

# Лабораторная работа №14

Операционные системы

---

Верниковская Е. А., НПИбд-01-23

10 мая 2024

Российский университет дружбы народов, Москва, Россия

# Вводная часть

---

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

## 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`.

- 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.

# Выполнение лабораторной работы

---

## Задание №1

Создаю файл для первого задания с расширением sh и делаю его исполняемым (рис. 1)

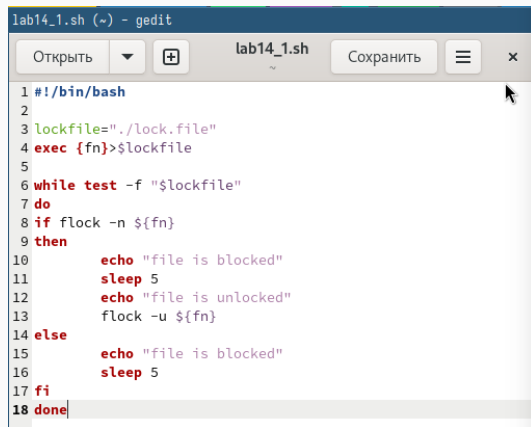
```
[eavernikovskaya@eavernikovskaya ~]$ touch lab14_1.sh  
[eavernikovskaya@eavernikovskaya ~]$ chmod +x lab14_1.sh  
[eavernikovskaya@eavernikovskaya ~]$
```

**Рис. 1:** Создание файла lab14\_1.sh и добавление прав на исполнение



Открываю файл `lab14_1.sh` в текстовом редакторе `gedit` и пишу командный файл, реализующий упрощённый механизм семафоров (подробнее см. в задании №1) (рис. 2)

# Задание №1



```
lab14_1.sh (~) - gedit
Открыть  lab14_1.sh  Сохранить  x
1 #!/bin/bash
2
3 lockfile="./lock.file"
4 exec {fn}>$lockfile
5
6 while test -f "$lockfile"
7 do
8   if flock -n ${fn}
9   then
10     echo "file is blocked"
11     sleep 5
12     echo "file is unlocked"
13     flock -u ${fn}
14   else
15     echo "file is blocked"
16     sleep 5
17   fi
18 done
```

Рис. 2: Написанная программа для lab14\_1.sh


## Задание №1

Далее запускаю файл с помощью `bash` и проверяю работу командного файла (рис. 3)

```
[eavernikovskaya@eavernikovskaya ~]$ bash lab14_1.sh
file is blocked
file is unlocked
file is blocked
file is unlocked
file is blocked
file is unlocked
file is blocked
^C
[eavernikovskaya@eavernikovskaya ~]$
```

**Рис. 3:** Проверка работы командного файла `lab14_1.sh`

Изучаю содержимое каталога `/usr/share/man/man1`. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд (рис. 4), (рис. 5)

A terminal window with a dark background. The prompt is `[eavernikovskaya@eavernikovskaya ~]` followed by the command `$ ls /usr/share/man/man1/`. A white cursor is at the end of the command.

```
[eavernikovskaya@eavernikovskaya ~]$ ls /usr/share/man/man1/
```

**Рис. 4:** Команда

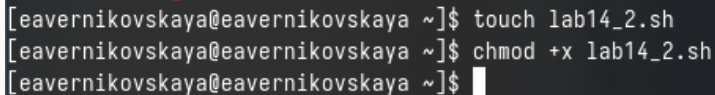
## Задание №2

```
mtx-metapost.1.gz
mtx-modules.1.gz
mtx-package.1.gz
mtx-patterns.1.gz
mtx-pdf.1.gz
mtx-plain.1.gz
mtx-profile.1.gz
mtx-rsync.1.gz
mtxrun.1.gz
mtx-scite.1.gz
mtx-server.1.gz
mtx-spell.1.gz
mtx-texworks.1.gz
mtx-timing.1.gz
mtx-tools.1.gz
mtx-unicode.1.gz
mtx-unzip.1.gz
mtx-update.1.gz
mtx-vscode.1.gz
mtx-watch.1.gz
zcmp.1.gz
zdiff.1.gz
zenity.1.gz
zforce.1.gz
zgrep.1.gz
zip.1.gz
zipcloak.1.gz
zipdetails.1.gz
zipgrep.1.gz
zipinfo.1.gz
zipnote.1.gz
zipsplit.1.gz
zless.1.gz
zmore.1.gz
znew.1.gz
zsoelim.1.gz
zvbi-atsc-cc.1.gz
zvbi-chains.1.gz
zvbid.1.gz
zvbi-ntsc-cc.1.gz
```

[eavernikovskaya@eavernikovskaya ~]\$

Рис. 5: Содержимое каталога /usr/share/man/man1

Создаю файл для второго задания с расширением sh и делаю его исполняемым (рис. 6)

