

Отчёт лабораторной работы №12

Дисциплина: Операционные системы

Касьянов Даниил Владимирович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Контрольные вопросы	13
4	Выводы	17
5	Библиография	18

Список таблиц

Список иллюстраций

1 Цель работы

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

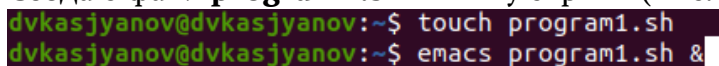
2 Выполнение лабораторной работы

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;
- `-C` — различать большие и малые буквы;
- `-n` — выдавать номера строк.

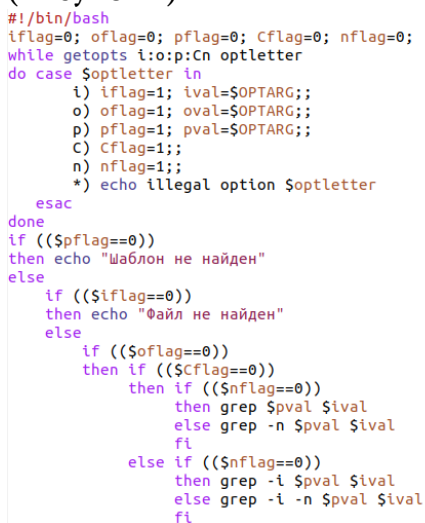
а затем ищет в указанном файле нужные строки, определяемые ключом `-р`.

Создаю файл **program1.sh** и пишу скрипт (Рис. 1, 2, 3):



```
dvkasjyanov@dvkasjyanov:~$ touch program1.sh
dvkasjyanov@dvkasjyanov:~$ emacs program1.sh &
```

(Рисунок 1)



```
#!/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))
                then grep $pval $ival
                else grep -n $pval $ival
                fi
            else if (($nflag==0))
                then grep -l $pval $ival
                else grep -i -n $pval $ival
                fi
        fi
    fi
fi
```

(Рисунок 2)

```
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
fi
```

(Рисунок 3)

Проверяю работу скрипта, используя различные опции (например, команда `./program1.sh -i first.txt -o second.txt -p capital -C -n`), предварительно добавив право на исполнение файла (команда `chmod +x program1.sh`) и создав 2 файла, которые необходимы для выполнения программы: **first.txt** и **second.txt** (Рисунок 4):

```
dvkasjyanov@dvkasjyanov:~$ touch first.txt second.txt
dvkasjyanov@dvkasjyanov:~$ chmod +x program1.sh
dvkasjyanov@dvkasjyanov:~$ cat first.txt
Moscow is the capital of Russia
London is the capital of Great Britain
Simple text
Rome is not the CAPITAL of Canada
dvkasjyanov@dvkasjyanov:~$ ./program1.sh -i first.txt -o second.txt -p capital -C -n
dvkasjyanov@dvkasjyanov:~$ cat second.txt
1:Moscow is the capital of Russia
2:London is the capital of Great Britain
4:Rome is not the CAPITAL of Canada
dvkasjyanov@dvkasjyanov:~$ ./program1.sh -i first.txt -o second.txt -p capital -n
dvkasjyanov@dvkasjyanov:~$ cat second.txt
1:Moscow is the capital of Russia
2:London is the capital of Great Britain
dvkasjyanov@dvkasjyanov:~$ ./program1.sh -i first.txt -C -n
Шаблон не найден
dvkasjyanov@dvkasjyanov:~$ ./program1.sh -o second.txt -p capital -C -n
Файл не найден
dvkasjyanov@dvkasjyanov:~$
```

(Рисунок 4)

2. Написать на языке **C** 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.

