

Федеральное государственное автономное
образовательное учреждение высшего образования
Российский Университет Дружбы Народов им. Патриса Лумумбы
Математический университет имени Никольского
Факультет Физико-математических и естественных наук
Кафедра Прикладной математики и информатики

Отчет по лабораторной работе № 10

“ Программирование в командном процессоре ОС UNIX. Ветвления и циклы”

Выполнил:

Студент группы НПМбв-02-20

Сарновский Даниил

Москва

2024 год

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

Задание

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами:
 - i - inputfile — прочитать данные из указанного файла;
 - o - outputfile — вывести данные в указанный файл;
 - p - шаблон — указать шаблон для поиска;
 - C — различать большие и малые буквы;
 - n — выдавать номера строк,

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

2. Написать 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 о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.
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).
4. Написать командный файл, который с помощью команды `tag` 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`).

Теоретическое введение

Переменные в языке программирования bash

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
```

переместит файл `afile` из текущего каталога в каталог с абсолютным полным именем `/usr/andy/bin`. Использование значения, присвоенного некоторой переменной, называется подстановкой. Для того чтобы имя переменной не сливалось с символами, которые могут

следовать за ним в командной строке, при подстановке в общем случае используется следующая форма записи:

```
${имя переменной}
```

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 Delaware Michigan "New Jersey"
```

Использование арифметических вычислений. Операторы let и read

Команда let берет два операнда и присваивает их переменной. Положительным моментом команды let можно считать то, что для идентификации переменной ей не нужен знак доллара; вы можете писать команды типа `let sum=x+7`, и let будет искать переменную x и добавлять к ней 7.

Команда let также расширяет другие выражения let, если они заключены в двойные круглые скобки. Таким способом вы можете создавать довольно сложные выражения. Команда let не ограничена простыми арифметическими выражениями.

Команда read позволяет читать значения переменных со стандартного ввода:

```
echo "Please enter Month and Day of Birth?"
read mon day trash
```

Командные файлы и функции

Последовательность команд может быть помещена в текстовый файл. Такой файл называется командным. Далее этот файл можно выполнить по команде:

