

# Отчет по лабораторной работе № 11

---

Ничипорова Елена Дмитриевна

19-05-22

РУДН, Москва

## Отчет

---

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

- Используя команды `getopts` `grep`, написала командный файл, который анализирует командную строку с ключами: `i`—прочитать данные из указанного файла; `o`—вывести данные в указанный файл; `r`—шаблон—указать шаблон для поиска; `C`—различать большие и малые буквы; `n`—выдавать номера строк. Для данной задачи создала файл `prog1.sh` (рис. 1) и написала следующие алгоритмы (рис. 2)

```
ednichiporova@dk6n64 ~ $ touch prog1.sh
ednichiporova@dk6n64 ~ $ emacs &
```

Figure 1: создание файла

```
#!/bin/bash
if [ $# -eq 0 ]; then
    echo "Usage: prog1.sh [-i file] [-o file] [-r pattern] [-C] [-n]"
    exit 1
fi

while getopts "i:o:r:C:n" opt; do
    case $opt in
        i) file="$OPTARG"; shift;;
        o) outfile="$OPTARG"; shift;;
        r) pattern="$OPTARG"; shift;;
        C) case="case"; shift;;
        n) lines="lines"; shift;;
        *) echo "Illegal option $OPTARG"
           exit 1;;
    esac
done

if [ $# -eq 0 ]; then
    echo "Usage: prog1.sh [-i file] [-o file] [-r pattern] [-C] [-n]"
    exit 1
fi

if [ $# -eq 1 ]; then
    file="$1"
    shift
fi

if [ $# -eq 2 ]; then
    outfile="$1"
    shift
fi

if [ $# -eq 3 ]; then
    pattern="$1"
    shift
fi

if [ $# -eq 4 ]; then
    case="$1"
    shift
fi

if [ $# -eq 5 ]; then
    lines="$1"
    shift
fi

if [ $# -eq 6 ]; then
    echo "Too many arguments"
    exit 1
fi

if [ $# -eq 0 ]; then
    echo "Usage: prog1.sh [-i file] [-o file] [-r pattern] [-C] [-n]"
    exit 1
fi

if [ $# -eq 1 ]; then
    file="$1"
    shift
fi

if [ $# -eq 2 ]; then
    outfile="$1"
    shift
fi

if [ $# -eq 3 ]; then
    pattern="$1"
    shift
fi

if [ $# -eq 4 ]; then
    case="$1"
    shift
fi

if [ $# -eq 5 ]; then
    lines="$1"
    shift
fi

if [ $# -eq 6 ]; then
    echo "Too many arguments"
    exit 1
fi
```

- проверила работу написанного скрипта, используя различные опции, предварительно добавив права на выполнение(рис. 3) и создав два файла, которые необходимы для выполнения программы. Скрипт работает корректно(рис. 4)

```
ednichiporova@dk6n64 ~ $ touch a1.txt a2.txt
ednichiporova@dk6n64 ~ $ chmod +x prog1.sh
```

Figure 3: предоставление прав доступа

```
ednichiporova@dk6n64 ~ $ cat a1.txt
water abc
abc
prog1
water water
ednichiporova@dk6n64 ~ $ ./prog1.sh -i a1.txt -o a2.txt -p water -m
ednichiporova@dk6n64 ~ $ cat a2.sh
cat: a2.sh: No such file or directory
ednichiporova@dk6n64 ~ $ cat a2.txt
1:water abc
4:water water
ednichiporova@dk6n64 ~ $ ./prog1.sh -i a1.txt -o a2.txt -p water -m
ednichiporova@dk6n64 ~ $ cat a2.txt
1:water abc
4:water water
ednichiporova@dk6n64 ~ $ ./prog1.sh -i a1.txt -C -m
ednichiporova@dk6n64 ~ $ ./prog1.sh -i a2.txt -p water -C -m
1:1:water abc
```

Figure 4: проверка работы программы

Написала 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. Для данной задачи я создала 2 файла (рис. 5) и написала соответствующие скрипты (рис. 6) (рис. 7)

```
ednichiporova@dk6n64 ~ $ touch chislo.c  
ednichiporova@dk6n64 ~ $ touch chislo.sh
```

Figure 5: создание файлов

```
#include <stdio.h>  
#include <stdlib.h>  
int main()  
{  
    printf("введите число\n");  
    int a;  
    scanf("%d", &a);  
    if (a<0) exit(0);  
    if (a>0) exit(1);  
    if (a==0) exit(2);  
    return 0;  
}
```

- Проверила работу написанных скриптов. Скрипты работают корректно(рис. 8)

```
edilichiporova@kdm4 ~$ chmod +x chislo.sh
edilichiporova@kdm4 ~$ ./chislo.sh
5
введите число
5
число больше нуля
edilichiporova@kdm4 ~$ ./chislo.sh
введите число
0
число равно нулю
edilichiporova@kdm4 ~$ ./chislo.sh
введите число
-2
число меньше нуля
edilichiporova@kdm4 ~$
```

Figure 8: Проверка скрипта №2

Написала командный файл,создающий указанное число файлов,пронумерованных последовательно от 1 до [?] (например1.tmp,2.tmp,3.tmp,4.tmpит.д.).Число файлов,которые необходимо создать,передается в аргументы командной строки.Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).Для данной задачи я создала новый файл (рис. 9) и написала соответствующий скрипт(рис. 10)