Создаю файлы **number.c** и **number.sh** и пишу скрипты (Рис. 5-7).

```
dvkasjyanov@dvkasjyanov:~$ touch number.c number.sh
dvkasjyanov@dvkasjyanov:~$ emacs number.c &
[2] 17409
dvkasjyanov@dvkasjyanov:~$ emacs number.sh &
[3] 17461
[1] Завершён      emacs 911.c
[2] Завершён      emacs number.c
dvkasjyanov@dvkasjyanov:~$
[3]+ Завершён      emacs number.sh
dvkasjyanov@dvkasjyanov:~$
```

(Рисунок 5)



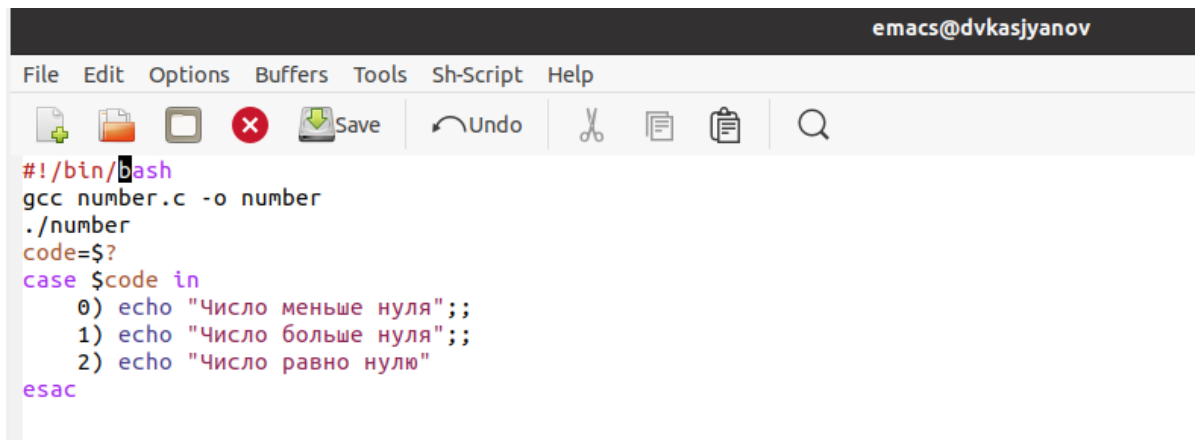
The screenshot shows the Emacs editor window titled "emacs@dvkasjyanov". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "C", and "Help". The toolbar contains icons for file operations (new, open, save, close), editing (undo, redo, cut, copy, paste), and a search icon. The main text area displays the following C code:

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

int main()
{
    printf("Введите число: ");
    int a;
    scanf("%d", &a);
    if (a<0) exit(0);
    if (a>0) exit(1);
    if (a==0) exit(2);

    return 0;
}
```

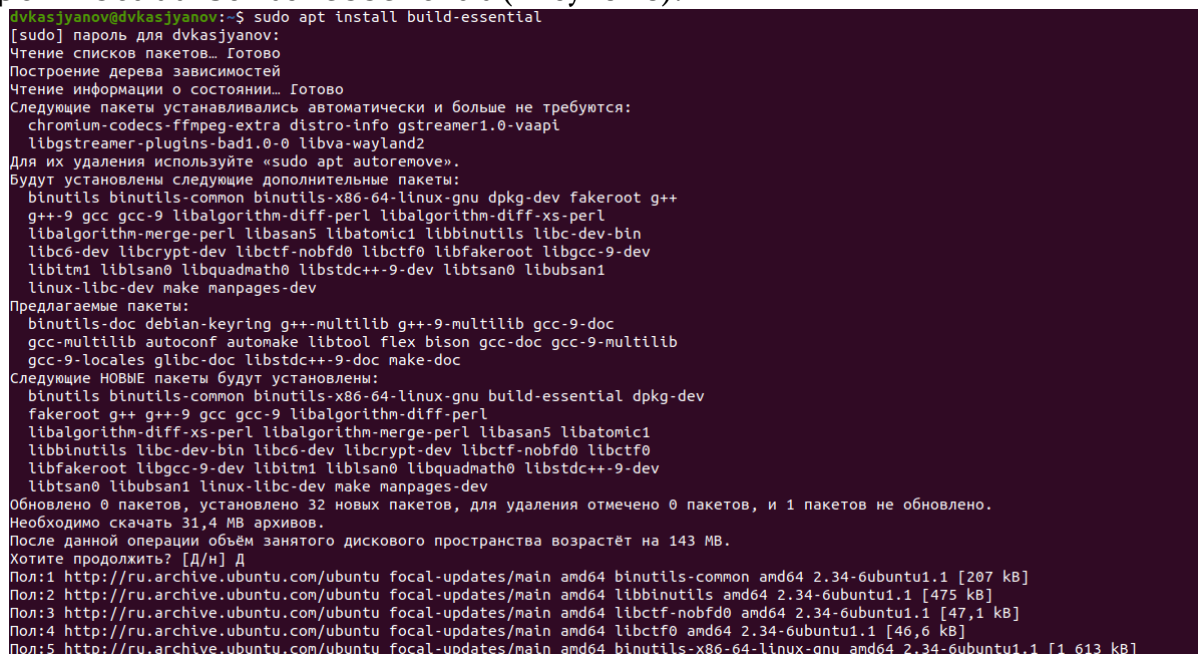
(Рисунок 6)