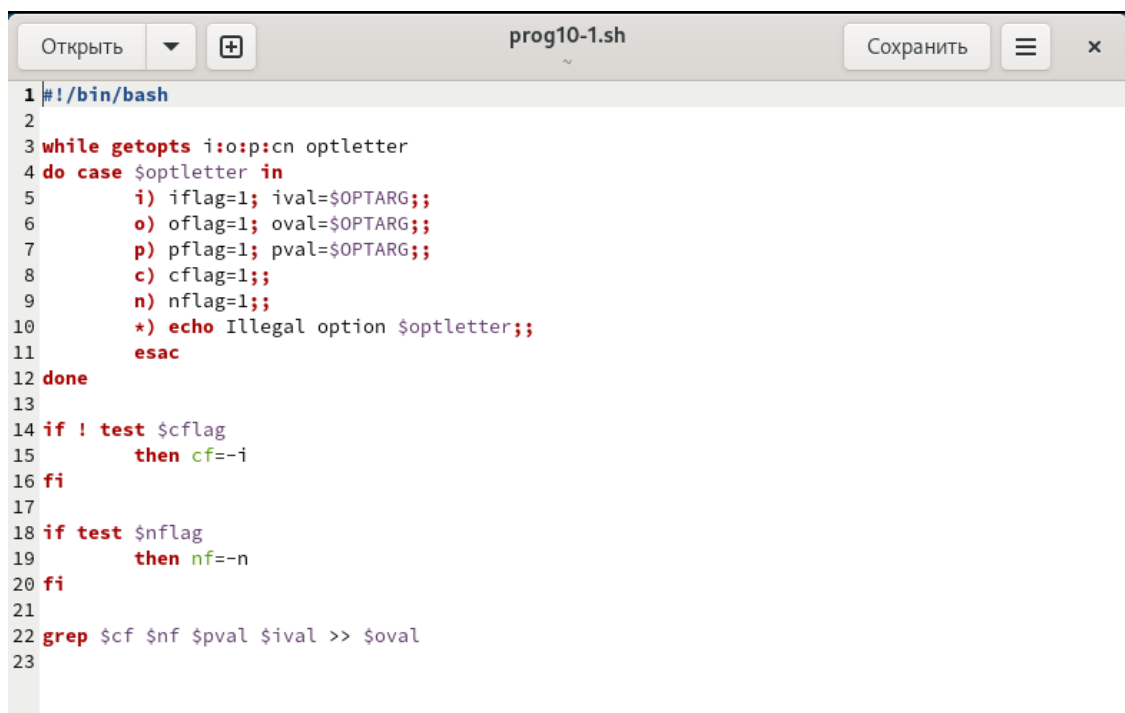
```
bash командный_файл [аргументы]
```

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
```

Выполнение лабораторной работы

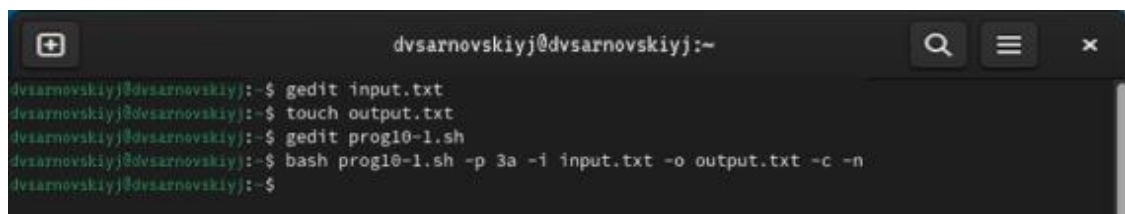
1. Используя команды `getopts` `grep`, напишем командный файл, который анализирует командную строку с ключами (`-i`, `-o`, `-p`, `-c`, `-n`), а затем ищет в указанном файле нужные строки, определяемые ключом `-p` (рис. 1):



```
1#!/bin/bash
2
3while getopts i:o:p:cn optletter
4do case $optletter in
5    i) iflag=1; ival=$OPTARG;;
6    o) oflag=1; oval=$OPTARG;;
7    p) pflag=1; pval=$OPTARG;;
8    c) cflag=1;;
9    n) nflag=1;;
10   *) echo Illegal option $optletter;;
11   esac
12done
13
14if ! test $cflag
15then cf=-i
16fi
17
18if test $nflag
19then nf=-n
20fi
21
22grep $cf $nf $pval $ival >> $oval
23
```

Рис. 1. Командный файл 1

Создадим один текстовый файл со стихотворением “input.txt” и файл, в который будет записываться результат “output.txt”. Делаем файл “prog10-1.sh” исполняемым и выводим результат (рис. 2), (рис. 3).



