

Лабораторная работа 12

**Программирование в командном процессоре ОС UNIX. Расширенное
программирование**

Кузнецов Василий Юрьевич

Содержание

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

1 Цель работы

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

2 Выполнение лабораторной работы

1. Скрипт 1 (рис. 1)

```
vykuznecov@dk3n54:~  
#!/bin/bash  
  
while getopts :i:o:p:cn opt  
do  
    case "${opt}" in  
        i)input=${OPTARG};;  
        o)output=${OPTARG};;  
        p)mask=${OPTARG};;  
        c)c=true;;  
        n)n=true;;  
    esac  
done  
  
if [ $c ]  
then  
    if [ $n ]  
    then  
        grep -n -i $mask $input > $output  
        exit 0  
    else  
        grep -i $mask $input > $output  
        exit 0  
    fi  
else  
    grep -n $mask $input > $output  
    exit 0  
fi
```

Figure 2.1: Скрипт 1

Запуск (рис. 2)

```
vykuznecov@dk3n54:~/lab12
lockfile="./locking.file"
exec (fn_>"$lockfile"
if test -f "$lockfile"
then
    while [ 1 != 0 ]
    do
        if flock -n ${fn}
        then
            echo "file was locked"
            sleep 4
            echo "unlocking"
            flock -u ${fn}
        else
            echo "file already locked"
            sleep 3
        fi
    done
fi
```

Figure 2.2: Скрипт 1 запуск

2. Скрипт 2 (рис. 3)

```
vykuznecov@dk3n54 ~/lab12 $ vi script1
vykuznecov@dk3n54 ~/lab12 $ vi script1
vykuznecov@dk3n54 ~/lab12 $ chmod 777 script1
vykuznecov@dk3n54 ~/lab12 $
```

Figure 2.3: Скрипт 2

Запуск (рис. 4)

```
command=""

while getopts :n: opt
do
case $opt in
n)command="$OPTARG";;
esac
done

if test -f "/usr/share/man/man1/$command.1.gz"
then less /usr/share/man/man1/$command.1.gz
else
echo "no such command"
fi
```

Figure 2.4: Скрипт 2 запуск

3. Скрипт 3 (рис. 5)

```

vykuznecov@dk3n54 ~/lab12 $ vi script2
vykuznecov@dk3n54 ~/lab12 $ ls
script1  script2
vykuznecov@dk3n54 ~/lab12 $ vi script1
vykuznecov@dk3n54 ~/lab12 $ vi script2
vykuznecov@dk3n54 ~/lab12 $ ./script2 -n touch
bash: ./script2: Отказано в доступе
vykuznecov@dk3n54 ~/lab12 $ chmod 777 script2
vykuznecov@dk3n54 ~/lab12 $ ./script2 -n touch
no such command
vykuznecov@dk3n54 ~/lab12 $ ./script2 -n qwerty
no such command
vykuznecov@dk3n54 ~/lab12 $ █

```

Figure 2.5: Скрипт 3

Запуск (рис. 6)

```

vykuznecov@dk3n54 ~/lab12 $ vi script3
vykuznecov@dk3n54 ~/lab12 $ chmod 777 script3
vykuznecov@dk3n54 ~/lab12 $ ./script3
bagbj
vykuznecov@dk3n54 ~/lab12 $ ./script3
becda
vykuznecov@dk3n54 ~/lab12 $ █

```

Figure 2.6: Скрипт 3 запуск

3 Вывод

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

4 Контрольные вопросы

4.1 Контрольные вопросы

1: Найдите синтаксическую ошибку в следующей строке: `while [$1 != "exit"]`

\$1. Так же между скобками должны быть пробелы. В противном случае скобки и рядом стоящие символы будут восприниматься как одно целое

2: Как объединить (конкатенация) несколько строк в одну?

`cat file.txt | xargs | sed -e 's/\./.\n/g'`

3: Найдите информацию об утилите `seq`. Какими иными способами можно реализовать её функционал при программировании на `bash`?

`seq` - выдает последовательность чисел. Реализовать ее функционал можно командой `for n in {1..5} do done`

4: Какой результат даст вычисление выражения `$((10/3))`?

3

5: Укажите кратко основные отличия командной оболочки `zsh` от `bash`.

`Zsh` очень сильно упрощает работу. Но существуют различия. Например, в `zsh` после `for` обязательно вставлять пробел, нумерация массивов в `zsh` начинается с 1 (что не особо удобно на самом деле). Если вы собираетесь писать скрипт, который легко будет запускать множество разработчиков, то я рекомендуется `Bash`. Если скрипты вам не нужны - `Zsh` (более простая работа с файлами, например)

6: Проверьте, верен ли синтаксис данной конструкции `for ((a=1; a <= LIMIT; a++))`

Верен

7: Сравните язык `bash` с какими-либо языками программирования. Какие преимущества у `bash` по сравнению с ними? Какие недостатки?

Bash позволяет очень легко работать с файловой системой без лишних конструкций (в отличие от обычного языка программирования). Но относительно обычных языков программирования `bash` очень сжат. Тот же Си имеет гораздо более широкие возможности для разработчика.