

Отчёт по лабораторной работе №10

Дисциплина: Операционные Системы

Шишук Владислав Олегович, НПМбд-03-21

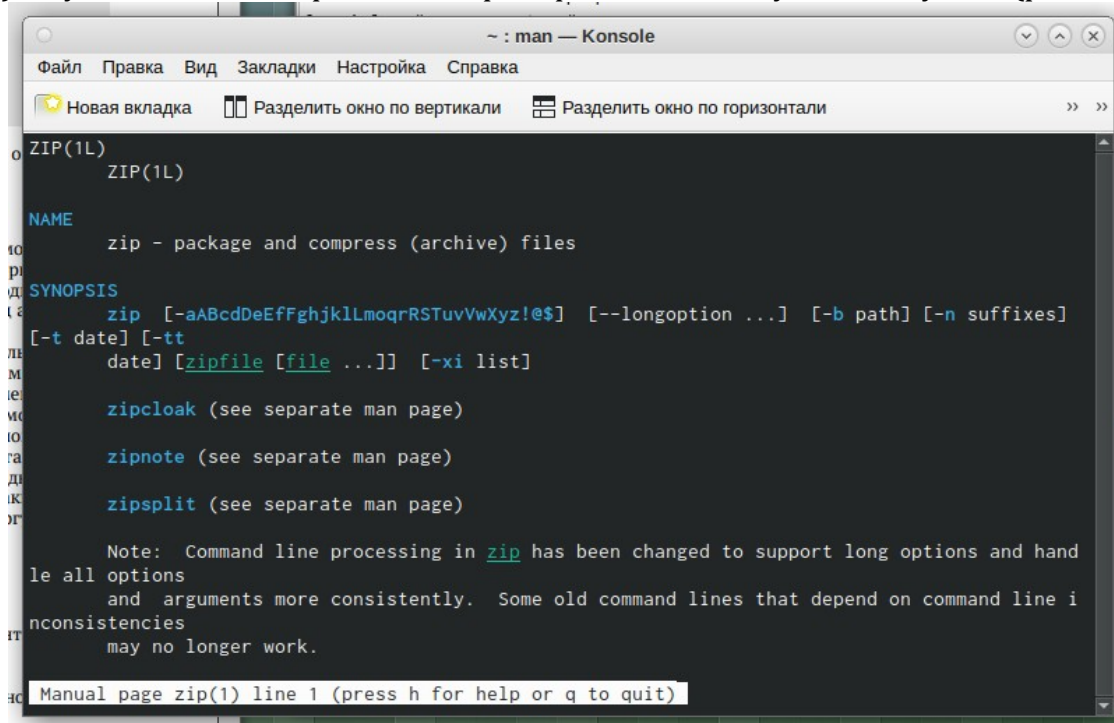
Содержание

Цель работы

Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

Выполнение лабораторной работы

1). Изучаем команды архивации zip, bzip2, tar, используя команду man (рис.1-3)



```
~ : man — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
+-----+
+ Новая вкладка  + Разделить окно по вертикали  + Разделить окно по горизонтали  +
+-----+
ZIP(1L)
ZIP(1L)

NAME
    zip - package and compress (archive) files

SYNOPSIS
    zip [-aABcdDeEfFghjklLmoqrRSTuvVwXyz!@$_] [--longoption ...] [-b path] [-n suffixes]
    [-t date] [-tt date] [zipfile [file ...]] [-xi list]

    zipcloak (see separate man page)
    zipnote (see separate man page)
    zipsplit (see separate man page)

Note: Command line processing in zip has been changed to support long options and handle all options and arguments more consistently. Some old command lines that depend on command line inconsistencies may no longer work.

Manual page zip(1) line 1 (press h for help or q to quit)
```

```
~ : man — Konsole
Файл Правка Вид Закладки Настройка Справка
Новая вкладка Разделить окно по вертикали Разделить окно по горизонтали >> >>

bzip2(1)                                General Commands Manual                                bzip2(1)

NAME
  bzip2, bunzip2 - a block-sorting file compressor, v1.0.8
  bzipcat - decompresses files to stdout
  bzip2recover - recovers data from damaged bzip2 files

SYNOPSIS
  bzip2 [ -cdfkqstvwVL123456789 ] [ filenames ... ]
  bunzip2 [ -fkvsVL ] [ filenames ... ]
  bzipcat [ -s ] [ filenames ... ]
  bzip2recover filename

DESCRIPTION