```
dvsarnovskiyj@dvsarnovskiyj:~
dvsarnovskiyj@dvsarnovskiyj:~$ gedit input.txt
dvsarnovskiyj@dvsarnovskiyj:~$ touch output.txt
dvsarnovskiyj@dvsarnovskiyj:~$ gedit prog10-1.sh
dvsarnovskiyj@dvsarnovskiyj:~$ bash prog10-1.sh -p 3a -i input.txt -o output.txt -c -n
dvsarnovskiyj@dvsarnovskiyj:~$
```

Рис. 2. Создание нужных файлов

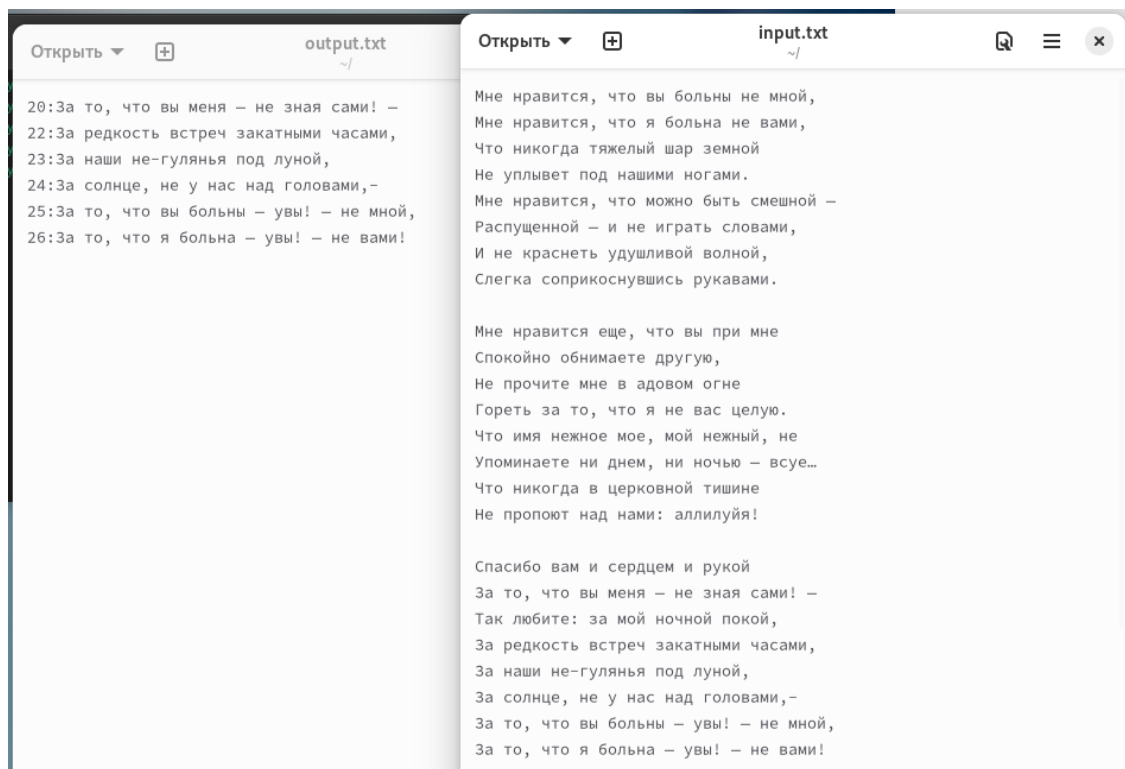


Рис. 3. Результат выполнения командного файла 1

2. Напишем 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 о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено (рис. 4), (рис. 5):

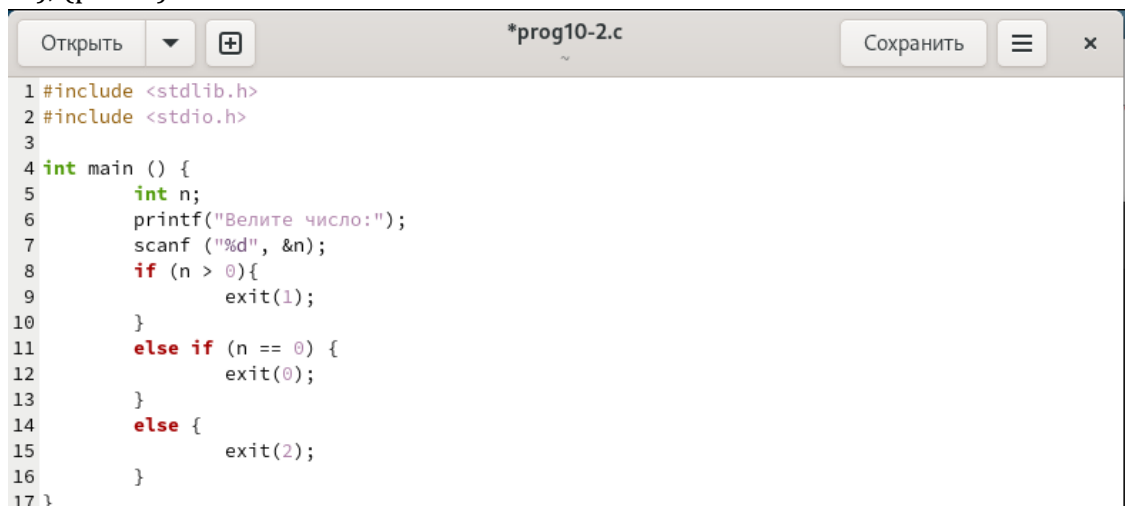
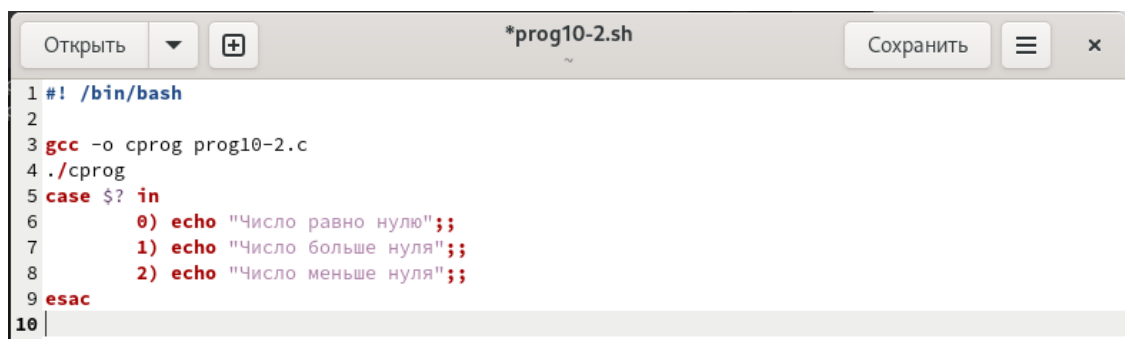


Рис. 4. Код на СИ

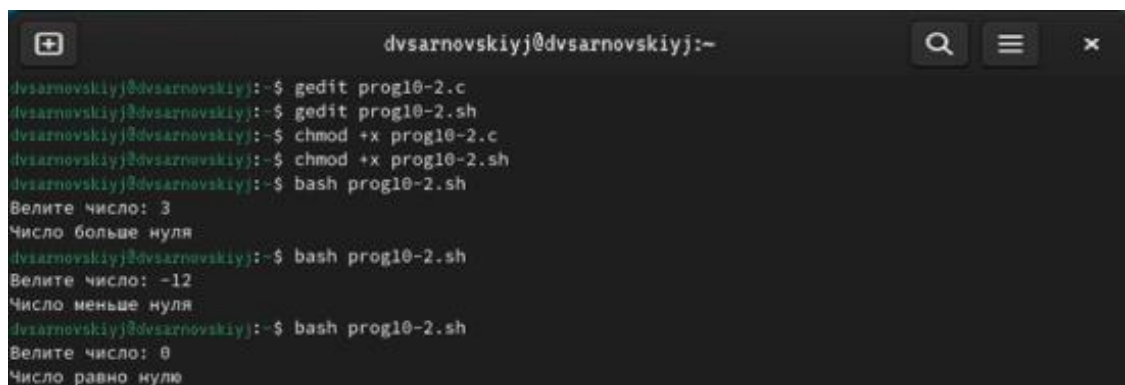
Код bash:



