

Отчет по лабораторной работе №13

Дисциплина: Операционные системы

Морозова Анастасия Владимировна

Содержание

1	Цель работы	4
2	Задачи	5
3	Выполнение лабораторной работы	6
4	Контрольные вопросы	15
5	Выводы	19
6	Библиография	20

Список иллюстраций

3.1	Написание первого скрипта	7
3.2	Проверка работы скрипта	8
3.3	Измененный скрипт	9
3.4	Измененный скрипт	10
3.5	Проверка работы скрипта	10
3.6	Содержимое каталога	11
3.7	Написание второго скрипта	12
3.8	Проверка работы скрипта	12
3.9	Справка ls	13
3.10	Справка mkdir	13
3.11	Написание третьего скрипта	14
3.12	Проверка работы скрипта	14

1 Цель работы

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

2 Задачи

1. Ознакомиться с логическими управляющими конструкций и циклов.
2. Написать 3 командных файла в ходе работы.
3. Выполнить отчет.

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). Создала файл `ex1.sh` и написала скрипт. (рис. -fig. 3.1)



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

U: *- ex1.sh All L12 (S

Рис. 3.1: Написание первого скрипта

Проверила работу скрипта (команда ./ex1.sh 2 6), добавив право на исполнение файла (команда chmod +x ex1.sh). Скрипт работает корректно. (рис. -fig. 3.2):

```
avmogozova@avmogozova:~$ touch ex1.sh
avmogozova@avmogozova:~$ emacs &
[1] 3605
avmogozova@avmogozova:~$ chmod +x ex1.sh
avmogozova@avmogozova:~$ ./ex1.sh 2 6
Ожидание
Ожидание
Выполнено
Выполнено
Выполнено
Выполнено
Выполнено
Выполнено
avmogozova@avmogozova:~$
```

Рис. 3.2: Проверка работы скрипта

Далее изменила скрипт так, чтобы его можно было выполнять в нескольких терминалах и проверила его работу (команда `./ex1.sh 2 5` Выполнение `> /dev/pts/2` & или команда `./ex1.sh 3 6` Выполнение `> /dev/tty2`). Но команды не сработали, выводя сообщение об отказе в доступе. При этом скрипт работает корректно (команда `./ex1.sh 3 6`). (рис. -fig. 3.3) (рис. -fig. 3.4)(рис. -fig. 3.5)


```
#!/bin/bash
function waiting{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=s2-s1))
    while ((t<t1))
    do
        echo "Ожидание"
        sleep 1
        s2=$(date +%s)
        ((t=s2-s1))
    done
}
function executing {
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=s2-s1))
    while ((t<t2))
    do
        echo "Выполнено"
        sleep 1
        s2=$(date +%s)
        ((t=s2-s1))
    done
}
```

Рис. 3.3: Измененный скрипт

```

}
t1=$1
t2=$2
command=$3
while true
do
    if [ "$command" == "Выход" ]
    then
        echo "Выход"
        exit 0
    fi
    if [ "$command" == "Ожидание" ]
    then waiting
    fi
    if [ "$command" == "Выполнение" ]
    then executing
    fi
    echo "Следующее действие: "
    read command
done

```

Рис. 3.4: Измененный скрипт

```

avmorofova@avmorofova:~$ ./ex1.sh 3 6 Выполнение>/dev/pts/2
bash: /dev/pts/2: Permission denied
avmorofova@avmorofova:~$ ./ex1/sh 3 6
bash: ./ex1/sh: No such file or directory
avmorofova@avmorofova:~$ ./ex1.sh 3 6
Следующее действие:
Выполнение
Выполнено
Выполнено
Выполнено
Выполнено
Выполнено
Выполнено
Следующее действие:
Ожидание
Ожидание
Ожидание
Ожидание
Следующее действие:
Выход
Выход
avmorofova@avmorofova:~$ █