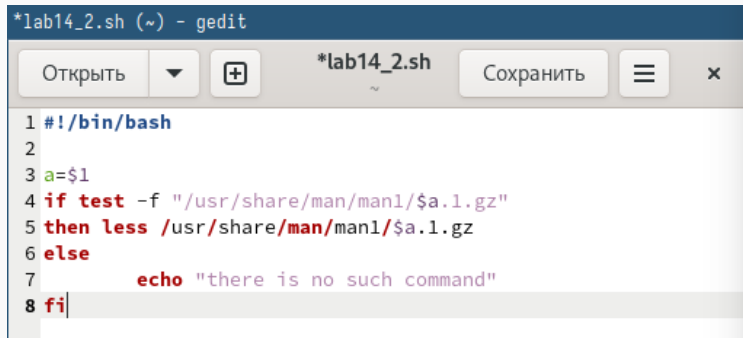
A terminal window with a dark background and light-colored text. The prompt is [eavernikovskaya@eavernikovskaya ~]\$. The first command is touch lab14\_2.sh. The second command is chmod +x lab14\_2.sh. The third line shows the prompt again with a cursor, indicating the command has been executed.

```
[eavernikovskaya@eavernikovskaya ~]$ touch lab14_2.sh  
[eavernikovskaya@eavernikovskaya ~]$ chmod +x lab14_2.sh  
[eavernikovskaya@eavernikovskaya ~]$
```

**Рис. 6:** Создание файла lab14\_2.sh и добавление прав на исполнение

## Задание №2

Открываю файл lab14\_2.sh в текстовом редакторе gedit и пишу командный файл, который будет реализовывать команду man (рис. 7)



The screenshot shows a gedit text editor window titled '\*lab14\_2.sh (~) - gedit'. The window has a toolbar with buttons for 'Открыть' (Open), a dropdown arrow, a '+' icon, the filename '\*lab14\_2.sh', a 'Сохранить' (Save) button, a menu icon, and a close 'x' button. The script content is as follows:

```
1 #!/bin/bash
2
3 a=$1
4 if test -f "/usr/share/man/man1/$a.1.gz"
5 then less /usr/share/man/man1/$a.1.gz
6 else
7     echo "there is no such command"
8 fi
```

Рис. 7: Написанная программа для lab14\_2.sh

Далее запускаю файл с помощью `bash` и проверяю его работу (рис. 8), (рис. 9), (рис. 10), (рис. 11)



## Задание №2

```
foot
ESC[4mLSESC[24m(1)                                User Commands
ESC[4mLSESC[24m(1)

ESC[1mNAMEESC[0m
ls - list directory contents

ESC[1mSYNOPSISESC[0m
ESC[1mls ESC[22m[ESC[4mOPTIONESC[24m]... [ESC[4mFILEESC[24m]...

ESC[1mDESCRIPTIONESC[0m
List information about the FILES (the current directory by default). Sort entries alphabetically if none of ESC[1m-cft
u-ESC[0m
ESC[1mvSUX ESC[22mnor ESC[1m--sort ESC[22mis specified.

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

ESC[1m-eESC[22m, ESC[1m--allESC[0m
do not ignore entries starting with .

ESC[1m-AESC[22m, ESC[1m--almost-allESC[0m
do not list implied . and ..

ESC[1m--authorESC[0m
with ESC[1m-lESC[22m, print the author of each file

ESC[1m-bESC[22m, ESC[1m--escapeESC[0m
print C-style escapes for nongraphic characters

ESC[1m--block-sizeESC[22m=ESC[4mSIZEESC[0m
with ESC[1m-lESC[22m, scale sizes by SIZE when printing them; e.g., '--block-size=M'; see SIZE format below

ESC[1m-BESC[22m, ESC[1m--ignore-backupsESC[0m
do not list implied entries ending with ~

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

Рис. 8: man команды ls (работа командного файла lab14\_2.sh)

## Задание №2

```
foot
ESC[4mCHMODESC[24m(1)                                User Commands
ESC[4mCHMODESC[24m(1)

ESC[1mNAMEESC[0m
  chmod - change file mode bits

ESC[1mSYNOPSISESC[0m
  ESC[1mchmod ESC[22m[ESC[4mOPTIONESC[24m]... ESC[4mMODEESC[24m[ESC[4m,MODEESC[24m]... ESC[4mFILEESC[24m...
  ESC[1mchmod ESC[22m[ESC[4mOPTIONESC[24m]... ESC[4mOCTAL-MODEESC[24m ESC[4mFILEESC[24m...
  ESC[1mchmod ESC[22m[ESC[4mOPTIONESC[24m]... ESC[4m--reference=RFILEESC[24m ESC[4mFILEESC[24m...

