

# **Лабораторная работа №12**

**Дисциплина: Операционные системы**

Алиева Милена Арифовна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>8</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
<b>5</b>	<b>Ответы на контрольные вопросы</b>	<b>13</b>
<b>6</b>	<b>Выводы</b>	<b>16</b>

## Список иллюстраций

4.1	Командный файл . . . . .	9
4.2	Запуск командного файла . . . . .	10
4.3	Содержимое каталога /usr/share/man/man1 . . . . .	10
4.4	Командный файл . . . . .	11
4.5	Запуск командного файла . . . . .	11
4.6	Командный файл . . . . .	12
4.7	Запуск файла ко третьему заданию . . . . .	12

## 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.

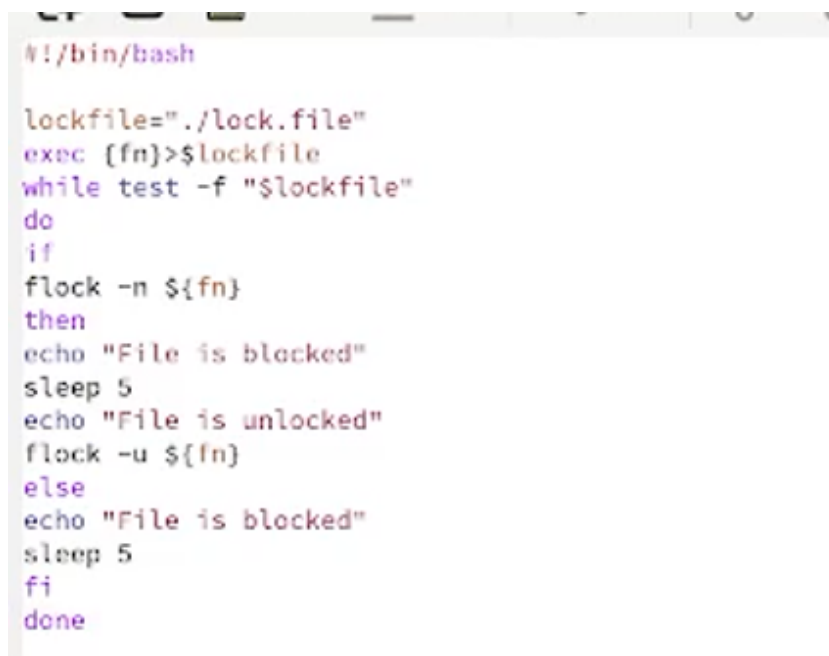
### 3 Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: – оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций; – C-оболочка (или csh) — надстройка на оболочке Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; – оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; – BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation). POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна



## 4 Выполнение лабораторной работы

1. Написали командный файл, реализующий упрощённый механизм семафоров (рис. [4.1])



```
#!/bin/bash

lockfile="./lock.file"
exec {fn}>$lockfile
while test -f "$lockfile"
do
if
flock -n ${fn}
then
echo "File is blocked"
sleep 5
echo "File is unlocked"
flock -u ${fn}
else
echo "File is blocked"
sleep 5
fi
done
```

Рис. 4.1: Командный файл

Запустили командный файл (рис. [4.2])

```

[maalieva@fedora ~]$ bash prog1.sh
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
^C
[maalieva@fedora ~]$

```

Рис. 4.2: Запуск командного файла

2. Изучили содержимое каталога /usr/share/man/man1 (рис. [4.3])

