

Лабораторная работа №13

Дисциплина: Операционные системы

Жибицкая Евгения Дмитриевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Контрольные вопросы	16
5	Выводы	17

Список иллюстраций

3.1	Создание файлов	7
3.2	Код на с#	8
3.3	Программа 2	8
3.4	Запуск 2	9
3.5	Программа 1	10
3.6	Запуск 1	12
3.7	Создание файла и наделение правами	12
3.8	Программа 3	13
3.9	Запуск 3	14
3.10	Программа 4	14
3.11	Запуск 4	15

Список таблиц

1 Цель работы

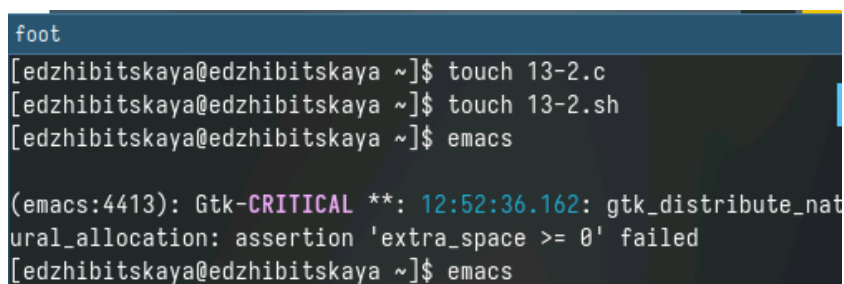
Изучение основ программирования в оболочке ОС UNIX. Приобретение навыков в написании более сложных командных файлов с использованием логических управляющих конструкций и циклов.

2 Задание

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; `-r` — 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 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.
3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до ∞ (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 `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`).

3 Выполнение лабораторной работы

Начнем выполнение задания со 2 пункта. Для этого создадим 2 файла - назовем их 13-2.sh и 13-2.c, запустим редактор emacs для написания самого кода(рис. 3.1). Не забудем прописать команду `chmod +x` для наделения файла правом на исполнение.



```
foot
[edzhibitskaya@edzhibitskaya ~]$ touch 13-2.c
[edzhibitskaya@edzhibitskaya ~]$ touch 13-2.sh
[edzhibitskaya@edzhibitskaya ~]$ emacs

(emacs:4413): Gtk-CRITICAL **: 12:52:36.162: gtk_distribute_natural_allocation: assertion 'extra_space >= 0' failed
[edzhibitskaya@edzhibitskaya ~]$ emacs
```

Рис. 3.1: Создание файлов

Реализуем задание - напишем код на `c#` и создадим командный файл, анализирующий этот код (рис. 3.2) (рис. 3.3).

```

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

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

```

Рис. 3.2: Код на с#

```

#!/bin/bash
gcc 13-2.c -o 13-2
./13-2
case $? in
    0) echo "Число равно нулю";;
    1) echo "Число больше нуля";;
    2) echo "Число меньше нуля";;
esac

```

Рис. 3.3: Программа 2

Программа 2

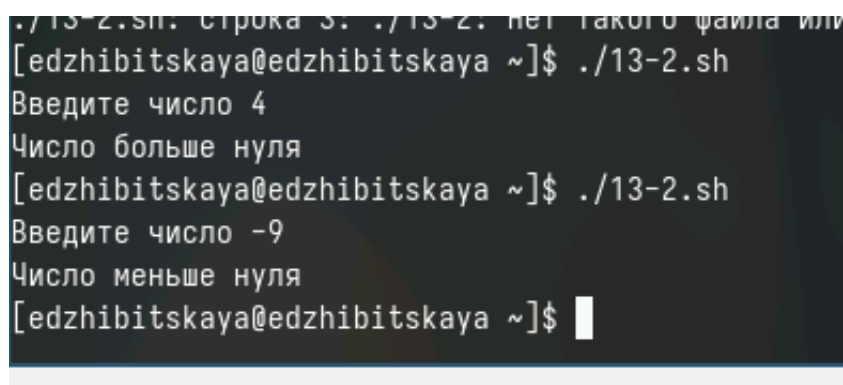
```

#!/bin/bash
gcc 13-2.c -o 13-2
./13-2
case $? in
    0) echo "Число равно нулю";;
    1) echo "Число больше нуля";;
    2) echo "Число меньше нуля";;

```


esac

Запустим файл и проверим работу программы(рис. 3.4).



```
./13-2.sh: строка 3: ./13-2: нет такого файла или каталога
[edzhibitskaya@edzhibitskaya ~]$ ./13-2.sh
Введите число 4
Число больше нуля
[edzhibitskaya@edzhibitskaya ~]$ ./13-2.sh
Введите число -9
Число меньше нуля
[edzhibitskaya@edzhibitskaya ~]$
```

Рис. 3.4: Запуск 2

Далее перейдем к пункту 1.

