

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Ростислав Хачикович Арзуманян НБИбд-01-20¹

26 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

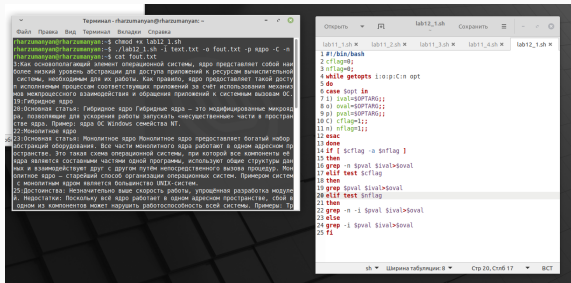
1 Выполнить 4 задания

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 two terminal windows and a text document. The left terminal window displays a shell script for creating a text file and a document explaining kernel types. The right terminal window shows a shell script for file operations. The text document in the background describes different types of operating system kernels.

```
Терминал -flagzlatyuu@flagzlatyuu: ~  
Файл Плавка Вид Терминал Вкладки Справка  
flagzlatyuu@flagzlatyuu:~$ smod xx lab12_1.sh  
flagzlatyuu@flagzlatyuu:~$ ./lab12_1.sh -l text.txt -o fout.txt -p karo -c -n  
flagzlatyuu@flagzlatyuu:~$ cat fout.txt  
3. Как основополагающий элемент операционной системы, ядро представляет собой наи-  
более низкий уровень абстракции для доступа приложений к ресурсам вычислительной  
системы, необходимым для их работы. Как правило, ядро предоставляет такой доступ  
в виде специальных процессов соответствующих приложений за счет использования механиз-  
мов межпроцессного взаимодействия и обращения приложений к системным вызовам ОС.  
19. Гибридное ядро  
20. Основная статья: Гибридное ядро Гибридное ядро – это модифицированные микроза-  
ра, позволяющие для ускорения работы запускать «несущественные» части в простран-  
стве ядра. Пример: ядро ОС Windows семейства NT.  
22. Монолитное ядро  
23. Основная статья: Монолитное ядро Монолитное ядро предоставляет богатый набор  
абстракций оборудования. Все части монолитного ядра работают в одном адресном про-  
странстве, это такая схема операционной системы, при которой все компоненты ядра  
являются составными частями одной программы, используют общие структуры дан-  
ных и взаимодействуют друг с другом путем непосредственного вызова процедур. Мо-  
нолитное ядро – старейший способ организации операционных систем. Примером систем  
с монолитным ядром является большинство UNIX-систем.  
25. Достоинства: Немного быстрее скорость работы, упрощенная разработка модулей  
я. Недостатки: Поскольку все ядро работает в одном адресном пространстве, сбой в  
одном из компонентов может нарушить работоспособность всей системы. Примеры: Tr
```

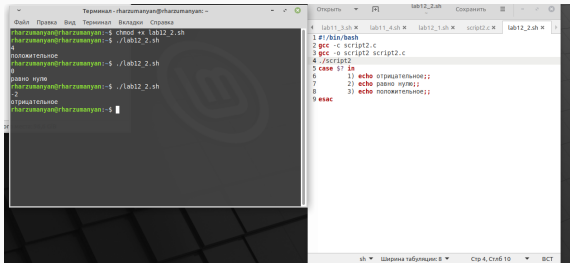
```
lab12_1.sh  
lab12_1.sh x  
1 #!/bin/bash  
2 cflag=0  
3 nflag=0  
4 while getopts i:o:p:c:n opt  
5 do  
6 case $opt in  
7 i) sval=$OPTARG;;  
8 o) oval=$OPTARG;;  
9 p) pval=$OPTARG;;  
10 c) cflag=1;;  
11 n) nflag=1;;  
12 esac  
13 done  
14 if [ $cflag -a $nflag ]  
15 then  
16 grep -n $pval $sval>$oval  
17 elif test $cflag  
18 then  
19 grep $oval $sval>$oval  
20 elif test $nflag  
21 then  
22 grep -n -i $pval $sval>$oval  
23 else  
24 grep -l $pval $sval>$oval  
25 fi
```