```

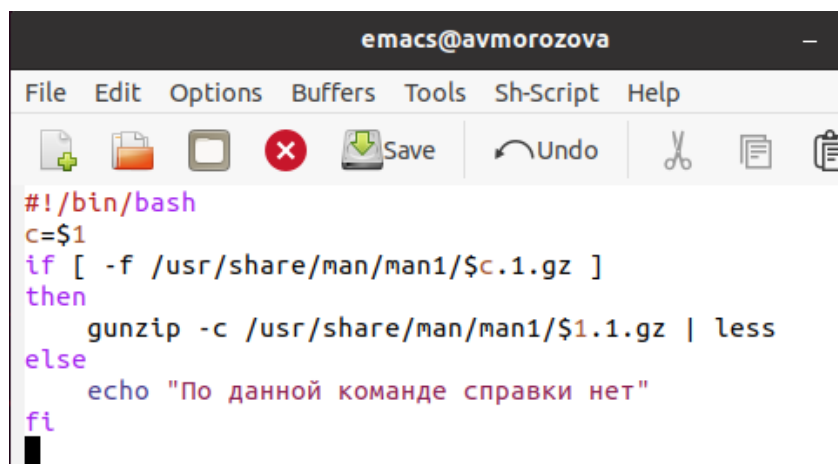
Рис. 3.5: Проверка работы скрипта

- 2) Реализовала команду `man` с помощью командного файла. Изучила 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`. (рис. -fig. 3.6)

```
avmorofova@avmorofova:~$ cd /usr/share/man/man1
avmorofova@avmorofova:/usr/share/man/man1$ ls
' [.1.gz'
411toppm.1.gz      mkindex.1.gz
a2ping.1.gz        mkisofs.1.gz
a5booklet.1.gz     mkjobtexmf.1.gz
aa-enabled.1.gz    mkmanifest.1.gz
aa-exec.1.gz       mknod.1.gz
aconnect.1.gz      mkocp.1.gz
add-apt-repository.1.gz
addr2line.1.gz     mkofm.1.gz
afm2afm.1.gz       mksquashfs.1.gz
afm2pl.1.gz        mktemp.1.gz
afm2tfm.1.gz       mktexfmt.1.gz
aleph.1.gz         mktexlsr.1.gz
allcm.1.gz         mktexmf.1.gz
alleg.1.gz         mktexpk.1.gz
allneeded.1.gz     mktextfm.1.gz
alsabat.1.gz       mkzftree.1.gz
alsactl.1.gz       mlabel.1.gz
alsaloop.1.gz      mllatex.1.gz
alsamixer.1.gz     mltex.1.gz
alsatplg.1.gz      mnafm.1.gz
alsaucm.1.gz       mmcli.1.gz
amidi.1.gz         mmd.1.gz
amixer.1.gz        mmount.1.gz
amstex.1.gz        mmove.1.gz
animate.1.gz       mmpfb.1.gz
animate-im6.1.gz   mogrify.1.gz
                   mogrify-im6.1.gz
                   mogrify-im6.q16.1.gz
```

Рис. 3.6: Содержимое каталога

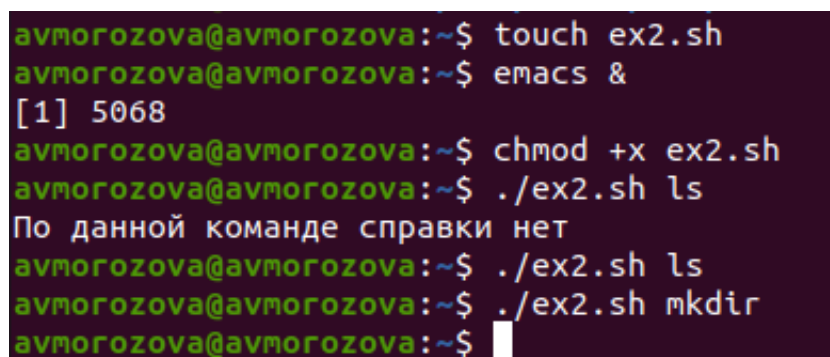
Создала файл `ex2.sh` и написала соответствующий скрипт. (рис. -fig. 3.7)



```
#!/bin/bash
c=$1
if [ -f /usr/share/man/man1/$c.1.gz ]
then
    gunzip -c /usr/share/man/man1/$1.1.gz | less
else
    echo "По данной команде справки нет"
fi
```

