

Отчет по лабораторной работе №13

Дисциплина: Операционные системы

Пермякова Елизавета Евгеньевна

Содержание

1	Цель работы	4
2	Задачи	5
3	Выполнение лабораторной работы	6
4	Контрольные вопросы	14
5	Выводы	18
6	Библиография	19

List of Figures

3.1	Первый скрипт	7
3.2	Проверка работы скрипта	8
3.3	Измененный скрипт	9
3.4	Проверка работы скрипта	10
3.5	Каталог /usr/share/man/man1	10
3.6	Содержимое каталога	11
3.7	Второй скрипт	11
3.8	Проверка работы скрипта	12
3.9	Третий скрипт	12
3.10	Проверка работы скрипта	13

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. Познакомиться с логическими управляющими конструкций и циклов.
2. В ходе работы написать 3 командных файла.
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). Для данной задачи я создала файл `prog1.sh` и написала соответствующий скрипт. (рис. 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
```

Figure 3.1: Первый скрипт

Далее я проверила работу написанного скрипта (команда «./prog1.sh 2 6»), предварительно добавив право на исполнение файла (команда «chmod +x prog1.sh»). Скрипт работает корректно. (рис. 3.2):

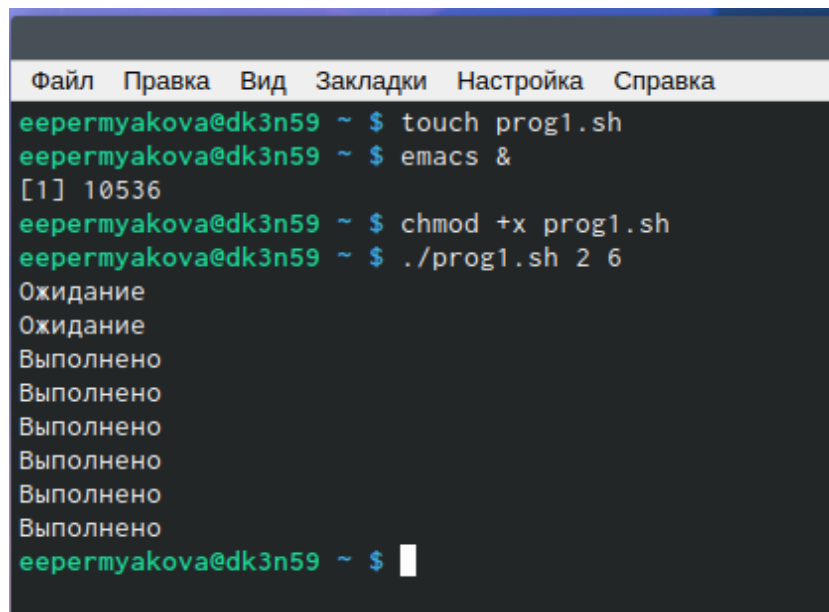
A screenshot of a terminal window with a dark background and light-colored text. At the top, there is a menu bar with the following items: "Файл", "Правка", "Вид", "Закладки", "Настройка", and "Справка". Below the menu bar, the terminal shows a series of commands and their outputs. The user's prompt is "ееермыakova@dk3n59 ~ \$". The commands and outputs are: 1. Command: "touch prog1.sh". 2. Command: "emacs &". Output: "[1] 10536". 3. Command: "chmod +x prog1.sh". 4. Command: "./prog1.sh 2 6". Output: "Ожидание". 5. Output: "Ожидание". 6. Output: "Выполнено". 7. Output: "Выполнено". 8. Output: "Выполнено". 9. Output: "Выполнено". 10. Output: "Выполнено". 11. The prompt "ееермыakova@dk3n59 ~ \$" is shown again with a cursor. The terminal window has a title bar at the top, and the menu bar is visible. The text is in a monospaced font.

Figure 3.2: Проверка работы скрипта

После этого я изменила скрипт так, чтобы его можно было выполнять в нескольких терминалах и проверила его работу (например, команда «./prog1.sh 2 5 Выполнение > /dev/pts/2 &» или команда «./prog1.sh 2 5 Выполнение > /dev/tty2»). Но ни одна команда не работала, выводя сообщение “Отказано в доступе”. При этом скрипт работает корректно (команда «./prog1.sh 3 6»). (рис. 3.3) (рис. 3.4)