```
#!/bin/bash
gcc number.c -o number
./number
code=$?
case $code in
  0) echo "Число меньше нуля";;
  1) echo "Число больше нуля";;
  2) echo "Число равно нулю";;
esac
```

(Рисунок 7)

Для дальнейшей работы необходимо скачать **gcc**, используя команду `sudo apt install build-essential` (Рисунок 8):



```
dvkasjyanov@dvkasjyanov:~$ sudo apt install build-essential
[sudo] пароль для dvkasjyanov:
Чтение списков пакетов... Готово
Построение дерева зависимостей
Чтение информации о состоянии... Готово
Следующие пакеты устанавливались автоматически и больше не требуются:
 chromium-codecs-ffmpeg-extra distro-info gstreamer1.0-vaapi
 libgstreamer-plugins-bad1.0-0 libva-wayland2
Для их удаления используйте «sudo apt autoremove».
Будут установлены следующие дополнительные пакеты:
 binutils binutils-common binutils-x86-64-linux-gnu dpkg-dev fakeroot g++
 g++-9 gcc gcc-9 libalgorithm-diff-perl libalgorithm-diff-xs-perl
 libalgorithm-merge-perl libasan5 libatomic1 libbinutils libc-dev-bin
 libc6-dev libcrypt-dev libctf-nobfd0 libctf0 libfakeroot libgcc-9-dev
 libitm1 liblsan0 libquadmath0 libstdc++-9-dev libtsan0 libubsan1
 linux-libc-dev make manpages-dev
Предлагаемые пакеты:
 binutils-doc debian-keyring g++-9-multilib gcc-9-doc
 gcc-multilib autoconf automake libtool flex bison gcc-doc gcc-9-multilib
 gcc-9-locales glibc-doc libstdc++-9-doc make-doc
Следующие НОВЫЕ пакеты будут установлены:
 binutils binutils-common binutils-x86-64-linux-gnu build-essential dpkg-dev
 fakeroot g++ g++-9 gcc gcc-9 libalgorithm-diff-perl
 libalgorithm-diff-xs-perl libalgorithm-merge-perl libasan5 libatomic1
 libbinutils libc-dev-bin libc6-dev libcrypt-dev libctf-nobfd0 libctf0
 libfakeroot libgcc-9-dev libitm1 liblsan0 libquadmath0 libstdc++-9-dev
 libtsan0 libubsan1 linux-libc-dev make manpages-dev
Обновлено 0 пакетов, установлено 32 новых пакетов, для удаления отмечено 0 пакетов, и 1 пакетов не обновлено.
Необходимо скачать 31,4 МБ архивов.
После данной операции объем занятого дискового пространства возрастёт на 143 МБ.
Хотите продолжить? [Д/н] Д
Пол:1 http://ru.archive.ubuntu.com/ubuntu focal-updates/main amd64 binutils-common amd64 2.34-6ubuntu1.1 [207 kB]
Пол:2 http://ru.archive.ubuntu.com/ubuntu focal-updates/main amd64 libbinutils amd64 2.34-6ubuntu1.1 [475 kB]
Пол:3 http://ru.archive.ubuntu.com/ubuntu focal-updates/main amd64 libctf-nobfd0 amd64 2.34-6ubuntu1.1 [47,1 kB]
Пол:4 http://ru.archive.ubuntu.com/ubuntu focal-updates/main amd64 libctf0 amd64 2.34-6ubuntu1.1 [46,6 kB]
Пол:5 http://ru.archive.ubuntu.com/ubuntu focal-updates/main amd64 binutils-x86-64-linux-gnu amd64 2.34-6ubuntu1.1 [1 613 kB]
```

(Рисунок 8)

Проверяю работу скрипта (Рисунок 9):

