

Шаблон отчёта по лабораторной работе номер 12

Дисциплина: Операционные системы

Крестененко Полина Александровна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	16

Список таблиц

Список иллюстраций

3.1	скрипт 1	7
3.2	скрипт 1	8
3.3	проверка	8
3.4	скрипт на С	9
3.5	скрипт 2	9
3.6	проверка	10
3.7	скрипт 3	11
3.8	проверка	11
3.9	скрипт 4	12
3.10	проверка	12

1 Цель работы

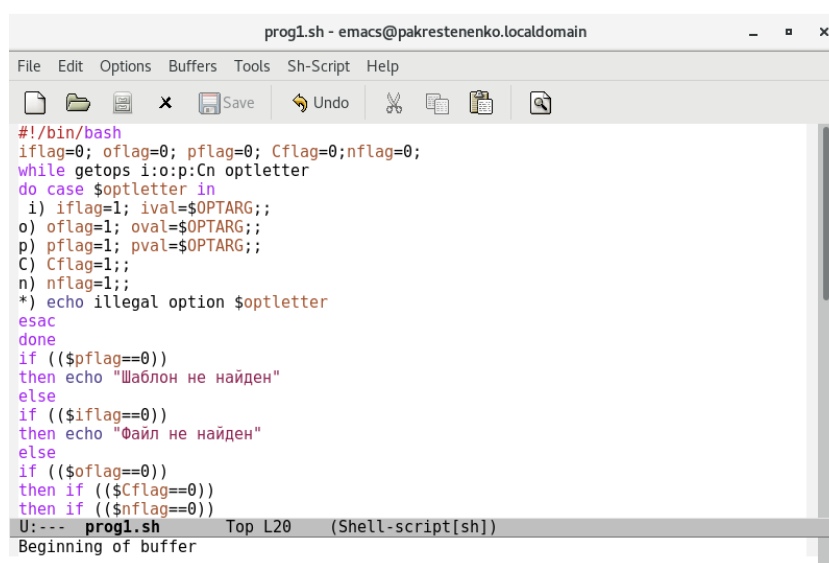
Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

2 Задание

Выполнить два задания, представленные в тескте файла лабораторной работы.

3 Выполнение лабораторной работы

- 1) 1) Используя команды `getopts` `grep`, написала командный файл, который анализирует командную строку с ключами: `-iinputfile` — прочитать данные из указанного файла; `-ooutputfile` — вывести данные в указанный файл; `-rшаблон` — 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`. Для данной задачи я создала файл `prog1.sh` и написала соответствующие скрипты. (рис. -fig. 3.1).



```
prog1.sh - emacs@pakrestenenko.localdomain
File Edit Options Buffers Tools Sh-Script Help
[Icons: Save, Undo, Cut, Copy, Paste, Find]

#!/bin/bash
iflag=0; oflag=0; pflag=0; Cflag=0; nflag=0;
while getopts i:o:p:Cn 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 ((pflag==0))
then echo "Шаблон не найден"
else
if ((iflag==0))
then echo "Файл не найден"
else
if ((oflag==0))
then if ((Cflag==0))
then if ((nflag==0))
U:--- prog1.sh Top L20 (Shell-script[sh])
Beginning of buffer
```

Рис. 3.1: скрипт1

(рис. -fig. 3.2).

```

else grep -n $pval $ival
fi
else if (($nflag==0))
then grep -i $pval $ival
else grep -i -n $pval $ival
fi
fi
else if (($cflag==0))
then if (($nflag==0))
then grep $pval $ival > $oval
else grep -n $pval $ival > $oval
fi
else if (($nflag==0))
then grep -i $pval $ival > $oval
else grep -i -n $pval $ival > $oval
fi
fi
fi
fi

