

Презентация по лабораторной работе №11

Дисциплина: Операционные системы

Шошина Е.А.

17 апреля 2023

Российский университет дружбы народов, Москва, Россия

Информация

- Шошина Евгения Александровна
- группа: НКАбд-03-22
- студент факультета физико-математических и естественных наук
- Российский университет дружбы народов
- 1132229532@pfur.ru
- <https://EAShoshina.github.io/ru/>



Вводная часть

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

Задание

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

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 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

Выполнение лабораторной работы

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами:
 - `-iinputfile` — прочитать данные из указанного файла;
 - `-ooutputfile` — вывести данные в указанный файл;
 - `-ршаблон` — 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. 1.11, 2.11, 3.11, 4.11)

Первая программа



```
1 #!/bin/bash
2 while getopts i:o:p:cn optletter
3 do
4 case $optletter in
5     i) iflag=1; ival=$OPTARG;;
6     o) oflag=1; oval=$OPTARG;;
7     p) pflag=1; pval=$OPTARG;;
8     c) cflag=1;;
9     n) nflag=1;;
10    *) echo Illegal option $optletter;;
11    esac
12 done
13 if ! test $cflag
14 then
15     cf=-i
16 fi
17 if test $nflag
18 then
19     nf=-n
20 fi
21 grep $cf $nf $pval $ival >> $oval
```

{ #fig:001 width=70% height=70%}

```
[eashoshina@fedora ~]$ gedit prog1.sh  
[eashoshina@fedora ~]$ bash prog1.sh -p B -i lab11_1.txt -o output.txt -c -n { #fig:002  
width=70% height=70%}
```

lab11_1.txt



Вы помните,
Вы всё, конечно, помните,
Как я стоял,
Приблизившись к стене,
Взволнованно ходили вы по комнате
И что-то резкое
В лицо бросали мне.

Вы говорили:
Нам пора расстаться,
Что вас измучила
Моя шальная жизнь,
Что вам пора за дело приниматься,
А мой удел —
Катиться дальше, вниз.

Любимая!
Меня вы не любили.
Не знали вы, что в соннике людском
Я был как лошадь, загнанная в мыле,
Припорошенная смелым ездоком.

Не знали вы,
Что я в сплошном дыму,
В развороченном бурей быте
С того и мучаюсь, что не пойму —
Куда несет нас рок событий.

Лицом к лицу
Лица не увидеть.
Большое видится на расстоянье.
Когда кипит морская гладь —
Корабль в плачевном состоянье.

output.txt

```
1:Вы помните,  
2:Вы всё, конечно, помните,  
5:Взволнованно ходили вы по комнате  
7:В лицо бросали мне.  
9:Вы говорили:  
25:В развороченном бурей быте  
38:В прямую гущу бурь и вьюг  
57:В угаре пьяном.  
62:В глазах усталых:  
68:В развороченном бурей быте  
80:В ударе нежных чувств.  
113:Вас помнящий всегда
```

{ #fig:004 width=70%

height=70%}

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.

(рис. 5.11. 6.11. 7.11)

A screenshot of a code editor window titled '*lab11_2.c'. The window has a toolbar with 'Открыть' (Open), a plus icon, 'Сохранить' (Save), a hamburger menu icon, and a close 'x' icon. The code is written in C and is as follows:

```
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 }
```

{ #fig:005 width=70% height=70%}

Вторая программа



```
1 #! /bin/bash
2
3 gcc -o cprog lab11_2.c
4 ./cprog
5 case $? in
6     0) echo "Число равно нулю";;
7     1) echo "Число больше нуля";;
8     2) echo "Число меньше нуля";;
9 esac
```

{ #fig:006 width=70% height=70%}

```
[eashoshina@fedora ~]$ bash prog2.sh 15
```

```
Введите число: 5
```

```
Число больше нуля
```

```
{ #fig:007 width=70% height=70%}
```

3. Написать 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. 8.11, 9.11)

Третья программа



```
1 #!/bin/bash
2 |
3 for((i=1; i<=*$; i++))
4 do
5 if test -f "$i".tmp
6 then rm "$i".tmp
7 else touch "$i.tmp"
8 fi
9 done
```

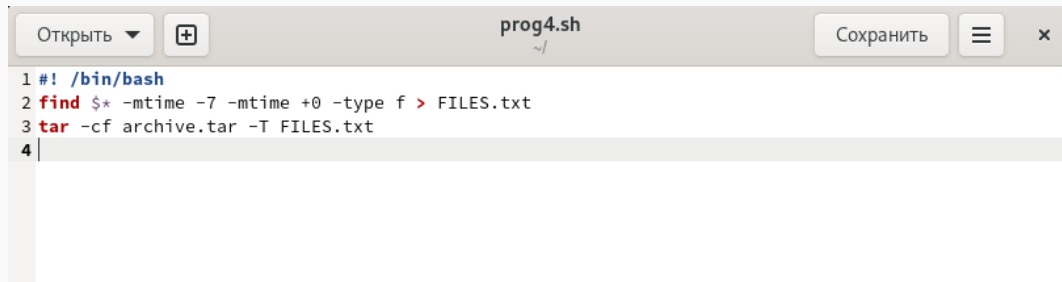
{ #fig:008 width=70% height=70%}

```
[eashoshina@fedora ~]$ gedit prog3.sh
[eashoshina@fedora ~]$ bash prog3.sh 3
[eashoshina@fedora ~]$ ls
✓ 1.tmp      cprog      input.txt
✓ 2.tmp      EAShoshina.github.io lab07.sh
✓ 3.tmp      equiplist2 lab07.sh~
australia   feathers   lab11
backup      file.txt   lab11_1.txt
bin         fun        lab11_2.c
blog        hello      main.cpp
```