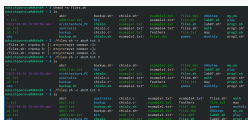
```
ednichiporova@dk6n64 ~ $ touch files.sh
ednichiporova@dk6n64 ~ $ emacs
```

Figure 9: создание файла

```
#!/bin/bash
opt=$1;
format=$2;
number=$3;
function Files()
{
    for (( i=1; i<=number; i++)) do
        file=$(echo $format | tr 'a' 'i')
        if [ $opt == "-r" ]
        then
            rm -f $file
        elif [ $opt == "-c" ]
        then
            touch $file
        fi
    done
}
Files
```



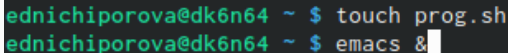
- Проверила работу написанного скрипта, предварительно добавив право на исполнение(рис. 11)



```
test1.sh: Permission denied
test2.sh: Permission denied
test3.sh: Permission denied
test4.sh: Permission denied
test5.sh: Permission denied
test6.sh: Permission denied
test7.sh: Permission denied
test8.sh: Permission denied
test9.sh: Permission denied
test10.sh: Permission denied
test11.sh: Permission denied
test12.sh: Permission denied
test13.sh: Permission denied
test14.sh: Permission denied
test15.sh: Permission denied
test16.sh: Permission denied
test17.sh: Permission denied
test18.sh: Permission denied
test19.sh: Permission denied
test20.sh: Permission denied
test21.sh: Permission denied
test22.sh: Permission denied
test23.sh: Permission denied
test24.sh: Permission denied
test25.sh: Permission denied
test26.sh: Permission denied
test27.sh: Permission denied
test28.sh: Permission denied
test29.sh: Permission denied
test30.sh: Permission denied
test31.sh: Permission denied
test32.sh: Permission denied
test33.sh: Permission denied
test34.sh: Permission denied
test35.sh: Permission denied
test36.sh: Permission denied
test37.sh: Permission denied
test38.sh: Permission denied
test39.sh: Permission denied
test40.sh: Permission denied
test41.sh: Permission denied
test42.sh: Permission denied
test43.sh: Permission denied
test44.sh: Permission denied
test45.sh: Permission denied
test46.sh: Permission denied
test47.sh: Permission denied
test48.sh: Permission denied
test49.sh: Permission denied
test50.sh: Permission denied
test51.sh: Permission denied
test52.sh: Permission denied
test53.sh: Permission denied
test54.sh: Permission denied
test55.sh: Permission denied
test56.sh: Permission denied
test57.sh: Permission denied
test58.sh: Permission denied
test59.sh: Permission denied
test60.sh: Permission denied
test61.sh: Permission denied
test62.sh: Permission denied
test63.sh: Permission denied
test64.sh: Permission denied
test65.sh: Permission denied
test66.sh: Permission denied
test67.sh: Permission denied
test68.sh: Permission denied
test69.sh: Permission denied
test70.sh: Permission denied
test71.sh: Permission denied
test72.sh: Permission denied
test73.sh: Permission denied
test74.sh: Permission denied
test75.sh: Permission denied
test76.sh: Permission denied
test77.sh: Permission denied
test78.sh: Permission denied
test79.sh: Permission denied
test80.sh: Permission denied
test81.sh: Permission denied
test82.sh: Permission denied
test83.sh: Permission denied
test84.sh: Permission denied
test85.sh: Permission denied
test86.sh: Permission denied
test87.sh: Permission denied
test88.sh: Permission denied
test89.sh: Permission denied
test90.sh: Permission denied
test91.sh: Permission denied
test92.sh: Permission denied
test93.sh: Permission denied
test94.sh: Permission denied
test95.sh: Permission denied
test96.sh: Permission denied
test97.sh: Permission denied
test98.sh: Permission denied
test99.sh: Permission denied
test100.sh: Permission denied
```


Figure 11: Проверка скрипта №3

- Написала командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировала его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find). Для данной программы я создала новый файл (рис. 12) и написала соответствующий скрипт (рис. 13)

A terminal window with a dark background. The prompt is 'ednichiporova@dk6n64 ~ \$'. The first command entered is 'touch prog.sh' and the second is 'emacs &'.

```
ednichiporova@dk6n64 ~ $ touch prog.sh
ednichiporova@dk6n64 ~ $ emacs &
```

Figure 12: создание файлов

A code block with a light background showing a shell script. The script uses 'find' to locate files changed within the last 7 days and then iterates over them to perform an action (partially visible).

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

- Проверила работу скрипта, предварительно добавив право на исполнение и создала новый каталог с несколькими файлами (рис. 14). Скрипт работает корректно (рис. 15)

```
ednichiporova@dk6n64 ~ $ chmod +x prog.sh
ednichiporova@dk6n64 ~ $ mkdir catalog1
ednichiporova@dk6n64 ~ $ cd ~/catalog1
```

Figure 14: создание каталога и предоставление прав доступа

```
ednichiporova@dk6n64 ~/catalog1 $ ls -l
total 2
-rw-r--r-- 1 ednichiporova studsci 22 may 19 14:26 a1.txt
-rw-r--r-- 1 ednichiporova studsci 26 may 19 14:26 a2.txt
ednichiporova@dk6n64 ~/catalog1 $ ./prog.sh
a1.txt
a2.txt
ednichiporova@dk6n64 ~/catalog1 $ tar -tf catalog1.tar
a1.txt
a2.txt
```

Figure 15: проверка скрипта №4

- В ходе выполнения данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов