

Лабораторная работа № 14

**Программирование в командном процессоре ОС UNIX.
Расширенное программирование**

Мальянц Виктория Кареновна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
3.1	Задание № 1	8
3.2	Задание № 2	9
3.3	Задание № 3	11
4	Выводы	13
5	Контрольные вопросы	14
	Список литературы	15

Список иллюстраций

3.1	Создание файла lab14-1.sh и открытие его	8
3.2	Редактирование файла	8
3.3	Право на исполнение файла lab14-1.sh и запуск этого файла	9
3.4	Просмотр содержимого /usr/share/man/man1	9
3.5	Создание файла lab14-2.sh и открытие его	9
3.6	Редактирование файла	10
3.7	Право на исполнение файла lab14-2.sh и запуск этого файла	10
3.8	Справка	11
3.9	Создание файла lab14-3.sh и открытие его	11
3.10	Редактирование файла	12
3.11	Право на исполнение файла lab14-3.sh и запуск этого файла	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. 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. Задание № 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. Задание № 3. Используя встроенную переменную `$RANDOM`, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтите, что `$RANDOM` выдаёт псевдослучайные числа

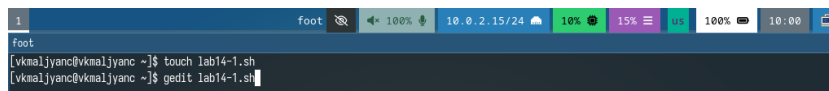
в диапазоне от 0 до 32767.

4. Контрольные вопросы

3 Выполнение лабораторной работы

3.1 Задание № 1

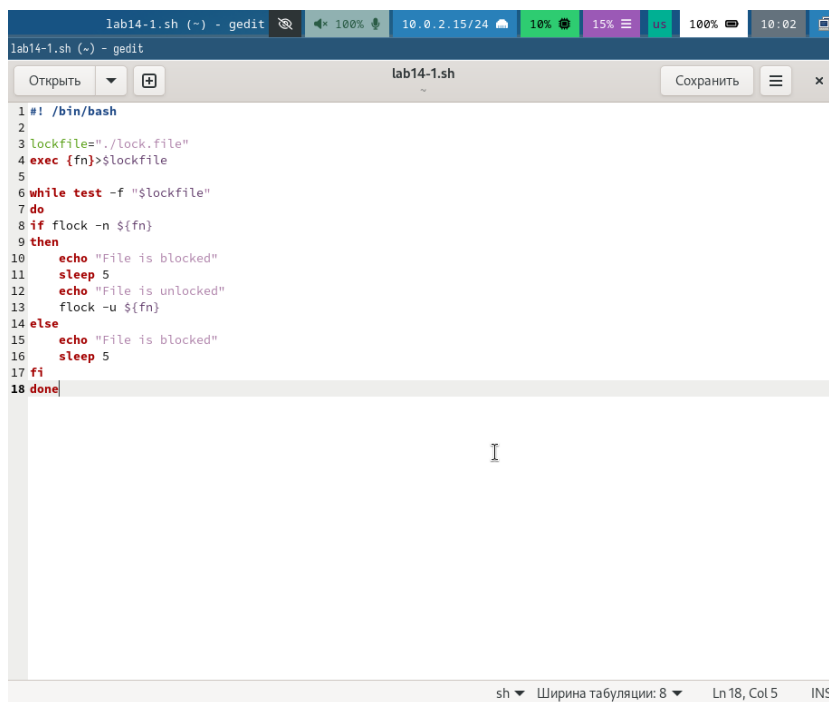
Создаю файл lab14-1.sh и открываю его (рис. 3.1).



```
1  
foot  
[vkmajjyanc@vkmajjyanc ~]$ touch lab14-1.sh  
[vkmajjyanc@vkmajjyanc ~]$ gedit lab14-1.sh
```

