

Лабораторная работа №12

Дисциплина: Операционные системы

Галанова Дарья Александровна

Содержание

1 Цель работы	5
2 Задание	6
3 Выполнение лабораторной работы	7
4 Библиография	17
5 Выводы	18

List of Tables

List of Figures

3.1	Создание файла	7
3.2	Скрипт №1.....	8
3.3	Проверка работы скрипта.....	8
3.4	Изменённый скрипт №1.....	9
3.5	Изменённый скрипт №1.....	9
3.6	Проверка работы скрипта.....	9
3.7	Реализация команды map	10
3.8	Реализация команды map	10
3.9	Создание файла	11
3.10	Скрипт №2.....	11
3.11	Проверка работы скрипта.....	12
3.12	Создание файла	12
3.13	Скрипт №3.....	13
3.14	Проверка работы скрипта.....	13

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. Сделать отчёт по лабораторной работе №12 в формате Markdown.
2. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

3 Выполнение лабораторной работы

1). Написала 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). Для данной задачи я создала файл: sem.sh (Рисунки 3.1) и написала соответствующий скрипт (алгоритм действий представлен на рис. 3.2).

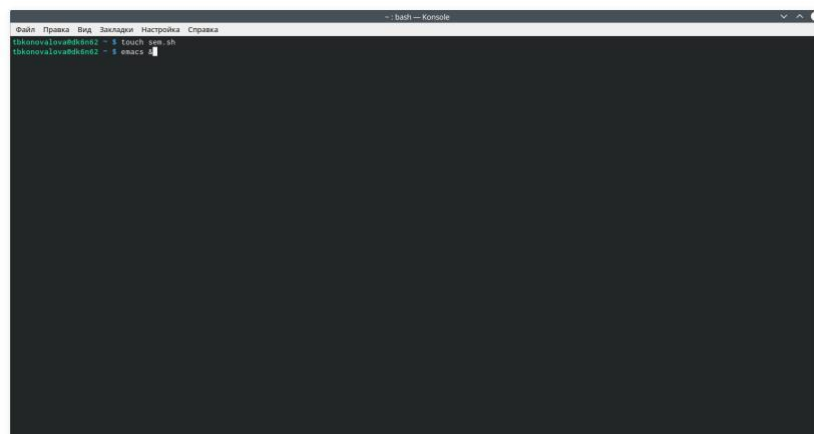


Figure 3.1: Создание файла

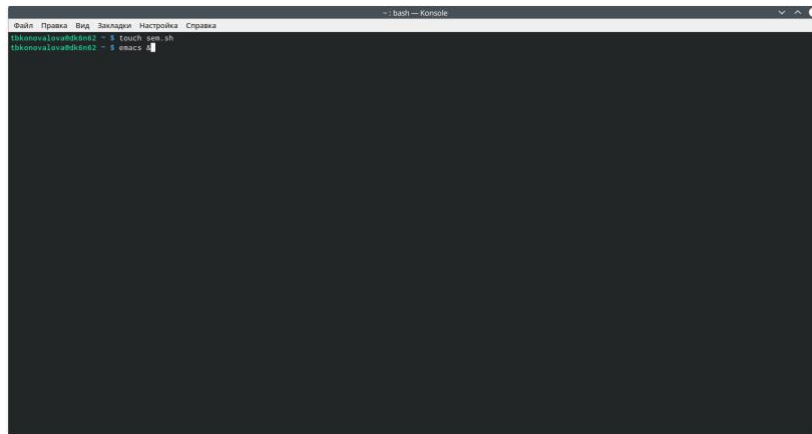


Figure 3.2: Скрипт №1

Далее я проверила работу написанного скрипта (команда «./sem.sh47»), пред-варительно добавив право на исполнение файла (команда «chmod+xsem.sh») (Скриншот 3.3). Скрипт работает корректно.

