

Отчёт по лабораторной работе №13

Дисциплина: Операционные системы

Джеффри Родригес Сантос

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
4	Контрольные вопросы	18
5	Выводы	21

Список таблиц

1 Цель работы

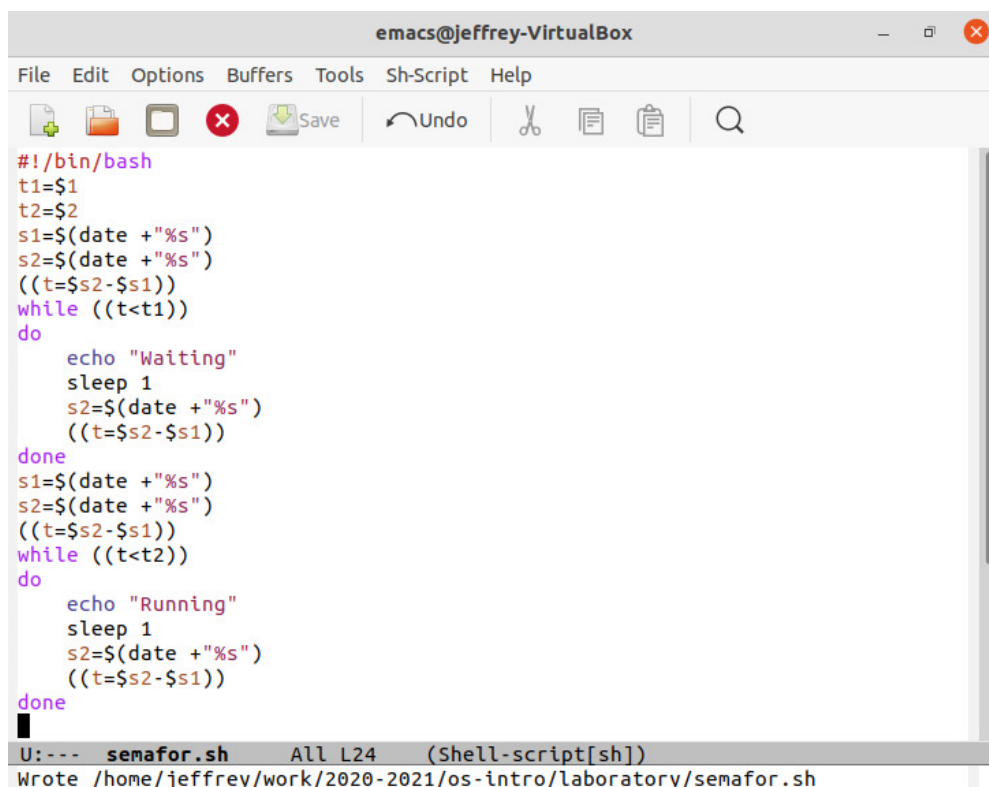
Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

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 фоновом, а в привилегированном режиме. Доработать программу так, чтобы имелась возможность взаимодействия трёх и более процессов.
2. Реализовать команду `man` с помощью командного файла. Изучите содержимое каталога `/usr/share/man/man1`. В нем 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 псевдослучайные числа в диапазоне от 0 до 32767.

3 Выполнение лабораторной работы

1. Написал 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 создал файл: `semafor.sh` и написал соответствующий скрипт (рис. -fig. 3.1).



```
#!/bin/bash
t1=$1
t2=$2
s1=$(date +%s)
s2=$(date +%s)
((t=$s2-$s1))
while ((t<t1))
do
    echo "Waiting"
    sleep 1
    s2=$(date +%s)
    ((t=$s2-$s1))
done
s1=$(date +%s)
s2=$(date +%s)
((t=$s2-$s1))
while ((t<t2))
do
    echo "Running"
    sleep 1
    s2=$(date +%s)
    ((t=$s2-$s1))
done
```

U: --- semafor.sh All L24 (Shell-script[sh])
Wrote /home/jeffrey/work/2020-2021/os-intro/laboratory/semafor.sh

Рис. 3.1: Создал файл и написал первый скрипт

Далее я проверил работу написанного скрипта (команда «`./semafor.sh 3 5`»), предварительно добавив право на исполнение файла (команда «`chmod +x semafor.sh`») (рис. -fig. 3.2). Скрипт работает корректно.

```

jeffrey@jeffrey-VirtualBox:~/work/2020-2021/04-Intro/Laboratory$ chmod +x semafor.sh
jeffrey@jeffrey-VirtualBox:~/work/2020-2021/04-Intro/Laboratory$ ./semafor.sh
jeffrey@jeffrey-VirtualBox:~/work/2020-2021/04-Intro/Laboratory$ ./semafor.sh 3
5
Waiting
Waiting
Waiting
Running
Running
Running
Running
jeffrey@jeffrey-VirtualBox:~/work/2020-2021/04-Intro/Laboratory$