```

Рис. 3.2: скрипт 1

Далее я проверила работу написанного скрипта, используя различные опции (например, команда «./prog.sh -I a1.txt -o a2.txt -p capital -C -n»), предварительно добавив право на исполнение файла (команда «chmod +x prog1.sh») и создав 2 файла, которые необходимы для выполнения программы: a1.txt и a2.txt (рис. -fig. 3.3).

```

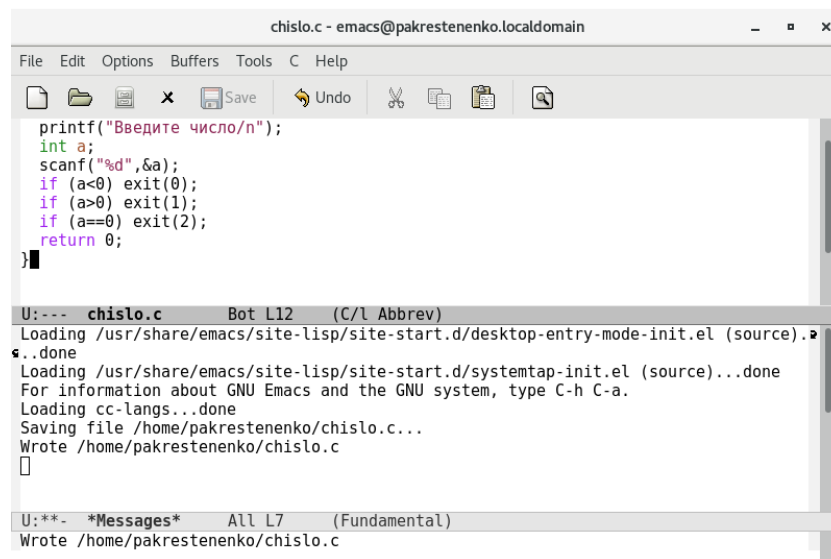
[pakrestenenko@pakrestenenko ~]$ touch prog1.sh
[pakrestenenko@pakrestenenko ~]$ emacs &
[1] 27070
[pakrestenenko@pakrestenenko ~]$ touch a1.txt a2.txt
[pakrestenenko@pakrestenenko ~]$ chmod +x prog1.sh
[pakrestenenko@pakrestenenko ~]$ cat a1.txt
[pakrestenenko@pakrestenenko ~]$ ./prog1.sh -i a1.txt -o a2.txt -p capital -C -n
./prog1.sh: line 3: getops: команда не найдена
Шаблон не найден
[pakrestenenko@pakrestenenko ~]$ cat a1.txt
Moscow is the capital of RUSSIA
Paris is the capital of France
Simple text
Rome is not the CAPITAL of Canada
[pakrestenenko@pakrestenenko ~]$ ./prog1.sh -i a1.txt -o a2.txt -p capital -C -n

```

Рис. 3.3: проверка

- 2) Написала 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. Для данной задачи я создала 2 файла: chislo.c и chislo.sh и написала соответствующие скрипты.(рис. -fig. 3.4).



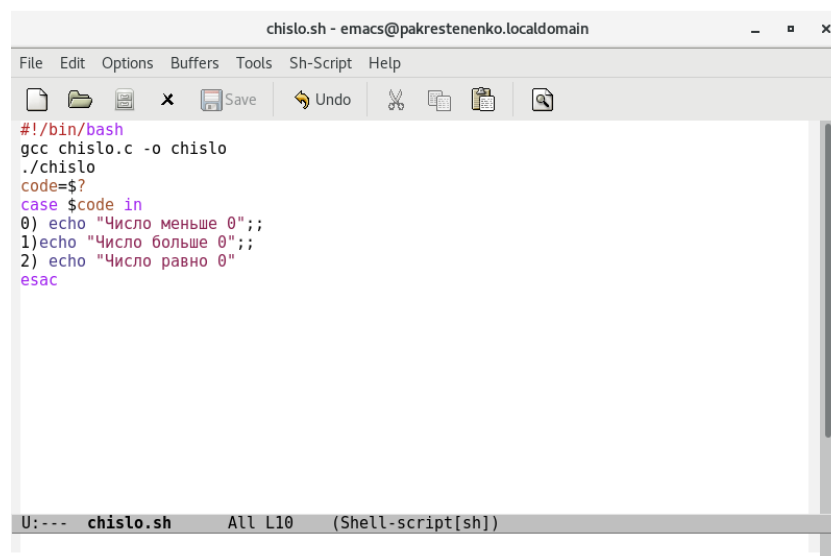
```
chislo.c - emacs@pakrestenenko.localdomain
File Edit Options Buffers Tools C Help
[Icons] Save Undo [Icons]
printf("Введите число\n");
int a;
scanf("%d",&a);
if (a<0) exit(0);
if (a>0) exit(1);
if (a==0) exit(2);
return 0;
}

U:--- chislo.c Bot L12 (C/L Abbrev)
Loading /usr/share/emacs/site-lisp/site-start.d/desktop-entry-mode-init.el (source)...done
Loading /usr/share/emacs/site-lisp/site-start.d/systemtap-init.el (source)...done
For information about GNU Emacs and the GNU system, type C-h C-a.
Loading cc-langs...done
Saving file /home/pakrestenenko/chislo.c...
Wrote /home/pakrestenenko/chislo.c

U:-- *Messages* All L7 (Fundamental)
Wrote /home/pakrestenenko/chislo.c
```