```

dvkasjyanov@dvkasjyanov:~$ chmod +x number.sh
dvkasjyanov@dvkasjyanov:~$ ./number.sh
Введите число: 0
Число равно нулю
dvkasjyanov@dvkasjyanov:~$ ./number.sh
Введите число: -666
Число меньше нуля
dvkasjyanov@dvkasjyanov:~$ ./number.sh
Введите число: 666
Число больше нуля
dvkasjyanov@dvkasjyanov:~$

```

(Рисунок 9)

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).

Создаю файл **K.sh** и пишу скрипт: Рис. (10, 11).

```

dvkasjyanov@dvkasjyanov:~$ touch K.sh
dvkasjyanov@dvkasjyanov:~$ emacs K.sh &

```

(Рисунок 10)

The screenshot shows the Emacs editor interface with the title bar 'emacs@dvkasjyanov'. The menu bar includes 'File', 'Edit', 'Options', 'Buffers', 'Tools', 'Sh-Script', and 'Help'. The toolbar contains icons for file operations and editing. The main text area displays the following script content:

```

#!/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

```

(Рисунок 11)

Проверяю работу скрипта (Рисунок 12):

```
dvkasjyanov@dvkasjyanov:~$ chmod +x K.sh
dvkasjyanov@dvkasjyanov:~$ ls
1.doc      2.pdf      4th.sh~    conf.txt   feathers    logfile     number.sh~  script2.sh  snap        Шаблоны
1.jpg      3.doc      5.jpg      conf.txt.save file.txt    logfile.txt main.cpp    script2.sh~ text.txt
1.pdf      3.jpg      '911.c#'   date.txt   file.txt.save first.txt   may         script3.sh~ work
1st.sh     3.pdf      911.c      Desktop    K.sh       K.sh       monthly     script3.sh~ Видео
1st.sh~    '3rd.sh#'  911.c~    ex2.sh     K.sh~      K.sh~      my_os       script4.sh~ Документы
1.txt      3rd.sh     911.sh     ex2.sh~    lab07.sh   lab07.sh   number       script4.sh~ Загрузки
2.doc      3rd.sh~    911.sh~    ex3.sh     lab07.sh~  lab07.sh~  number.c     script.sh~  Изображения
2.jpg      4.doc      abc1       ex3.sh~    lab08.sh   lab08.sh   number.c~    script.sh~  Музыка
2nd.sh     4.jpg      australia  ex.sh      lab08      lab08      number.c-    second.txt  'Общедоступные
2nd.sh~    4th.sh     backup     ex.sh~     l.log      l.log      number.sh    'script2.sh#  'Рабочий стол'

