

Отчет по лабораторной работе 11

Операционные системы

Скрипникова София Дмитриевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
4	Контрольные вопросы	15
5	Выводы	19

Список иллюстраций

3.1	Создание файла	8
3.2	Создание файла	9
3.3	Результат работы программы	9
3.4	Создание файла	10
3.5	Создание файла	10
3.6	Результат работы программы	10
3.7	Создание файла	11
3.8	Создание файла	12
3.9	Результат работы программы	12
3.10	Создание файла	13
3.11	Создание файла	13
3.12	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 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).
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. Создала файл для программы 1 (рис. 3.1).

```
[sdscripnikova@sdscripnikova ~]$ touch pr1.sh
[sdscripnikova@sdscripnikova ~]$ chmod u+x pr1.sh
[sdscripnikova@sdscripnikova ~]$ ls
abcl      hello      program1.sh  Документы
backup    '#lab07.sh#' program2.sh  еучеюече
bin       lab07.sh   program3.sh  Загрузки
blog      letters   program4.sh  Изображения
'cd ~'    memos     sdscripnikova.github.io Музыка
'cd ~/.pub' misk      ski.plases  Общедоступные
conf.txt  my_os     text.txt    'Рабочий стол'
feathers   play      work        Шаблоны
file.txt  pr1.sh    Видео
```

```
[sdscripnikova@sdscripnikova ~]$ gedit pr1.sh
```

Рис. 3.1: Создание файла

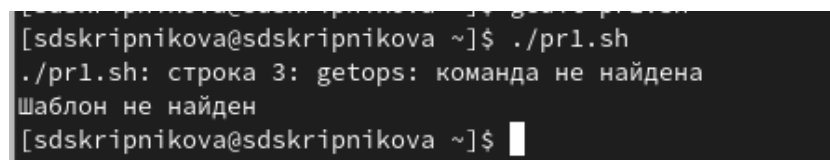
2. Написала текст программы 1 (рис. 3.2).



```
1 #!/bin/bash
2 iflag=0; oflag=0; pflag=0; cflag=0; nflag=0;
3 while getopts i:o:p:Cn optletter
4 do case $optletter in
5 i) iflag=1; ival=$OPTARG;;
6 o) oflag=1; oval=$OPTARG;;
7 p) pflag=1; pval=$OPTARG;;
8 C) cflag=1;;
9 *) echo illegal option $optletter
10
11 esac
12
13 done
14
15 if (($pflag==0))
16 then echo "Шаблон не найден"
17 else
18 if (($iflag==0))
19 then echo "Файл не найден"
20 exit
21 else
22 if (($oflag==0))
23 then if (($cflag==0))
24 then if (($nflag==0))
25 then grep $pval $ival
26 else grep -n $pval $ival
27 fi
28 else if (($nflag=0))
29 then grep -i $pval $ival
30 else grep -i -n $pval $ival
31 fi
32 else if (($cflag==0))
33 then if (($nflag==0))
```

Рис. 3.2: Создание файла

3. Проверила работу написанной программы (рис. 3.3).



```
[sdscripnikova@sdscripnikova ~]$ ./pr1.sh
./pr1.sh: строка 3: getopts: команда не найдена
Шаблон не найден
[sdscripnikova@sdscripnikova ~]$
```

Рис. 3.3: Результат работы программы

4. Создала файл для программы 2 (рис. 3.4).

```

[sdskripnikova@sdskripnikova ~]$ touch pr2.sh
[sdskripnikova@sdskripnikova ~]$ chmod u+x pr2.sh
[sdskripnikova@sdskripnikova ~]$ ls
abc1      hello      pr2.sh      Видео
backup    '#lab07.sh#' program1.sh  Документы
bin        lab07.sh   program2.sh  еучеюече
blog      letters    program3.sh  Загрузки
'cd ~'     memos      program4.sh  Изображения
'cd ~/.pub' misk       sdskripnikova.github.io Музыка
conf.txt   my_os      ski.plases   Общедоступные
feathers    play       text.txt     'Рабочий стол'
file.txt    pr1.sh     work         Шаблоны
[sdskripnikova@sdskripnikova ~]$

```

Рис. 3.4: Создание файла

5. Написала текст программы 2 (рис. 3.5).

```

1 #!/bin/bash
2
3
4
5
6 gcc prog2.c -o prog2
7
8 ./prog2
9
10 code=$?
11
12 case $code in
13
14     0) echo "Число меньше 0";;
15
16     1) echo "Число больше 0";;
17
18     2) echo "Число равно 0";;
19
20 esac

```

Рис. 3.5: Создание файла

6. Проверила работу написанной программы (рис. 3.6).

```

[sdskripnikova@sdskripnikova ~]$ ./pr2.sh
cc1: фатальная ошибка: pr2.c: Нет такого файла или каталога
компиляция прервана.
./pr2.sh: строка 3: ./pr2: Нет такого файла или каталога
[sdskripnikova@sdskripnikova ~]$

