

Лабораторная работа-12

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

Световидова Полина НБИбд-04-22

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
4	Выводы	17
5	Контрольные вопросы	18

Список иллюстраций

figno1	создание файла	8
figno2	код	9
figno3	проверка программы	10
figno4	создание нового файла	10
figno5	код для map	11
figno6	проверка кода	11
figno7	map less	13
figno8	создание нового файла	14
figno9	код random	15
figno10	проверка кода с random	16

Список таблиц

1 Цель работы

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

2 Задание

1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t_1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени $t_2 < t_1$, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (`> /dev/tty#`, где `#` — номер терминала куда перенаправляется вывод), в котором также запущен этот файл, но не в фоновом, а в привилегированном режиме. Доработать программу так, чтобы имела возможность взаимодействия трёх и более процессов.
2. Реализовать команду `man` с помощью командного файла. Изучите содержимое каталога `/usr/share/man/man1`. В нём находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой `less` сразу же просмотрев содержимое справки. Командный файл должен получать в виде аргумента командной строки название команды и в виде результата выдавать справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге `man1`.
3. Используя встроенную переменную `$RANDOM`, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтите, что `$RANDOM` выдаёт псевдо случайные числа в диапазоне от

0 до 32767.

3 Выполнение лабораторной работы

Создаю файл для выполнения работы и написания кода

3.1

```
[root@10 ~]# touch 12lab.sh  
[root@10 ~]# chmod +x 12lab.sh
```

создание файла

3.2

написание кода по заданию в emacs

3.3

```
Открыть ▾ 12lab.sh Стр. 14, Поз. 17
/goot (Администратор)

1  #!/bin/bash
2  t1=$1
3  t2=$2
4  s1=$(date +%s)
5  s2=$(date +%s)
6  ((t=$s2-$s1))
7  while ((t<t1))
8  do
9      echo "ожидание"
10     sleep 1
11     s2=$(date +%s)
12     ((t=$s2-$s1))
13 done
14 s1=$(date +%s)
15 s2=$(date +%s)
16 ((t=$s2-$s1))
17 while ((t<t2))
18 do
19     echo "выполнение"
20     sleep 1
21     s2=$(date +%s)
22     ((t=$s2-$s1))
23 done
24
```

КОД

3.4

проверяю написанную мной программу

3.5

```
root@10 ~]# ./12lab.sh  
root@10 ~]#
```

проверка программы

3.6

создаю новый файл для реализации команды map

3.7

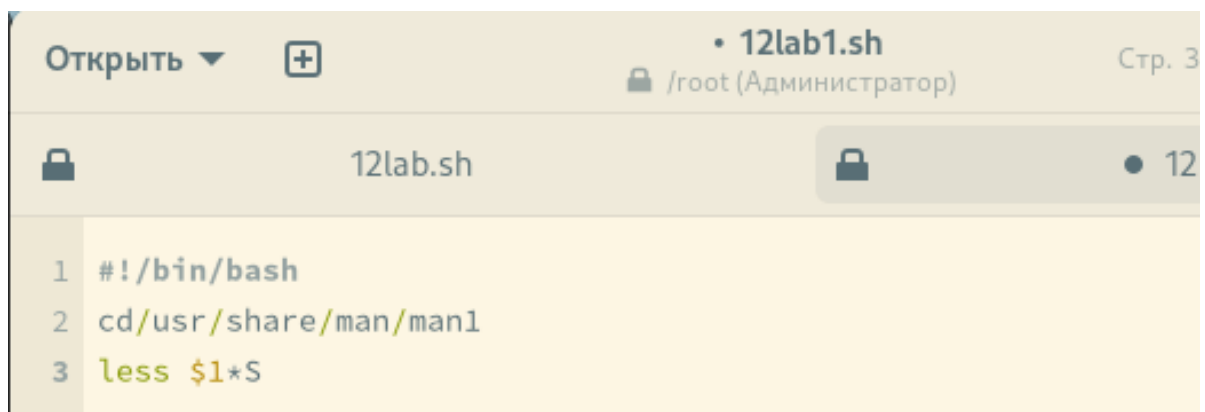
```
root@10 ~]# ./12lab.sh  
root@10 ~]# touch 12lab1.sh  
root@10 ~]# chmod +x 12lab1.sh  
root@10 ~]#
```