ESC[1mDESCRIPTIONESC[0m
  This manual page documents the GNU version of ESC[1mchmodESC[22m. ESC[1mchmod ESC[22mchanges the file mode bits of each
  given file according
  to ESC[4mmodeESC[24m, which can be either a symbolic representation of changes to make, or an octal number representing
  the bit
  pattern for the new mode bits.

  The format of a symbolic mode is [ESC[1mugoaESC[22m...][[ESC[1m+=ESC[22m][ESC[4mpermsESC[24m]...], where ESC[4mperm
  ESC[24m is either zero or more letters from the
  set ESC[1mrwxstESC[22m, or a single letter from the set ESC[1mugoESC[22m. Multiple symbolic modes can be given, separated
  by commas.

  A combination of the letters ESC[1mugoaESC[22mcontrols which users' access to the file will be changed: the user who
  owns it
  (ESC[1muESC[22m), other users in the file's group (ESC[1mgESC[22m), other users not in the file's group (ESC[1moESC[22m),
  or all users (ESC[1maESC[22m). If none of
  these are given, the effect is as if (ESC[1maESC[22m) were given, but bits that are set in the umask are not affected.

  The operator ESC[1m+ESC[22mcauses the selected file mode bits to be added to the existing file mode bits of each file;
  ESC[1m-ESC[22mcauses
  them to be removed; and ESC[1m=ESC[22mcauses them to be added and causes unmentioned bits to be removed except that
  a direc-
  tory's unmentioned set user and group ID bits are not affected.
  /usr/share/man/man1/chmod.1.gz
```

Рис. 9: man команды chmod (работа командного файла lab14\_2.sh)

## Задание №2

```
foot
ESC[4mBASH_BUILTINSESC[24m(1)                                General Commands Manual    ESC[4mB
ASH_BUILTINSESC[24m(1)

ESC[1mNAMEESC[0m
: , . , [ , alias, bg, bind, break, builtin, caller, cd, command, compgen, complete, compopt, continue, declare, dirs,
dismown, echo, enable, eval, exec, exit, export, false, fc, fg, getopts, hash, help, history, jobs, kill, let, local,
logout, mapfile, popd, printf, pushd, pwd, read, readarray, readonly, return, set, shift, shopt, source, suspend,
test, times, trap, true, type, typeset, ulimit, umask, unalias, unset, wait - bash built-in commands, see ESC[1mbashESC[
22m(1)

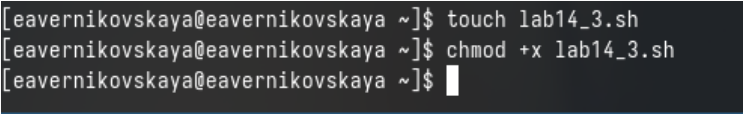
ESC[1mBASH BUILTIN COMMANDSESC[0m
Unless otherwise noted, each builtin command documented in this section as accepting options preceded by ESC[1m-- ESC[2
2maccepts
ESC[1m-- ESC[22mto signify the end of the options. The ESC[1mESC[22m, ESC[1mtrueESC[22m, ESC[1mfalseESC[22m, and ESC[1
mtestESC[22m/ESC[1mESC[22mbuiltins do not accept options and do not
treat ESC[1m-- ESC[22mspecially. The ESC[1mexitESC[22m, ESC[1mlogoutESC[22m, ESC[1mreturnESC[22m, ESC[1mbreakESC[22m,
ESC[1mcontinueESC[22m, ESC[1mletESC[22m, and ESC[1mshift ESC[22mbuiltins accept and process arguments
beginning with ESC[1mESC[22mwithout requiring ESC[1m--ESC[22m. Other builtins that accept arguments but are not spec
ified as accepting op
tions interpret arguments beginning with ESC[1mESC[22mas invalid options and require ESC[1m-- ESC[22mto prevent this i
nterpretation.
ESC[1mESC[22m[ESC[4margumentsESC[24m]
No effect; the command does nothing beyond expanding ESC[4margumentsESC[24m and performing any specified redi
rections.
The return status is zero.
ESC[1mESC[4mESC[22mfilenameESC[24m [ESC[4margumentsESC[24m]
ESC[1msource ESC[4mESC[22mfilenameESC[24m [ESC[4margumentsESC[24m]
Read and execute commands from ESC[4mfilenameESC[24m in the current shell environment and return the exit stat
us of the
last command executed from ESC[4mfilenameESC[24m. If ESC[4mfilenameESC[24m does not contain a slash, filenames i
n ESC[1mPATH ESC[22mare used to
find the directory containing ESC[4mfilenameESC[24m, but ESC[4mfilenameESC[24m does not need to be executable.
The file searched for
/usr/share/man/man1/cd.1.gz
```

Рис. 10: man команды cd (работа командного файла lab14\_2.sh)