Рис. 3.1: Создание файла lab14-1.sh и открытие его

Ввожу код в файл lab14-1.sh (рис. 3.2).



```
lab14-1.sh (-) - gedit  
lab14-1.sh (-) - gedit  
Открыть 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  
sh Ширина табуляции: 8 Ln 18, Col 5 INS
```

Рис. 3.2: Редактирование файла

Даю право на исполнение файла `lab14-1.sh` и запускаю его. Убеждаюсь в том, что программа работает корректно (рис. 3.3).

```
[vkmajyanc@vkmajyanc ~]$ chmod +x lab14-1.sh
[vkmajyanc@vkmajyanc ~]$ 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
File is unlocked
File is blocked
File is unlocked
File is blocked_
```

Рис. 3.3: Право на исполнение файла lab14-1.sh и запуск этого файла

3.2 Задание № 2

Просматриваю содержимое /usr/share/man/man1 (рис. 3.4).

```
[kmaljanc@kmaljanc ~]$ ls /usr/share/man/man1
.:1.gz
'[:1.gz
7z.1.gz
n2ping.1.gz
nc.1.gz
nconnect.1.gz
addr2line.1.gz
nfa2fwa.1.gz
afa2pl.1.gz
afa2fwa.1.gz
albatross.1.gz
aloph.1.gz
alias.1.gz
allcm.1.gz
altec.1.gz
allneeded.1.gz
alsactl.1.gz
alsa-info.sh.1.gz
alsaloop.1.gz
alsamixer.1.gz
alsaumute.1.gz
alt-java.1.gz
alt-java-java-2l-openjdk.1.gz
amidi.1.gz
amixer.1.gz
amstex.1.gz
animate.1.gz
nplay.1.gz
nplaymidi.1.gz
nplaymidi2.1.gz
nppstreamcl.1.gz
aprosop.1.gz
git-write-tree.1.gz
glib-compile-schemas.1.gz
gm.1.gz
gnake.1.gz
gnaeqn.1.gz
gnome-keyring.1.gz
gnome-keyring-3.1.gz
gnome-keyring-daemon.1.gz
gnroff.1.gz
gfd-tool.1.gz
gp-archive.1.gz
gpaswd.1.gz
gp-collect-app.1.gz
gp-display-htal.1.gz
gp-display-src.1.gz
gp-display-text.1.gz
gpg.1.gz
gpg2.1.gz
gpg-agent.1.gz
gpg-card.1.gz
gpg-check-pattern.1.gz
gpgconf.1.gz
gpg-connect-agent.1.gz
gpgpgpasmal.1.gz
gpg-preset-passphrase.1.gz
gpgsw.1.gz
gpgtar.1.gz
gpgv.1.gz
gpgv2.1.gz
gpg-wks-client.1.gz
gpg-wks-server.1.gz
gpic.1.gz
openssl-x509.1.gz
openvt.1.gz
opl2fwa.1.gz
optex.1.gz
ot2kpx.1.gz
otangle.1.gz
otinfo.1.gz
otftotfm.1.gz
otp2ocp.1.gz
outocp.1.gz
ovf2ovp.1.gz
ovp2ovf.1.gz
pacat.1.gz
pacmd.1.gz
pactl.1.gz
padsp.1.gz
pagelayoutapi.1.gz
pamon.1.gz
pamphiletangler.1.gz
panelctl.1.gz
pango-view.1.gz
paper.1.gz
paperconf.1.gz
papi.1.gz
paps.1.gz
parec.1.gz
parecord.1.gz
passwd.1.gz
passwd.1.gz
paste.1.gz
pasuspender.1.gz
patch.1.gz
```

Рис. 3.4: Просмотр содержимого `/usr/share/man/man1`

Создаю файл lab14-2.sh и открываю его (рис. 3.5).

```
[vkmajyanc@vkmajyanc ~]$ touch lab14-2.sh
[vkmajyanc@vkmajyanc ~]$ gedit lab14-2.sh
```

Рис. 3.5: Создание файла lab14-2.sh и открытие его

Ввожу код в файл lab14-2.sh (рис. 3.6).