Стр 20, Стрб 17 ВСТ

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



The image shows a terminal window on the left and a script editor on the right. The terminal window displays the execution of a script named `lab12_2.sh`. The user runs `chmod +x lab12_2.sh` and then `./lab12_2.sh`. The script prompts for a number and outputs the result: "положительное" (positive), "равно нулю" (equal to zero), or "отрицательное" (negative). The script editor on the right shows the source code of `lab12_2.sh`, which is a C program that reads an integer and uses a `case` statement to print the appropriate message.

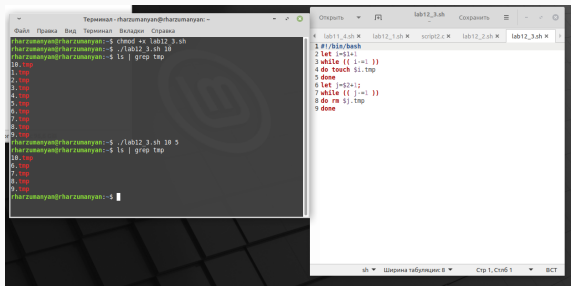
```
Терминал - rharzumanyan@rharzumanyan: ~  
Файл Правка Вид Терминал Выход Остановка  
rharzumanyan@rharzumanyan:~$ chmod +x lab12_2.sh  
rharzumanyan@rharzumanyan:~$ ./lab12_2.sh  
4  
положительное  
rharzumanyan@rharzumanyan:~$ ./lab12_2.sh  
0  
равно нулю  
rharzumanyan@rharzumanyan:~$ ./lab12_2.sh  
-2  
отрицательное  
rharzumanyan@rharzumanyan:~$
```

```
lab12_2.sh  
1 #!/bin/bash  
2 gcc -c script2.c  
3 gcc -o script2 script2.c  
4 ./script2  
5 case $? in  
6   1) echo отрицательное;;  
7   2) echo равно нулю;;  
8   3) echo положительное;;  
9 esac
```

Figure 2: Задание 2

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

Выполнение работы



The image shows a terminal window and a file editor window. The terminal window displays the execution of a shell script named lab12_3.sh, which creates a file named \$1.tmp and prints the numbers 1 through 9. The file editor window shows the content of the script, which is a shell script that creates a file named \$1.tmp and prints the numbers 1 through 9.

```
Terminal: rharzumanyan@rharzumanyan:~  
rharzumanyan@rharzumanyan:~$ cd / && lab12_3.sh  
rharzumanyan@rharzumanyan:~$ ./lab12_3.sh 10  
rharzumanyan@rharzumanyan:~$ ls | grep tmp  
1.tmp  
2.tmp  
3.tmp  
4.tmp  
5.tmp  
6.tmp  
7.tmp  
8.tmp  
9.tmp  
rharzumanyan@rharzumanyan:~$ ./lab12_3.sh 10 5  
rharzumanyan@rharzumanyan:~$ ls | grep tmp  
10.tmp  
6.tmp  
7.tmp  
8.tmp  
9.tmp  
rharzumanyan@rharzumanyan:~$
```

```
lab12_3.sh  
1 #!/bin/bash  
2 let i=1  
3 while (( i <= 10 ))  
4 do touch $i.tmp  
5 done  
6 let i+=1  
7 while (( i <= 10 ))  
8 do rm $i.tmp  
9 done
```

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

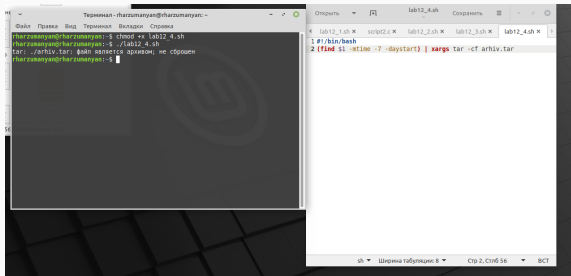


Figure 4: Задание 4

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.