```

Рис. 3.2: Проверил первый скрипт

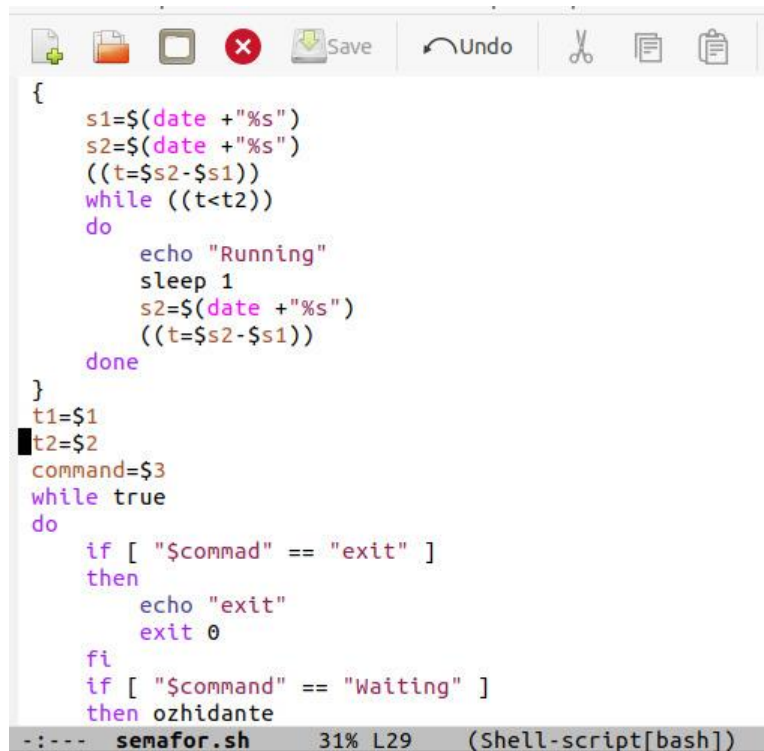
После этого я изменил скрипт так, чтобы его можно было выполнять в нескольких терминалах (рис. -fig. 3.3) (рис. -fig. 3.4) и проверил его работу (например, команда «./semafor.sh 2 4 Ожидание > /dev/pts/1 &»).

```

emacs@jeffrey-VirtualBox
File Edit Options Buffers Tools Sh-Script Help
+ Save Undo
#!/bin/bash
function ozhidate
{
    s1=$(date +%s")
    s2=$(date +%s")
    ((t=$s2-$s1))
    while ((t<t1))
    do
        echo "Waiting"
        sleep 1
        s2=$(date +%s")
        ((t=$s2-$s1))
    done
}
function vipolnenie
{
    s1=$(date +%s")
    s2=$(date +%s")
    ((t=$s2-$s1))
    while ((t<t2))
    do
        echo "Running"
        sleep 1
        s2=$(date +%s")
    done
}
U:--- semafor.sh Top L24 (Shell-script[sh])
Beginning of buffer

```

Рис. 3.3: изменил первый скрипт



```
{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=s2-s1))
    while ((t<t2))
    do
        echo "Running"
        sleep 1
        s2=$(date +%s)
        ((t=s2-s1))
    done
}
t1=$1
t2=$2
command=$3
while true
do
    if [ "$command" == "exit" ]
    then
        echo "exit"
        exit 0
    fi
    if [ "$command" == "Waiting" ]
    then ozhidante
    fi
done
```

-:--- semafor.sh 31% L29 (Shell-script[bash])



```
File Edit Options Buffers Tools Sh-Script Help
while true
do
    if [ "$command" == "exit" ]
    then
        echo "exit"
        exit 0
    fi
    if [ "$command" == "Waiting" ]
    then ozhidante
    fi
    if [ "$command" == "Running" ]
    then vipolnenie
    fi
    echo "Next action"
    read command
done
```