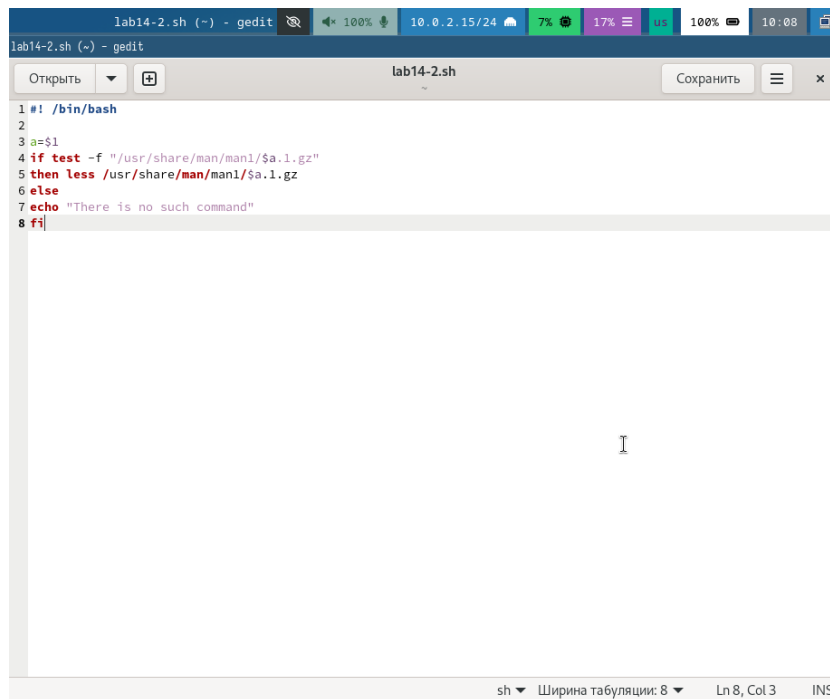


Рис. 3.6: Редактирование файла

Даю право на исполнение файла lab14-2.sh и запускаю его (рис. 3.7).

```
[vkmajjyanc@vkmajjyanc ~]$ chmod +x lab14-2.sh
[vkmajjyanc@vkmajjyanc ~]$ bash lab14-2.sh cd
```

Рис. 3.7: Право на исполнение файла lab14-2.sh и запуск этого файла

Убеждаюсь в том, что программа работает корректно (рис. 3.8).