dvkasjyanov@dvkasjyanov:~$ ./K.sh -c abc#.txt 3
dvkasjyanov@dvkasjyanov:~$ ls
1.doc      3.doc      '911.c#'   conf.txt   file.txt    main.cpp    program1.sh  script4.sh  Загрузки
1.jpg      3.jpg      911.c      conf.txt.save file.txt.save may         program1.sh~ script4.sh~ Изображения
1.pdf      3.pdf      911.c~    date.txt   first.txt   monthly     'PROGRAMMING.sh#' script.sh~  Музыка
1st.sh     '3rd.sh#'  911.sh     Desktop    K.sh       my_os       PROGRAMMING.sh script.sh~  'Общедоступные
1st.sh~    3rd.sh     911.sh~    ex2.sh     K.sh~      number      PROGRAMMING.sh~ second.txt  'Рабочий стол'
1.txt      3rd.sh~    abc1       ex2.sh~    lab07.sh   number.c~   reports      ski.places  Шаблоны
2.doc      4.doc      abc1.txt   ex3.sh     lab07.sh~  number.c-   'script2.sh#' text.txt
2.jpg      4.jpg      abc2.txt   ex3.sh~    lab08.sh   l.log       script2.sh   snap
2nd.sh     4th.sh     abc3.txt   ex.sh      lab08      l.log       script2.sh   text.txt
2nd.sh~    4th.sh~    australia ex.sh~     log07      logfile     script2.sh~ work
2.pdf      5.jpg      backup     ex.sh~     logfile.txt print        script3.sh~ Видео
dvkasjyanov@dvkasjyanov:~$ ./K.sh -r abc#.txt 3
dvkasjyanov@dvkasjyanov:~$ ls
1.doc      2.pdf      4th.sh~    conf.txt   feathers    logfile     number.sh~  script2.sh  snap        Шаблоны
1.jpg      3.doc      5.jpg      conf.txt.save file.txt    logfile.txt main.cpp    script2.sh~ text.txt
1.pdf      3.jpg      '911.c#'   date.txt   file.txt.save first.txt   may         script3.sh~ work
1st.sh     3.pdf      911.c      Desktop    K.sh       K.sh       monthly     script3.sh~ Видео
1st.sh~    '3rd.sh#'  911.c~    ex2.sh     K.sh~      K.sh~      my_os       script4.sh~ Документы
1.txt      3rd.sh     911.sh     ex2.sh~    lab07.sh   lab07.sh   number       script4.sh~ Загрузки
2.doc      3rd.sh~    911.sh~    ex3.sh     lab07.sh~  lab07.sh~  number.c     script.sh~  Изображения
2.jpg      4.doc      abc1       ex3.sh~    lab07.sh~  lab07.sh~  number.c~    script.sh~  Музыка
2nd.sh     4.jpg      australia  ex.sh      lab08      lab08      number.c-    second.txt  'Общедоступные
2nd.sh~    4th.sh     backup     ex.sh~     l.log      l.log      number.sh    'script2.sh#  'Рабочий стол'

dvkasjyanov@dvkasjyanov:~$
```

(Рисунок 12)

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`).

Создаю файл **prog4.sh** и пишу скрипт: Рис. (13, 14).

```
dvkasjyanov@dvkasjyanov:~$ touch prog4.sh
dvkasjyanov@dvkasjyanov:~$ emacs prog4.sh &
```

(Рисунок 13)

```
emacs@dvkasjyanov
File Edit Options Buffers Tools Sh-Script Help
[Icons] Save Undo [Icons]

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

(Рисунок 14)

