Front matter

lang: ru-RU title: "Отчет по лабораторной работе №12" subtitle: "по дисциплине: Операционные системы" author: "Трефилова Мария Андреевна"

Formatting

toc-title: "Содержание" toc: true # Table of contents toc_depth: 2 lof: false # List of figures lot: false # List of tables fontsize: 12pt linestretch: 1.5 papersize: a4paper documentclass: scrreprt polyglossia-lang: russian polyglossia-otherlangs: english mainfont: PT Serif romanfont: PT Serif sansfont: PT Sans monofont: PT Mono mainfontoptions: Ligatures=TeX romanfontoptions: Ligatures=TeX sansfontoptions: Ligatures=TeX,Scale=MatchLowercase monofontoptions: Scale=MatchLowercase indent: true pdf-engine: lualatex header-includes: - \linepenalty=10 # the penalty added to the badness of each line within a paragraph (no associated penalty node) Increasing the value makes tex try to have fewer lines in the paragraph. - \interlinepenalty=0 # value of the penalty (node) added after each line of a paragraph. - \hyphenpenalty=50 # the penalty for line breaking at an automatically inserted hyphen - \exhyphenpenalty=50 # the penalty for line breaking at an explicit hyphen - \binoppenalty=700 # the penalty for breaking a line at a binary operator - \relpenalty=500 # the penalty for breaking a line at a relation - \clubpenalty=150 # extra penalty for breaking after first line of a paragraph - \widowpenalty=150 # extra penalty for breaking before last line of a paragraph - \displaywidowpenalty=50 # extra penalty for breaking before last line before a display math - \brokenpenalty=100 # extra penalty for page breaking after a hyphenated line - \predisplaypenalty=10000 # penalty for breaking before a display - \postdisplaypenalty=0 # penalty for breaking after a display - \floatingpenalty = 20000 # penalty for splitting an insertion (can only be split footnote in standard LaTeX) - \raggedbottom # or \flushbottom - \usepackage{float} # keep figures where there are in the text

