

Отчет по лабораторной работе №11

**Программирование в командном процессоре ОС UNIX. Ветвления и
циклы**

Сагдеров Камал

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
5	Выводы	14
6	Контрольные вопросы	15
	Список литературы	19

Список иллюстраций

4.1	Первая программа	8
4.2	Результат	9
4.3	Результат	9
4.4	Вызов программы в терминале	9
4.5	Вторая программа	10
4.6	Результат	11
4.7	Третья программа	11
4.8	Результат	12
4.9	Четвертая программа	12
4.10	Результат	12
4.11	Результат	13

1 Цель работы

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

2 Задание

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; `-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`.
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 ∞ (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 `tag` 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`).

3 Теоретическое введение

Оболочка операционной системы (от англ. shell — оболочка) — интерпретатор команд операционной системы (ОС), обеспечивающий интерфейс для взаимодействия пользователя с функциями системы. Зайдя в систему, вы увидите приглашение командной строки — строку, содержащую символ \$ (далее этот символ будет обозначать командную строку). Задача командного интерпретатора состоит в передаче передавать ваши команды операционной системе и прикладным программам, а их ответы — вам. По своим задачам ему соответствует `command.com` в DOS, но функционально оболочки UNIX несравненно богаче. При помощи командных интерпретаторов можно писать небольшие программы — сценарии (скрипты). В Linux доступны следующие командные оболочки:

- Bash — самая распространённая оболочка под Linux. Она ведёт историю команд и предоставляет возможность их редактирования;
- `pdksh` — клон `korn shell`, хорошо известной оболочки в системах UNIX;
- `tcsh` — улучшенная версия `>C shell`;
- `zsh` — новейшая из перечисленных здесь оболочек; реализует улучшенное дополнение и другие удобные функции.

Командный интерпретатор исполняет команды своего языка, заданные в командной строке или поступающие из стандартного ввода или указанного файла.

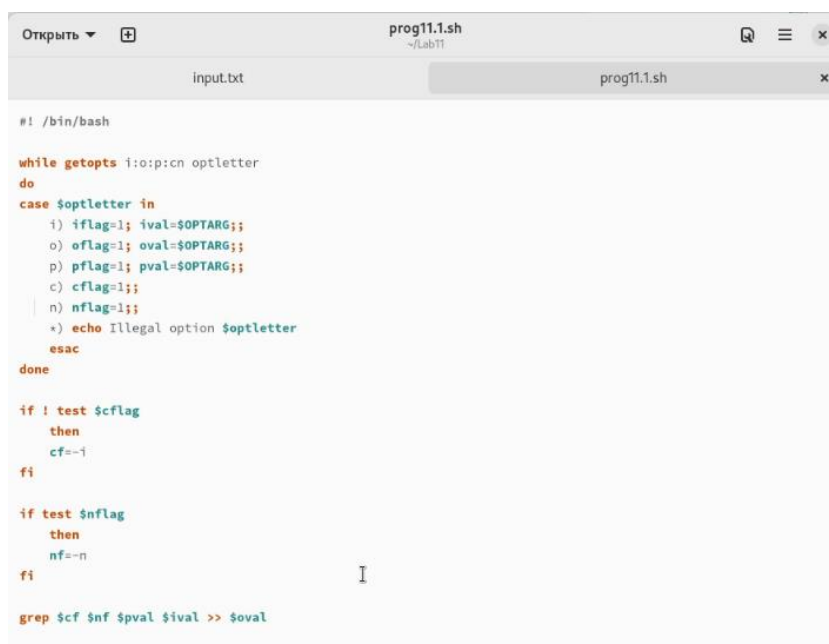
[1]

В качестве команд интерпретируются вызовы системных или прикладных утилит, а также управляющие конструкции. Кроме того, оболочка отвечает за раскрытие шаблонов имен файлов и за перенаправление и связывание ввода-вывода утилит.

