

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Лабораторная работа No 11.

Козлова Нонна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Ответы на контрольные вопросы	13
5	Выводы	15
	Список литературы	16

Список иллюстраций

3.1	Код	8
3.2	Код	9
3.3	Код	9
3.4	Работает	10
3.5	Код	10
3.6	Работает	11
3.7	Использую команду find	12

Список таблиц

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

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; `-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`. 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 \times (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`).

3 Выполнение лабораторной работы

1. Используя команды `getopts` `grep`, написала 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`. (рис. 3.1).

```
#!/bin/bash
iflag=0; oflag=0; cflag=0; nflag=0;
while getopts i:o:p:C:n 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 $optletter
    esac
done
if (($pflag==0))
then echo "шаблон не найден"
else
    if (($iflag==0))
    then echo "шаблон не найден"
    else
        if (($oglag==0))
        then if (($Cflag==0))
            then if (($nflag==0))
                then grep -i $pval $ival
                else grep -n $pval $ival
            fi
        fi
    fi
fi
fi
fi
```

Рис. 3.1: Код

2. Написала на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю.(рис. 3.2), (рис. 3.3).



```
Открыть ▼ [иконка] *prog2.c
~/work/study/2022-2023/Операционные с...

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

Рис. 3.2: Код



```
Открыть ▼ [иконка] prog2.sh
~/work/study/2022-2023/Операционные с...

1 #!/bin/bash
2
3 gcc prog2.c -o prog2
4 ./prog2
5 code=$?
6 case $code in
7     0) echo "число меньше 0";;
8     1) echo "число больше 0";;
9     2) echo "число равно 0";;
10 esac
```

Рис. 3.3: Код

3. Проверяю работу программы (рис. 3.4).

```

nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/1
ab11 $ ./prog2.sh
Введите число: 3
число больше 0
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab11 $ ./prog2.sh
Введите число: -5
число меньше 0
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab11 $ ./prog2.sh
Введите число: 0
число равно 0
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab11 $

```

Рис. 3.4: Работает

- Пишу командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до ∞ (рис. 3.5).

```

1 #!/bin/bash
2
3 opt=$1;
4 form=$2;
5 num=$3;
6 function Files() {
7     for ((i=1; i<$num; i++)) do
8         file=$(echo $form | tr '#' "$i")
9         if [ $opt == "-r" ]
10            then
11             rm -f &file
12         elif [ $opt == "c" ]
13            then
14             touch $file
15         fi
16     done
17 }
18 Files

```

Рис. 3.5: Код

- Проверяю работу программы (рис. 3.6).

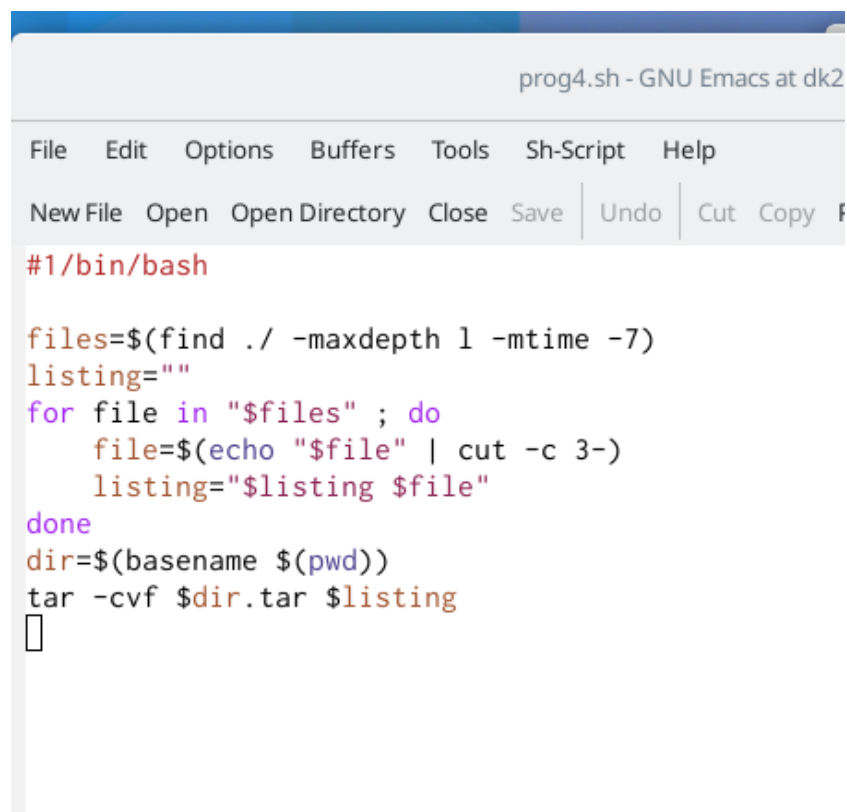
```