```
Приложения  Места  GNU Emacs
File Edit Options Buffers Tools Sh-Script Help
#!/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-$s1))
    while ((t<t2))
    do
        echo "Выполнение"
        sleep 1
        s2=$(date +%s)
        ((t=$s2-$s1))
    done
}
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
U:--- prog1.sh All L45 (Shell-script[sh]) Ср мая 26 12:13
Beginning of buffer
```

Figure 3.3: Измененный скрипт

```

eeperryakova@dk3n59 ~ $ ./prog1.sh 3 6 > /dev/tty2
bash: /dev/tty2: Отказано в доступе
eeperryakova@dk3n59 ~ $ ./prog1.sh 3 6 > /dev/pts/1 &
[2] 18093
eeperryakova@dk3n59 ~ $ bash: /dev/pts/1: Отказано в доступе
^C
[2]+  Выход 1          ./prog1.sh 3 6 > /dev/pts/1
eeperryakova@dk3n59 ~ $ ./prog1.sh 3 6
Следующее действие:
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
Следующее действие:
J;blfybt
Следующее действие:
Ожидание
Ожидание
Ожидание
Ожидание
Следующее действие:
Выход
Выход
eeperryakova@dk3n59 ~ $ ./prog1.sh 2 5 Выполнение > /dev/pts/2 &
[2] 18194
eeperryakova@dk3n59 ~ $ bash: /dev/pts/2: Отказано в доступе
^C
[2]+  Выход 1          ./prog1.sh 2 5 Выполнение > /dev/pts/2
eeperryakova@dk3n59 ~ $ ./prog1.sh 2 5 Выполнение > /dev/tty2
bash: /dev/tty2: Отказано в доступе

```

Figure 3.4: Проверка работы скрипта

- 2) Реализовала команду mancs 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. 3.5) (рис. 3.6)

