

Лабораторная работа 13

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

Кузнецов Василий Юрьевич

Содержание

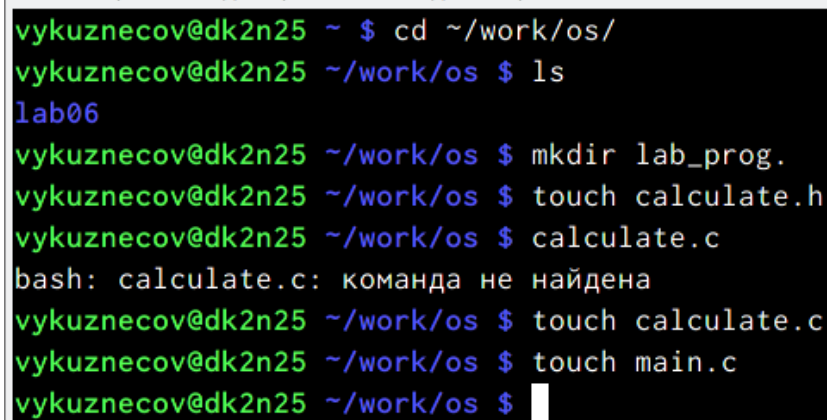
1	Цель работы	3
2	Выполнение лабораторной работы	4
3	Вывод	9
4	Контрольные вопросы	10

1 Цель работы

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

2 Выполнение лабораторной работы

1. Скрипт 1 (рис. 1)



```
vykuznecov@dk2n25 ~ $ cd ~/work/os/  
vykuznecov@dk2n25 ~/work/os $ ls  
lab06  
vykuznecov@dk2n25 ~/work/os $ mkdir lab_prog.  
vykuznecov@dk2n25 ~/work/os $ touch calculate.h  
vykuznecov@dk2n25 ~/work/os $ calculate.c  
bash: calculate.c: команда не найдена  
vykuznecov@dk2n25 ~/work/os $ touch calculate.c  
vykuznecov@dk2n25 ~/work/os $ touch main.c  
vykuznecov@dk2n25 ~/work/os $
```

Figure 2.1: Скрипт 1

Запуск (рис. 2)

```

vykuznecov@dk2n25 ~ $ cd ~/work/os/
vykuznecov@dk2n25 ~/work/os $ ls
lab06
vykuznecov@dk2n25 ~/work/os $ mkdir lab_prog.
vykuznecov@dk2n25 ~/work/os $ touch calculate.h
vykuznecov@dk2n25 ~/work/os $ calculate.c
bash: calculate.c: команда не найдена
vykuznecov@dk2n25 ~/work/os $ touch calculate.c
vykuznecov@dk2n25 ~/work/os $ touch main.c
vykuznecov@dk2n25 ~/work/os $

```

Figure 2.2: Скрипт 1 запуск

2. Скрипт 2 (рис. 3)

```

vykuznecov@dk2n25 ~/work/os $ vi main.
vykuznecov@dk2n25 ~/work/os $ ls
calculate.c calculate.h calculate.o lab06 lab_prog. main. main.c main.o
vykuznecov@dk2n25 ~/work/os $ rm main.
vykuznecov@dk2n25 ~/work/os $ vi mian.c
vykuznecov@dk2n25 ~/work/os $ vi main.o
vykuznecov@dk2n25 ~/work/os $ rm main.c
vykuznecov@dk2n25 ~/work/os $ vi main.c
vykuznecov@dk2n25 ~/work/os $ vi calculate.c
vykuznecov@dk2n25 ~/work/os $ gcc -c calculate.c
vykuznecov@dk2n25 ~/work/os $ gcc -c main.c
main.c: В функции «main»:
main.c:13:9: предупреждение: формат «%s» ожидает аргумент типа «char *», но аргумент 2 имеет тип «char (*)[4]» [-Wformat=]
   13 | scanf("%s",&Operation);
      |         ~^ ~~~~~
      |         | |
      |         | char (*)[4]
      |         char *
vykuznecov@dk2n25 ~/work/os $ gcc calculate.o main.o -o calcul -lm