В совокупности с набором утилит, оболочка представляет собой операционную среду, полноценный язык программирования и мощное средство решения как системных, так и некоторых прикладных задач, в особенности, автоматизации часто выполняемых последовательностей команд

4 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; `-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`. (рис. 4.1), (рис. 4.2), (рис. 4.4), (рис. 4.3).



```
#!/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
```

Рис. 4.1: Первая программа



Рис. 4.2: Результат



Рис. 4.3: Результат

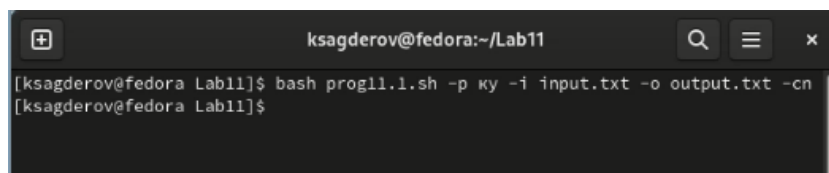


Рис. 4.4: Вызов программы в терминале

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. 4.5),(рис. 4.6).

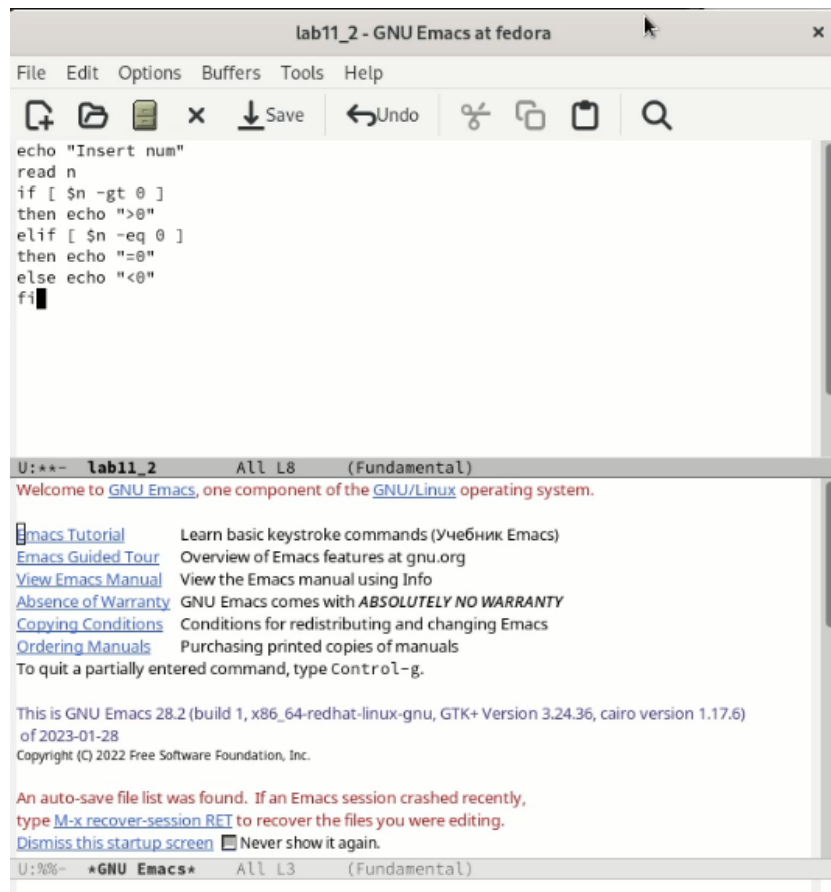
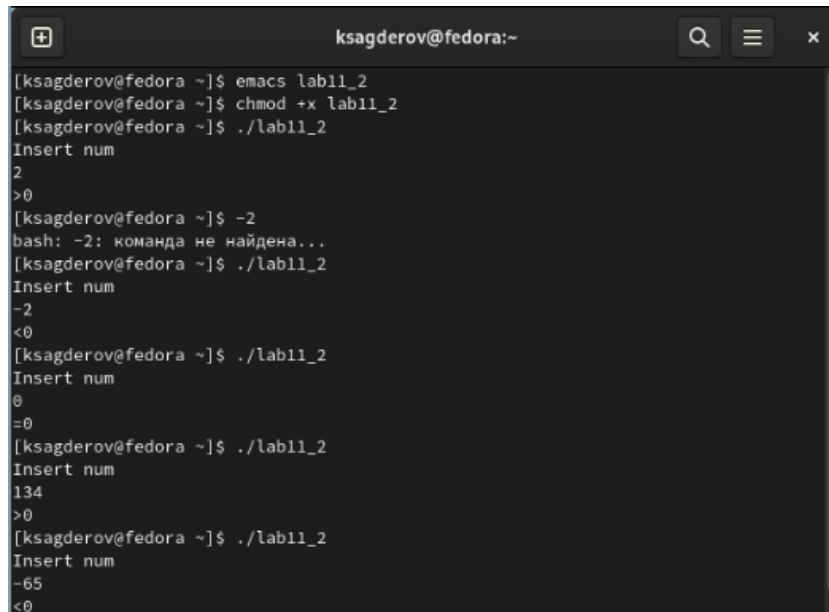


