

Лабораторная работа №13

Операционные системы

Панявкина И.В.

10 мая 2025

Российский университет дружбы народов, Москва, Россия

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

1. Используя команды `grep`, написать командный файл, который анализирует командную строку с ключами:
 - `-i`inputfile — прочитать данные из указанного файла;
 - `-o`outputfile — 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 `-r`.

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. Команд- ный 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`).

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек:

оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций; C-оболочка (или csh) — надстройка на оболочке Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).

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. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке `bash`. В других оболочках большинство команд будет совпадать с описанными ниже.

Создаю файл с разрешением на исполнение (рис. 1).

```
[irinapanyavkina@irinapanyavkina ~]$ touch 111.sh  
[irinapanyavkina@irinapanyavkina ~]$ chmod +x 111.sh  
[irinapanyavkina@irinapanyavkina ~]$ bash 111.sh -p улит -i input.txt -o output.txt -c -n  
[irinapanyavkina@irinapanyavkina ~]$
```

Рис. 1: Создание файла

Командный файл, с командами `getopts` и `grep`, который анализирует командную строку с ключами: - `-iinputfile` — прочитать данные из указанного файла; - `-ooutputfile` — вывести данные в указанный файл; - `-rшаблон` — 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` (рис. 2).

Выполнение лабораторной работы

```
#!/bin/bash

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 ! test $cflag
then
    cf=-i
fi

if test $nflag
then
    nf=-n
fi

grep $cf $nf $pval $ival >> $oval
```

Результат работы программы в файле output.txt (рис. 3).

```
1:Заходит однажды в бар улитка и говорит:  
3:- Простите, но мы не обслуживаем улиток.  
5:Через неделю заходит опять эта улитка и спрашивает:
```

U:--- **output.txt** All L1 (Text)

```
Заходит однажды в бар улитка и говорит:  
-Можно виски с колой?  
- Простите, но мы не обслуживаем улиток.  
И бармен вышвырнул ее за дверь.  
..
```

Создаю исполняемый файл для второй программы, также создаю файл 12.c для программы на Си (рис. 4).

```
[irinapanyavkina@irinapanyavkina ~]$ touch 112.sh  
[irinapanyavkina@irinapanyavkina ~]$ chmod +x 112.sh  
[irinapanyavkina@irinapanyavkina ~]$ touch 12.cpp  
[irinapanyavkina@irinapanyavkina ~]$ bash 112.sh
```

Рис. 4: Создание файла

Пишу программу на языке Си, 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 (рис. 5).

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

int main () {
    int n;
    printf ("Введите число: "); scanf ("%d", &n);
    if(n>0){
        exit(1);
    }
    else if (n==0) {
        exit(0);
    }
    else {
        exit(2);
    }
}
```

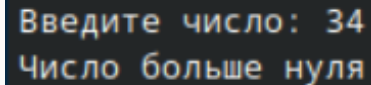
Рис. 5: Код программы на Си

Командный файл должен вызывать эту программу и, проанализировав с помощью команды \$?, выдать сообщение о том, какое число было введено (рис. 6).

```
#!/bin/bash

gcc -o cprog 12.c
./cprog
case $? in
0) echo "Число равно нулю";;
1) echo "Число больше нуля";;
2) echo "Число меньше нуля";;
esac
```

Программа работает корректно (рис. 7).

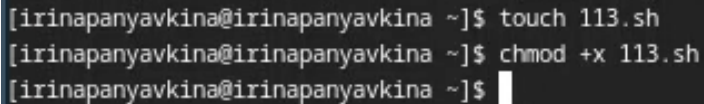


Введите число: 34
Число больше нуля

A screenshot of a terminal window with a dark background. The text is displayed in a light blue or cyan monospaced font. The first line shows the prompt 'Введите число:' followed by the user input '34'. The second line shows the program's output 'Число больше нуля'.

Рис. 7: Результат работы программы

Создаю исполняемый файл для третьей программы (рис. 8).

A terminal window with a dark background and light-colored text. It shows three lines of commands being executed in a shell. The first line creates a file named 113.sh using the 'touch' command. The second line sets execute permissions for the file using 'chmod +x 113.sh'. The third line shows the prompt with a cursor, indicating the command has finished.

```
[irinapanyavkina@irinapanyavkina ~]$ touch 113.sh  
[irinapanyavkina@irinapanyavkina ~]$ chmod +x 113.sh  
[irinapanyavkina@irinapanyavkina ~]$
```

Рис. 8: Создание файла

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) (рис. 9).

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

Проверяю, что программа создала файлы и удалила их при соответствующих запросах (рис. 10).

```
[irinapanyavkina@irinapanyavkina ~]$ bash 113.sh 4
[irinapanyavkina@irinapanyavkina ~]$ ls
111.sh  1.tmp  abc1      'd (1-я копия).md'  dotfiles-template  git-extended  lab07.sh  my_os
112.sh  2.tmp  australia d.md               feathers            gitflow       lab07.sh~  output.txt
113.sh  3.tmp  backup    dotfl              file.txt           iloveos       may        pandoc-3.1.11.1
12.cpp  4.tmp  bin       dotfiles           fun                input.txt     monthly    pandoc-3.1.11.1-linux
[irinapanyavkina@irinapanyavkina ~]$
```

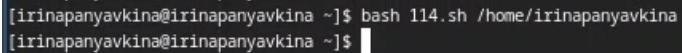
Рис. 10: Результат работы программы

Создаю исполняемый файл для четвертой программы. Это 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`) (рис. 11).


```
#!/bin/bash  
find $* -mtime -7 -mtime +0 -type f > FILES.txt  
tar -cf archive.tar -T FILES.txt
```

Рис. 11: Код программы

Проверяю работу программы (рис. 12).

A terminal window with a dark background. The first line shows the prompt [irinapanyavkina@irinapanyavkina ~]\$ followed by the command bash 114.sh /home/irinapanyavkina. The second line shows the prompt [irinapanyavkina@irinapanyavkina ~]\$ followed by a white cursor block.

```
[irinapanyavkina@irinapanyavkina ~]$ bash 114.sh /home/irinapanyavkina  
[irinapanyavkina@irinapanyavkina ~]$
```

Рис. 12: Результат работы программы

Выводы

При выполнении данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX, научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Список литературы

1. Лабораторная работа №13 [Электронный ресурс]

URL:<https://esystem.rudn.ru/mod/resource/view.php?id=1224393>