

Лабораторная работа №11

Операционные системы

Голощапов Ярослав

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	8
4	Выполнение лабораторной работы	9
5	Выводы	15

Список иллюстраций

4.1	Создание файлов	9
4.2	Тело программы	10
4.3	Запуск программы	10
4.4	Исходный текстовый файл	11
4.5	Результат работы программы	11
4.6	Код основной программы	11
4.7	Код подпрограммы	12
4.8	Проверка работы	12
4.9	Создание третьего файла	13
4.10	Запуск программы	13
4.11	Созданные файлы	13
4.12	Файл 4	13
4.13	Запуск	14
4.14	Результат работы программы	14

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;
- `-ршаблон` — указать шаблон для поиска;
- `-С` — различать большие и малые буквы;
- `-п` — выдавать номера строк.

а затем ищет в указанном файле нужные строки, определяемые ключом `-р`.

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

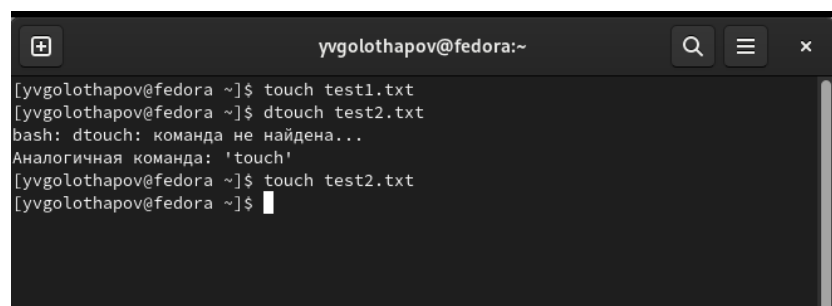
запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).

3 Теоретическое введение

При перечислении имён файлов текущего каталога можно использовать следующие символы: – * — соответствует произвольной, в том числе и пустой строке; – ? — соответствует любому одинарному символу; – [c1-c1] — соответствует любому символу, лексикографически находящемуся между символами c1 и c2. Например, – echo * — 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. 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 \$, ' , , ". Например, – echo * выведет на экран символ , – echo ab'|'cd выведет на экран строку ab|*cd.

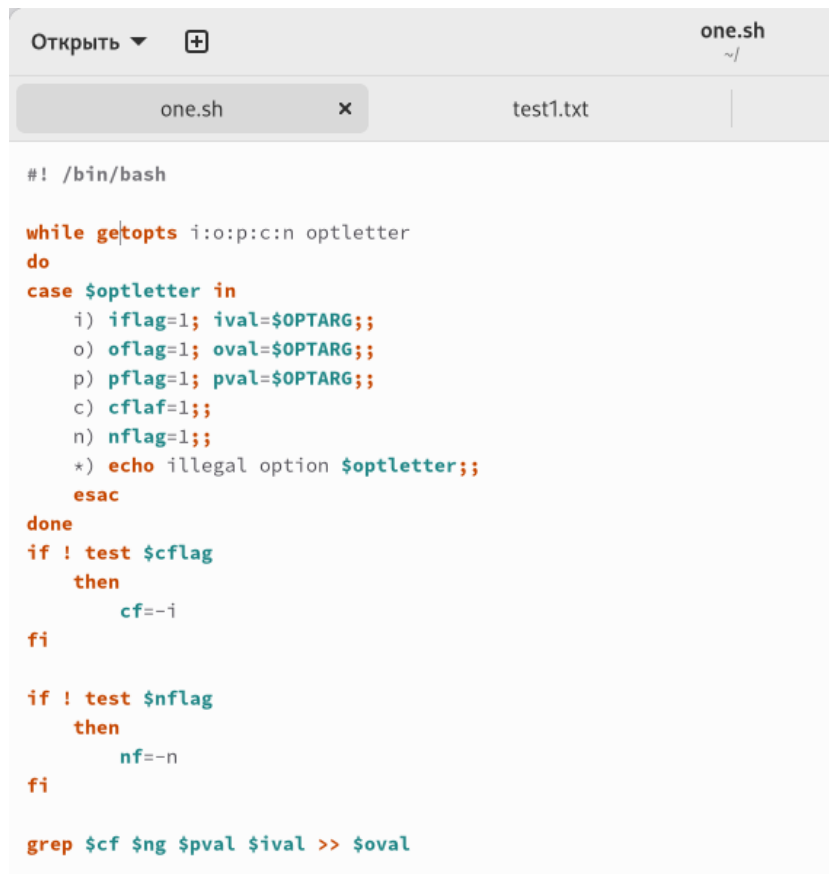
4 Выполнение лабораторной работы

Я создал файл one.sh для первой программы, а также текстовые файлы, откуда будет браться информация (рис. fig. 4.1). Написал код программы (рис. fig. 4.2).



```
yvgolothapov@fedora:~  
[yvgolothapov@fedora ~]$ touch test1.txt  
[yvgolothapov@fedora ~]$ dtouch test2.txt  
bash: dtouch: команда не найдена...  
Аналогичная команда: 'touch'  
[yvgolothapov@fedora ~]$ touch test2.txt  
[yvgolothapov@fedora ~]$
```

