

Лабораторная работа №13

Операционные системы

Тойчубекова Асель Нурлановна

04 мая 2024

Российский университет дружбы народов, Москва, Россия

Информация

- Тойчубекова Асель Нурлановна
- Студент НПИбд-02-23
- факультет физико-математических и естественных наук
- Российский университет дружбы народов
- 1032235033@rudn.ru
- <https://aseltoichubekova.github.io/ru/>

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

Задание

1. Используя команды `grep`, написать командный файл, 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 `-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. 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 заданного числа (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`).

Теоретическое введение

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 оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд;

- оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;
- BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).

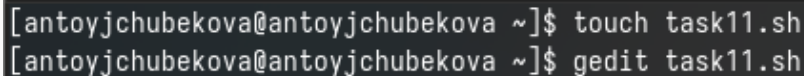
POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.

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.

Выполнение лабораторной работы

Для начало я создаю первый файл, в котором буду писать программу и открою его в редакторе gedit.

A terminal window with a dark background and light gray text. It shows two commands being executed in a shell. The first command is 'touch task11.sh' and the second is 'gedit task11.sh'. Both commands are preceded by the prompt '[antoyjchubekova@antoyjchubekova ~]\$'.

```
[antoyjchubekova@antoyjchubekova ~]$ touch task11.sh  
[antoyjchubekova@antoyjchubekova ~]$ gedit task11.sh
```

Рис. 1: Создание файла

Редактирую файл, пишу командный файл, который используя команды `getopts` `grep` анализирует командную строку с некоторыми ключами. Используя цикл `while` программа анализирует все флаги, записывая данные в нужные файлы. В конце проверяются использования опций `s` и `n` и присваиваются определенным переменным. Осуществляется команда `grep`, которая в данном случае берет и записывает в новый файл текст который совпал с шаблоном.

Выполнение лабораторной работы

```
1 #!/bin/bash
2 while getopts i:o:p:cn optletter
3 do
4 case $optletter in
5 i) i_flag=1; i_file=$OPTARG;;
6 o) o_flag=1; o_file=$OPTARG;;
7 p) p_flag=1; p_file=$OPTARG;;
8 c) c_flag=1;;
9 n) n_flag=1;;
10 *) echo Нет такой опции как $optletter;;
11 esac
12 done
13 if ! test $c_flag
14 then
15 cf=-i
16 fi
17 if test $n_flag
18 then
19 nf=-n
20 fi
21 grep $cf $nf $p_file $i_file >> $o_file
22 |
```


Даю права на выполнение и запускаю программу.

```
[antoychubekova@antoychubekova ~]$ chmod +x task11.sh  
[antoychubekova@antoychubekova ~]$ bash task11.sh -p Черный -i output.txt -o input.txt -c -n
```

Рис. 3: Запуск программы

Выполнение лабораторной работы

Мы видим, что программа удачно сработала и строки с шаблоном “Черный” из текста в файле output.txt записался в файл input.txt.

*input.txt (~) - gedit

Открыть ▾ + *input.txt ~ Сохранить ≡ ×

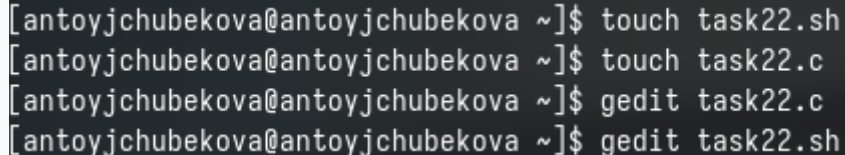
1 12:Черный человек,
2 13:Черный, черный,
3 14:Черный человек
4 16:Черный человек
5 18:Черный человек
6 25:Черный человек
7 26:Черный, черный...S

output.txt (~) - gedit

Открыть ▾ + output.... ~ Сохранить ≡ ×

1 Друг мой, друг мой,
2 Я очень и очень болен.
3 Сам не знаю, откуда взялась эта боль.
4 То ли ветер свистит
5 Над пустым и безлюдным полем,
6 То ль, как рошу в сентябрь,
7 Осыпает мозги алкоголь.
8 Голова моя машет ушами,
9 Как крыльями птица.
10 Ей на шее ноги
11 Маячить больше невмочь.
12 Черный человек,
13 Черный, черный,
14 Черный человек
15 На кровать ко мне садится,
16 Черный человек
17 Спать не дает мне всю ночь.
18 Черный человек
19 Водит пальцем по мерзкой книге
20 И, гнусая надо мной,
21 ...