-:--- semafor.sh Bot L31 (Shell-scri

Рис. 3.4: Изменил первый скрипт

Но ни одна команда не работала, так как мне было “Отказано в доступе” (рис.-fig. 3.5). При этом скрипт работает корректно (команда «./semafor.sh 2 4 Ожидание»).

```

jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ ./semafor.sh 2
4 Waiting > &
bash: syntax error near unexpected token `&'
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ sudo ./semafor
.sh 2 4 Waiting > /dev/pts/1 &
[1] 2933
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ ./semafor.sg 2
4 Waiting > /dev/pts/1 &
[2] 2940

[1]+  Stopped                  sudo ./semafor.sh 2 4 Waiting > /dev/pts/1
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ bash: ./semafo
r.sg: No such file or directory

[2]-  Exit 127                  ./semafor.sg 2 4 Waiting > /dev/pts/1
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ ./semafor.sg 3
4 Waiting > /dev/pts/1 &
[2] 2942
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ bash: ./semafo
r.sg: No such file or directory

[2]-  Exit 127                  ./semafor.sg 3 4 Waiting > /dev/pts/1
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ ./semafor.sg 2
5 Running > /dev/pts/1 &
[2] 2943

jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ ./semafor.sg 3
4 Waiting > /dev/pts/1 &
[2] 2942
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ bash: ./semafo
r.sg: No such file or directory

[2]-  Exit 127                  ./semafor.sg 3 4 Waiting > /dev/pts/1
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ ./semafor.sg 2
5 Running > /dev/pts/1 &
[2] 2943
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ bash: ./semafo
r.sg: No such file or directory

[2]-  Exit 127                  ./semafor.sg 2 5 Running > /dev/pts/1
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ ./semafor.sg 3
5 exit > /dev/pts/1 &
[2] 2945
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ bash: ./semafo
r.sg: No such file or directory

[2]-  Exit 127                  ./semafor.sg 3 5 exit > /dev/pts/1
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ ./semafor.sg 2
6 Running > /dev/pts/3 &
[2] 2947
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ bash: /dev/pts
/3: Permission denied

[2]-  Exit 1                    ./semafor.sg 2 6 Running > /dev/pts/3
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$

```

Рис. 3.5: проверил первый скрипт повторно

2. Реализовал команду man с помощью командного файла. Изучил содержимое каталога /usr/share/man/man1 (рис. -fig. 3.6). В нем 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.

```
jeffrey@jeffrey-VirtualBox:~/work/2020-2021/oa-intro/laboratory$ cd /usr/share/
man/man1
jeffrey@jeffrey-VirtualBox:/usr/share/man/man1$ ls
'.1.gz'
aa-enabled.1.gz
aa-exec.1.gz
aconnect.1.gz
add-apt-repository.1.gz
addr2line.1.gz
alsabat.1.gz
alsactl.1.gz
alsaloop.1.gz
alsamixer.1.gz
alsatplg.1.gz
alsaucm.1.gz
amidi.1.gz
amixer.1.gz
apg.1.gz
apgbfm.1.gz
aplay.1.gz
aplaymidi.1.gz
apport-bug.1.gz
apport-cli.1.gz
apport-collect.1.gz
apport-unpack.1.gz
appres.1.gz
appstreamcli.1.gz
apropos.1.gz
apt-add-repository.1.gz
```

Рис. 3.6: Содержимое каталога /usr/share/man/man1

Для данной задачи я создал файл: man.sh и написал соответствующий скрипт (рис. -fig. 3.7).

```
emacs@jeffrey-VirtualBox
File Edit Options Buffers Tools Sh-Script Help
[Icons: New, Open, Save, Undo, Cut, Copy, Paste, Find]
#!/bin/bash
a=$1
if [ -f /usr/share/man/man1/$a.1.gz ]
then
    gunzip -c /usr/share/man/man1/$1.1.gz | less
else
    echo "There is no help for this command"
fi
```

Рис. 3.7: Создал файл и написал второй скрипт

Далее я проверил работу написанного скрипта (команды «./man.sh mkdir», «./man.sh rm» и «./man.sh cat»), предварительно добавив право на исполнение файла (команда «chmod +x man.sh») (рис. -fig. 3.8) (рис. -fig. 3.9) (рис. -fig. 3.10).

Скрипт работает корректно.

```
jeffrey@jeffrey-VirtualBox:~/work/2020-2021/04-Intro/Laboratory$ chmod +x man.sh
jeffrey@jeffrey-VirtualBox:~/work/2020-2021/04-Intro/Laboratory$ ./man.sh mkdir
jeffrey@jeffrey-VirtualBox:~/work/2020-2021/04-Intro/Laboratory$ ./man.sh rm
jeffrey@jeffrey-VirtualBox:~/work/2020-2021/04-Intro/Laboratory$ ./man.sh cat
```