  bzip2 compresses files using the Burrows-Wheeler block sorting text compression algorithm, and Huffman coding. Compression is generally considerably better than that achieved by more conventional LZ77/LZ78-based compressors, and approaches the performance of the PPM family of statistical compressors.

  The command-line options are deliberately very similar to those of GNU gzip, but they are not identical.

  bzip2 expects a list of file names to accompany the command-line flags. Each file
Manual page bzip2(1) line 1 (press h for help or q to quit)
```

```
~ : man — Konsole
Файл Правка Вид Закладки Настройка Справка
Новая вкладка Разделить окно по вертикали Разделить окно по горизонтали >> >>

TAR(1)                                GNU TAR Manual                                TAR(1)

NAME
  tar - an archiving utility

SYNOPSIS
  Traditional usage
    tar {A|c|d|r|t|u|x}[GnSkUWompsMBiajJzZhPlRvwo] [ARG...]

  UNIX-style usage
    tar -A [OPTIONS] ARCHIVE ARCHIVE

    tar -c [-f ARCHIVE] [OPTIONS] [FILE...]

    tar -d [-f ARCHIVE] [OPTIONS] [FILE...]

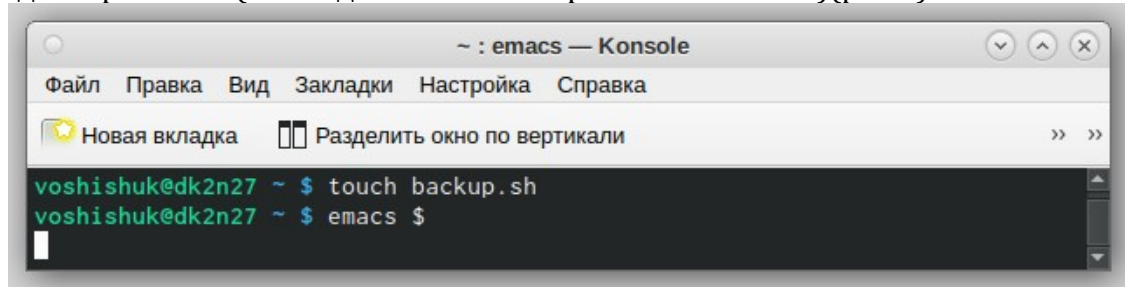
    tar -t [-f ARCHIVE] [OPTIONS] [MEMBER...]

    tar -r [-f ARCHIVE] [OPTIONS] [FILE...]

    tar -u [-f ARCHIVE] [OPTIONS] [FILE...]

    tar -x [-f ARCHIVE] [OPTIONS] [MEMBER...]
Manual page tar(1) line 1 (press h for help or q to quit)
```

- создаем файл, в котором будем писать первый скрипт, и открываем его в редакторе emacs (команды «touch backup.sh» и «emacs &»)(рис.4)

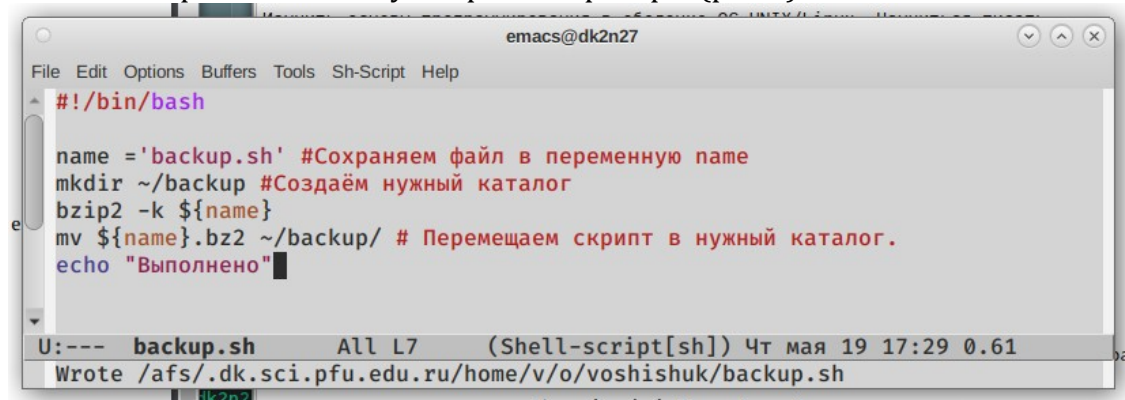


```

~ : emacs — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
Новая вкладка  Разделить окно по вертикали
voshishuk@dk2n27 ~ $ touch backup.sh
voshishuk@dk2n27 ~ $ emacs &