```
[eavernikovskaya@eavernikovskaya ~]$ ./lab14_2.sh ls  
[eavernikovskaya@eavernikovskaya ~]$ ./lab14_2.sh chmod  
[eavernikovskaya@eavernikovskaya ~]$ ./lab14_2.sh cd  
[eavernikovskaya@eavernikovskaya ~]$
```

Рис. 11: Проверка работы командного файла lab14\_2.sh

Создаю файл для третьего задания с расширением sh и делаю его исполняемым (рис. 12)

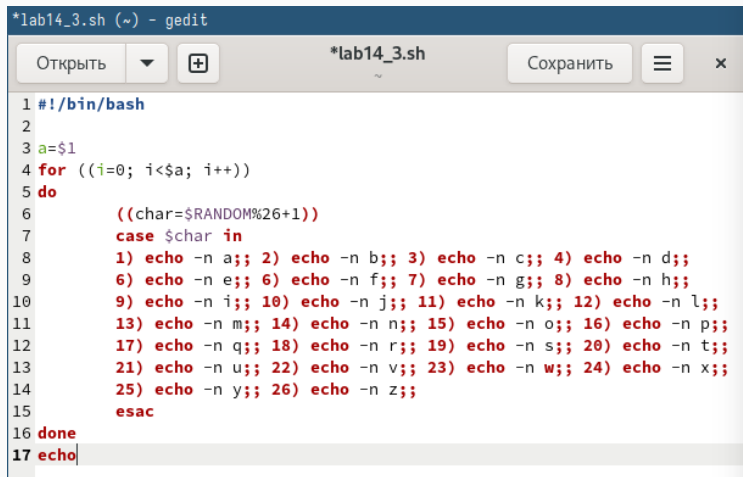
A terminal window with a dark background and light-colored text. It shows three lines of commands being executed in a shell. The first line creates a file named 'lab14\_3.sh' using the 'touch' command. The second line adds execute permissions to the file using 'chmod +x'. The third line shows the prompt with a cursor, indicating the command has finished.

```
[eavernikovskaya@eavernikovskaya ~]$ touch lab14_3.sh  
[eavernikovskaya@eavernikovskaya ~]$ chmod +x lab14_3.sh  
[eavernikovskaya@eavernikovskaya ~]$
```

**Рис. 12:** Создание файла lab13\_3.sh и добавление прав на исполнение

Открываю файл `lab14_3.sh` в текстовом редакторе `gedit` и пишу командный файл, генерирующий случайную последовательность букв латинского алфавита (рис. 13)

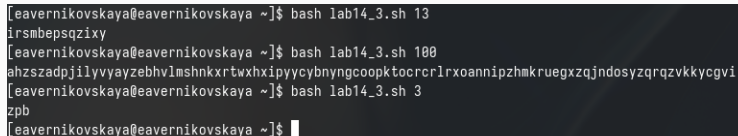
## Задание №3



```
*lab14_3.sh (~) - gedit
Открыть  *lab14_3.sh  Сохранить
1 #!/bin/bash
2
3 a=$1
4 for ((i=0; i<$a; i++))
5 do
6     ((char=$RANDOM%26+1))
7     case $char in
8         1) echo -n a;; 2) echo -n b;; 3) echo -n c;; 4) echo -n d;;
9         6) echo -n e;; 6) echo -n f;; 7) echo -n g;; 8) echo -n h;;
10        9) echo -n i;; 10) echo -n j;; 11) echo -n k;; 12) echo -n l;;
11        13) echo -n m;; 14) echo -n n;; 15) echo -n o;; 16) echo -n p;;
12        17) echo -n q;; 18) echo -n r;; 19) echo -n s;; 20) echo -n t;;
13        21) echo -n u;; 22) echo -n v;; 23) echo -n w;; 24) echo -n x;;
14        25) echo -n y;; 26) echo -n z;;
15    esac
16 done
17 echo
```

Рис. 13: Написанная программа для lab14\_3.sh

Далее запускаю файл с помощью `bash` и проверяю его работу (рис. 14)



```
[eavernikovskaya@eavernikovskaya ~]$ bash lab14_3.sh 13  
irmsbepsqzixy  
[eavernikovskaya@eavernikovskaya ~]$ bash lab14_3.sh 100  
ahzszadpjilyvyayzebhlmsnkrxrtwxhixpyycybnynngcoopktocrclrxoannipzhmkruegxzqjndosyzqrqzvkkycgvi  
[eavernikovskaya@eavernikovskaya ~]$ bash lab14_3.sh 3  
zpb  
[eavernikovskaya@eavernikovskaya ~]$
```

**Рис. 14:** Проверка работы командного файла `lab14_3.sh`



## **Подведение итогов**

---

В ходе выполнения лабораторной работы мы изучили основы программирования в оболочке ОС UNIX а также научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Не пользовалась сайтами.