Создаю файлы для второй программы и открываю их в редакторе.

A terminal window with a dark background and light-colored text. It shows four lines of commands being executed in a shell. The prompt is [antoyjchubekova@antoyjchubekova ~]\$ for each line. The commands are: touch task22.sh, touch task22.c, gedit task22.c, and gedit task22.sh.

```
[antoyjchubekova@antoyjchubekova ~]$ touch task22.sh  
[antoyjchubekova@antoyjchubekova ~]$ touch task22.c  
[antoyjchubekova@antoyjchubekova ~]$ gedit task22.c  
[antoyjchubekova@antoyjchubekova ~]$ gedit task22.sh
```

Рис. 5: Создание файла

В файле с расширением .c пишу программу на си, которая вводит число и определяет, является ли оно больше нуля, меньше нуля, или равно нулю. Затем завершающаяся с помощью функции `exit(n)`, передавая информацию о коде завершения в оболочку. Для этого с помощью `if` проверяю числа и вывожу соответствующий `exit`.

Выполнение лабораторной работы

```
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int main (){
5     int n;
6     printf ("Введите число:");
7     scanf ("%d",&n);
8     if(n>0){
9         exit(1);
10    }
11    else if (n==0){
12        exit(0);
13    }
14    else {
15        exit(2);
16    }
17 }
```

Открываю файл с расширением .sh и создаю командный файл, который анализирует программу на си с помощью команды \$? и выдает сообщения о том, какое число было введено. Для этого я использую команду gcc -o чтобы программа на си закомпилировалась в новый файл, затем с помощью case проверяю какие значения в этом файле и вывожу соответствующее сообщение.

Выполнение лабораторной работы

```
1 #!/bin/bash
2 gcc -o newtask task22.c
3 ./newtask
4 case $? in
5 0) echo "Число=0";;
6 1) echo "Число >0";;
7 2) echo "Число <0";;
8 esac
9
```

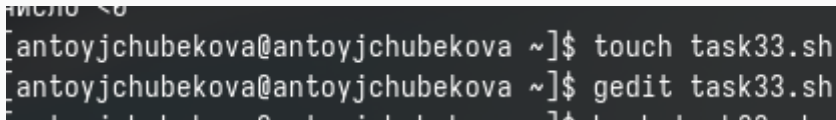
Рис. 7: Редактирование файла

Выполнение лабораторной работы

Даю право на выполнение и запускаю программу с числами 4,0,-8, мы видим, что все работает корректно.

```
[antoyjchubekova@antoyjchubekova ~]$ chmod +x task22.sh
[antoyjchubekova@antoyjchubekova ~]$ bash task22.sh
Введите число:4
Число >0
[antoyjchubekova@antoyjchubekova ~]$ bash task22.sh
Введите число:0
Число=0
[antoyjchubekova@antoyjchubekova ~]$ bash task22.sh
Введите число:-8
Число <0
[antoyjchubekova@antoyjchubekova ~]$
```


Создаю третий файл для написания командного файла и открываю его в редакторе.

A terminal window with a dark background and light-colored text. The prompt is 'antoyjchubekova@antoyjchubekova ~]'. The first command entered is 'touch task33.sh' and the second is 'gedit task33.sh'.

```
antoyjchubekova@antoyjchubekova ~]$ touch task33.sh  
antoyjchubekova@antoyjchubekova ~]$ gedit task33.sh
```

Рис. 9: Создание файла

Создаю командный файл, который создает указанное количество файлов, пронумерованные последовательно от 1 до n , число файлов же передается в аргументы командной строки. Также этот же командный файл должен удалять файлы с похожими именами, если они есть до создания новых. Для этого я запрашиваю у пользователя количество файлов, которые необходимо создать и прохожусь от 1 до этого числа параллельно проверяя есть ли уже файлы с такими именами если да, то удаляю их, а если нет создаю их.