```

- Пишем скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в нашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. При написании скрипта используем архиватор bzip2. (рис.5)



```

emacs@dk2n27
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash

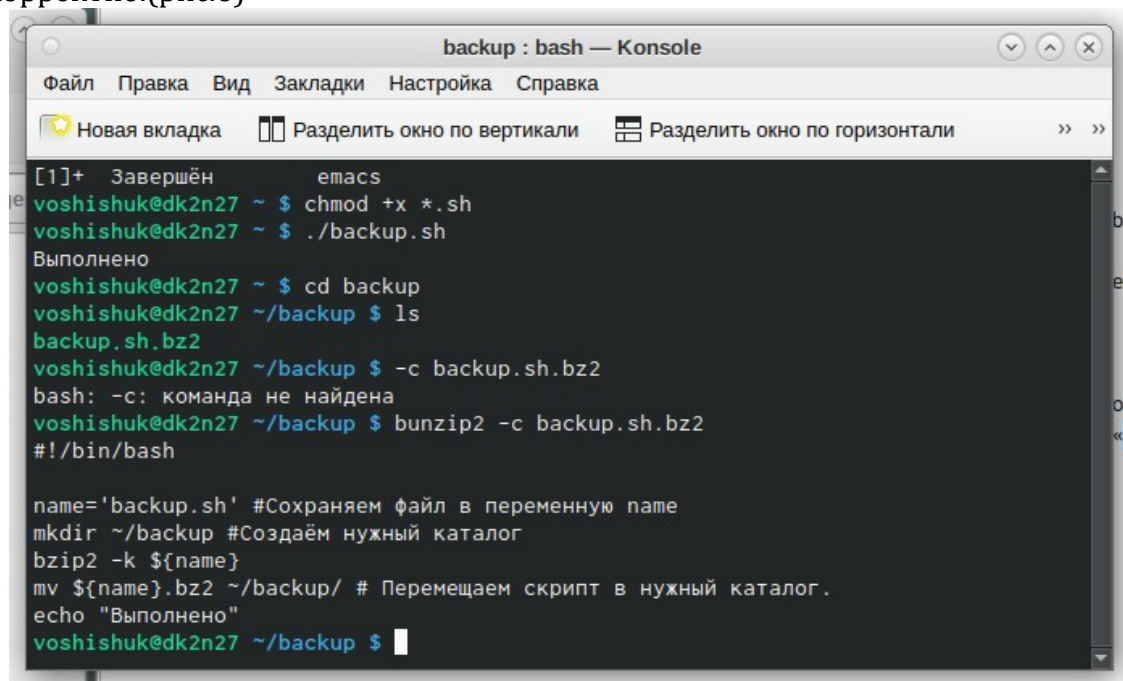
name='backup.sh' #Сохраняем файл в переменную name
mkdir ~/backup #Создаём нужный каталог
bzip2 -k ${name}
mv ${name}.bz2 ~/backup/ # Перемещаем скрипт в нужный каталог.
echo "Выполнено"

```

U:--- backup.sh All L7 (Shell-script[sh]) Чт мая 19 17:29 0.61
Wrote /afs/.dk.sci.pfu.edu.ru/home/v/o/voshishuk/backup.sh

- Проверяем работу скрипта (команда «./backup.sh»), перед этим добавив для него право на выполнение (команда «chmod +x *.sh»). Проверяем, появился ли каталог backup/, перейдя в него (команда «cd backup/»), просматриваем содержимое архива (команда «bunzip2 -c backup.sh.bz2»). Скрипт работает

корректно.(рис.6)



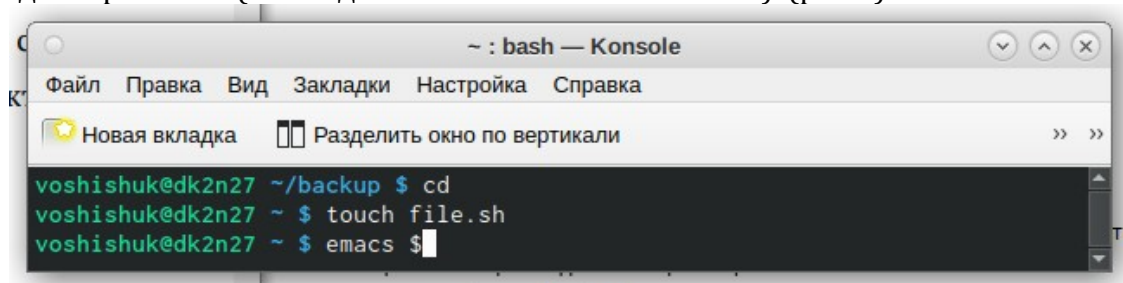
```
backup : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
Новая вкладка  Разделить окно по вертикали  Разделить окно по горизонтали  >> >>

[1]+  Завершён      emacs
voshishuk@dk2n27 ~ $ chmod +x *.sh
voshishuk@dk2n27 ~ $ ./backup.sh
Выполнено
voshishuk@dk2n27 ~ $ cd backup
voshishuk@dk2n27 ~/backup $ ls
backup.sh.bz2
voshishuk@dk2n27 ~/backup $ -c backup.sh.bz2
bash: -c: команда не найдена
voshishuk@dk2n27 ~/backup $ bunzip2 -c backup.sh.bz2
#!/bin/bash

name='backup.sh' #Сохраняем файл в переменную name
mkdir ~/backup #Создаём нужный каталог
bzip2 -k ${name}
mv ${name}.bz2 ~/backup/ # Перемещаем скрипт в нужный каталог.
echo "Выполнено"
voshishuk@dk2n27 ~/backup $
```

2).