```

Рис. 3.6: Результат работы программы

7. Создала файл для программы 3 (рис. 3.7).

```
[sdscripnikova@sdscripnikova ~]$ touch pr3.sh
[sdscripnikova@sdscripnikova ~]$ chmod u+x pr3.sh
[sdscripnikova@sdscripnikova ~]$ ls
abcl      '#lab07.sh#'  program1.sh  еучеюече
backup    lab07.sh      program2.sh  Загрузки
bin       letters      program3.sh  Изображения
blog      memos        program4.sh  Музыка
'cd ~'    misk         sdscripnikova.github.io  Общедоступные
'cd ~/.pub'  my_os       ski.places   'Рабочий стол'
conf.txt  play         text.txt     Шаблоны
feathers  pr1.sh       work
file.txt  pr2.sh       Видео
hello     pr3.sh       Документы
[sdscripnikova@sdscripnikova ~]$
```

Рис. 3.7: Создание файла

8. Написала текст программы 3 (рис. 3.8).

```

1 #!/bin/bash
2
3
4
5
6 opt=$1;
7
8 form=$2;
9
10 num=$3;
11
12 function Files() {
13     for ((i=; i<=$num; i++)) do
14 file=$(echo $form | tr '#' "Si")
15
16 if [ $opt == "-c" ]
17
18 then
19
20     touch $file
21
22 fi
23     done
24 }
25
26 Files
27

```

Рис. 3.8: Создание файла

9. Проверила работу написанной программы (рис. 3.9).

```

[sdskripnikova@sdskripnikova ~]$ chmod +x pr3.sh
[sdskripnikova@sdskripnikova ~]$ ./pr3.sh
./pr3.sh: строка 13: ((: i=: синтаксическая ошибка: ожидается операнд (неверный
маркер «=»)
[sdskripnikova@sdskripnikova ~]$

```

Рис. 3.9: Результат работы программы

10. Создала файл для программы 4 (рис. 3.10).

```
[sdscripnikova@sdscripnikova ~]$ touch pr4.sh
[sdscripnikova@sdscripnikova ~]$ chmod u+x pr4.sh
[sdscripnikova@sdscripnikova ~]$ ls
abcl      '#lab07.sh#'  pr4.sh          Документы
backup    lab07.sh       program1.sh      еучеёече
bin        letters       program2.sh      Загрузки
blog       memos          program3.sh      Изображения
'cd ~'     misk           program4.sh      Музыка
'cd ~/.pub' my_os          sdscripnikova.github.io  Общедоступные
conf.txt   play           ski.places       'Рабочий стол'
feathers    pr1.sh         text.txt         Шаблоны
file.txt   pr2.sh         work
hello      pr3.sh         Видео
```

Рис. 3.10: Создание файла

11. Написала текст программы 4 (рис. 3.11).

```
1 #!/bin/bash
2 files=$(find ./-maxdepth 1 -mtime -7)
3 listing=""
4 for file in "$files" ; do
5     file=$(echo "$file" | cut -c 3-)
6     listing="$listing $file"
7 done
8
9 dir=$(basename $(pwd))
10 tar -cvf $dir.tar $listing
```

Рис. 3.11: Создание файла

12. Проверила работу написанной программы (рис. 3.12).

```
[sdscripnikova@sdscripnikova ~]$ chmod +x pr4.sh
[sdscripnikova@sdscripnikova ~]$ ./pr4.sh
find: './-maxdepth': Нет такого файла или каталога
find: '1': Нет такого файла или каталога
tar: Робкий отказ от создания пустого архива
Попробуйте «tar --help» или «tar --usage» для
получения более подробного описания.
[sdscripnikova@sdscripnikova ~]$
```

Рис. 3.12: Результат работы программы

4 Контрольные вопросы

1. Команда `getopts` 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, для команды `ls` флагом может являться `-F`.

Строка опций `option-string` – это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, то она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введённые данные с помощью оператора `case`.

Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента.

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

2. При перечислении имён файлов текущего каталога можно использовать следующие символы:
- `-` – соответствует произвольной, в том числе и пустой строке;

- `?` – соответствует любому одинарному символу;
 - `[c1-c2]` – 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.
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 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, являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда.
- 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.

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 перестаёт быть правильным.
Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.
5. Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования `bash`: это 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).

Примеры бесконечных циклов:

```
while true
do echo hello andy
done
until false
do echo hello mike
done
```

6. Строка `if test -f mans/i.s, mans/i.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь). 7. Выполнение оператора цикла `while` сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется по-

следовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь).

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.

5 Выводы

В ходе выполнения данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX и научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.