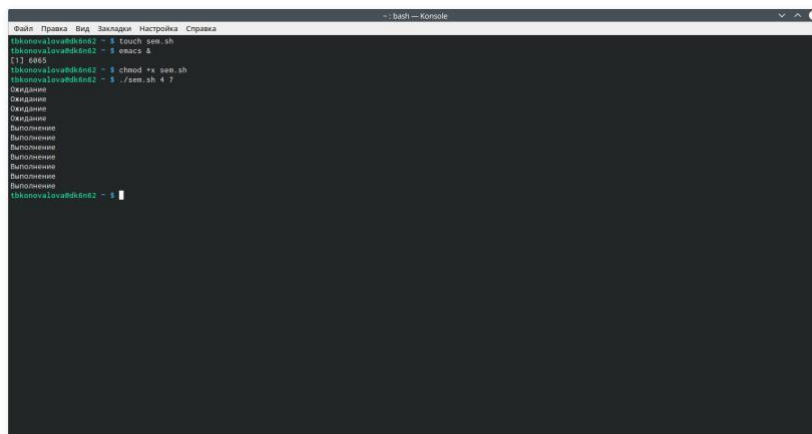


Figure 3.3: Проверка работы скрипта

После этого я изменила скрипт так, чтобы его можно было выполнять в нескольких терминалах и проверила его работу (например, команда «./sem.sh2 3 Ожидание > /dev/pts/1 &») (алгоритм действий представлен на рис. 3.4 , 3.5 , 3.6). Однако у меня не получилось проверить работу скрипта, так как было отказно в доступе.


```

File Edit Options Buffers Tools ShellScript Help
~/bin/bash
function ogidania
{
  s1=$(date +%s)
  s2=$(date +%s)
  ((t=${s2}-${s1}))
  while ((t < 12))
  do
    echo "Ожидание"
    sleep 1
    s1=$(date +%s)
    s2=$(date +%s)
  done
}

function vipolnenie
{
  s1=$(date +%s)
  s2=$(date +%s)
  ((t=${s2}-${s1}))
  while ((t < 12))
  do
    echo "Выполнение"
    sleep 1
    s1=$(date +%s)
    s2=$(date +%s)
  done
}

t1=$1
t2=$2
<command>
while true
do
  if [ "$command" == "Buzug" ]
  then
    echo "Buzug"
    exit 0
  fi
  if [ "$command" == "Ozhdaniye" ]
  then
    ogidania
  fi
  if [ "$command" == "Vykonaniye" ]
  then
    vipolnenie
  fi
  echo "Cmagnete podstrane: "
  read command
done

U===== sen.sh Top L27 (Shell-script[sh]) On Mon 31 14:38 0.88
Warning (initialization): An error occurred while loading "/.emacs":
error: Package "fira-code-mode-" is unavailable
To ensure normal operation, you should investigate and remove the
cause of the error in your initialization file. Start Emacs with
the "--debug-init" option to view a complete error backtrace.
[]

U|<= warnings All 18 (Special) On Mon 31 14:38 0.88
Write /afs/db.ccl.pfu.edu.ru/home/ru/ibkonova/sem.sh

```

Figure 3.4: Изменённый скрипт №1

```

File Edit Options Buffers Tools ShellScript Help
~/bin/bash
t1=$1
t2=$2
<command>
while true
do
  if [ "$command" == "Buzug" ]
  then
    echo "Buzug"
    exit 0
  fi
  if [ "$command" == "Ozhdaniye" ]
  then
    ogidania
  fi
  if [ "$command" == "Vykonaniye" ]
  then
    vipolnenie
  fi
  echo "Cmagnete podstrane: "
  read command
done

U===== sen.sh Bot L27 (Shell-script[sh]) On Mon 31 14:38 0.88
Warning (initialization): An error occurred while loading "/.emacs":
error: Package "fira-code-mode-" is unavailable
To ensure normal operation, you should investigate and remove the
cause of the error in your initialization file. Start Emacs with
the "--debug-init" option to view a complete error backtrace.
[]

U|<= warnings All 18 (Special) On Mon 31 14:38 0.88
End of buffer