- Создаем файл, в котором буду писать второй скрипт, и открываем его в редакторе emacs (команды «touch file.sh» и «emacs &»). (рис.7)

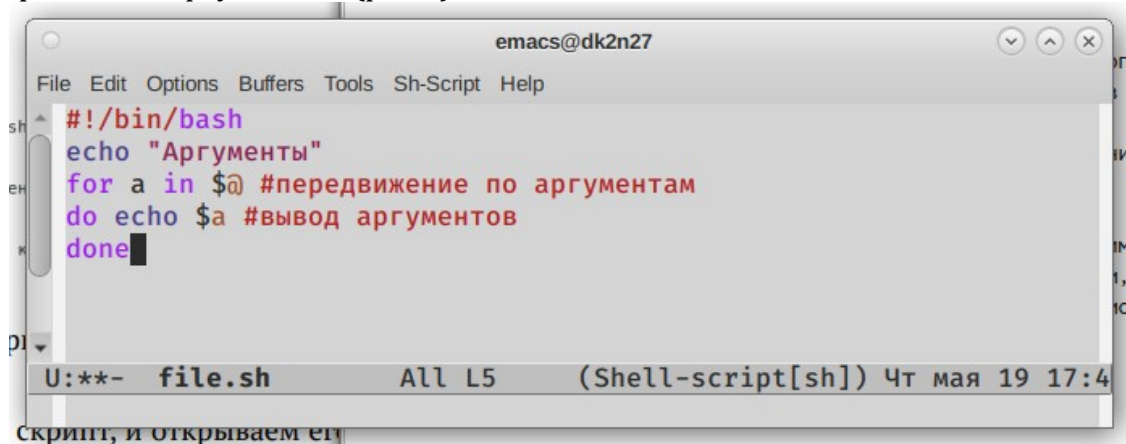


```
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
Новая вкладка  Разделить окно по вертикали  >> >>

voshishuk@dk2n27 ~/backup $ cd
voshishuk@dk2n27 ~ $ touch file.sh
voshishuk@dk2n27 ~ $ emacs $
```

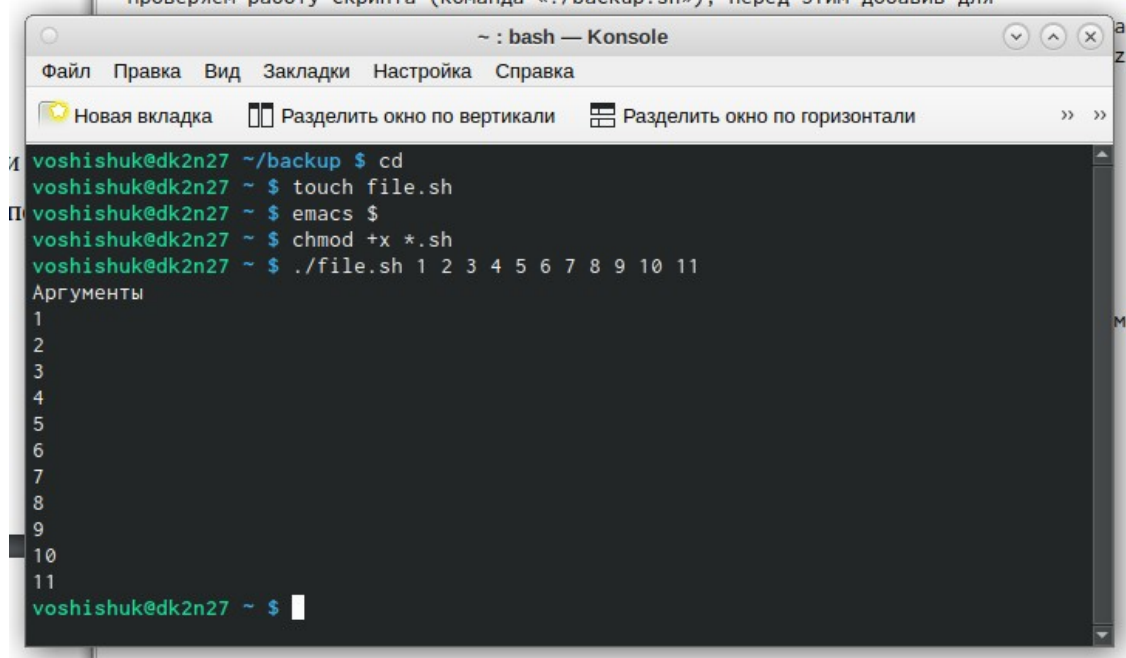
- Пишем пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех

переданных аргументов. (рис.8)



```
emacs@dk2n27
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
echo "Аргументы"
for a in $@ #передвижение по аргументам
do echo $a #вывод аргументов
done
U:***- file.sh All L5 (Shell-script[sh]) Чт мая 19 17:4
```

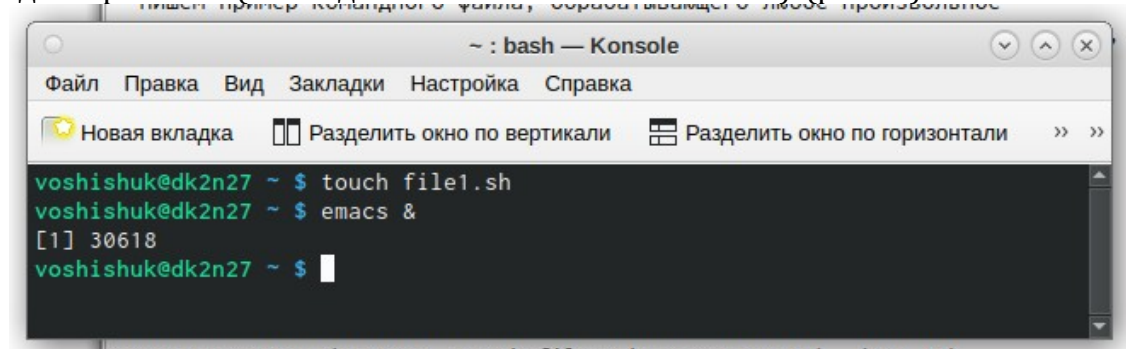
- Проверил работу написанного скрипта (рис.9)



```
~ : bash — Konsole
Файл Правка Вид Закладки Настройка Справка
Новая вкладка Разделить окно по вертикали Разделить окно по горизонтали
voshishuk@dk2n27 ~/backup $ cd
voshishuk@dk2n27 ~ $ touch file.sh
voshishuk@dk2n27 ~ $ emacs $
voshishuk@dk2n27 ~ $ chmod +x *.sh
voshishuk@dk2n27 ~ $ ./file.sh 1 2 3 4 5 6 7 8 9 10 11
Аргументы
1
2
3
4
5
6
7
8
9
10
11
voshishuk@dk2n27 ~ $
```

3).

- Создаем файл, в котором буду писать третий скрипт, и открываем его в редакторе emacs (команды «touch file1.sh» и «emacs &»). (рис.10)



```
~ : bash — Konsole
Файл Правка Вид Закладки Настройка Справка
Новая вкладка Разделить окно по вертикали Разделить окно по горизонтали
voshishuk@dk2n27 ~ $ touch file1.sh
voshishuk@dk2n27 ~ $ emacs &
[1] 30618
voshishuk@dk2n27 ~ $
```


- Пишем командный файл – аналог команды ls (без использования самой этой команды и команды dir). Он должен выдавать информацию о нужном каталоге и выводить информацию о возможностях доступа к файлам этого каталога (рис.11)

```
#!/bin/bash
a="$1" #сохраняем путь до каталога
for i in ${a}/* #цикл по файлам каталога
do
    echo "$i"

    if test -f $i #проверка обычности файлов
    then echo "Обычный файл"
    fi

    if test -d $i #проверка каталога
    then echo "Каталог"
    fi

    if test -r $i #право на чтение
    then echo "Чтение разрешено"
    fi

    if test -w $i #право на изменение
    then echo "Запись разрешена"
    fi
done
```

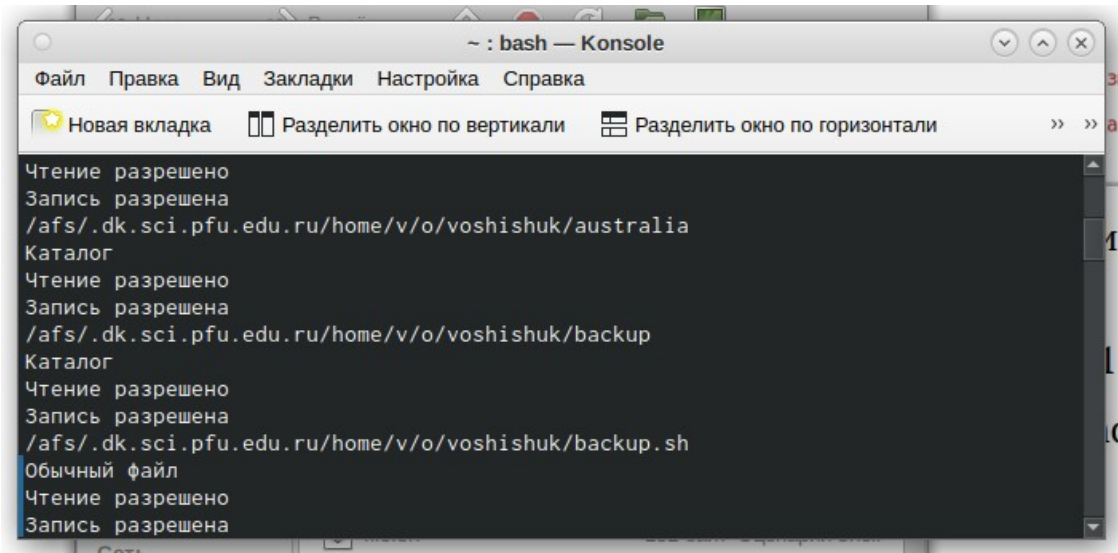
U:--- file1.sh All L1 (Shell-script[bash]) Чт мая 19 18:12 1.09