```
1 #!/bin/bash
2
3 gcc -o cprog prog10-2.c
4 ./cprog
5 case $? in
6     0) echo "Число равно нулю";;
7     1) echo "Число больше нуля";;
8     2) echo "Число меньше нуля";;
9 esac
10
```

Рис. 5. Код bash

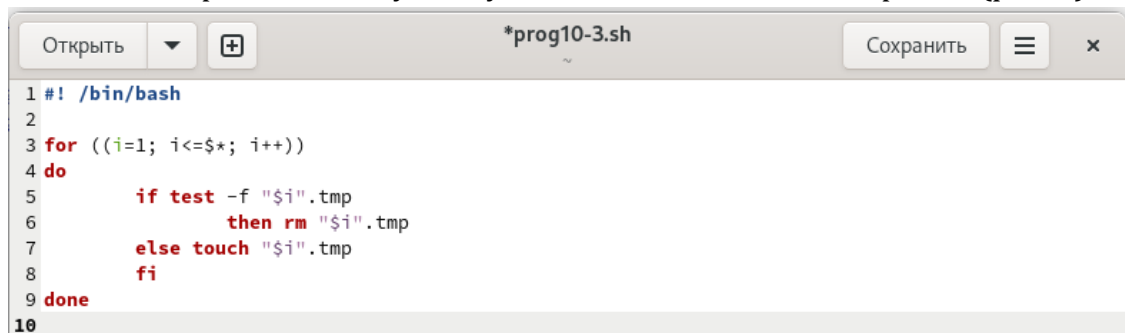
Делаем файлы исполняемыми и выводим результат (рис. 6).