{ #fig:009 width=70% height=70%}

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`). (рис. 10.11, 11.11, 12.11)



The screenshot shows a code editor window with a title bar containing the filename 'prog4.sh' and a tilde icon. On the left of the title bar are buttons for 'Открыть' (Open) and a plus icon. On the right are buttons for 'Сохранить' (Save), a hamburger menu icon, and a close 'x' icon. The editor area contains a shell script with four lines of code, each preceded by a line number. The script uses 'find' to locate files and 'tar' to create an archive.

```
1 #! /bin/bash
2 find $* -mtime -7 -mtime +0 -type f > FILES.txt
3 tar -cf archive.tar -T FILES.txt
4 |
```

{ #fig:010 width=70% height=70%}

```
[eashoshina@fedora ~]$ bash prog4.sh /home/eashoshina  
tar: Удаляется начальный '/' из имен объектов  
tar: Удаляются начальные '/' из целей жестких ссылок
```

{ #fig:011 width=70% height=70%}


```
FILES.txt
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/permanent/chrome/idb/1657114595Amcateirvtistv.sqlite
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++github.com/.metadata-v2
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++github.com/ls/data.sqlite
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++github.com/ls/usage
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++www.youtube.com*partitionKey=%28https%2Civan-shamae
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++www.youtube.com*partitionKey=%28https%2Civan-shamae
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++www.youtube.com*partitionKey=%28https%2Civan-shamae
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++esystem.rudn.ru/ls/data.sqlite
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++esystem.rudn.ru/ls/usage
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++forums.ivaniti.com/idb/940727097r9e-c6o5rPdVG.sqlite
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++www.youtube.com/idb/4239436100Co7g%u0026t2aab6aas
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++www.youtube.com/idb/2085183176vCt7G%u0026Cf7C%u0026naf61ag
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++www.youtube.com/idb/128499467yCt7-wiCt7-Nr2eas6pao.
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++www.youtube.com*partitionKey=%28https%2Crudn.ru%29/.
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++www.youtube.com*partitionKey=%28https%2Crudn.ru%29/.
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++www.youtube.com*partitionKey=%28https%2Crudn.ru%29/.
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++www.youtube.com*partitionKey=%28https%2Crudn.ru%29/1
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++www.youtube.com*partitionKey=%28https%2Crudn.ru%29/1
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++rutube.ru/.metadata-v2
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++rutube.ru/cache/caches.sqlite
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++rutube.ru/cache/morgue/163/(7f384656-024b-48ed-8855
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++rutube.ru/cache/.padding
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++rutube.ru/ls/data.sqlite
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++rutube.ru/ls/usage
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++rutube.ru/idb/3422673420Ruhac.sqlite
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++rutube.ru/idb/3713173747_s_qdmban.sqlite
/home/sashoshina/.mozilla/firefox/0qlthcn4.default-release/storage/default/https+++rutube.ru/idb/32633619981ddibc_ra.sqlite
```

Рис. 1: Рис. 12.11: Результат

Выводы

В процессе 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.

Контрольные вопросы

1. Каково предназначение команды `getopts`?

- Осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ...]` Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину.
- Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Предположим, необходимо распознать командную строку следующего формата: `testprog -ifile_in.txt -ofile_out.doc -L -t -r` Вот как выглядит использование оператора `getopts` в этом случае: `while getopts o:i:Ltr optletter do case`

2. Какое отношение метасимволы имеют к генерации имён файлов?

- При перечислении имён файлов текущего каталога можно использовать следующие символы: `-` – соответствует произвольной, в том числе и пустой строке; `?` – соответствует любому одинарному символу; `[c1-c2]` – соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`. Например, `echo *` – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; `ls .c` – выведет все файлы с последними двумя символами, совпадающими с `.c`. `echo prog.?` – выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.` `[a-z]` – соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

3. Какие операторы управления действиями вы знаете?

- Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости отрезультатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути, являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.

4. Какие операторы используются для прерывания цикла?

- Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестаёт быть правильным. Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.

5. Для чего нужны команды `false` и `true`?

- Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования `bash`: это команда `true`, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда `false`, которая всегда возвращает код завершения, не равный нулю (т. е. ложь).

6. Что означает строка `if test -f man /i.$s`, встреченная в командном файле?

- Строка `if test -f man /i.`, `??s/?.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).

7. Объясните различия между конструкциями `while` и `until`.

- Выполнение оператора цикла `while` сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь). При замене в операторе цикла `while` служебного слова `while` на `until` условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла `while` и оператор цикла `until` идентичны.