

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 11
Программирование в командном процессоре ОС UNIX. Командные файлы
дисциплина: Операционные системы

Студент: Тазаева Анастасия Анатольевна
Группа: НПИбд-02-20

МОСКВА 2021г.

Цель работы:

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

Ход работы:

1. *Написать скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации необходимо узнать, изучив справку*

Командной строкой создала файл first_task.sh (emacs first_task.sh)(рис.1), в который и был написан скрипт (рис.2). Далее с помощью команды ls проверила архивируется файл или нет.(рис.3) Чтобы выполнить командный файл пришлось дать права на выполнение(chmod u+x first_task.sh)

```
aatazaeva@dk6n61 ~ $ emacs first_task.sh
```

(рис.1)



```
File Edit Options Buffers Tools Sh-Script Help
res='first_task.sh'
cp "$0" "$res"
tar -cf first_task.tar $res

-:--- first_task.sh All L1 (Shell-script[sh]) Пт мая 28 14:30 0.56
```

(рис.2)

```

aatazaeva@dk6n61 ~ $ ls
'!'          lab07.sh      wshb
abc1         may          Видео
australia   monthly     Документы
conf.txt    my_os       Загрузки
feathers     OS          Изображения
file.txt    public      Музыка
'#first_task.sh#' public_html  Общедоступные
first_task.sh reports      пробный_код_с_векторами
first_task.sh~ ski.places  пробный_код_с_векторами.ср
first_task.tar text.txt    'Рабочий стол'
GNUstep     tmp        Шаблоны
'#lab07.sh#' work

```

(рис.3)

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

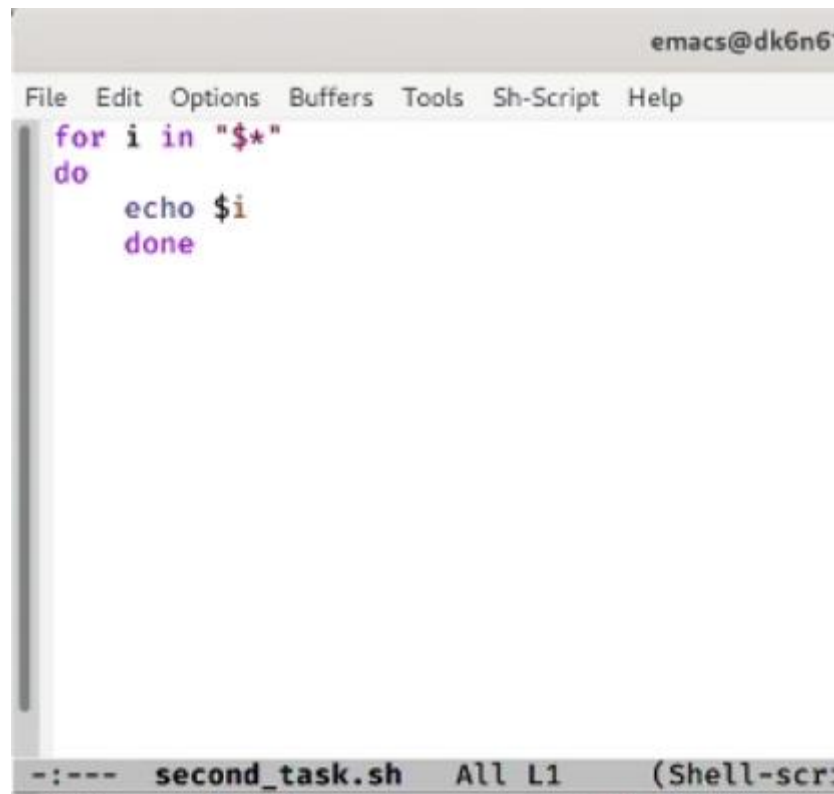
Аналогично создала файл для второго задания (emacs second_task.sh)(рис.4), в нем и был написан код,(рис.5) Чтобы выполнить файл дали права(chmod u+x second_task.sh) (рис.6) и проверили работу командного файла (рис.7)

```

aatazaeva@dk6n61 ~ $ emacs second_task.sh

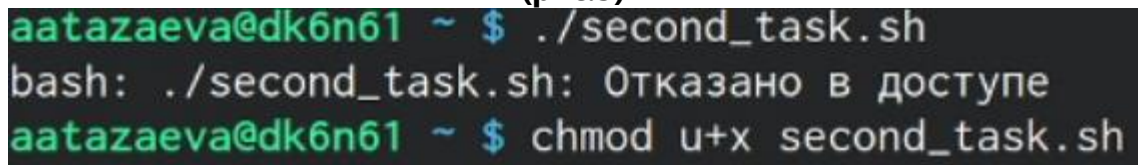
```

(рис.4)



```
emacs@dk6n61
File Edit Options Buffers Tools Sh-Script Help
for i in "$*"
do
    echo $i
done
-:--- second_task.sh All L1 (Shell-scr)
```

(рис.5)



```
aatazaeva@dk6n61 ~ $ ./second_task.sh
bash: ./second_task.sh: Отказано в доступе
aatazaeva@dk6n61 ~ $ chmod u+x second_task.sh
```

(рис.6)



```
aatazaeva@dk6n61 ~ $ ./second_task.sh 1 2 3 4 5 6 q w e r t y
1
2
3
4
5
6
q
w
e
r
t
y
```

(рис.7)

3. Написать командный файл — аналог команды *ls* (без использования самой этой команды и команды *dir*). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога

Создала файл(рис.8), с написанным скриптом(рис.9_1), не забываем дать права юзеру на выполнение(рис.9). После чего смотрим, как работает программа(рис.10). Для того, чтобы понять работает она правильно или нет, пропишем команду *ls -l*(рис.11). Сравнивая выполнение командного файла и данные домашней директории, убедимся что скрипт верный. На примере файла *abc1*, скрипт вывел нам, что это файл, есть права на чтение и выполнение, сверяем, *-rw*, действительно это так.

aatazaeva@dk6n61 ~ \$ emacs third_task.sh

```
for i in *
do if test -d $i
  then echo $i: 'Dir'
  else echo -n $i: 'File'
    if test -w $i
    then echo "Ready to write"
      if test -r $i
      then echo "Ready to read"
      else echo "Not ready to anything"
      fi
    fi
  fi
done
```

~:--- third_task.sh All L1 (Shell-script[

(pic 9.1)

aatazaeva@dk6n61 ~ \$ chmod u+x third_task.sh

(pic 9)

aatazaeva@dk6n61 ~ \$./third_task.sh

```
!: FileReady to write
Ready to read
abc1: FileReady to write
Ready to read
australia: Dir
conf.txt: FileReady to write
Ready to read
feathers: FileReady to write
Ready to read
file.txt: FileReady to write
Ready to read
#first_task.sh#: FileReady to write
```

(pic.10)

```

aatazaeva@dk6n61 ~ $ ls -l
итого 79
-rw-r--r-- 1 aatazaeva studsci 105 мая 19 16:41 '!'
-rw-rw-r-- 1 aatazaeva studsci 0 мая 19 12:40 abc1
drwxr--r-- 3 aatazaeva studsci 2048 мая 19 16:00 australia
-rw-r--r-- 1 aatazaeva studsci 1233 мая 19 14:15 conf.txt
-rw-rw-r-- 1 aatazaeva studsci 0 мая 19 13:09 feathers
-rw-r--r-- 1 aatazaeva studsci 122 мая 19 14:45 file.txt
-rwxr--r-- 1 aatazaeva studsci 64 мая 28 14:17 '#first_task.sh#'
-rwxr--r-- 1 aatazaeva studsci 63 мая 28 14:24 first_task.sh
-rwxr--r-- 1 aatazaeva studsci 68 мая 28 14:23 first_task.sh~
-rw-r--r-- 1 aatazaeva studsci 10240 мая 28 14:30 first_task.tar
drwxr-xr-x 3 aatazaeva studsci 2048 мая 14 16:27 GNUstep
-rw-r--r-- 1 aatazaeva studsci 97 мая 21 16:06 '#lab07.sh#'
-rw-r--r-- 1 aatazaeva studsci 98 мая 21 15:39 lab07.sh
-rw-r--r-- 1 aatazaeva studsci 0 мая 19 12:20 may
drwx--x--x 2 aatazaeva studsci 2048 мая 19 12:05 monthly
-r-xr--r-- 1 aatazaeva studsci 0 мая 19 13:06 my_os

```

(рис.11)

4. Написать 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.

Для 4 задания создали файл,(рис.12) в нем же и написали скрипт,(рис.13) и проверили(рис.14)

```

aatazaeva@dk6n61 ~ $ emacs fourth_task.sh

```

(рис.12)



```
emacs@
File Edit Options Buffers Tools Sh-Script Help
find $1 -name "*. $2" -type f | wc -l
-:--- fourth_task.sh All L1 (Shell)
```

(рис.13)



```
aatazaeva@dk6n61 ~ $ ./fourth_task.sh ~ txt
43
```

(рис.14)

Контрольные вопросы:

Контрольные вопросы:

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются? Программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. Оболочка Борна - стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций. C-оболочка - надстройка над оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд. Оболочка Корна - напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна. BASH - сокращение от Bourne Again Shell, в основе своей совмещает свойства оболочек C и Корна.
2. Что такое POSIX? Набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ
3. Как определяются переменные и массивы в языке программирования bash? Переменная/=значение. set -A (переменная), (список значений)
4. Каково назначение операторов let и read? let - берет два операнда и присваивает их переменной. read - чтение значения переменных со стандартного ввода.

5. Какие арифметические операции можно применять в языке программирования bash? Операции логики, умножение, деление, сложение, вычитание.
6. Что означает операция (())? Условия оболочки bash
7. Какие стандартные имена переменных Вам известны? PATH, IFS, MAIL, TERM, LOGNAME.
8. Что такое метасимволы? Символы ' < > * ? | \ " &, являются метасимволами и имеют для командного процессора отличный от обычных символом смысл (они технически влияют на поведение программы).
9. Как экранировать метасимволы? 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 \$, ' , , " .
10. Как создавать и запускать командные файлы? bash <командный_файл> [аргументы] chmod +x <командный_файл> ./командный_файл
11. Как определяются функции в языке программирования bash? Ключевое слово function <fun_name>{тело функции}
12. Каким образом можно выяснить, является файл каталогом или обычным файлом? – test -d file — истина, если файл file является каталогом.
13. Каково назначение команд set, typeset и unset? Оболочка bash позволяет работать с массивами. Для создания массива используется команда set с флагом -A typeset является встроенной инструкцией и предназначена для наложения ограничений на переменные С помощью команды unset можно изъять переменную из программы
14. Как передаются параметры в командные файлы? При вызове командного файла на выполнение параметры ему могут быть переданы точно таким же образом, как и выполняемой программе. С точки зрения командного файла эти параметры являются позиционными. Символ \$ является метасимволом 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