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.

U:%*- *Warnings* Bot L8 (Special) Чт мая 19 18:12 1.09

Beginning of buffer

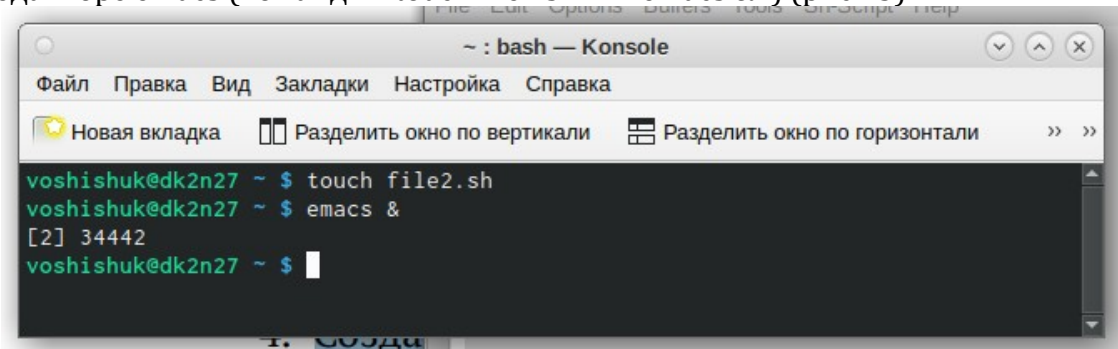
- Далее проверяем работу скрипта (команда «./file1.sh ~»), предварительно добавив для него право на выполнение (команда «chmod +x *.sh»). Скрипт работает корректно. (рис.12)



```
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
Новая вкладка  Разделить окно по вертикали  Разделить окно по горизонтали
Чтение разрешено
Запись разрешена
/afs/.dk.sci.pfu.edu.ru/home/v/o/voshishuk/australia
Каталог
Чтение разрешено
Запись разрешена
/afs/.dk.sci.pfu.edu.ru/home/v/o/voshishuk/backup
Каталог
Чтение разрешено
Запись разрешена
/afs/.dk.sci.pfu.edu.ru/home/v/o/voshishuk/backup.sh
Обычный файл
Чтение разрешено
Запись разрешена
```

4).

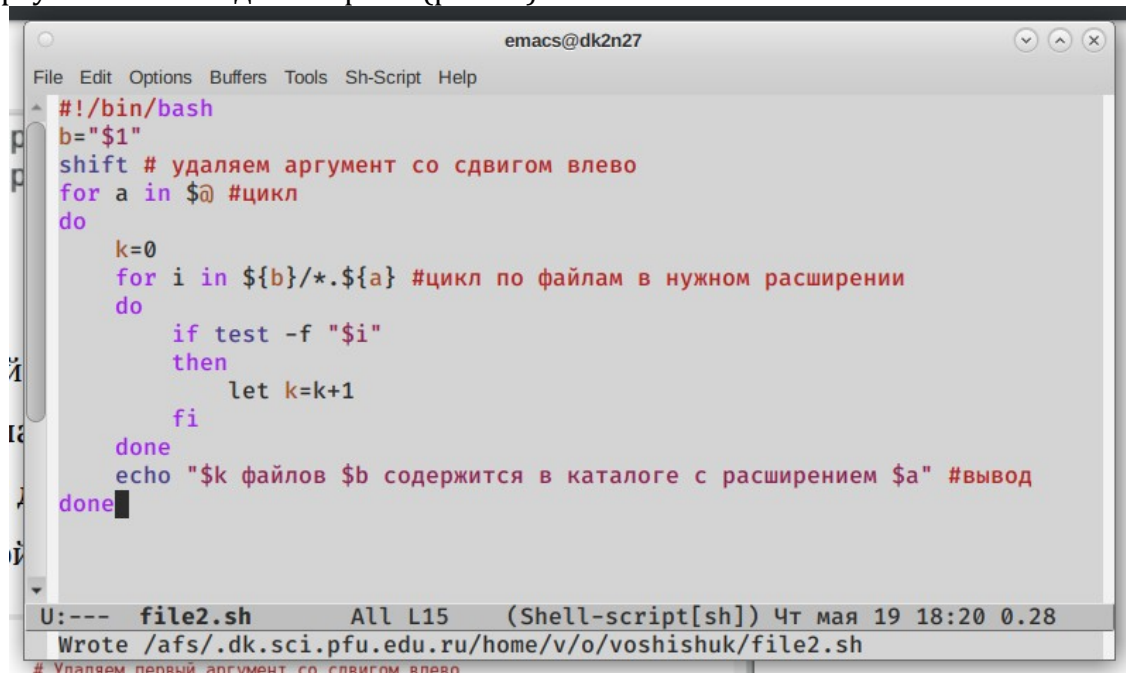
- Создаем файл, в котором буду писать третий скрипт, и открываем его в редакторе emacs (команды «touch file2.sh» и «emacs &»).(рис.13)



```
~ : bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
Новая вкладка  Разделить окно по вертикали  Разделить окно по горизонтали
voshishuk@dk2n27 ~ $ touch file2.sh
voshishuk@dk2n27 ~ $ emacs &
[2] 34442
voshishuk@dk2n27 ~ $
```