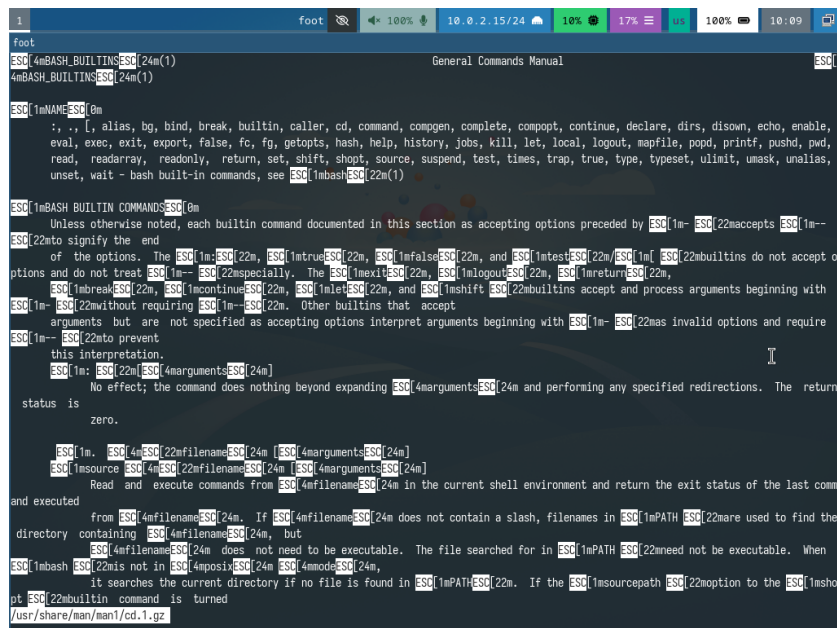


Рис. 3.8: Справка

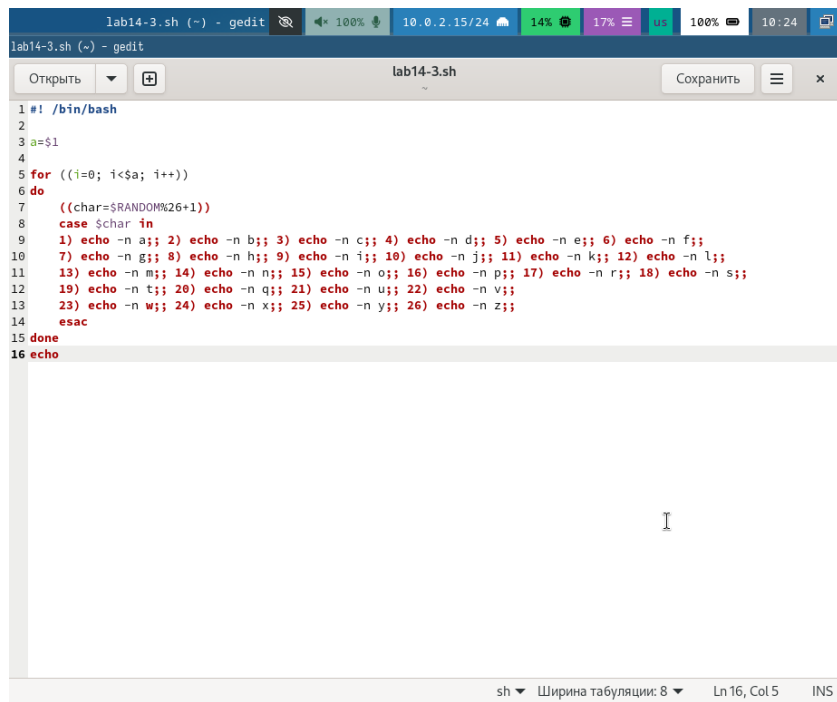
3.3 Задание № 3

Создаю файл lab14-3.sh и открываю его (рис. 3.9).



Рис. 3.9: Создание файла lab14-3.sh и открытие его

Ввожу код в файл lab14-3.sh (рис. 3.10).



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

Рис. 3.10: Редактирование файла

Даю право на исполнение файла lab14-3.sh и запускаю его. Убеждаюсь в том, что программа работает корректно (рис. 3.11) [1].



```
[vkmajjyanc@vkmajjyanc ~]$ chmod +x lab14-3.sh
[vkmajjyanc@vkmajjyanc ~]$ bash lab14-3.sh 15
ulojaipzxuujlpb
[vkmajjyanc@vkmajjyanc ~]$
```

Рис. 3.11: Право на исполнение файла lab14-3.sh и запуск этого файла

4 Выводы

Я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

5 Контрольные вопросы

1. Пробел должен быть между [и условием.
2. В bash можно объединить строки, используя оператор + или просто ставя строки рядом.
3. Утилита seq используется для генерации последовательностей чисел. Иные способы: использование цикла for, использование printf.
4. Результат будет 3, так как в bash происходит целочисленное деление.
5. Отличия командной оболочки zsh от bash: расширенные возможности автозаполнения, темы и плагины, маски, глобальные алиасы.
6. Верен.
7. Преимущества: простота использования, интеграция с системными утилитами, низкий уровень. Недостатки: ограниченная функциональность, отсутствие строгой типизации, медлительность (по сравнению с компилируемыми языками, такими как C или Go).

Список литературы

1. Лабораторная работа № 14.