Рис. 4.1: Создание файлов



```
#!/bin/bash

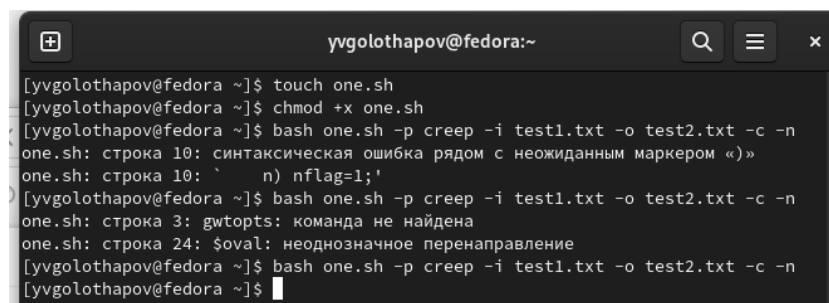
while getopts i:o:p:c:n optletter
do
case $optletter in
    i) iflag=1; ival=$OPTARG;;
    o) oflag=1; oval=$OPTARG;;
    p) pflag=1; pval=$OPTARG;;
    c) cflag=1;;
    n) nflag=1;;
    *) echo illegal option $optletter;;
    esac
done
if ! test $cflag
then
    cf=-i
fi

if ! test $nflag
then
    nf=-n
fi

grep $cf $ng $pval $ival >> $oval
```

Рис. 4.2: Тело программы

Запустил программу и проверил, работает ли она (рис. fig. 4.3). Программа работает (рис. fig. 4.4), (рис. fig. 4.5).



```
yvgolothapov@fedora:~
[yvgolothapov@fedora ~]$ touch one.sh
[yvgolothapov@fedora ~]$ chmod +x one.sh
[yvgolothapov@fedora ~]$ bash one.sh -p creep -i test1.txt -o test2.txt -c -n
one.sh: строка 10: синтаксическая ошибка рядом с неожиданным маркером «)»
one.sh: строка 10: `n) nflag=1;`
[yvgolothapov@fedora ~]$ bash one.sh -p creep -i test1.txt -o test2.txt -c -n
one.sh: строка 3: gwtOpts: команда не найдена
one.sh: строка 24: $oval: неоднозначное перенаправление
[yvgolothapov@fedora ~]$ bash one.sh -p creep -i test1.txt -o test2.txt -c -n
[yvgolothapov@fedora ~]$
```

Рис. 4.3: Запуск программы

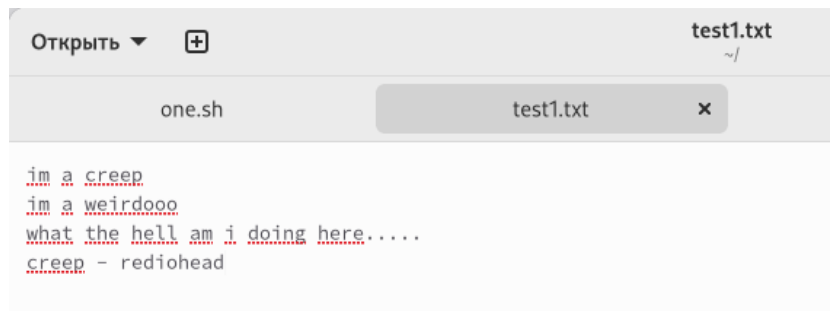


Рис. 4.4: Исходный текстовый файл



Рис. 4.5: Результат работы программы

Создал файлы для второй программы: two.sh и two.c и заполнил код (рис. fig. 4.6), (рис. fig. 4.7).

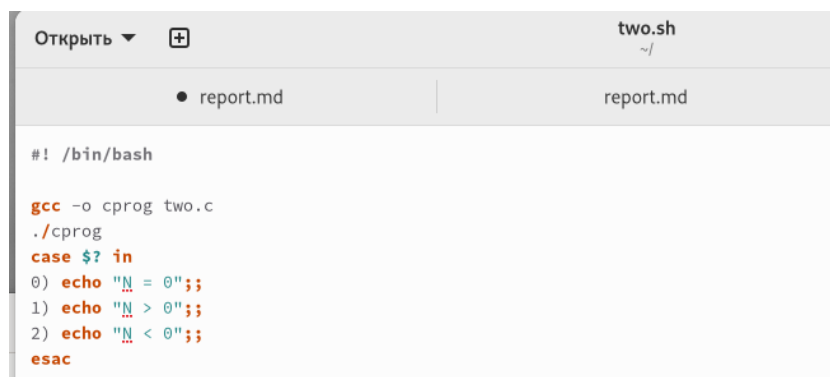


Рис. 4.6: Код основной программы

```
Открыть ▾ + two.c ~/
● report.md

#include <stdio.h>
#include <stdlib.h>

int main () {
    int n;
    printf("Enter n: ");
    scanf("%d", &n);
    if(n > 0) {
        exit(1);
    }
    else if (n == 0) {
        exit(0);
    }
    else {
        exit(2);
    }
}
```