- Пишем командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде

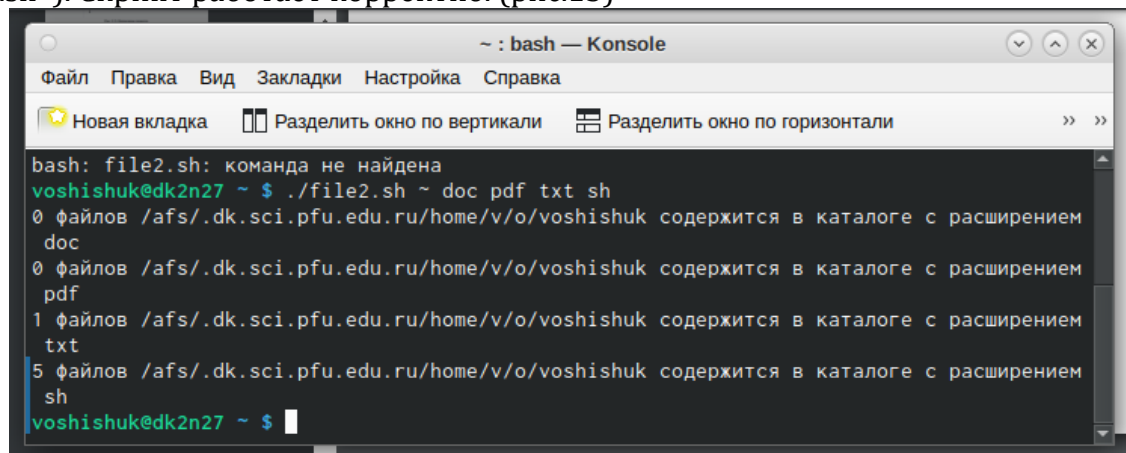
аргумента командной строки.(рис.14)



```
#!/bin/bash
b="$1"
shift # удаляем аргумент со сдвигом влево
for a in $@ #цикл
do
    k=0
    for i in ${b}/*.${a} #цикл по файлам в нужном расширении
    do
        if test -f "$i"
        then
            let k=k+1
        fi
    done
    echo "$k файлов $b содержится в каталоге с расширением $a" #вывод
done
```

U:--- file2.sh All L15 (Shell-script[sh]) Чт мая 19 18:20 0.28
Wrote /afs/.dk.sci.pfu.edu.ru/home/v/o/voshishuk/file2.sh
Удаляем первый аргумент со сдвигом влево

- Проверяем работу написанного скрипта (команда «./file.sh ~ doc pdf txt sh»), предварительно добавив для него право на выполнение (команда «chmod +x *.sh»). Скрипт работает корректно. (рис.15)



```
bash: file2.sh: команда не найдена
voshishuk@dk2n27 ~ $ ./file2.sh ~ doc pdf txt sh
0 файлов /afs/.dk.sci.pfu.edu.ru/home/v/o/voshishuk содержится в каталоге с расширением doc
0 файлов /afs/.dk.sci.pfu.edu.ru/home/v/o/voshishuk содержится в каталоге с расширением pdf
1 файлов /afs/.dk.sci.pfu.edu.ru/home/v/o/voshishuk содержится в каталоге с расширением txt
5 файлов /afs/.dk.sci.pfu.edu.ru/home/v/o/voshishuk содержится в каталоге с расширением sh
voshishuk@dk2n27 ~ $
```

Контрольные вопросы

1. Командный процессор (командная оболочка, интерпретатор команд shell) – это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера.
2. POSIX (Portable Operating System Interface for Computer Environments) – набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux подобных операционных систем и

переносимости прикладных программ на уровне исходного кода.
POSIXсовместимые оболочки разработаны на базе оболочки Корна.