Рис. 4.5: Вторая программа



```
ksagderov@fedora:~  
[ksagderov@fedora ~]$ emacs lab11_2  
[ksagderov@fedora ~]$ chmod +x lab11_2  
[ksagderov@fedora ~]$ ./lab11_2  
Insert num  
2  
>0  
[ksagderov@fedora ~]$ -2  
bash: -2: команда не найдена...  
[ksagderov@fedora ~]$ ./lab11_2  
Insert num  
-2  
<0  
[ksagderov@fedora ~]$ ./lab11_2  
Insert num  
0  
=0  
[ksagderov@fedora ~]$ ./lab11_2  
Insert num  
134  
>0  
[ksagderov@fedora ~]$ ./lab11_2  
Insert num  
-65  
<0
```

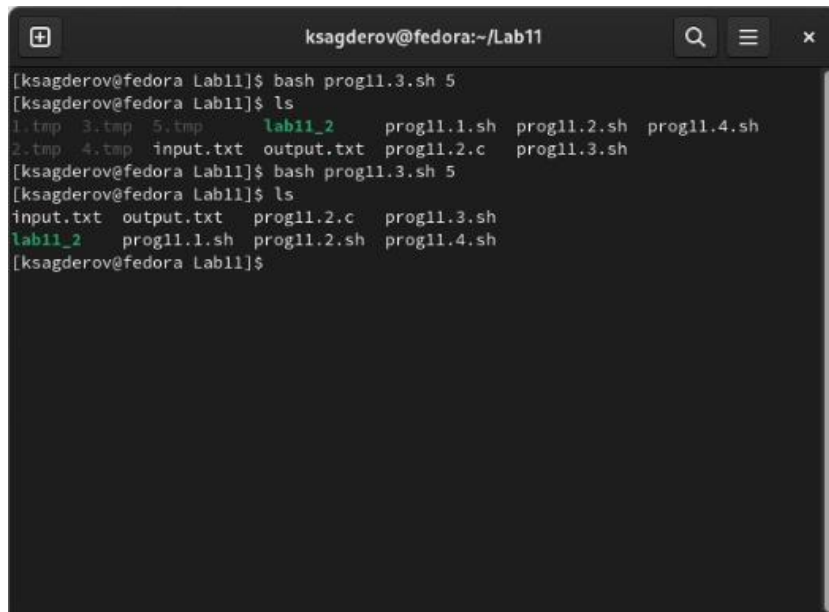
Рис. 4.6: Результат

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) (рис. 4.7), (рис. 4.8).



```
Открыть ▾ + prog11.3.sh  
~/Lab11  
#!/bin/bash  
for ((i=1; i<=9; i++))  
do  
if test -f "$i".tmp  
then rm "$i".tmp  
else touch "$i.tmp"  
fi  
done
```

Рис. 4.7: Третья программа



```
[ksagderov@fedora Lab11]$ bash prog11.3.sh 5
[ksagderov@fedora Lab11]$ ls
1.tmp 3.tmp 5.tmp lab11_2 prog11.1.sh prog11.2.sh prog11.4.sh
2.tmp 4.tmp input.txt output.txt prog11.2.c prog11.3.sh
[ksagderov@fedora Lab11]$ bash prog11.3.sh 5
[ksagderov@fedora Lab11]$ ls
input.txt output.txt prog11.2.c prog11.3.sh
lab11_2 prog11.1.sh prog11.2.sh prog11.4.sh
[ksagderov@fedora Lab11]$
```

Рис. 4.8: Результат

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. 4.9),(рис. 4.10),(рис. 4.11).

