

Лабораторная работа №12

Дисциплина: Операционные системы

Галиев Казиз Жарылкасымович

Содержание

Цель работы	5
Выполнение лабораторной работы	6
Выводы	11
Контрольные вопросы	12
Список литературы	14

Список иллюстраций

0.1	Первый командный файл	6
0.2	Результат выполнения первого командного файла	7
0.3	Программа на языке Си	7
0.4	Текст второго командного файла	8
0.5	Результат выполнения второго командного файла	8
0.6	Текст третьего командного файла	9
0.7	Результат выполнения третьего командного файла	9
0.8	Текст четвертого командного файла	9

Список таблиц

Цель работы

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

Выполнение лабораторной работы

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-i` `inputfile` — прочитать данные из указанного файла; `-o` `outputfile` — 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 `-p` (рис. @fig:001), (рис. @fig:002).

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

Рис. 0.1: Первый командный файл

```
[kzgaliev@fedora report]$ emacs probl
[kzgaliev@fedora report]$ ./probl -i conf.txt -o output.txt -p h -c -n
[kzgaliev@fedora report]$ ls
81b conf.txt image Makefile output.txt pandoc probl probl- report.md
[kzgaliev@fedora report]$ cat output.txt
1:anthy-unicode.conf
5:chrony.conf
13:host.conf
27:nsswitch.conf
41:Trolltech.conf
45:usb_modeswitch.conf
47:whois.conf
[kzgaliev@fedora report]$
```

Рис. 0.2: Результат выполнения первого командного файла

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 (рис. @fig:003) .

```
#include <iostream>
using namespace std;
int main(int argument, char *arg[]){
    if (atoi(arg[1]) > 0) {
        exit(1);
    }
    else if (atoi(arg[1]) == 0){
        exit(2);
    }
    else {
        exit(3);
    }
    return 0;
}
```

Рис. 0.3: Программа на языке Си

Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено (рис. @fig:004) , (рис. @fig:005) .

```
#!/bin/bash
CC=g++
EXEC=compare
SRC=compare.cpp
if [ "SRC" -nt "$EXEC" ]
then
echo "Rebuilding $EXEC ....."
$CC -o $EXEC $SRC
fi
./$EXEC $1
ec=$?
if [ "$ec" == "1" ]
then
echo "argument > 0"
fi
if [ "$ec" == "2" ]
then
echo "argument = 0"
fi
if [ "$ec" == "3" ]
then
echo "argument < 0"
fi
```

Рис. 0.4: Текст второго командного файла

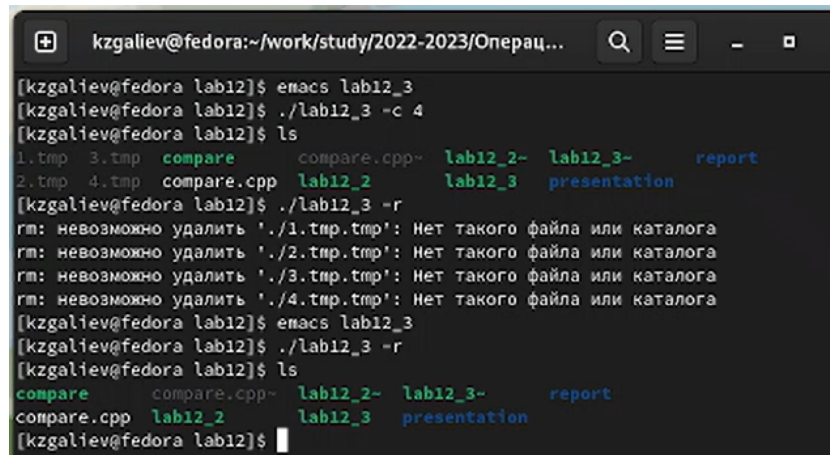
```
[kzgaliev@fedora lab12]$ ./lab12_2 -9
argument < 0
[kzgaliev@fedora lab12]$ ./lab12_2 8
argument > 0
[kzgaliev@fedora lab12]$ ./lab12_2 0
argument = 0
```

Рис. 0.5: Результат выполнения второго командного файла

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) , (рис. @fig:007)


```
#!/bin/bash
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. 0.6: Текст третьего командного файла



```
kzgaliev@fedora:~/work/study/2022-2023/Операц...
[kzgaliev@fedora lab12]$ emacs lab12_3
[kzgaliev@fedora lab12]$ ./lab12_3 -c 4
[kzgaliev@fedora lab12]$ ls
1.tmp  3.tmp  compare  compare.cpp~  lab12_2~  lab12_3~  report
2.tmp  4.tmp  compare.cpp  lab12_2  lab12_3  presentation
[kzgaliev@fedora lab12]$ ./lab12_3 -r
rm: невозможно удалить './1.tmp.tmp': Нет такого файла или каталога
rm: невозможно удалить './2.tmp.tmp': Нет такого файла или каталога
rm: невозможно удалить './3.tmp.tmp': Нет такого файла или каталога
rm: невозможно удалить './4.tmp.tmp': Нет такого файла или каталога
[kzgaliev@fedora lab12]$ emacs lab12_3
[kzgaliev@fedora lab12]$ ./lab12_3 -r
[kzgaliev@fedora lab12]$ ls
compare  compare.cpp~  lab12_2~  lab12_3~  report
compare.cpp  lab12_2  lab12_3  presentation
[kzgaliev@fedora lab12]$
```

Рис. 0.7: Результат выполнения третьего командного файла

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), (рис. @fig:009) .

```
#!/bin/bash
while getopts :d opt;
do case $opt in
d)dir="$OPTARG"
esac
done
find $dir -mtime -7 -mtime +0 -type f > arch.txt | tar -cf archive_lab12_4.tar -T arch.txt
```

Рис. 0.8: Текст четвертого командного файла

```
[kzgaliev@fedora lab12]$ emacs lab12_4
[kzgaliev@fedora lab12]$ ./lab12_4 /home
[kzgaliev@fedora lab12]$ ./lab12_4 /home
[kzgaliev@fedora lab12]$ ls
archive_lab12_4.tar  compare.cpp  lab12_2-  lab12_4-  report
arch.txt             compare.cpp~ lab12_3   lab12_4~
compare              lab12_2     lab12_3-  presentation
```

Выводы

В результате проделанной лабораторной работы я изучил основы программирования в оболочке ОС Unix и научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Контрольные вопросы

1. Каково предназначение команды `getopts`?

Осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных.

2. Какое отношение метасимволы имеют к генерации имён файлов?

Такие символы, как `' < > * ? | " &`, являются метасимволами и имеют для командного процессора специальный смысл. Снятие специального смысла с метасимвола называется экранированием метасимвола. Экранирование может быть осуществлено с помощью предшествующего метасимволу символа `\`, который, в свою очередь, является метасимволом. Для экранирования группы метасимволов нужно заключить её в одинарные кавычки. Строка, заключённая в двойные кавычки, экранирует все метасимволы, кроме `$, ', , "`.

3. Какие операторы управления действиями вы знаете?

Язык программирования `bash` предоставляет возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`, `until`. 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.

4. Какие операторы используются для прерывания цикла?

Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов.

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

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), используются только совместно с управляющими конструкциями языка программирования `bash`

6. Объясните различия между конструкциями `while` и `until`. 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. В обобщённой форме оператор цикла `until` выглядит следующим образом: `until` список-команд `do` список-команд `done`.

Список литературы