Также создадим файл, назовем его 13-1.sh и наделим его правами на исполнение. Далее создадим 2 текстовых файла(в один добавим текст, а второй нужен для записи) и перейдем к написанию программы(рис. 3.5).

```
#!/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 "Not found"
else
    if (($iflag==0))
    then echo "File not found"
    else
        if((oflag==0))
        then if (($Cflag==0))
            then if (($nflag==0))
                then grep $pval $inal
                else
                    grep -n $pval $ival
                fi
            else if (($nflag==0))
                then grep -i $pval $ival
                else grep -i -n $pval $ival
                fi
            fi
        else if (($Cflag==0))
            then if (($nflag==0))
```

Рис. 3.5: Программа 1

Программа 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 (($pflag==0))
then echo "Not found"
else
    if (($iflag==0))
    then echo "File not found"
    else
        if((oflag==0))
        then if (($Cflag==0))
            then if ((nflag==0))
                then grep $pval $inal
                else
                    grep -n $pval $ival
                fi
            else if (($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
fi
fi
fi
```

Запустим наш файл и посмотрим на вывод(рис. 3.6).

```
[edzhibitskaya@edzhibitskaya ~]$ touch f1.txt f2.
txt
[edzhibitskaya@edzhibitskaya ~]$ gedit f1.txt
[edzhibitskaya@edzhibitskaya ~]$ chmod +x 13-1.sh
[edzhibitskaya@edzhibitskaya ~]$ ./13-1.sh -i f1.
txt -o f2.txt -p корабль -C -n
[edzhibitskaya@edzhibitskaya ~]$ cat f2.txt
2:Вез корабль карамель, наскочил корабль на мель,
матросы две недели карамель на мели ели.
[edzhibitskaya@edzhibitskaya ~]$ gedit f1.txt
[edzhibitskaya@edzhibitskaya ~]$ ./13-1.sh -i f1.
txt -o f2.txt -p мама -C -n
[edzhibitskaya@edzhibitskaya ~]$ cat f2.txt
1:мама мыла раму
[edzhibitskaya@edzhibitskaya ~]$
```

Рис. 3.6: Запуск 1

Затем создадим файл для 3й программы. Наделим правами доступ и перейдем к написанию кода(рис. 3.7) и (рис. 3.8).

```
[edzhibitskaya@edzhibitskaya ~]$ touch 13-3.sh
[edzhibitskaya@edzhibitskaya ~]$ emacs
[edzhibitskaya@edzhibitskaya ~]$ chmod +x 13-3.sh
[edzhibitskaya@edzhibitskaya ~]$ ./13-3.sh 3
```

Рис. 3.7: Создание файла и наделение правами

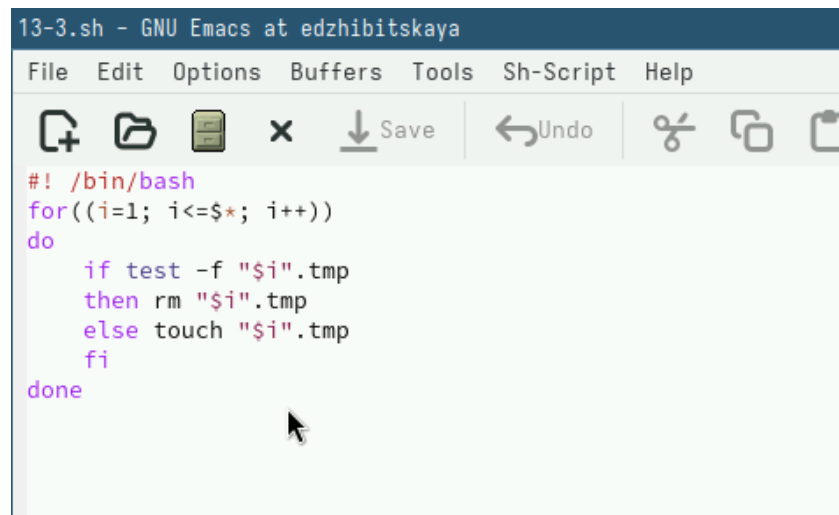


Рис. 3.8: Программа 3

Программа 3

```
#!/bin/bash
for((i=1; i<=*$*; i++))
do
    if test -f "$i".tmp
    then rm "$i".tmp
    else touch "$i".tmp
    fi
done
```

Запустим программу, убедимся, что все работает корректно - файлы создаются и удаляются(рис. 3.9).