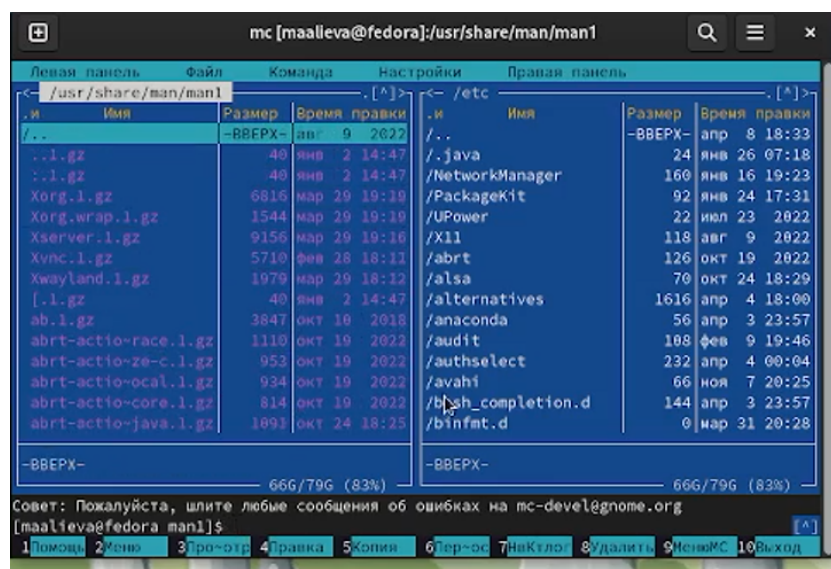


Рис. 4.3: Содержимое каталога /usr/share/man/man1

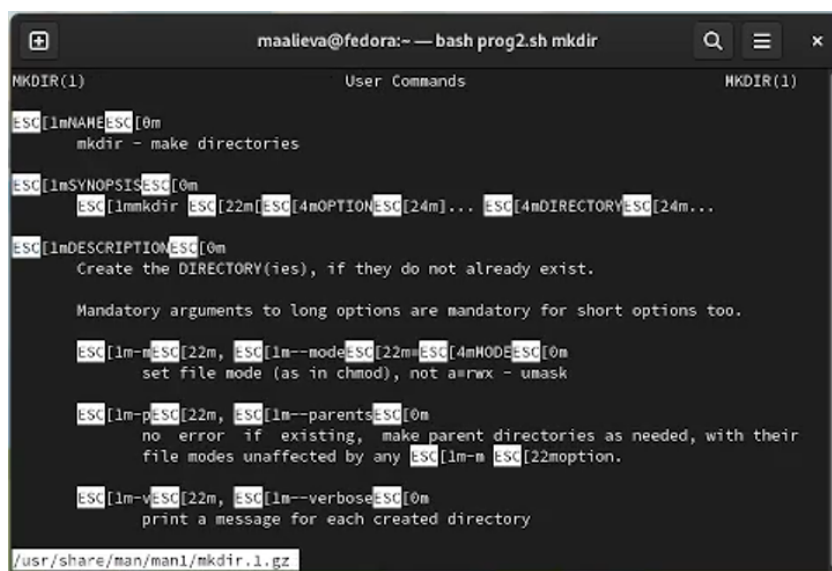
Реализовали команду man с помощью командного файла (рис. [4.4])

```
#!/bin/bash

a=$1
if test -f "/usr/share/man/man1/$a.1.gz"
then
less /usr/share/man/man1/$a.1.gz
else
echo "Нет такой команды"
fi
```

Рис. 4.4: Командный файл

Запустили командный файл, посмотрели команду mkdir (рис. [4.5])



```
maalleva@fedora:~ — bash prog2.sh mkdir
MKDIR(1) User Commands MKDIR(1)

ESC[1mNAMEESC[0m
mkdir - make directories

ESC[1mSYNOPSISESC[0m
ESC[1mmkdir ESC[22mESC[4mOPTIONESC[24m... ESC[4mDIRECTORYESC[24m...

ESC[1mDESCRIPTIONESC[0m
Create the DIRECTORY(ies), if they do not already exist.

Mandatory arguments to long options are mandatory for short options too.

ESC[1m-mESC[22m, ESC[1m--modeESC[22mESC[4mMODEESC[0m
set file mode (as in chmod), not a=rwx - umask

ESC[1m-pESC[22m, ESC[1m--parentsESC[0m
no error if existing, make parent directories as needed, with their
file modes unaffected by any ESC[1m-m ESC[22moption.

ESC[1m-vESC[22m, ESC[1m--verboseESC[0m
print a message for each created directory

/usr/share/man/man1/mkdir.1.gz
```

Рис. 4.5: Запуск командного файла

3. Используя встроенную переменную \$RANDOM, написали командный файл, генерирующий случайную последовательность букв латинского алфавита (рис. [4.6])

```
#!/bin/bash
a=1
for ((i=0; i<80; 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
===== prog3.sh All 11 (Shell=script(bash))
Welcome to GNU Emacs, one component of the GNU/Linux operating system.

Emacs Tutorial Learn basic keyboard commands of Emacs Emacs
Emacs Quick Tour Overview of Emacs features at gnu.org
View the Emacs manual using info
GNU Emacs comes with ABSOLUTELY NO WARRANTY
Please see the file COPYING for details of the GNU General Public License
```

Рис. 4.6: Командный файл

Запуск командного файла (рис. [4.7])

```
[maalieva@fedora ~]$ bash prog3.sh 4
hmbg
[maalieva@fedora ~]$ bash prog3.sh 467
ycchxgqktimbtpwhwozjinvvynoavxfybxbndxsniwzhdefvryryefuvkzygttprnmraeufmxiaeixlyalwtiom
qvglvxxemqtbharhmigqnbfbzvgbkedppahtwberqibvpiiehihuydbrjigxmbgssccvlsxrdaknofhkslseqmia
rofphfmravzvwzhcqvalghsldddckcfjizlzhmlysygrnkyopsbdbqbusxojxlobqqtpmwbxoziihvljniosa
gykescmowdgfsycscunbidgbqsmpejpcrztvpzejgavbmwhawhsisuzlpgqfmbcwqneuqauvdxczqhyhqtwi
jyanxpyoeqacgskejxpohmkdkzjykenshvsiirbyeyshjjmwowuvajzprmbpdgjtgcckjsreazntnkjposugaivam
apyxupetnuojdltpysanhxcgl
[maalieva@fedora ~]$
```

Рис. 4.7: Запуск файла ко третьему заданию

## 5 Ответы на контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке: `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. Как объединить (конкатенация) несколько строк в одну?

Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами: Первый: `VAR1="Hello," VAR2=" World" VAR3="$VAR1$VAR2" echo "$VAR3"` : *Hello, World* : `$VAR1 = "Hello," $VAR1 += "World" echo $VAR1`  
Результат: *Hello, World*

3. Найдите информацию об утилите seq. Какими иными способами можно реализовать её функционал при программировании на bash?

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. Какой результат даст вычисление выражения  $\$((10/3))$ ?

Результатом данного выражения  $\$((10/3))$  будет 3, потому что это целочисленное деление без остатка.

5. Укажите кратко основные отличия командной оболочки zsh от bash.

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. Проверьте, верен ли синтаксис данной конструкции 1 for ((a=1; a <= LIMIT; a++))

for ((a=1; a <= LIMIT; a++)) синтаксис верен, потому что, используя двойные круглые скобки, можно не писать \$ перед переменными ().

7. Сравните язык bash с какими-либо языками программирования. Какие преимущества у bash по сравнению с ними? Какие недостатки?

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

## **6 Выводы**

В ходе 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