаблон — указать шаблон для поиска;
- `-C` — различать большие и малые буквы;
- `-n` — выдавать номера строк.

```
while getopts "i:o:p:C:n" 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"
```

Рис. 2.1: Код скрипта1

Сделаем файл исполняемым. (рис. 2.2)

```
palebedeva@dk8n52 ~ $ chmod +x lab11_1
palebedeva@dk8n52 ~ $ ./lab11_1 -i conf.txt -o output.txt -p n etconf -C -n
palebedeva@dk8n52 ~ $ emacs output.txt
```

Рис. 2.2: Исполняем файл

Проверим результат работы программы. (рис. 2.3)

```
1:appstream.conf
2:brltty.conf
3:ca-certificates.conf
4:cachedfilesd.conf
5:cfg-update.conf
6:cpufreq-bench.conf
7:dconf
8:dhcpcd.conf
9:dispatch-conf.conf
10:dley-na-server-service.conf
11:dnsmasq.conf
12:e2fsck.conf
13:e2scrub.conf
14:etc-update.conf
15:fluidsynth.conf
16:fuse.conf
```

Рис. 2.3: Результат работы

2. Написали на языке 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 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. 2.4) (рис. 2.5)


```

echo "Insert number"
read n
if [ $n -gt 0 ]
then echo ">0"
elif [ $n -eq 0 ]
then echo "=0"
else echo "<0"
fi

```

Рис. 2.4: Код скрипта2

```

oalebedeva@dk8n52 ~ $ chmod +x lab11_2
oalebedeva@dk8n52 ~ $ ./lab11_2
Insert number
7
>0
oalebedeva@dk8n52 ~ $ ./lab11_2
Insert number
0
=0
oalebedeva@dk8n52 ~ $ ./lab11_2
Insert number
-12
<0

```

Рис. 2.5: Результат работы

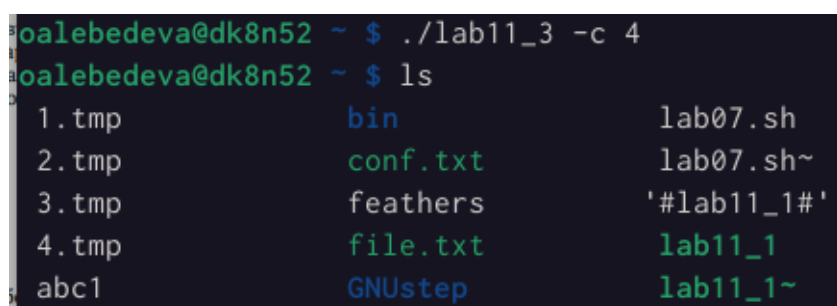
3. Написали командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до ∞ (например 1.tmp, 2.tmp, 3.tmp, 4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы

командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют). (рис. 2.6)

```
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
```

Рис. 2.6: Код скрипта3

Задали программе создать 4 файла tmp. (рис. 2.7)



The terminal shows the user running the script `./lab11_3 -c 4`. After running `ls`, the output shows four new files: `1.tmp`, `2.tmp`, `3.tmp`, and `4.tmp`, along with other existing files like `bin`, `conf.txt`, `feathers`, `file.txt`, `abc1`, `GNUstep`, `lab07.sh`, and `lab11_1`.

Рис. 2.7: Результат работы

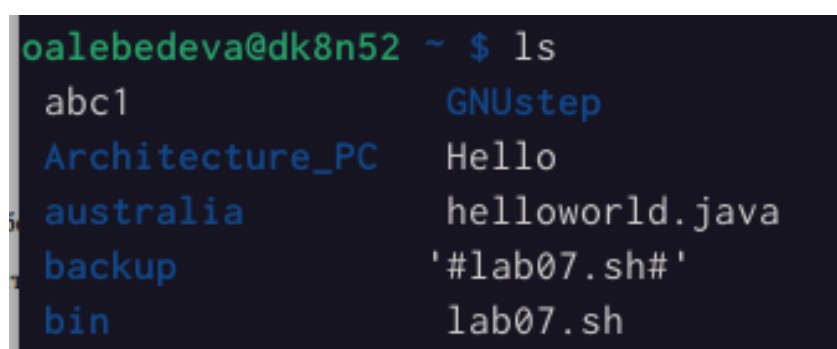
Теперь одной командой удалим созданные файлы. (рис. 2.8)



The terminal shows the user running the script `./lab11_3 -r` to remove the files created in the previous step.

Рис. 2.8: Команда удаления

Проверяем содержание домашнего каталога, файлы были удалены. (рис. 2.9)



The terminal shows the user running `ls` again. The output now shows only the files that were not removed: `abc1`, `Architecture_PC`, `australia`, `backup`, `bin`, `GNUstep`, `Hello`, `helloworld.java`, and `lab07.sh`. The four `.tmp` files are no longer present.

Рис. 2.9: Результат работы

4. Написали 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. 2.10)

```
while getopts ":p:" opt; do
case $opt in
p)dir="$OPTARG";;
esac
done

find $dir -mtime +0 -mtime -7 -print0 | xargs -0 tar -cf archive.tar
```

Рис. 2.10: Код скрипта4

Файлы, которые были изменены менее недели тому назад: (рис. 2.11)

backup	lab07.sh	my_os	script3~
bin	lab07.sh~	my_os1	script4
conf.txt	'#lab11_1#'	output.txt	ski.plases

Рис. 2.11: Результат работы

3 Вывод

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

4 Контрольные вопросы

1. Каково предназначение команды `getopts`?

Команда `getopts` выводит сообщение об ошибке в стандартный протокол, когда встречается опция, не включенная в цепочку_опций.

2. Какое отношение метасимволы имеют к генерации имён файлов?

При генерации имен используют метасимволы:

? один произвольный символ;

[...] любой из символов, указанных в скобках перечислением и/или с указанием диапазона;

`cat f*` выдаст все файлы каталога, начинающиеся с “f”;

`cat f` выдаст все файлы, содержащие “f”;

`cat program.?` выдаст файлы данного каталога с однобуквенными расширениями, скажем “`program.c`” и “`program.o`”, но не выдаст “`program.com`”;

`cat [a-d]*` выдаст файлы, которые начинаются с “a”, “b”, “c”, “d”. Аналогичный эффект дадут и команды “`cat [abcd]`” и “`cat [bdac]`”.

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.

4. Какие операторы используются для прерывания цикла?

Оператор `break` используется для выхода из текущего цикла. Оператор `continue` используется для выхода из текущей итерации цикла и начала следующей итерации.

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?

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

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

Цикл `while` выполняет тело цикла пока условие истинно.

Цикл `until` выполняет тело цикла пока условие ложно. Другими словами цикл `until` выполняется до тех пор пока условие не станет истинным.