```
dvsarnovskiy@dvsarnovskiyj:~$ gedit prog10-2.c
dvsarnovskiy@dvsarnovskiyj:~$ gedit prog10-2.sh
dvsarnovskiy@dvsarnovskiyj:~$ chmod +x prog10-2.c
dvsarnovskiy@dvsarnovskiyj:~$ chmod +x prog10-2.sh
dvsarnovskiy@dvsarnovskiyj:~$ bash prog10-2.sh
Велите число: 3
Число больше нуля
dvsarnovskiy@dvsarnovskiyj:~$ bash prog10-2.sh
Велите число: -12
Число меньше нуля
dvsarnovskiy@dvsarnovskiyj:~$ bash prog10-2.sh
Велите число: 0
Число равно нулю
```

Рис. 6. Результат выполнения командного файла 2

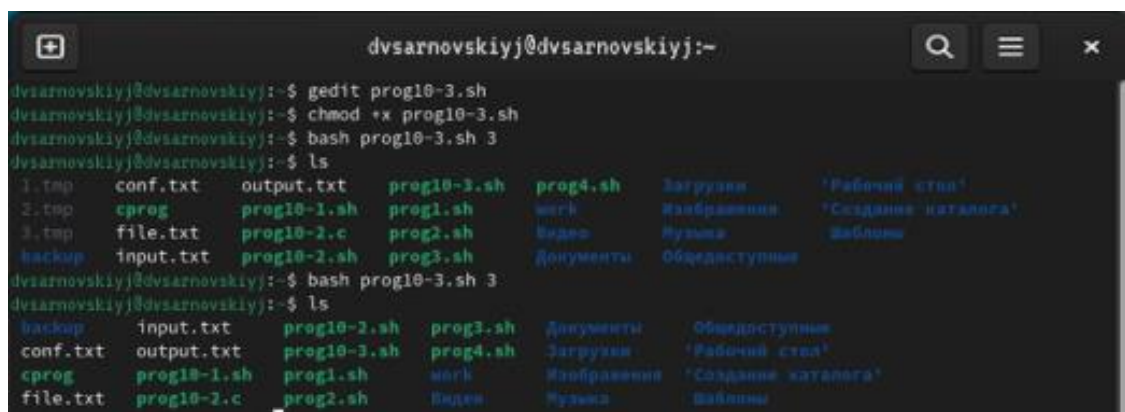
3. Напишем командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N. Число файлов, которые необходимо создать, передается в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (рис. 7):