3. Командный процессор `bash` обеспечивает возможность использования переменных типа строка символов. Имена переменных могут быть выбраны пользователем. Пользователь имеет возможность присвоить переменной значение некоторой строки символов. Например, команда `mark=/usr/andy/bin` присваивает значение строки символов `/usr/andy/bin` переменной `mark` типа строка символов. Значение, присвоенное некоторой переменной, может быть впоследствии использовано. Для этого в соответствующем месте командной строки должно быть употреблено имя этой переменной, которому предшествует метасимвол `$`. Например, команда `mv afile ${mark}` переместит файл `afile` из текущего каталога в каталог с абсолютным полным именем `/usr/andy/bin`. Оболочка `bash` позволяет работать с массивами. Для создания массива используется команда `set` с флагом `-A`. За флагом следует имя переменной, а затем список значений, разделённых пробелами. Например, `set -A states Delaw -are Michigan "New Jersey"` Далее можно сделать добавление в массив, например, `states[49]=Alaska`. Индексация массивов начинается с нулевого элемента.
4. Оболочка `bash` поддерживает встроенные арифметические функции. Команда `let` является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Простейшее выражение – это единичный терм (`term`), обычно целочисленный. Команда `let` берет два операнда и присваивает их переменной. Команда `read` позволяет читать значения переменных со стандартного ввода: `echo "Please enter Month and Day of Birth ?"` `read mon day trash` В переменные `mon` и `day` будут считаны соответствующие значения, введённые с клавиатуры, а переменная `trash` нужна для того, чтобы отобразить всю избыточно введённую информацию.
5. В языке программирования `bash` можно применять такие арифметические операции как сложение (+), вычитание (-), умножение (*), целочисленное деление (/) и целочисленный остаток от деления (%).
6. В (()) можно записывать условия оболочки `bash`, а также внутри двойных скобок можно вычислять арифметические выражения и возвращать результат.ю и игнорировать её.
- 7.
8. Такие символы, как ' < > * ? | " &, являются метасимволами и имеют для командного процессора специальный смысл.
9. Снятие специального смысла с метасимвола 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`.
10. Последовательность команд может быть помещена в текстовый файл. Такой файл называется командным. Далее этот файл можно выполнить по команде:

«bash командный_файл [аргументы]» Чтобы не вводить каждый раз последовательности символов bash, необходимо изменить код защиты этого командного файла, обеспечив доступ к этому файлу по выполнению. Это может быть сделано с помощью команды «chmod +x имя_файла» Теперь можно вызывать свой командный файл на выполнение, просто вводя его имя с терминала так, как будто он является выполняемой программой. Командный процессор распознает, что в Вашем файле на самом деле хранится не выполняемая программа, а программа, написанная на языке программирования оболочки, и осуществит её интерпретацию.

11. Группу команд можно объединить в функцию. Для этого существует ключевое слово `function`, после которого следует имя функции и список команд, заключённых в фигурные скобки. Удалить функцию можно с помощью команды `unset` с флагом `-f`.
12. Чтобы выяснить, является ли файл каталогом или обычным файлом, необходимо воспользоваться командами «`test -f [путь до файла]`» (для проверки, является ли обычным файлом) и «`test -d [путь до файла]`» (для проверки, является ли каталогом).
13. Команду «`set`» можно использовать для вывода списка переменных окружения. В системах Ubuntu и Debian команда «`set`» также выведет список функций командной оболочки после списка переменных командной оболочки. Поэтому для ознакомления со всеми элементами списка переменных окружения при работе с данными системами рекомендуется использовать команду «`set | more`». Команда «`typeset`» предназначена для наложения ограничений на переменные. Команду «`unset`» следует использовать для удаления переменной из окружения командной оболочки.
14. При вызове командного файла на выполнение параметры ему могут быть переданы точно таким же образом, как и выполняемой программе. С точки зрения командного позиционными. Символ `$` файла является эти параметры метасимволом являются командного процессора. Он используется, в частности, для ссылки на параметры, точнее, для получения их значений в командном файле. В командный файл можно передать до девяти параметров. При использовании где-либо в командном файле комбинации символов `$i`, где $0 < i < 10$, вместо неё будет осуществлена подстановка значения параметра с порядковым номером i , т. е. аргумента командного файла с порядковым номером i . Использование комбинации символов `$0` приводит к подстановке вместо неё имени данного командного файла.

Выводы

Я изучил основы программирования в оболочке ОС UNIX/Linux и научился писать небольшие командные файлы.