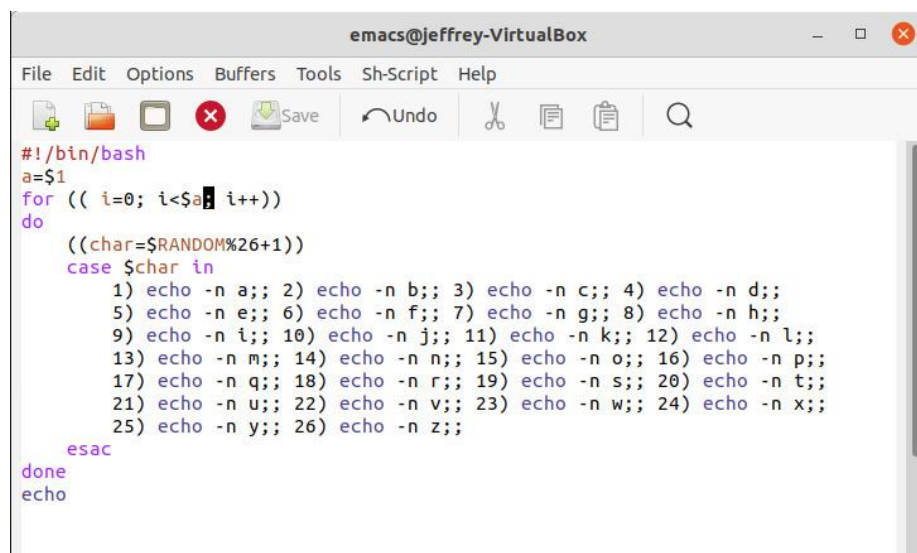
Рис. 3.8: Проверил второй скрипт

```
.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.47.3.
.TH MKDIR "1" "September 2019" "GNU coreutils 8.30" "User Commands"
.SH NAME
mkdir \- make directories
.SH SYNOPSIS
.B mkdir
[\\fI\\,OPTION\\fR]... \\fI\\,DIRECTORY\\fR...
.SH DESCRIPTION
.\" Add any additional description here
.PP
Create the DIRECTORY(ies), if they do not already exist.
.PP
Mandatory arguments to long options are mandatory for short options too.
.TP
\\fB\\-m\\fR, \\fB\\-\\-mode\\fR=\\fI\\,MODE\\fR
set file mode (as in chmod), not a=rwx \\- umask
.TP
\\fB\\-p\\fR, \\fB\\-\\-parents\\fR
no error if existing, make parent directories as needed
.TP
\\fB\\-v\\fR, \\fB\\-\\-verbose\\fR
print a message for each created directory
.TP
\\fB\\-Z\\fR
set SELinux security context of each created directory
to the default type
.TP
\\fB\\-\\-context\\fR=\\fI\\,CTX\\fR
:
```

```
.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.47.3.
.TH RM "1" "September 2019" "GNU coreutils 8.30" "User Commands"
.SH NAME
rm \- remove files or directories
.SH SYNOPSIS
.B rm
[\fI\,OPTION\|\fR]... [\fI\,FILE\|\fR]...
.SH DESCRIPTION
This manual page
documents the GNU version of
.BR rm .
.B rm
removes each specified file. By default, it does not remove
directories.
.P
If the \fI\-I\|fR or \fI\-\-interactive=once\|fR option is given,
and there are more than three files or the \fI\-r\|fR, \fI\-R\|fR,
or \fI\-\-recursive\|fR are given, then
.B rm
prompts the user for whether to proceed with the entire operation. If
the response is not affirmative, the entire command is aborted.
.P
Otherwise, if a file is unwritable, standard input is a terminal, and
the \fI\-f\|fR or \fI\-\-force\|fR option is not given, or the
\fI\-i\|fR or \fI\-\-interactive=always\|fR option is given,
.B rm
prompts the user for whether to remove the file. If the response is
not affirmative, the file is skipped.
:█
```

Рис. 3.10: Проверил второй скрипт

3. Используя встроенную переменную \$RANDOM, написал командный файл, генерирующий случайную последовательность букв латинского алфавита. Для данной задачи я создал файл: random.sh и написал соответствующий скрипт (рис. -fig.3.11).



```
#!/bin/bash
a=$1
for (( i=0; i<$a; 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
```

Рис. 3.11: Создал файл и написал третий скрипт

Далее я проверил работу написанного скрипта (команда «./random.sh 15»), предварительно добавив право на исполнение файла (команда «chmod +x random.sh») (рис. -fig. 3.12). Скрипт работает корректно.

```
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ chmod +x random.sh
jeffrey@jeffrey-VirtualBox: ~/work/2020-2021/os-intro/laboratory$ ./random.sh 15
atwdvphiuxgsyl
```

Рис. 3.12: Проверил третий скрипт

4 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,"
```

```
VAR3="VAR1VAR2"  
VAR2="World"  
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` 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

5 Выводы

В ходе выполнения данной лабораторной работы я изучил основы программирования в оболочке ОС UNIX, а также научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.