Рис. 3.4: скрипт на С

(рис. -fig. 3.5).



```
chislo.sh - emacs@pakrestenenko.localdomain
File Edit Options Buffers Tools Sh-Script Help
[Icons] Save Undo [Icons]
#!/bin/bash
gcc chislo.c -o chislo
./chislo
code=$?
case $code in
0) echo "Число меньше 0";;
1) echo "Число больше 0";;
2) echo "Число равно 0";;
esac

U:--- chislo.sh All L10 (Shell-script[sh])
```

Рис. 3.5: скрипт 2

далее я проверила работу написанных скриптов (команда «./chislo.sh»), предварительно добавив право на исполнение файла (команда «chmod +xchislo.sh»)(рис.

-fig. 3.6).

```
[pakrestenenko@pakrestenenko ~]$ touch chislo.c
[pakrestenenko@pakrestenenko ~]$ touch chislo.sh
[pakrestenenko@pakrestenenko ~]$ emacs &
[1] 27513
[pakrestenenko@pakrestenenko ~]$ emacs &
[2] 27591
[pakrestenenko@pakrestenenko ~]$ chmod +x chislo.sh
[pakrestenenko@pakrestenenko ~]$ ./chislo.sh
Введите число/n 0
./chislo.sh: line 5: syntax error near unexpected token `&'
./chislo.sh: line 5: `case &code in'
[pakrestenenko@pakrestenenko ~]$ ./chislo.sh
Введите число/n 0
./chislo.sh: line 5: syntax error near unexpected token `&'
./chislo.sh: line 5: `case &code in'
[pakrestenenko@pakrestenenko ~]$ ./chislo.sh
Введите число/n 0
Число равно 0
[pakrestenenko@pakrestenenko ~]$ ./chislo.sh
Введите число/n 5
Число больше 0
[pakrestenenko@pakrestenenko ~]$ ./chislo.sh
Введите число/n -5
Число меньше 0
```

Рис. 3.6: проверка

- 3) Написала 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). Для данной задачи я создала файл: files.sh и написала соответствующий скрипт.(рис. -fig. 3.7).

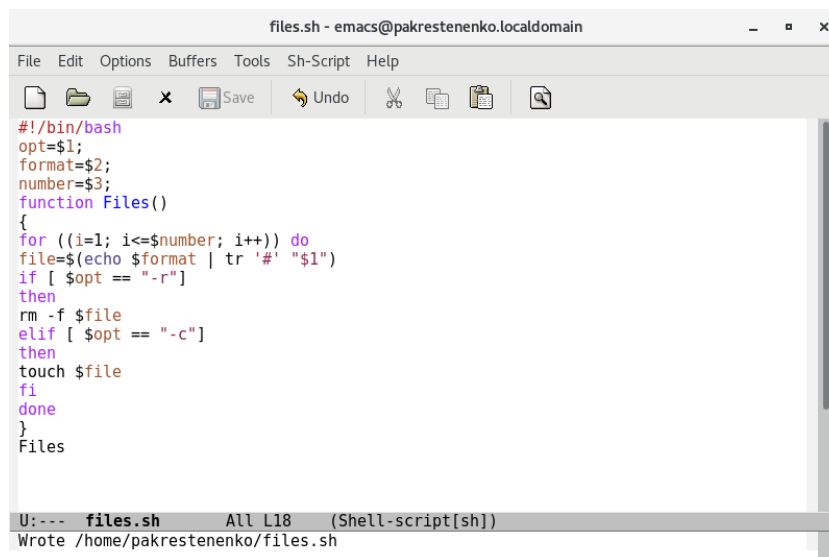


Рис. 3.7: скрипт 3

Далее я проверила работу написанного скрипта (команда «./files.sh»), предварительно добавив право на исполнение файла (команда «chmod +xfiles.sh»). Сначала я создала три файла (команда «./files.sh -c abc#.txt 3»), удовлетворяющие условию задачи, а потом удалила их (команда «./files.sh -r abc#.txt 3»)(рис. -fig. 3.8).

```
[pakrestenenko@pakrestenenko ~]$ touch files.sh
[pakrestenenko@pakrestenenko ~]$ emacs &
[3] 27782
[pakrestenenko@pakrestenenko ~]$ chmod +x files.sh
[pakrestenenko@pakrestenenko ~]$ ls
10 a1.txt      chislo      example1.txt  files.sh~    work        Музыка
11 a2.txt      chislo.c    example2.txt  lab010.sh    Видео       Общедоступные
12 backup     chislo.c~   example3.txt  lab010.sh~   Документы   Рабочий стол
3  backup.sh   chislo.sh   example4.txt  prog1.sh     Загрузки    Шаблоны
9  backup.sh~  chislo.sh~  files.sh      prog1.sh~    Изображения
[pakrestenenko@pakrestenenko ~]$ ./files.sh -c abc#.txt 3
```

Рис. 3.8: проверка

- 4) Написала командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировала его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find). Для данной задачи я создала файл: prog4.sh и написала соответствующий скрипт.(рис. -fig. 3.9).

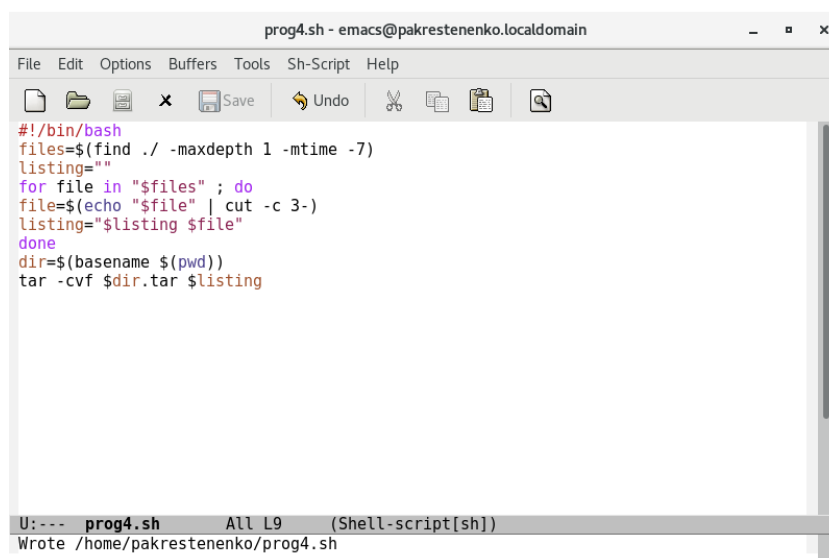


Рис. 3.9: скрипт 4

Далее я проверила работу написанного скрипта (команды «`sudo ~/prog4.sh`» и «`tar -tf Catalog1.tar`»), предварительно добавив право на исполнение файла (команда «`chmod +x prog4.sh`») и создав отдельный Catalog1 с несколькими файлами. Как видно из Рисунков 10, 11, файлы, измененные более недели назад, заархивированы не были. Скрипт работает корректно.(рис. -fig. 3.10).

```
[pakrestenenko@pakrestenenko ~]$ chmod +x prog4.sh
[pakrestenenko@pakrestenenko ~]$ cd ~/Catalog1
bash: cd: /home/pakrestenenko/Catalog1: Нет такого файла или каталога
[pakrestenenko@pakrestenenko ~]$ mkdir Catalog1
[pakrestenenko@pakrestenenko ~]$ cd ~/Catalog1
[pakrestenenko@pakrestenenko Catalog1]$ ls -l
итого 0
[pakrestenenko@pakrestenenko Catalog1]$ sudo ~/prog4.sh
[sudo] пароль для pakrestenenko:
```

Рис. 3.10: проверка

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 введенные 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 введенных 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 отрезультатов 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 последовательность 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.

4 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.