создание нового файла

3.8

пишу сам код для реализации задумки

3.9



The screenshot shows a file editor window with a title bar containing "Открыть", a plus icon, "• 12lab1.sh", and "Стр. 3". Below the title bar is a toolbar with a lock icon, the filename "12lab1.sh", another lock icon, and a page indicator "• 12". The main editing area contains three lines of code:

```
1 #!/bin/bash
2 cd/usr/share/man/man1
3 less $1*S
```

код для man

3.10

проверяю новую программу на работу

3.11



The screenshot shows a terminal window with the following text:

```
[root@10 ~]# ./12lab1.sh less
```

проверка кода

3.12

```
root@10:/usr
LESS(1)                                General Commands Manual                                LESS(1)

NAME
    less - opposite of more

SYNOPSIS
    less -?
    less --help
    less -V
    less --version
    less [-[+]aABcDeEfFgGiIJKLmMnNqQrRsSuUVwWX~]
        [-b space] [-h lines] [-j line] [-k keyfile]
        [-{o0} logfile] [-p pattern] [-P prompt] [-t tag]
        [-T tagsfile] [-x tab,...] [-y lines] [-[z] lines]
        [-# shift] [+[[cmd] [--] [filename]...]
    (See the OPTIONS section for alternate option syntax with long option
    names.)

DESCRIPTION
    less is a program similar to more(1), but which allows backward move-
    ment in the file as well as forward movement. Also, less does not have
    to read the entire input file before starting, so with large input
    files it starts up faster than text editors like vi(1). Less uses
    termcap (or terminfo on some systems), so it can run on a variety of
    terminals. There is even limited support for hardcopy terminals. (On
    a hardcopy terminal, lines which should be printed at the top of the
    screen are prefixed with a caret.)

    Commands are based on both more and vi. Commands may be preceded by a
    decimal number, called N in the descriptions below. The number is used
    by some commands, as indicated.

COMMANDS
    In the following descriptions, ^X means control-X. ESC stands for the
    ESCAPE key; for example ESC-v means the two character sequence "ES-
    CAPE", then "v".

    h or H Help: display a summary of these commands. If you forget all
        the other commands, remember this one.

    SPACE or ^V or f or ^F
        Scroll forward N lines, default one window (see option -z be-
        low). If N is more than the screen size, only the final screen-
        ful is displayed. Warning: some systems use ^V as a special
    Manual page less(1) line 1 (press h for help or q to quit)
```

man less

3.13

создаю новый текстовый файл и делаю его исполняемым

3.14

```
[root@10 ~]# touch 12lab2.sh  
[root@10 ~]# chmod +x 12lab2.sh  
[root@10 ~]#
```

создание нового файла

3.15

пишу код, генерирующий случайную последовательность букв латинского алфавита

3.16

```
1  #!/bin/bash
2  M=10
3  c=1
4  d=1
5  echo
6  echo "10 случайных слов"
7  while (($c!=($M+1)))
8  do
9      echo $(for((i=1; i<=10; i++));
10         do
11             printf '%s' "${RANDOM:0:1}"
12         done
13         tr '[0-9]' '[a-z]'
14         echo $d
15         ((c+=1))
16         ((d+=1))
17     done
```

код random

3.17

проверяю программу на работу

3.18

```
[root@10 ~]# ./12lab2.sh  
  
10 случайных слов  
есссбсчсбј  
1  
јсссеіесббј  
2  
сббсссбсbeb  
3  
hbgіhсfсcd  
4  
ссббссссссhс  
5  
ebgbcсbgbd  
6  
сббсссddсbс  
7  
bbіbсссbfcg  
8  
dbссhсbсссb  
9  
сbdјfејhdb  
10  
[root@10 ~]#
```