udy/2022-2023/Операционные системы/os-intro/labs/lab11 $ touch prog3.sh
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab11 $ chmod +x *.sh
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab11 $ ls
lab11.sh lab11.txt presentation prog2 prog2.c prog2.sh prog3.sh report
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab11 $ ./prog3.sh -r a#.txt 3
Usage: file [-bcCdEhikLlNnprsSvzZ0] [--apple] [--extension] [--mime-encoding]
          [--mime-type] [-e <testname>] [-F <separator>] [-f <namefile>]
          [-m <magicfiles>] [-P <parameter=value>] [--exclude-quiet]
          <file> ...
          file -C [-m <magicfiles>]
          file [--help]
Usage: file [-bcCdEhikLlNnprsSvzZ0] [--apple] [--extension] [--mime-encoding]
          [--mime-type] [-e <testname>] [-F <separator>] [-f <namefile>]
          [-m <magicfiles>] [-P <parameter=value>] [--exclude-quiet]
          <file> ...
          file -C [-m <magicfiles>]
          file [--help]
Usage: file [-bcCdEhikLlNnprsSvzZ0] [--apple] [--extension] [--mime-encoding]
          [--mime-type] [-e <testname>] [-F <separator>] [-f <namefile>]
          [-m <magicfiles>] [-P <parameter=value>] [--exclude-quiet]
          <file> ...
          file -C [-m <magicfiles>]
          file [--help]
nykozlova@dk2n26 ~/work/study/2022-2023/Операционные системы/os-intro/labs/lab11 $ ls
lab11.sh lab11.txt presentation prog2 prog2.c prog2.sh prog3.sh report

```

Рис. 3.6: Работает

6. Пишу командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицирую его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (рис. 3.7).

A screenshot of a GNU Emacs editor window. The title bar at the top reads "prog4.sh - GNU Emacs at dk2". Below the title bar is a menu bar with the following items: "File", "Edit", "Options", "Buffers", "Tools", "Sh-Script", and "Help". Below the menu bar is a toolbar with icons for "New File", "Open", "Open Directory", "Close", "Save", "Undo", "Cut", "Copy", and "Paste". The main text area contains a shell script with the following code:

```
#!/bin/bash

files=$(find ./ -maxdepth 1 -mtime -7)
listing=""
for file in "$files" ; do
    file=$(echo "$file" | cut -c 3-)
    listing="$listing $file"
done
dir=$(basename $(pwd))
tar -cvf $dir.tar $listing
```

The script is written in a shell script syntax with some color coding: the shebang is red, keywords like "for", "do", "done", and "tar" are purple, and variable names and file names are in various colors (brown, green, blue). The script finds files in the current directory with a maximum depth of 1 and a maximum age of 7 days, then creates a tar archive of them.

Рис. 3.7: Используя команду find

4 Ответы на контрольные вопросы

1. Команда `getopts` является встроенной командой командной оболочки `bash`, предназначенной для разбора параметров сценариев. Она обрабатывает исключительно однобуквенные параметры как с аргументами, так и без них и этого вполне достаточно для передачи сценариям любых входных данных.

2. При генерации имен используют метасимволы:

произвольная (возможно пустая) последовательность символов; `?` один произвольный символ; `[...]` любой из символов, указанных в скобках перечислением и/или с указанием диапазона; `cat f*` выдаст все файлы каталога, начинающиеся с `"f"`; `cat f` выдаст все файлы, содержащие `"f"`; `cat program.?` выдаст файлы данного каталога с однобуквенными расширениями, скажем `"program.c"` и `"program.o"`, но не выдаст `"program.com"`; `cat [a-d]*` выдаст файлы, которые начинаются с `"a"`, `"b"`, `"c"`, `"d"`. Аналогичный эффект дадут и команды `"cat [abcd]"` и `"cat [bdac]"`. Операторы `&&` и `||` являются управляющими операторами. Если в командной строке стоит `command1 && command2`, то `command2` выполняется в том, и только в том случае, если статус выхода из команды `command1` равен нулю, что говорит об успешном ее завершении. Аналогично, если командная строка имеет вид `command1 || command2`, то команда `command2` выполняется тогда, и только тогда, когда статус выхода из команды `command1` отличен от нуля.

3. Оператор `break` завершает выполнение ближайшего включающего цикла или условного оператора, в котором он отображается.

4. Команда `true` всегда возвращает ноль в качестве выходного статуса для индикации успеха. Команда `false` всегда возвращает не-ноль в качестве выходного статуса для индикации неудачи. Во всех управляющих конструкциях в качестве логического значения используется код возврата из программы, указанной в качестве условия. Код возврата 0 – истина, любое другое значение – ложь. Программа `true` – всегда завершается с кодом 0, `false` – всегда завершается с кодом 1.
5. Введенная строка означает условие существования файла `mans/i.$s`
6. Цикл `While` выполняется до тех пор, пока указанное в нем условие истинно. Когда указанное условие становится ложным - цикл завершается. Цикл `Until` выполняется до тех пор, пока указанное в нем условие ложно.

5 Выводы

В ходе лабораторной работы, я изучила основы программирования в оболочке ОС UNIX, нааучилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Список литературы