Проверяю работу скрипта (Рисунок 15):

```
dvkasjyanov@dvkasjyanov:~$ chmod +x K.sh
dvkasjyanov@dvkasjyanov:~$ ls
1.doc      2.pdf      4th.sh~   conf.txt  feathers  logfile   number.sh~  script2.sh  snap      Шаблоны
1.jpg      3.doc      5.jpg     conf.txt.save file.txt  logfile.txt main.cpp    script2.sh~  text.txt
1.pdf      3.jpg      '911.c#'  date.txt  file.txt.save first.txt  may        script3.sh~  work
1st.sh     3.pdf      911.c     Desktop   first.txt  K.sh      monthly    script3.sh~  Видео
1st.sh~    '3rd.sh#'  911.c~    ex2.sh    ex2.sh~   K.sh~     my_os      script4.sh~  Документы
1.txt      3rd.sh     911.sh    ex2.sh~   ex3.sh~   lab07.sh  number     script4.sh~  Загрузки
2.doc      3rd.sh~    911.sh~   ex3.sh    ex3.sh~   lab07.sh~ number.c~   script.sh~   Изображения
2.jpg      4.doc      abc1       ex3.sh~   ex.sh~    lab08     number.c~  script.sh~   Музыка
2nd.sh     4.jpg      australia  ex.sh     ex.sh~    l.log     number.c~  second.txt   Общиедоступные
2nd.sh~    4th.sh     backup     ex.sh~    ex.sh~    l.log     number.sh  'script2.sh#' ski.plases   'Рабочий стол'

dvkasjyanov@dvkasjyanov:~$ ./K.sh -c abc#.txt 3
dvkasjyanov@dvkasjyanov:~$ ls
1.doc      3.doc      '911.c#'  conf.txt  file.txt  main.cpp  program1.sh  script4.sh  Загрузки
1.jpg      3.jpg      911.c     conf.txt.save file.txt.save first.txt  monthly     script4.sh~  Изображения
1.pdf      3.pdf      911.c~    date.txt  file.txt.save first.txt  my_os      script.sh   Музыка
1st.sh     '3rd.sh#'  911.sh    Desktop   K.sh      number     PROGRAMMING.sh~ second.txt  Общиедоступные
1st.sh~    3rd.sh     911.sh~   ex2.sh    ex2.sh~   lab07.sh~ reports     script.sh~  'Рабочий стол'
1.txt      3rd.sh~    abc1       ex2.sh~   ex3.sh~   lab08     'script2.sh#' ski.plases  snap
2.doc      4.doc      abc1.txt   ex3.sh    ex3.sh~   lab07.sh~ script2.sh  text.txt   work
2.jpg      4.jpg      abc2.txt   ex3.sh~   ex.sh~    lab08     script2.sh~ work       Видео
2nd.sh     4th.sh     abc3.txt   ex.sh     ex.sh~    l.log     script3.sh~ script3.sh~ Документы
2nd.sh~    4th.sh~    australia  ex.sh~    ex.sh~    logfile.txt play        script3.sh~
2.pdf      5.jpg      backup     feathers  logfile.txt print
dvkasjyanov@dvkasjyanov:~$ ./K.sh -r abc#.txt 3
dvkasjyanov@dvkasjyanov:~$ ls
1.doc      2.pdf      4th.sh~   conf.txt  feathers  logfile   number.sh~  script2.sh  snap      Шаблоны
1.jpg      3.doc      5.jpg     conf.txt.save file.txt  logfile.txt main.cpp    script2.sh~  text.txt
1.pdf      3.jpg      '911.c#'  date.txt  file.txt.save first.txt  may        script3.sh~  work
1st.sh     3.pdf      911.c     Desktop   first.txt  K.sh      monthly    script3.sh~  Видео
1st.sh~    '3rd.sh#'  911.c~    ex2.sh    ex2.sh~   K.sh~     my_os      script4.sh~  Документы
1.txt      3rd.sh     911.sh    ex2.sh~   ex3.sh~   lab07.sh  number     script4.sh~  Загрузки
2.doc      3rd.sh~    911.sh~   ex3.sh    ex3.sh~   lab07.sh~ number.c~   script.sh   Изображения
2.jpg      4.doc      abc1       ex3.sh~   ex.sh~    lab08     number.c~  script.sh~   Музыка
2nd.sh     4.jpg      australia  ex.sh     ex.sh~    l.log     number.c~  second.txt   Общиедоступные
2nd.sh~    4th.sh     backup     ex.sh~    ex.sh~    l.log     number.sh  'script2.sh#' ski.plases   'Рабочий стол'

dvkasjyanov@dvkasjyanov:~$
```

(Рисунок 15)

3 Контрольные вопросы

3. 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 man$/$i.$s` проверяет, существует ли файл `man$/$i.$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

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

5 Библиография

[Лабораторная работа №12 - “Программирование в командном процессоре ОС UNIX. Ветвления и циклы”](<https://esystem.rudn.ru/mod/resource/view.php?id=718607>)