```
1 #!/bin/bash
2
3 for ((i=1; i<=$*; i++))
4 do
5     if test -f "$i".tmp
6     then rm "$i".tmp
7     else touch "$i".tmp
8     fi
9 done
10
```

Рис. 7. Командный файл 3

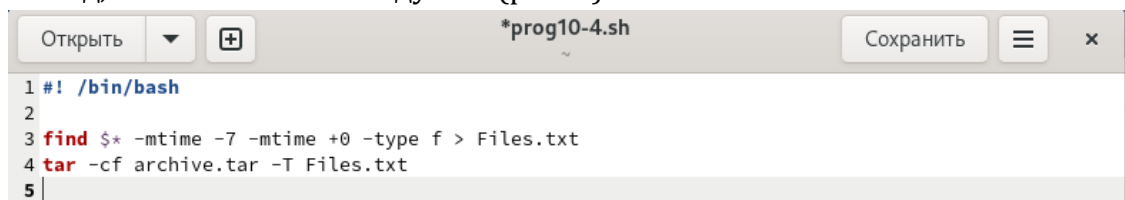
Делаем файлы исполняемыми и выводим результат (рис. 8).



```
dvsnarnovskijj@dvsnarnovskijj:~$ gedit prog10-3.sh
dvsnarnovskijj@dvsnarnovskijj:~$ chmod +x prog10-3.sh
dvsnarnovskijj@dvsnarnovskijj:~$ bash prog10-3.sh 3
dvsnarnovskijj@dvsnarnovskijj:~$ ls
1.tmp  conf.txt  output.txt  prog10-3.sh  prog4.sh  Загрузки  'Рабочий стол'
2.tmp  cprog    prog10-1.sh  prog1.sh    work      Изображения  'Создание каталога'
3.tmp  file.txt  prog10-2.c  prog2.sh    Видео     Музыка       Шаблоны
backup input.txt  prog10-2.sh  prog3.sh    Документы Общедоступные
dvsnarnovskijj@dvsnarnovskijj:~$ bash prog10-3.sh 3
dvsnarnovskijj@dvsnarnovskijj:~$ ls
backup  input.txt  prog10-2.sh  prog3.sh  Документы  Общедоступные
conf.txt  output.txt  prog10-3.sh  prog4.sh  Загрузки   'Рабочий стол'
cprog    prog10-1.sh  prog1.sh    work      Изображения 'Создание каталога'
file.txt  prog10-2.c  prog2.sh    Видео     Музыка     Шаблоны
```

Рис. 8. Результат выполнения командного файла 3

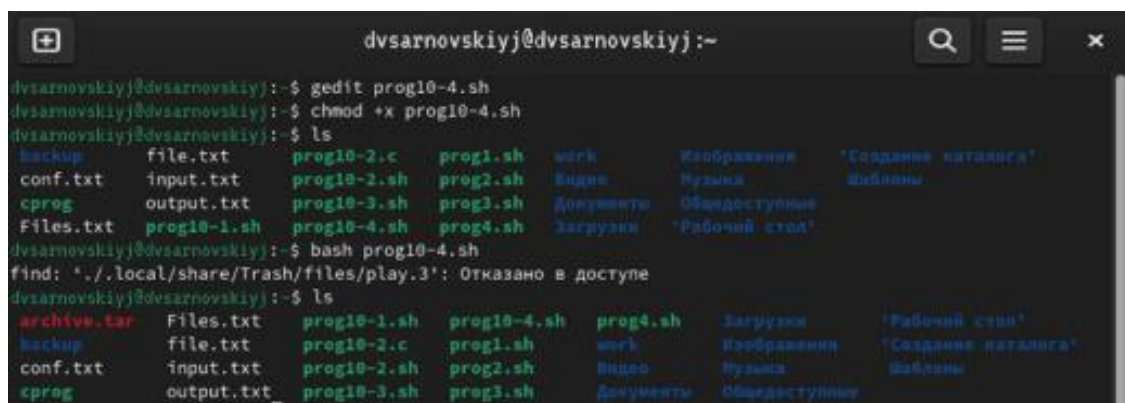
4. Напишем командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад, используя команду find (рис. 9):



```
Открыть  *prog10-4.sh  Сохранить
1 #!/bin/bash
2
3 find $* -mtime -7 -mtime +0 -type f > Files.txt
4 tar -cf archive.tar -T Files.txt
5
```

Рис. 9. Командный файл 4

Делаем файлы исполняемыми и выводим результат (рис. 10).