```

Figure 2.3: Скрипт 2

Запуск (рис. 4)

```

vykuznecov@dk2n25 ~/work/os $ vi main.
vykuznecov@dk2n25 ~/work/os $ ls
calculate.c calculate.h calculate.o lab06 lab_prog. main. main.c main.o
vykuznecov@dk2n25 ~/work/os $ rm main.
vykuznecov@dk2n25 ~/work/os $ vi mian.c
vykuznecov@dk2n25 ~/work/os $ vi main.o
vykuznecov@dk2n25 ~/work/os $ rm main.c
vykuznecov@dk2n25 ~/work/os $ vi main.c
vykuznecov@dk2n25 ~/work/os $ vi calculate.c
vykuznecov@dk2n25 ~/work/os $ gcc -c calculate.c
vykuznecov@dk2n25 ~/work/os $ gcc -c main.c
main.c: В функции «main»:
main.c:13:9: предупреждение: формат «%s» ожидает аргумент типа «char *», но аргумент 2 имеет тип «char (*)[4]» [-Wformat=]
   13 | scanf("%s",&Operation);
      |         ~^ ~~~~~
      |         | |
      |         | char (*)[4]
      |         char *
vykuznecov@dk2n25 ~/work/os $ gcc calculate.o main.o -o calcul -lm

```

Figure 2.4: Скрипт 2 запуск

3. Скрипт 3 (рис. 5)

```
Файл Правка Вид Терминал Вкладки Справка
vykuznecov@dk2n25 ~/work/os $ vi Makefile
vykuznecov@dk2n25 ~/work/os $ gdb ./calcul
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcul...
(No debugging symbols found in ./calcul)
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/v/y/vykuznecov/work/os/calcul
Число: 4
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 6
    10.00
[Inferior 1 (process 117596) exited normally]
(gdb)
```

Figure 2.5: Скрипт 3

Запуск (рис. 6)

```
Файл Правка Вид Терминал Вкладки Справка
vykuznecov@dk2n25 ~/work/os $ vi Makefile
vykuznecov@dk2n25 ~/work/os $ gdb ./calcul
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcul...
(No debugging symbols found in ./calcul)
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/v/y/vykuznecov/work/os/calcul
Число: 4
Операция (+,-,*,/,pow,sqrt,sin,cos,tan): +
Второе слагаемое: 6
  10.00
[Inferior 1 (process 117596) exited normally]
(gdb)
```

Figure 2.6: Скрипт 3 запуск

3 Вывод

Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

4 Контрольные вопросы

1. Весьма необходимой при программировании является команда `getopts`, которая 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 -infile_in.txt -outfile_out.doc -L -t -r` Вот как выглядит использование оператора `getopts` в этом случае:

```
while getopts o:i:Ltr optletter do
  case $optletter in
    o) iflag=1; ival=$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 равна `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, следовательно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

2. При перечислении имен файлов текущего каталога можно использовать следующие символы: `*` — соответствует произвольной, в том числе и пустой строке; `?` — соответствует любому одному символу; `[c1-c1]` — соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`. `echo *` — выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; `ls .c` — *выведет все файлы с последними двумя символами, равными .c*. `echo prog.?` — *выдаст 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. Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости от результатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет Вам 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 дела являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда.
4. 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 перестает быть правильным. Пример бесконечного цикла `while`, с прерыванием в момент, когда файл перестает существовать: `while true do if [! -f $file] then break fi sleep 10 done`

5. 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.
6. Введенная строка означает условие существования файла `man s/i.$s`
7. Если речь идет о 2-х параллельных действиях, то это `while`. когда мы показываем, что сначала делается 1-е действие. потом оно заканчивается при наступлении 2-го действия, применяем `until`.