```

Figure 3.5: Изменённый скрипт №1

```

Файл Правка Вид Закладки Настройка Справка
~/bin/bash
ibkonova@lova8d8n2 ~ $ ./sem.sh 2 3 Ожидание > /dev/pts/1 &
[2] 1880
ibkonova@lova8d8n2 ~ $ bash: /dev/pts/1: Отказано в доступе

[1]* Выход 1 ./sem.sh 2 3 Ожидание > /dev/pts/1
ibkonova@lova8d8n2 ~ $ ./sem.sh 2 3 Ожидание > /dev/pts/2 &
[2] 1881
ibkonova@lova8d8n2 ~ $ bash: /dev/pts/2: Отказано в доступе

[1]* Выход 1 ./sem.sh 2 3 Ожидание > /dev/pts/2
ibkonova@lova8d8n2 ~ $ ./sem.sh 2 5 Выполнение > /dev/pts/2 &
[2] 1880
ibkonova@lova8d8n2 ~ $ bash: /dev/pts/2: Отказано в доступе

[1]* Выход 1 ./sem.sh 2 5 Выполнение > /dev/pts/2
ibkonova@lova8d8n2 ~ $

```

Figure 3.6: Проверка работы скрипта

2). Реализовала команду `man` с помощью командного файла. Изучила содержимое каталога `/usr/share/man/man1` (Рисунки 3.7 , 3.8). В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в си-стеме 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`.

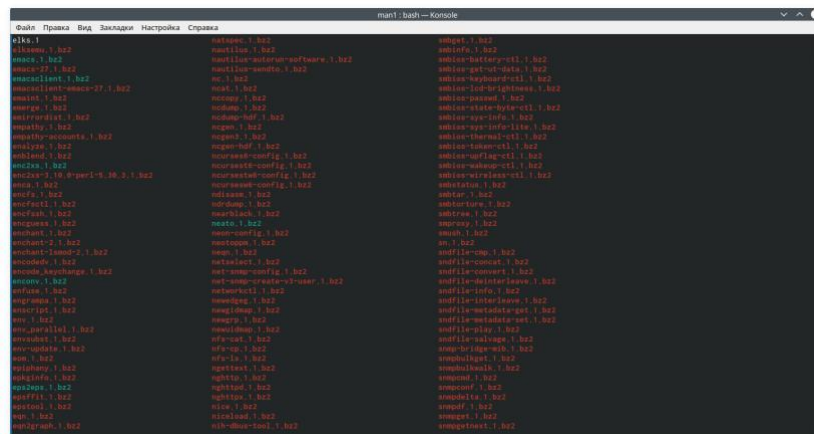


Figure 3.7: Реализация команды `man`

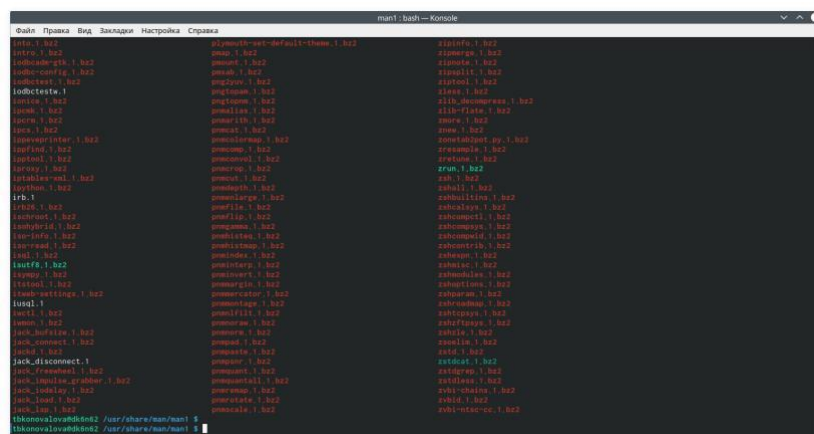


Figure 3.8: Реализация команды `man`

Для данной задачи я создала файл: `man.sh` (Рисунки 3.9) и написала соответ-

ствующий скрипт.

```
- $ touch man.sh
- $ emacs &
```

Figure 3.9: Создание файла

Далее я проверила работу написанного скрипта (команды «./man.shls» и «./man.sh mkdir»), предварительно добавив право на исполнение файла (команда «chmod +x man.sh») (Скриншот 3.10). Скрипт работает корректно.