Рис. 3.7: Написание второго скрипта

Проверила работу написанного скрипта (команды `./ex2.sh ls`, `./ex2.sh mkdir` и т.д.), предварительно добавив право на исполнение файла (команда `chmod +x ex2.sh`). Скрипт сработал. (рис. -fig. 3.8)(рис. -fig. 3.9)(рис. -fig. 3.10)



```
avmorofova@avmorofova:~$ touch ex2.sh
avmorofova@avmorofova:~$ emacs &
[1] 5068
avmorofova@avmorofova:~$ chmod +x ex2.sh
avmorofova@avmorofova:~$ ./ex2.sh ls
По данной команде справки нет
avmorofova@avmorofova:~$ ./ex2.sh ls
avmorofova@avmorofova:~$ ./ex2.sh mkdir
avmorofova@avmorofova:~$
```

Рис. 3.8: Проверка работы скрипта

```
avmorofova@avmorofova: ~
.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.47.3.
.TH LS "1" "September 2019" "GNU coreutils 8.30" "User Commands"
.SH NAME
ls \- list directory contents
.SH SYNOPSIS
.B ls
[\\fI\\,OPTION\\/\\fR]... [\\fI\\,FILE\\/\\fR]...
.SH DESCRIPTION
.\" Add any additional description here
.PP
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of \\fB\\-cftuvSUX\\fR nor \\fB\\-\\-sort\\fR is s
pecified.
.PP
Mandatory arguments to long options are mandatory for short options too.
.TP
\\fB\\-a\\fR, \\fB\\-\\-all\\fR
do not ignore entries starting with .
.TP
\\fB\\-A\\fR, \\fB\\-\\-almost\\-all\\fR
do not list implied . and ..
.TP
\\fB\\-\\-author\\fR
with \\fB\\-l\\fR, print the author of each file
.TP
\\fB\\-b\\fR, \\fB\\-\\-escape\\fR
print C\\-style escapes for nongraphic characters
.TP
:
```

Рис. 3.9: Справка ls

```
avmorofova@avmorofova: ~
.\" 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]
:
```

Рис. 3.10: Справка mkdir

3) Используя встроенную переменную \$RANDOM, написала командный файл,

генерирующий случайную последовательность букв латинского алфавита.

Создала файл ex3.sh и написала соответствующий скрипт. (рис. -fig. 3.11)

```
#!/bin/bash
k=$1
for ((i=0; i<$k; 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: Написание третьего скрипта

Проверила работу написанного скрипта (команды ./ex3.sh 6, ./ex3.sh 23 и ./ex3.sh 14), предварительно добавив право на исполнение файла (команда «chmod +x ex3.sh»). Скрипт работает корректно. (рис. -fig. 3.12)

```
avmorofova@avmorofova:~$ touch ex3.sh
avmorofova@avmorofova:~$ emacs &
[2] 5492
avmorofova@avmorofova:~$ chmod +x ex3.sh
avmorofova@avmorofova:~$ ./ex3.sh 6
xyenmo
avmorofova@avmorofova:~$ ./ex3.sh 23
uyffdrctybdglopfdphscktu
avmorofova@avmorofova:~$ ./ex3.sh 14
uoszflllfsujmsh
avmorofova@avmorofova:~$
```

Рис. 3.12: Проверка работы скрипта

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,
```

```
"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 `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.

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.

6 Библиография

1. https://esystem.rudn.ru/pluginfile.php/1142096/mod_resource/content/2/010-lab_shell_prog_3.pdf
2. Кулябов Д.С. Операционные системы: лабораторные работы: учебное пособие / Д.С. Кулябов, М.Н. Геворкян, А.В. Королькова, А.В. Демидова. — М. : Изд-во РУДН, 2016. — 117 с. — ISBN 978-5-209-07626-1 : 139.13; То же [Электронный ресурс]. — URL: <http://lib.rudn.ru/MegaPro2/Download/MObject/6118>.
3. Робачевский А.М. Операционная система UNIX [текст] : Учебное пособие / А.М. Робачевский, С.А. Немнюгин, О.Л. Стесик. — 2-е изд., перераб. и доп. — СПб. : БХВ-Петербург, 2005, 2010. — 656 с. : ил. — ISBN 5-94157-538-6 : 164.56. (ЕТ 60)
4. Таненбаум Эндрю. Современные операционные системы [Текст] / Э. Таненбаум. — 2-е изд. — СПб. : Питер, 2006. — 1038 с. : ил. — (Классика Computer Science). — ISBN 5-318-00299-4 : 446.05. (ЕТ 50)