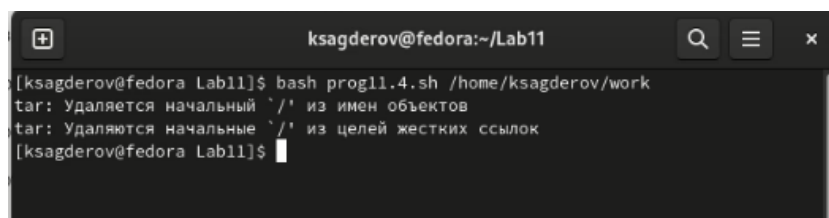


```
Открыть ▾ + prog11.4.sh
~/Lab11

#!/bin/bash

find $* -mtime -7 -mtime +0 -type f > FILES.txt
tar -cf archive.tar -T FILES.txt
```

Рис. 4.9: Четвертая программа



```
[ksagderov@fedora Lab11]$ bash prog11.4.sh /home/ksagderov/work
tar: Удаляется начальный '/' из имен объектов
tar: Удаляются начальные '/' из целей жестких ссылок
[ksagderov@fedora Lab11]$
```

Рис. 4.10: Результат

```
Открыть FILES.txt
~/Lab11

prog11.4.sh FILES.txt

/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/refs/heads/master
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/refs/remotes/origin/master
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/8c/
98e6e5c928a7b6d78945d69498ad9c65b95e
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/7b/
7796ebf3c558018a348773f15528ca2bd62add
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/8f/
fcb59f0b1a31723c9f1738823db8efe7b34f3
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/
72/4e2c285c4d93a492d36578bfee29ca6d3e3356
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/fc/
8d55e91ac7698423285a284358b18c363d8c5
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/5a/
713d9ce0ecd9648e769ebd7b3328693678e72b
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/4c/
35c4ea72a9d0an15c1a9f42b0b1ad7b797179e
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/b9/
b999ed344a96ab8c979c2251d991da46f8d6c9
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/
ba/8c736787d38bbf5112173833e7a77888a94c0b
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/
92/2cdf56c64d2c4eb995131124024274a190f1d5
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/50/
f48e5c5e3a6929889884af9f912b341daa189a
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/
84/9985e656c2d8dd6d01d6186954d3c35963d8c3
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/cf/
8be87474e116185d45ba3183543b232558d6b2
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/
22/44a788592a792c64f0973468be31dc9bb1eb24
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/an/
8af4b6a0661201f591cb36bdf93c276199e183
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/de/
ad5e4c81e9499061c9ce32c2457206fcc67ca8
/home/ksagderov/work/study/2022-2023/Операционные системы/qs-intra/.git/objects/
33/8577e7778e3d549441779ec2e483bb4839f77d
```

Рис. 4.11: Результат

5 Выводы

В процессе выполнения данной лабораторной работы я изучил основы программирования в оболочке ОС UNIX. Научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

6 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 $optletter in
o) iflag = 1; oval =OPTARG;;
i) iflag=1; ival=$OPTARG;;
L) Lflag=1;;
t) tflag=1;;
r) rflag=1;;
*) echo Illegal option
$optletter
esac
done
```

 Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента (будет равно `file_in.txt` для опции `i` и `file_out.doc` для опции `o`). `OPTIND` является числовым индексом на упомянутый аргумент. Функция `getopts` также понимает переменные типа массив, следова-

тельно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

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; [с1-с2] – соответствует любому символу, лексикографически находящемуся между символами с1 и с2. 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 mans/i.$s`, встреченная в командном файле?

Строка `if test -f mans/i.s, mans/i.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.

Список литературы

1. Командные оболочки (shells) [Электронный ресурс]. Free Software Foundation. URL: <https://docs.altlinux.org/ru-RU/archive/2.2/html-single/master/install-html/ch06s04.html>.