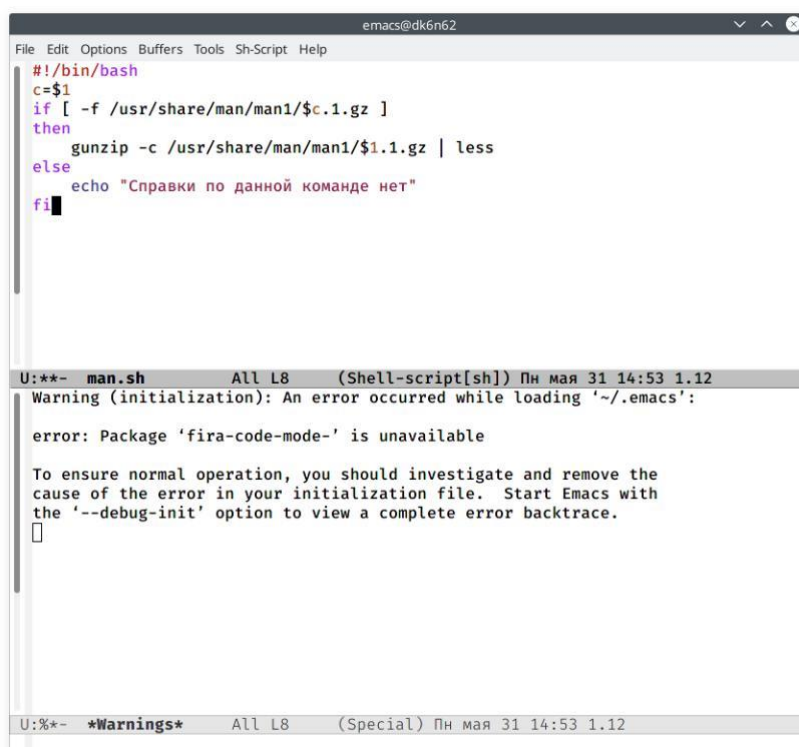
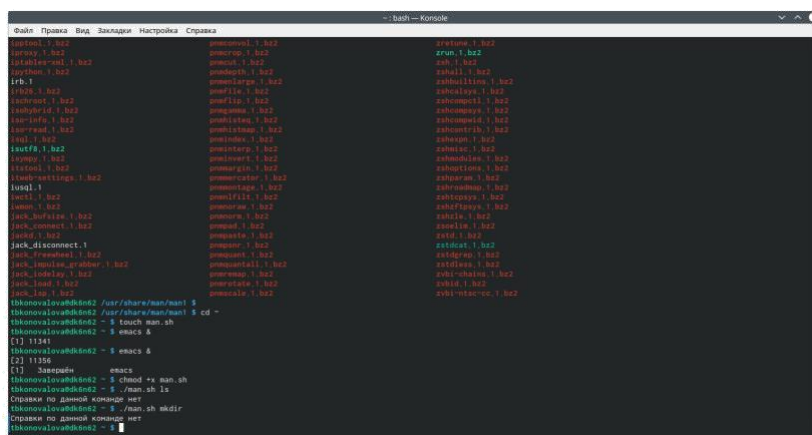


Figure 3.10: Скрипт №2



3). Используя встроенную переменную \$RANDOM, написала командный файл, генерирующий случайную последовательность букв латинского алфавита. Для данной задачи я создала файл: random.sh (Рисунок 3.12) и написала соответствующий скрипт (Рисунки 3.13).

