

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Никита Иванов НБИбд-01-20¹

27 мая, 2021, Москва, Россия

¹Российский Университет Дружбы Народов

Цели и задачи работы

Цель лабораторной работы

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

Задачи лабораторной работы

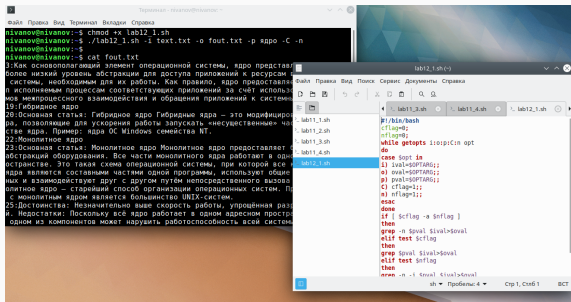
1 Выполнить 4 задания

Процесс выполнения лабораторной работы

1. Используя команды `getopts` `grep` напишем командный файл, который анализирует командную строку с ключами и выполним его: `-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

Выполнение работы



The image shows a terminal window on the left and a file editor on the right. The terminal window has a title bar 'Терминал - root@kali:~/workshop' and shows the following commands and output:

```
ivanov@ivanov:~$ chmod +x lab12_1.sh
ivanov@ivanov:~$ ./lab12_1.sh -i text.txt -o fout.txt -p ядро -C -n
ivanov@ivanov:~$
ivanov@ivanov:~$ cat fout.txt
1:Как основополагающий элемент операционной системы, ядро предоставляет более низкий уровень абстракции для доступа приложения к ресурсам системы, необходимым для их работы. Как правило, ядро предоставляет исполняемым процессам соответствующих приложений за счёт использования межпроцессного взаимодействия и обращения приложений к системным вызовам.
19:Гибридное ядро
20:Основная статья: Гибридное ядро Гибридные ядра – это модифицированные ядра, позволяющие для ускорения работы запускать «несущественные» части ядра. Пример: ядра ОС Windows семейства NT.
22:Монолитное ядро
23:Основная статья: Монолитное ядро Монолитное ядро предоставляет абстракцию оборудования. Все части монолитного ядра работают в одном пространстве. Это такая схема операционной системы, при которой все ядра являются составными частями одной программы, используют общие данные и взаимодействуют друг с другом путём непосредственного вызова системных вызовов. Монолитное ядро – старейший способ организации операционных систем. По сравнению с монолитным ядром является большинство UNIX-систем.
25:Достоинства: Незначительно выше скорость работы, упрощённая развёртка. Недостатки: Поскольку всё ядро работает в одном адресном пространстве, один из компонентов может нарушить работоспособность всей системы.
```

The file editor window has a title bar 'lab12_1.sh (-)' and shows the following code:

```
#!/bin/bash
cflag=0
nflag=0
while getopts itoipcin opt
do
case $opt in
i) sval=$OPTARG;;
o) sval=$OPTARG;;
p) sval=$OPTARG;;
c) cflag=1;;
n) nflag=1;;
esac
done
if [ $cflag -a $nflag ]
then
grep -n $sval $sval=$sval
elif test $cflag
then
grep $sval $sval=$sval
elif test $nflag
then
nrun -n -i $sval $sval=$sval
fi
done
```

Figure 1: Задание 1

2. Напишем сначала на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем завершим программу при помощи функции `exit(n)`, передавая информацию о коде завершения в оболочку. Командный файл вызовет эту программу и, проанализировав с помощью команды `$?`, выдаст сообщение о том, какое число было введено

Выполнение работы

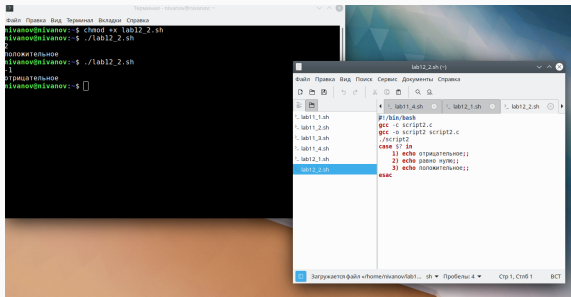


Figure 2: Задание 2

3. Напишем командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N

Выполнение работы

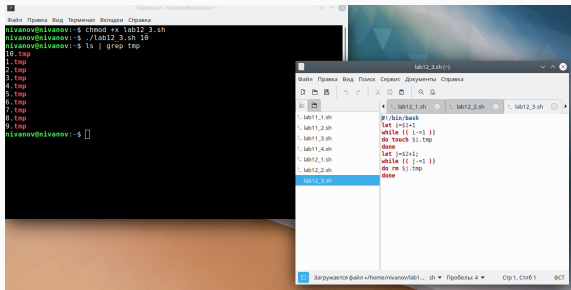


Figure 3: Задание 3

4. Напишем командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории. Модифицируем его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад.

Выполнение работы

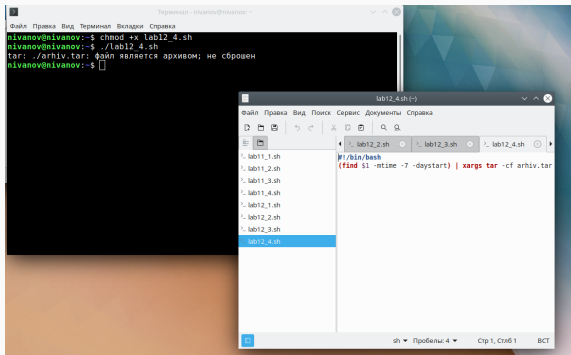


Figure 4: Задание 4

Выводы по проделанной работе

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