

Лабораторная работа 11

Отчет по лабораторной работе 11

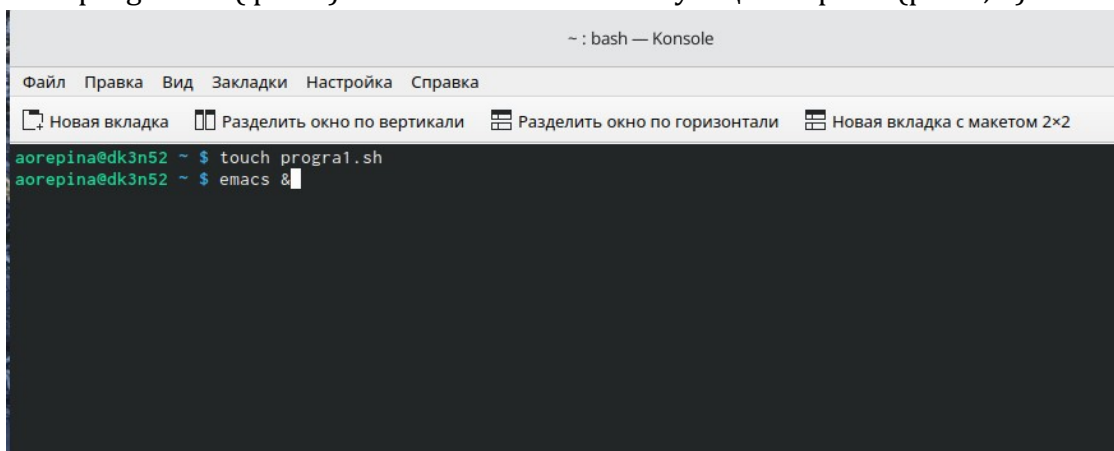
Репина Ангелина Олеговна

Цель работы

Цель данной работы - изучение основ программирования в оболочке ОС UNIX. Освоение навыков по написанию более сложных командных файлов с использованием логических управляющих конструкций и циклов.

Выполнение лабораторной работы

- 1) Используя команды `getopts` `grep`, написала командный файл, который анализирует командную строку с ключами: `-i` — прочитать данные из указанного файла; `-o` — вывести данные в указанный файл; `-r` — 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 `progra1.sh` (рис 1) и написала соответствующий скрипт (рис 2, 3)



The screenshot shows a terminal window titled '~ : bash — Konsole'. The menu bar includes 'Файл', 'Правка', 'Вид', 'Закладки', 'Настройка', and 'Справка'. Below the menu bar, there are icons and labels for window management: 'Новая вкладка', 'Разделить окно по вертикали', 'Разделить окно по горизонтали', and 'Новая вкладка с макетом 2x2'. The terminal content shows the user 'aorepina@dk3n52' at the prompt '~ \$' executing the command 'touch progra1.sh'. The next line shows the user entering 'emacs &' at the prompt '~ \$'.

```
emacs@dk3n52
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
iflag=0; oflag=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))
        then if (grep $pval $ival
U:--- progra1.sh Top L21 (Shell-script[sh]) Чт мая 12 11:11 0.76
Warning (initialization): An error occurred while loading '~/.emacs':

error: Package 'fira-code-mode-' is unavailable

To ensure normal operation, you should investigate and remove the
cause of the error in your initialization file. Start Emacs with
the '--debug-init' option to view a complete error backtrace.
█
```

2

```
emacs@dk3n52
File Edit Options Buffers Tools Sh-Script Help
    then if (($Cflag==0))
    then if (($nflag==0))
    then grep $pval $ival
    else grep -n $pval $ival
    fi
    else if (($nflag==0))
    then grep -i $pval $ival
    else grep -i -n $pval $ival
    fi
    else if (($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
U:--- progra1.sh 46% L21 (Shell-script[sh]) Чт мая 12 11:11 0.76
Warning (initialization): An error occurred while loading '~/.emacs':

error: Package 'fira-code-mode-' is unavailable
```

3

Проверила работу написанного скрипта, используя различные опции (например команду ./progra1.sh -i a1.txt -o a2.txt -C -n), предварительно добавив право на исполнение файла (chmod +x progra1.sh) и создав 2 файла, которые необходимы для выполнения программы (a1. txt a2.txt) (рис 4, 5, 6).Скрипт работает корректно

```
[1] 3764
aorepina@dk3n52 ~ $ touch a1.txt a2.txt
aorepina@dk3n52 ~ $ chmod +x progra1.sh
aorepina@dk3n52 ~ $ cat a1.txt
hello
goodbye goodbye
hello123
hello hello hello
aorepina@dk3n52 ~ $ ./progra1.sh -i a1.txt -o a2.txt -n hello -n
```

4

```

1:prog1.sh: строка 10: ((: не является оператором: ожидается операнд (неверный паркер: 0)
aorepina@dk3n52 ~ $ ./progra1.sh -i a1.txt -o a2.txt -p hello -n
aorepina@dk3n52 ~ $ cat a2.txt
1:hello
3:hello123
4:hello hello hello
aorepina@dk3n52 ~ $ ./progra1.sh -i a1.txt -o a2.txt -p hello -n
aorepina@dk3n52 ~ $ cat a2.txt
1:hello
3:hello123
4:hello hello hello
aorepina@dk3n52 ~ $ ./progra.sh -i a1.txt -o a2.txt -p hello -C -n
bash: ./progra.sh: Нет такого файла или каталога
aorepina@dk3n52 ~ $ ./progra1.sh -i a1.txt -o a2.txt -p hello -C -n
aorepina@dk3n52 ~ $ cat a2.txt
1:hello
3:hello123
4:hello hello hello
aorepina@dk3n52 ~ $ ./progra1.sh -i a1.txt -C -n

```

5

```

aorepina@dk3n52 ~ $ ./progra1.sh -o a2.txt -p hello -C -n
Файл не найден
aorepina@dk3n52 ~ $

```

6

- 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: `chlslo.c` `chlslo.sh` (рис 7) и написала соответствующие скрипты (рис 8,9)

```

aorepina@dk3n52 ~ $ touch chslo.c
aorepina@dk3n52 ~ $ touch chislo.sh
aorepina@dk3n52 ~ $ emacs &

```

7

```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
gcc chslo.c -o chslo
./chslo
code=$?
case $code in
    0) echo "Число меньше 0" ;;
    1) echo "Число больше 0" ;;
    2) echo "Число равно 0"
esac
```

8

```
File Edit Options Buffers Tools C Help
#include <stdio.h>
#include <stdlib.h>
int main() {
    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;
}
```

9

Проверила работу написанных скриптов (команда ./chlso.sh), предварительно добавив право на исполнение файла (chmod +x chlso.sh) (рис 10). Скрипты работают корректно

```

aorepina@dk3n52 ~ $ chmod +x chlso.sh
aorepina@dk3n52 ~ $ ./chlso.sh
Введите число
0
Число равно 0
aorepina@dk3n52 ~ $ ./chlso.sh
Введите число
3
Число больше 0
aorepina@dk3n52 ~ $ ./chlso.sh
Введите число
-10
Число меньше 0
aorepina@dk3n52 ~ $

```

10

- 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 (рис 11) и написала соответствующий скрипт (рис 12)

```

aorepina@dk3n52 ~ $ touch files.sh
aorepina@dk3n52 ~ $ emacs &

```

11

```

File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
opt=$1;
format=$2;
number=$3;
function Files()
{
    for (( i=1; i<=$number; i++ )) do
        file=$(echo $format | tr '#' "$i")
        if [ $opt == "-r" ]
        then
            rm -f $file
        elif [ $opt == "-c" ]
        then
            touch $file
        fi
    done
}
Files

```

12

[illegible]

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 содала файл pr4.sh (рис 14) и написала соответствующий скрипт (рис 15)

14

```
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
files=$(find ./ -maxdepth 1 -mtime -7)
listing=""
for file in "$files"; do
    files=$(echo "$file" | cut -c 3-)
    listing="$listing $file"
done
dir=$(basename $(pwd))
tar -cvf $dir.tar $listing
```

15

Далее я проверила работу написанного скрипта, предварительно добавив право на исполнение файла и создав отдельный каталог с несколькими файлами (рис 16)

```
aorepina@dk3n52 ~$ chmod +x pr4.sh
aorepina@dk3n52 ~$ mkdir Catalog1
aorepina@dk3n52 ~$ cd Catalog1
aorepina@dk3n52 ~/Catalog1$ ./pr4.sh
./
./a1.txt
./a2.txt
./abc1
tar: ./Catalog1.tar: файл является архивом; не сброшен
./a1.txt
./a2.txt
aorepina@dk3n52 ~/Catalog1$ tar -tf Catalog1.tar
./
./a1.txt
./a2.txt
./abc1
./a1.txt
./a2.txt
aorepina@dk3n52 ~/Catalog1$ ./pr4.sh
bash: ./pr4.sh: Нет такого файла или каталога
aorepina@dk3n52 ~/Catalog1$ ./pr4.sh
bash: ./pr4.sh: Нет такого файла или каталога
aorepina@dk3n52 ~/Catalog1$ ./pr4.sh
./
./a1.txt
./a2.txt
./abc1
tar: ./Catalog1.tar: файл является архивом; не сброшен
./a1.txt
./a2.txt
tar: ./Catalog1.tar: файл является архивом; не сброшен
aorepina@dk3n52 ~/Catalog1$ tar -tf Catalog1.tar
./
./a1.txt
./a2.txt
./abc1
./a1.txt
./a2.txt
aorepina@dk3n52 ~/Catalog1$
```

16

Выводы

В ходе выполнения лабораторной работы я изучила основы программирования в оболочке ОС UNIX и научилась писать небольшие командные файлы.

Итоговые 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:

1. `*` – соответствует произвольной, в том числе и пустой строке;
2. `?` – соответствует любому одинарному символу;
3. `[c1-c2]` – соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`. Например, `1.1 echo` – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; `1.2. ls.c` – выведет все файлы с последними двумя символами, совпадающими с `c`. `1.3. echoprog.?` – выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog`. `1.4. [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, неравный 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 mani.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.