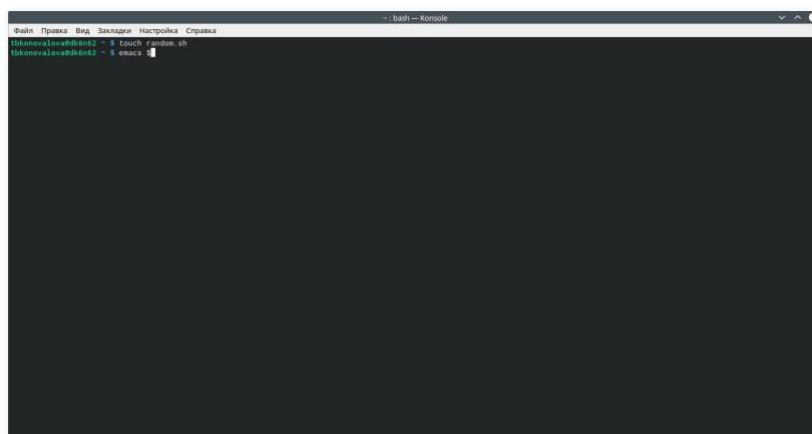


Figure 3.12: Создание файла

```
#!/bin/bash
i=$1
for (( i=0; i<$i; i++ ))
do
    (( char=$((RANDOM%26+1)) ))
    case $char in
        1) echo -n a;; 2) echo -n b;; 3) echo -n c;; 4) echo -n d;; 5) echo -n e;; 6) echo -n f;; 7) echo -n g;; 8) echo -n h;; 9) echo -n i;;
        10) echo -n j;; 11) echo -n k;; 12) echo -n l;; 13) echo -n m;; 14) echo -n n;; 15) echo -n o;; 16) echo -n p;; 17) echo -n q;; 18) echo -n r;; 19) echo -n s;; 20) echo -n t;;
        21) echo -n u;; 22) echo -n v;; 23) echo -n w;; 24) echo -n x;; 25) echo -n y;; 26) echo -n z;;
    esac
done
echo
```

Ubuntu 22.04 LTS [gnome@ubuntu:~]\$./random.sh 7

Welcome to Ubuntu, one component of the GNU/Linux operating system.

Basic Tutorial Learn basic keyboard commands (readme Emacs)
Emacs Quickstart Overview of Emacs features at gnu.org
View Emacs Manual View the Emacs manual using info
About Emacs GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Emacs Conditions for redistributing and changing Emacs
Getting Manuals Purchasing printed copies of manuals
To use a partially formatted command, type Ctrl+q

This is GNU Emacs 27.1 (build 1, x86_64-pc-linux-gnu, GTK+ Version 3.24.22, cairo version 1.18.0)
Copyright © 2000 Free Software Foundation, Inc.
Consult this screen to know how to report bugs and how to request new features.

gnome@ubuntu:~\$./random.sh 15

gnome@ubuntu:~\$

Figure 3.13: Скрипт №3

Далее я проверила работу написанного скрипта (команды «./random.sh 7»и «./random.sh 15»), предварительно добавив право на исполнение файла (команда «chmod +x random.sh») (Скриншот 3.14). Скрипт работает корректно.

```
gnome@ubuntu:~$ chmod +x random.sh
gnome@ubuntu:~$ ./random.sh 7
gnome@ubuntu:~$ ./random.sh 15
gnome@ubuntu:~$
```

Figure 3.14: Проверка работы скрипта

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="${VAR1} ${VAR2}"  
  
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 is 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` 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.

4 Библиография

1. Программное обеспечение GNU/Linux. Лекция 7. Вопросы лицензирования (Г. Курячий, МГУ);
2. Программное обеспечение GNU/Linux. Лекция 9. Хранилище и дистрибутив (Г. Курячий, МГУ);
3. Программное обеспечение GNU/Linux. Лекция 10. Минимальный набор знаний (Г. Курячий, МГУ);
4. Электронный ресурс: <https://infopedia.su/24x10498.html>
5. Электронный ресурс: <http://5fan.ru/wie>

5 Выводы

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