```
edzhibitskaya@edzhibitskaya ~]$ ./13-3.sh 3
edzhibitskaya@edzhibitskaya ~]$ ls
13-1.sh  13-2.sh~  abc1      Downloads  '#lab06.sh#'
13-1.sh~ 13-3.sh  backup    f1.txt     '#lab07.sh#'
13-2     13-3.sh~ code.cpp  f2.txt     lab07.sh
13-2.c   1.tmp    config    feathers    LICENSE
13-2.c~  2.tmp    conf.txt  file.txt    may
13-2.sh  3.tmp    Documents git-extended monthly
edzhibitskaya@edzhibitskaya ~]$ ./13-3.sh 3
edzhibitskaya@edzhibitskaya ~]$ ls
13-1.sh  13-2.sh~  config    feathers    LICENSE
13-1.sh~ 13-3.sh  conf.txt  file.txt    may
13-2     13-3.sh~ Documents git-extended monthly
13-2.c   abc1      Downloads '#lab06.sh#' my_os
13-2.c~  backup    f1.txt    '#lab07.sh#' newdir
13-2.sh  code.cpp  f2.txt    lab07.sh    noname.gv
edzhibitskaya@edzhibitskaya ~]$
```

Рис. 3.9: Запуск 3

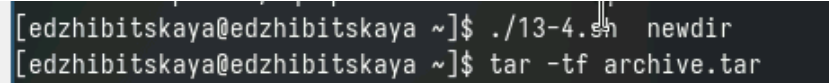
Наконец перейдем к 4 пункту. Выполним те же действия, создав и написав программу для исполняемого файла(рис. 3.10) и (рис. 3.11).

```
#!/bin/bash
find $* -mtime -7 -type f > FILES.txt
tar -cf archive.tar -T FILES.txt
```

Рис. 3.10: Программа 4

Программа 4

```
#!/bin/bash  
find $* -mtime -7 -type f > FILES.txt  
tar -cf archive.tar -T FILES.txt
```

A terminal window screenshot showing two lines of command execution. The first line shows a user running a script named '13-4.sh' in the current directory, which creates a new directory. The second line shows the user running 'tar -tf archive.tar' to list the contents of the archive.

```
[edzhibitskaya@edzhibitskaya ~]$ ./13-4.sh newdir  
[edzhibitskaya@edzhibitskaya ~]$ tar -tf archive.tar
```

Рис. 3.11: Запуск 4

В результате запуска создается архив, с данными из указанного каталога, причем с изменениями до 7 дней.

4 Контрольные вопросы

1. Команда `getopts` используется для обработки аргументов командной строки в скриптах Shell, чтобы сделать их более гибкими и управляемыми.
2. Метасимволы используются для шаблонного поиска и обработки файлов в командной оболочке UNIX. Они позволяют задавать шаблоны для поиска файлов с определенными именами или расширениями.
3. Операторы управления действиями включают в себя условные операторы (`if-then-else`), операторы цикла (`for`, `while`, `until`), операторы выбора (`case`).
4. Для прерывания цикла используются операторы `break` и `continue`.
5. Команда `false` возвращает ложное значение (код ошибки), а команда `true` - истинное значение (код успеха). Они могут быть использованы для управления потоком выполнения в скриптах.
6. Строка `if test -f mans/i.$s` проверяет, существует ли файл с именем, указанным в переменных `$s` и `$i`. Опция `-f` утилиты `test` указывает на проверку существования обычного файла.
7. Конструкция `while` выполняет цикл до тех пор, пока условие истинно, а конструкция `until` выполняет цикл до тех пор, пока условие ложно. Разница заключается в том, что в `while` условие проверяется перед выполнением тела цикла, а в `until` - после.

5 Выводы

В ходе работы мы изучили основные программирования в оболочке UNIX, написали несколько более сложных командных файлов.