```
dvsnarnovskijj@dvsnarnovskijj:~$ gedit prog10-4.sh
dvsnarnovskijj@dvsnarnovskijj:~$ chmod +x prog10-4.sh
dvsnarnovskijj@dvsnarnovskijj:~$ ls
backup  file.txt  prog10-2.c  prog1.sh  work      Изображения  'Создание каталога'
conf.txt  input.txt  prog10-2.sh  prog2.sh  Видео     Музыка       Шаблоны
cprog    output.txt  prog10-3.sh  prog3.sh  Документы Общедоступные
Files.txt  prog10-1.sh  prog10-4.sh  prog4.sh  Загрузки   'Рабочий стол'
dvsnarnovskijj@dvsnarnovskijj:~$ bash prog10-4.sh
find: './.local/share/Trash/files/play.3': Отказано в доступе
dvsnarnovskijj@dvsnarnovskijj:~$ ls
archive.tar  Files.txt  prog10-1.sh  prog10-4.sh  prog4.sh  Загрузки  'Рабочий стол'
backup      file.txt  prog10-2.c  prog1.sh    work      Изображения 'Создание каталога'
conf.txt    input.txt  prog10-2.sh  prog2.sh    Видео     Музыка     Шаблоны
cprog       output.txt  prog10-3.sh  prog3.sh    Документы Общедоступные
```

Рис. 10. Результат выполнения командного файла 4

Контрольные вопросы

1. Каково предназначение команды getopts?

Команда `getopts` используется для обработки аргументов командной строки. Она позволяет извлекать опции и их значения из списка аргументов.

2. Какое отношение метасимволы имеют к генерации имён файлов?

Метасимволы используются в генерации имён файлов для сопоставления шаблонов. Например, звездочка (*) сопоставляет любое количество символов, а знак вопроса (?) сопоставляет любой один символ.

3. Какие операторы управления действиями вы знаете?

Операторы управления действиями используются для изменения потока выполнения скрипта. Вот некоторые из наиболее распространенных операторов управления действиями:

- **if...then...else:** выполняет блок кода, если условие истинно. Если условие ложно, выполняется блок кода `else` (необязательно).
 - **case...esac:** выполняет блок кода в зависимости от значения переменной.
 - **for...do...done:** выполняет блок кода для каждого элемента в списке.
 - **while...do...done:** выполняет блок кода, пока условие истинно.
 - **until...do...done:** выполняет блок кода, пока условие ложно.
4. Какие операторы используются для прерывания цикла?
- **break:** немедленно выходит из цикла.
 - **continue:** переходит к следующей итерации цикла, пропуская оставшиеся операторы в текущей итерации.
5. Для чего нужны команды `false` и `true`?

Команды `false` и `true` используются для возврата кода выхода, указывающего на успех (`true`) или неудачу (`false`).

6. Что означает строка `if test -f mans/i.$s`, встреченная в командном файле?

Эта строка проверяет, существует ли файл с именем `mans/i.$s`. Если файл существует, выполняется оператор `then`.

7. Объясните различия между конструкциями `while` и `until`.

- **while:** выполняет блок кода, пока условие истинно.
- **until:** выполняет блок кода, пока условие ложно.

Выводы

В данной лабораторной работе мы изучили основы программирования в оболочке ОС UNIX, а также научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Список литературы

1. Руководство к лабораторной работе №10.