Рис. 4.7: Код подпрограммы

Проверил работу командного файла (рис. fig. 4.8).

```
two.sh: строка 4: ./cprog: Нет такого файла или каталога
[yvgolothapov@fedora ~]$ bash two.sh
Enter n: 10
N > 0
[yvgolothapov@fedora ~]$ bash two.sh
Enter n: 0
N = 0
[yvgolothapov@fedora ~]$ bash two.sh
Enter n: -5
N < 0
[yvgolothapov@fedora ~]$
```

Рис. 4.8: Проверка работы

Создал третий файл three.sh и написал код (рис. fig. 4.9).

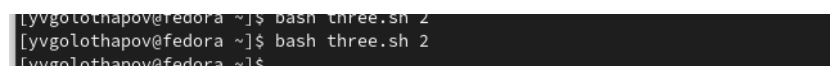


```
Открыть ▾ + three.sh ~/
● report.md

#!/bin/bash
for ((i=1; i<=$*; i++))
do
if test -f "$i".tmp
then rm "$i".tmp
else touch "$i.tmp"
fi
done
```

Рис. 4.9: Создание третьего файла

Запустил файл. Программа работает (рис. fig. 4.10), (рис. fig. 4.11).



```
[yvgolothapov@fedora ~]$ bash three.sh 2
[yvgolothapov@fedora ~]$ bash three.sh 2
[yvgolothapov@fedora ~]$
```

Рис. 4.10: Запуск программы

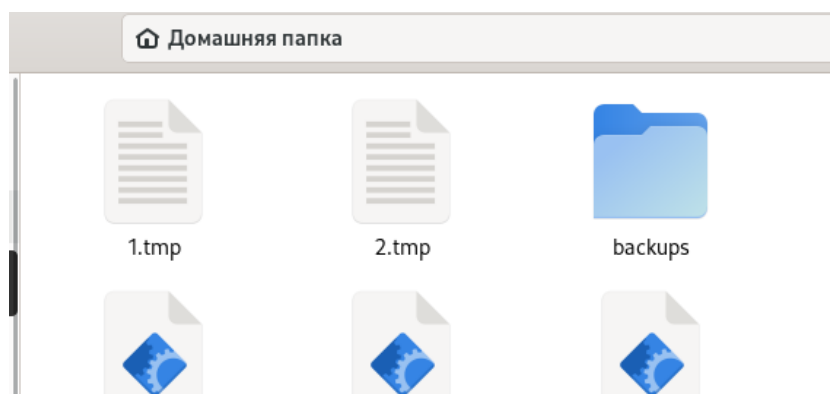


Рис. 4.11: Созданные файлы

Создал файл four.sh (рис. fig. 4.12).



```
Открыть ▾ + four.sh ~/
● report.md

#!/bin/bash

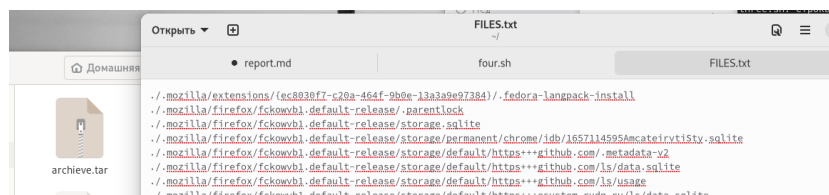
find $* -mtime -7 -mtime +0 -type f > FILES.txt
tar -cf archive.tar -T FILES.txt
```

Рис. 4.12: Файл 4

Программа работает (рис. fig. 4.13), (рис. fig. 4.14).

```
[yvgolothapov@fedora ~]$ touch four.sh  
[yvgolothapov@fedora ~]$ chmod +x four.sh  
[yvgolothapov@fedora ~]$ bash four.sh #! /home/yvgolothapov/test  
[yvgolothapov@fedora ~]$
```

Рис. 4.13: Запуск



```
./mozilla/extensions/(ec8038f7-c20a-464f-bb0a-k2a2a9e9738d)/.fedora-langpack-install  
./mozilla/firefox/fckowvbl.default-release/.parentlock  
./mozilla/firefox/fckowvbl.default-release/storage/sqlite  
./mozilla/firefox/fckowvbl.default-release/storage/permanent/chrome/idb/1657114595Amcateirvtisty.sqlite  
./mozilla/firefox/fckowvbl.default-release/storage/default/https+++github.com/.metadata-v2  
./mozilla/firefox/fckowvbl.default-release/storage/default/https+++github.com/ls/data.sqlite  
./mozilla/firefox/fckowvbl.default-release/storage/default/https+++github.com/ls/usage  
./mozilla/firefox/fckowvbl.default-release/storage/default/https+++system.rudn.ru/ls/data.sqlite
```

Рис. 4.14: Результат работы программы

5 Выводы

Я изучил основы программирования в оболочке ОС UNIX, а также научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

...