- \floatplacement{figure}{H} # keep figures where there are in the text

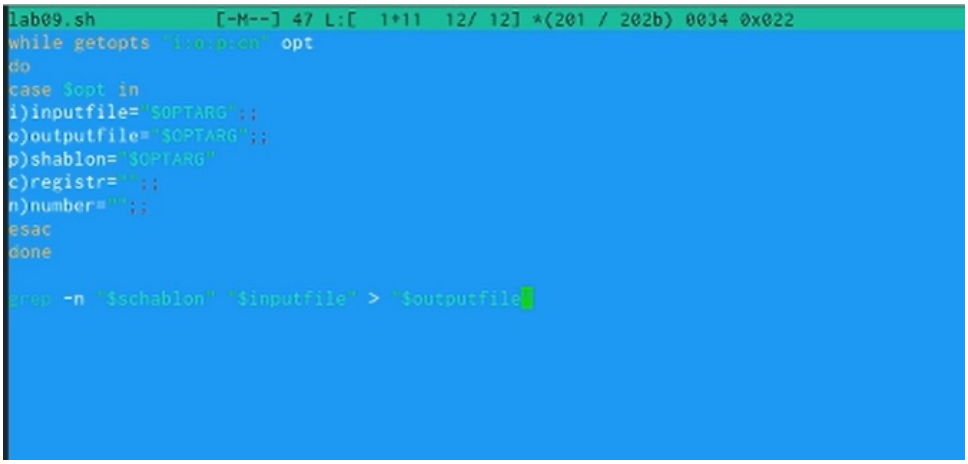
Цель работы

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

Выполнение лабораторной работы

Начальный этап

1. Используя команды getopts grep, напишем командный файл, который анализирует командную строку с ключами: -i — прочитать данные из указанного файла; -o — вывести данные в указанный файл; -p — указать шаблон для поиска; -C — различать большие и малые буквы (по умолчанию включен) -n — выдавать номера строк.



```
matrefilova@dk6n54 ~ $ mcedit lab09.sh

matrefilova@dk6n54 ~ $ chmod ugo+rwx lab09.sh
matrefilova@dk6n54 ~ $ ./lab09.sh
./lab09.sh: строка 7: синтаксическая ошибка рядом с неожиданным маркером «)»
./lab09.sh: строка 7: `c)registr="";;'
matrefilova@dk6n54 ~ $ mcedit lab09.sh

matrefilova@dk6n54 ~ $ chmod ugo+rwx script4.sh
matrefilova@dk6n54 ~ $ ./lab09.sh
./lab09.sh: строка 12: : Нет такого файла или каталога
matrefilova@dk6n54 ~ $ ./lab09.sh -i conf.txt -o test9.txt -p n
grep: conf.txt: Нет такого файла или каталога
matrefilova@dk6n54 ~ $ cat test9.txt
matrefilova@dk6n54 ~ $ ./lab09.sh -i conf.txt -o test9.txt -p n
grep: conf.txt: Нет такого файла или каталога
matrefilova@dk6n54 ~ $ ./lab09.sh -i conf.txt -o test9.txt -p n etconf -c -n
grep: conf.txt: Нет такого файла или каталога
matrefilova@dk6n54 ~ $ ls
 1.odt                  '2021-05-14 14-36-35.mkv'   GNUstep       lab09.sh   lab5.lst   lab7       public_html   script3.sh   tmp          Загрузки      Общедоступные                                  Шаблоны
 1.txt                  '2021-05-14 15-51-46.mkv'   home          lab5       lab6       lab7.asm   script1.sh    script4.sh   Видео        Изображения   'Рабочий стол'
'2021-05-14 14-13-51.mkv'  australia                katalog.txt   lab5.asm   lab6.asm   public     script2.sh    test9.txt    Документы    Музыка        'Снимок экрана от 2020-09-22 15-09-21.png'
matrefilova@dk6n54 ~ $ ./lab09.sh -i 1.txt -o test9.txt -p n etconf -c -n
matrefilova@dk6n54 ~ $ cat test9.txt
1:net
2:Вы помните,
3:Вы всё, конечно, помните,
4:Как я стоял,
5:Приблизившись к стене,
6:Взволнованно ходили вы по комнате
7:И что-то резкое
8:В лицо бросали мне.
9:
10:Вы говорили:
11:
12:Что вас измучила
13:Моя шальная жизнь,
14:Что вам пора за дело приниматься,
15:А мой удел –
16:
17:Катиться дальше, вниз.
18:
19:
20:
21:Любимая!
22:Меня вы не любили.
23:Не знали вы, что в сонмище людском
24:Я был, как лошадь загнанная в мыле,
25:Пришпоренная смелым ездоком.
26:net
matrefilova@dk6n54 ~ $
```

2. Напишем на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю.

```
c1.cpp            [----]  9 L:[  1+ 9   10/ 10] *(192 / 192b) <EOF>
#include <stdlib.h>
#include <stdio.h>
int main()

    int a;
    scanf("%d", &a);
    if (a<0) printf("%d < ", a);
    else if (a>0) printf("%d > ", a);
    else printf("%d is ", a);
return 0;
```

```
lab9_2.sh          [-M--] 11 L:[  1+ 1    2/  2] *(16
./c1
echo -e "0"
```

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

```
lab9_3.sh          [-M--]  2 L:[  1+28  29/ 2
while getops "dn" opt
do
case $pt in
    d)d="GETARG";;
    n)n="GETARG";;
    esac
done

if ["%d"!="0"] then
    while ($filename!=($n+1))
    do
------>for i in $filename
------>    do
------>------>touch $i.tmp
------>    done
------>(name+=1)
    done
else
    filename=1
    while($filename!=($n+1))
    do
------>for i in $filename
------>    do
------>------>rm
------>------>$i.tmp
------>    done
------>(filename+=1)
    done
fi
```

4. Написал командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировал его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад

```
lab9_4.sh          [-M--] 11 L:[  1+ 9  10/ 10] *(142 / 142b) <EOF>
a=0
b=0
read a
read b
if [ -e $a ]
then
    rm $a
fi
find "$b" -maxdepth 1 -ctime -7 -type f -print0 | xargs -0 tar rvf @$a.tar" >
gzip $a.tar
```

```
matrefilova@dk6n54 ~ $ mcedit lab9_4.sh

matrefilova@dk6n54 ~ $ chmod ugo+rwx lab9_4.sh
matrefilova@dk6n54 ~ $ ./lab9_4.sh
0
0
./lab9_4.sh: строка 9: синтаксическая ошибка рядом с неожиданным маркером «newline»
./lab9_4.sh: строка 9: `find "$b" -maxdepth 1 -ctime -7 -type f -print0 | xargs -0 tar rvf "$a.tar" >'
matrefilova@dk6n54 ~ $
```

Выводы

Таким образом, я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

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 -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 $optletter in o) oflag=1; oval=$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 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 на ходящемуся 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