```

eeperryakova@dk3n59 ~ $ cd /usr/share/man/man1
eeperryakova@dk3n59 /usr/share/man/man1 $ ls

```

Figure 3.5: Каталог /usr/share/man/man1

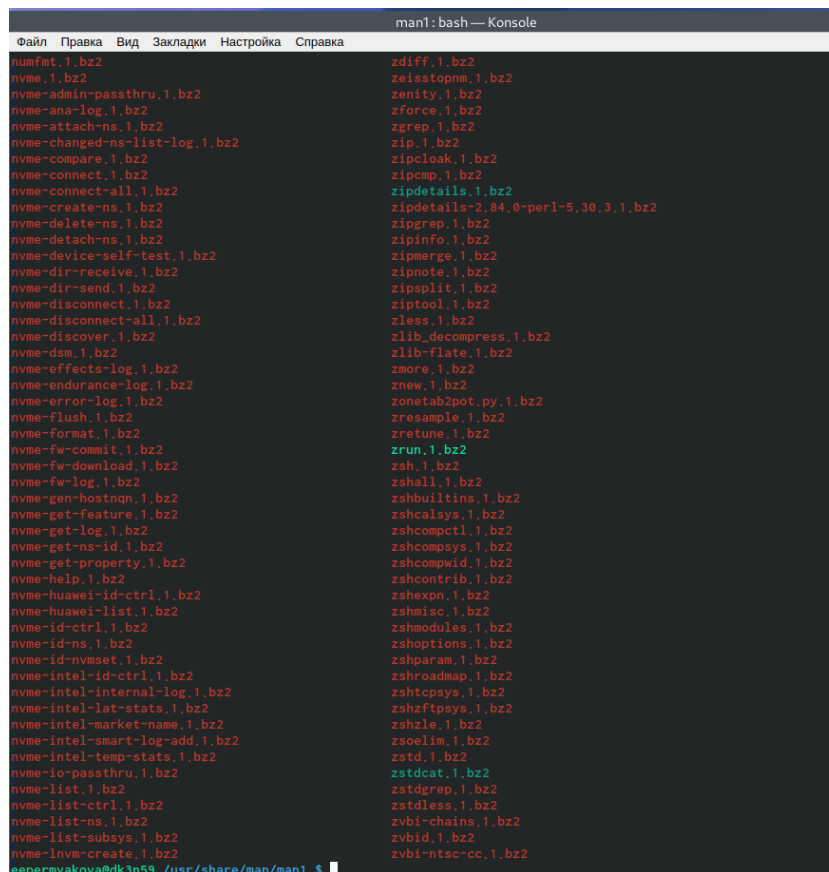


Figure 3.6: Содержимое каталога

Для данной задачи я создала файл prog2.sh и написала соответствующий скрипт. (рис. 3.7)

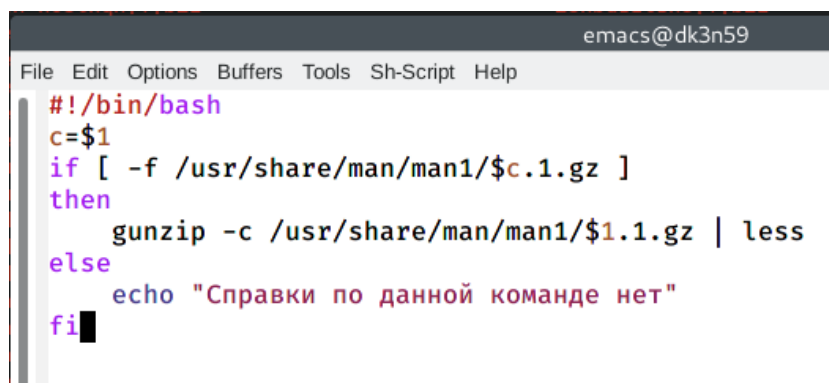


Figure 3.7: Второй скрипт

Далее я проверила работу написанного скрипта(команды «./prog2.sh ls»,

«./prog2.sh mkdir» и т. д.), предварительно добавив право на исполнение файла (команда «chmod +x prog2.sh»). Скрипт сработал и вывел, что по данным командам справок нет. (рис. 3.8)

```
eeperryakova@dk3n59 ~ $ chmod +x prog2.sh
eeperryakova@dk3n59 ~ $ ./prog2.sh ls
Справки по данной команде нет
eeperryakova@dk3n59 ~ $ ./prog2.sh mkdir
Справки по данной команде нет
eeperryakova@dk3n59 ~ $ ./prog2.sh cd
Справки по данной команде нет
eeperryakova@dk3n59 ~ $ ./prog2.sh make
Справки по данной команде нет
eeperryakova@dk3n59 ~ $ ./prog2.sh ls
Справки по данной команде нет
```

Figure 3.8: Проверка работы скрипта

3) Используя встроенную переменную \$RANDOM, написала командный файл, генерирующий случайную последовательность букв латинского алфавита.

Для данной задачи я создала файл prog3.sh и написала соответствующий скрипт. (рис. 3.9)

```
File Edit Options Buffers Tools SH-Script Help
#!/bin/bash
i=$1
for ((i=0; i<$1; i++))
do
  (( char=$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
```

Figure 3.9: Третий скрипт

Далее я проверила работу написанного скрипта (команды «./prog3.sh 4», «./prog3.sh 23» и «./prog3.sh 26»), предварительно добавив право на исполнение файла (команда «chmod +x prog3.sh»). Скрипт работает корректно. (рис. 3.10)

```
[3] 20573
eepermyakova@dk3n59 ~ $ chmod +x prog3.sh
eepermyakova@dk3n59 ~ $ ./prog3.sh 4
pukyeeepermyakova@dk3n59 ~ $ ./prog3.sh 3
msm
eepermyakova@dk3n59 ~ $ ./prog3.sh 23
zkiqbpwfqzonjkemazdkep
eepermyakova@dk3n59 ~ $ ./prog3.sh 26
bnmiucmoqscfmgqajxxmdlwadg
eepermyakova@dk3n59 ~ $
```

Figure 3.10: 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)