проверка кода с random

3.19

4 Выводы

В ходе выполнения лабораторной работы №12 я изучила основы программирования в оболочке ОС Linux, а так же научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов

5 Контрольные вопросы

1. `while [$1 != "exit"]` В данной строчке допущены следующие ошибки:
 - не хватает пробелов после первой скобки [и перед второй скобкой]
 - выражение `$1` необходимо взять в `"`, потому что эта переменная может содержать пробелы. Таким образом, правильный вариант должен выглядеть так: `while ["$1" != "exit"]`
2. Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами:
 - Первый: `VAR1="Hello," VAR2=" World" VAR3="VAR1VAR2" echo "$VAR3"`
Результат: Hello, World
 - Второй: `VAR1="Hello," VAR1+= " World" echo "$VAR1"` Результат: Hello, World
3. Команда `seq` в Linux используется для генерации чисел от ПЕРВОГО до ПОСЛЕДНЕГО шага INCREMENT. Параметры:
 - `seq LAST`: если задан только один аргумент, он создает числа от 1 до LAST с шагом шага, равным 1. Если LAST меньше 1, значение не выдает.
 - `seq FIRST LAST`: когда заданы два аргумента, он генерирует числа от FIRST до LAST с шагом 1, равным 1. Если LAST меньше FIRST, он не выдает никаких выходных данных.
 - `seq FIRST INCREMENT LAST`: когда заданы три аргумента, он генерирует числа от FIRST до LAST на шаге INCREMENT. Если LAST меньше, чем FIRST, он не производит вывод.

- `seq -f «FORMAT» FIRST INCREMENT LAST`: эта команда используется для генерации последовательности в форматированном виде. `FIRST` и `INCREMENT` являются необязательными.
 - `seq -s «STRING» ПЕРВЫЙ ВКЛЮЧЕНО`: Эта команда используется для `STRING` для разделения чисел. По умолчанию это значение равно `/n`. `FIRST` и `INCREMENT` являются необязательными.
 - `seq -w FIRST INCREMENT LAST`: эта команда используется для выравнивания ширины путем заполнения начальными нулями. `FIRST` и `INCREMENT` являются необязательными.
4. Результатом данного выражения `$((10/3))` будет 3, потому что это целочисленное деление без остатка.
 5. Отличия командной оболочки `zsh` от `bash`:
 - В `zsh` более быстрое автодополнение для `cd` с помощью `Tab`
 - В `zsh` существует калькулятор `zcalc`, способный выполнять вычисления внутри терминала
 - В `zsh` поддерживаются числа с плавающей запятой
 - В `zsh` поддерживаются структуры данных «хэш»
 - В `zsh` поддерживается раскрытие полного пути на основе неполных данных
 - В `zsh` поддерживается замена части пути
 - В `zsh` есть возможность отображать разделенный экран, такой же как разделенный экран `vim`
 6. `for ((a=1; a <= LIMIT; a++))` синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать `$` перед переменными `()`.
 7. Преимущества скриптового языка `bash`:
 - Один из самых распространенных и ставится по умолчанию в большинстве дистрибутивах Linux, MacOS

- Удобное перенаправление ввода/вывода
- Большое количество команд для работы с файловыми системами Linux
- Можно писать собственные скрипты, упрощающие работу в Linux
- Недостатки скриптового языка bash:
- Дополнительные библиотеки других языков позволяют выполнить больше действий
- Bash не является языком общего назначения
- Утилиты, при выполнении скрипта, запускают свои процессы, которые, в свою очередь, отражаются на скорости выполнения этого скрипта
- Скрипты, написанные на bash, нельзя запустить на других операционных системах без дополнительных действий