Выполнение лабораторной работы

```
1 #!/bin/bash  
2 echo "Введите число файлов"  
3 read n  
4 for((i=1;i<=$n;i++))  
5 do  
6 if test -f "$i".tmp  
7 then  
8 rm "$i".tmp  
9 else touch "$i".tmp  
10 fi  
11 done
```

Рис. 10: Редактирование файла

Даю право на выполнение и запускаю программу с числом 4.

```
[antoyjchubekova@antoyjchubekova ~]$ chmod +x task33.sh
[antoyjchubekova@antoyjchubekova ~]$ bash task33.sh
Введите число файлов
4
[antoyjchubekova@antoyjchubekova ~]$
```

Рис. 11: Запуск программы

Зайдя в свой домашний каталог, можно видеть, что файлы успешно созданы.

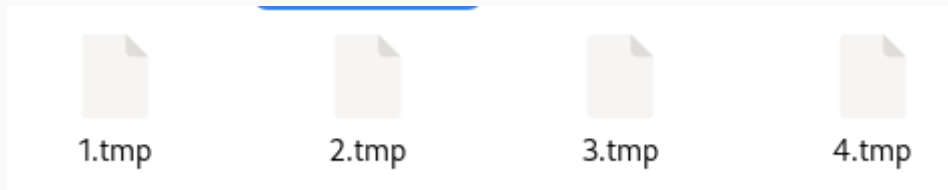
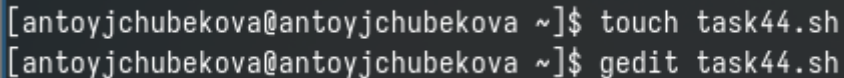


Рис. 12: Новые файлы

Создаю четвертый файл для написания командного файла и открываю его в редакторе.

A terminal window with a dark background and light-colored text. It shows two commands being executed in a shell. The first command is 'touch task44.sh' and the second is 'gedit task44.sh'. Both commands are preceded by a prompt indicating the user is 'antoyjchubekova' at a machine named 'antoyjchubekova' in the home directory (~).

```
[antoyjchubekova@antoyjchubekova ~]$ touch task44.sh  
[antoyjchubekova@antoyjchubekova ~]$ gedit task44.sh
```

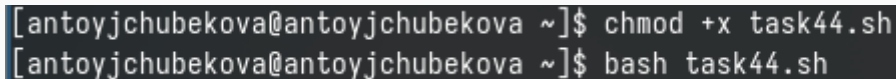
Рис. 13: Создание файла

Редактирую файл, пишу командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории, также модифицирует его так, чтобы запаковывались только те файлы, которые были изменены менее недели назад . Для этого я использую команду `find` с опцией `-mtime -7`(чтобы указать, что менее 7 дней) и `-mtime +0`(чтобы программа не брала в учет сегодняшние файлы), а также `type -f`(для архивации исключительно файлов) вывод этих команд записываю в новый файл `2xfile.txt`. Далее архивирую все файлы с помощью команды `tar -cf` и записываю вывод в файл `archive2x.tar`.

```
1 #!/bin/bash
2 find $* -mtime -7 -mtime +0 -type f > 2xfile.txt
3 tar -cf archive2x.tar -T 2xfile.txt
```

Рис. 14: Редактирование файла

Даю право на выполнение и запускаю программу.

A screenshot of a terminal window with a dark background and light-colored text. It shows two lines of commands being entered at a prompt. The first line is 'chmod +x task44.sh' and the second line is 'bash task44.sh'. The prompt is '[antoyjchubekova@antoyjchubekova ~]\$'.

```
[antoyjchubekova@antoyjchubekova ~]$ chmod +x task44.sh  
[antoyjchubekova@antoyjchubekova ~]$ bash task44.sh
```

Рис. 15: Запуск программы

Перейдя в домашний каталог видим, что программа сработала корректно.

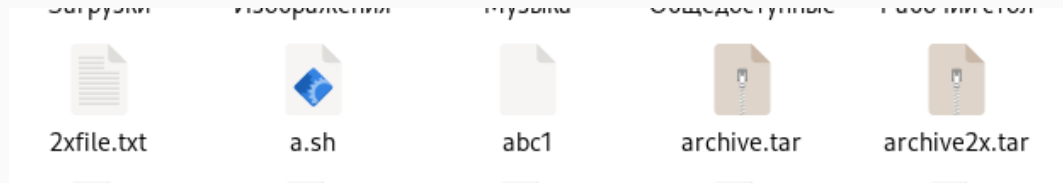


Рис. 16: Архив файлов

Выводы

В ходе выполнения лабораторной работы № 13 я изучила основы программирования в оболочке ОС UNIX. Также научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов. Создала четыре командных файла и проверила их работу.