
Front matter

title: "Лабораторная работа №12" subtitle: "Программирование в командном процессоре ОС UNIX. Расширенное программирование" author: "Данзанова Саяна Зоригтеевна"

Generic options

lang: ru-RU toc-title: "Содержание"

Bibliography

bibliography: bib/cite.bib csl: pandoc/csl/gost-r-7-0-5-2008-numeric.csl

PDF output format

toc: true # Table of contents toc-depth: 2 lof: true # List of figures lot: true # List of tables fontsize: 12pt
linestretch: 1.5 papersize: a4 documentclass: scrreprt

LaTeX polyglossia

polyglossia-lang: name: russian options:

- spelling=modern
- babelshorthands=true polyglossia-otherlangs: name: english

LaTeX babel

babel-lang: russian babel-otherlangs: english

Fonts

mainfont: PT Serif romanfont: PT Serif sansfont: PT Sans monofont: PT Mono mainfontoptions: Ligatures=TeX
romanfontoptions: Ligatures=TeX sansfontoptions: Ligatures=TeX,Scale=MatchLowercase monofontoptions:
Scale=MatchLowercase,Scale=0.9

BibLaTeX

biblatex: true biblio-style: "gost-numeric" biblatexoptions:

- parenttracker=true
- backend=biber
- hyperref=auto
- language=auto
- autolang=other*
- citestyle=gost-numeric

Pandoc-crossref LaTeX customization

figureTitle: "Рис." tableTitle: "Таблица" listingTitle: "Листинг" lofTitle: "Список иллюстраций" lotTitle: "Список таблиц" lolTitle: "Листинги"

Misc options

indent: true header-includes:

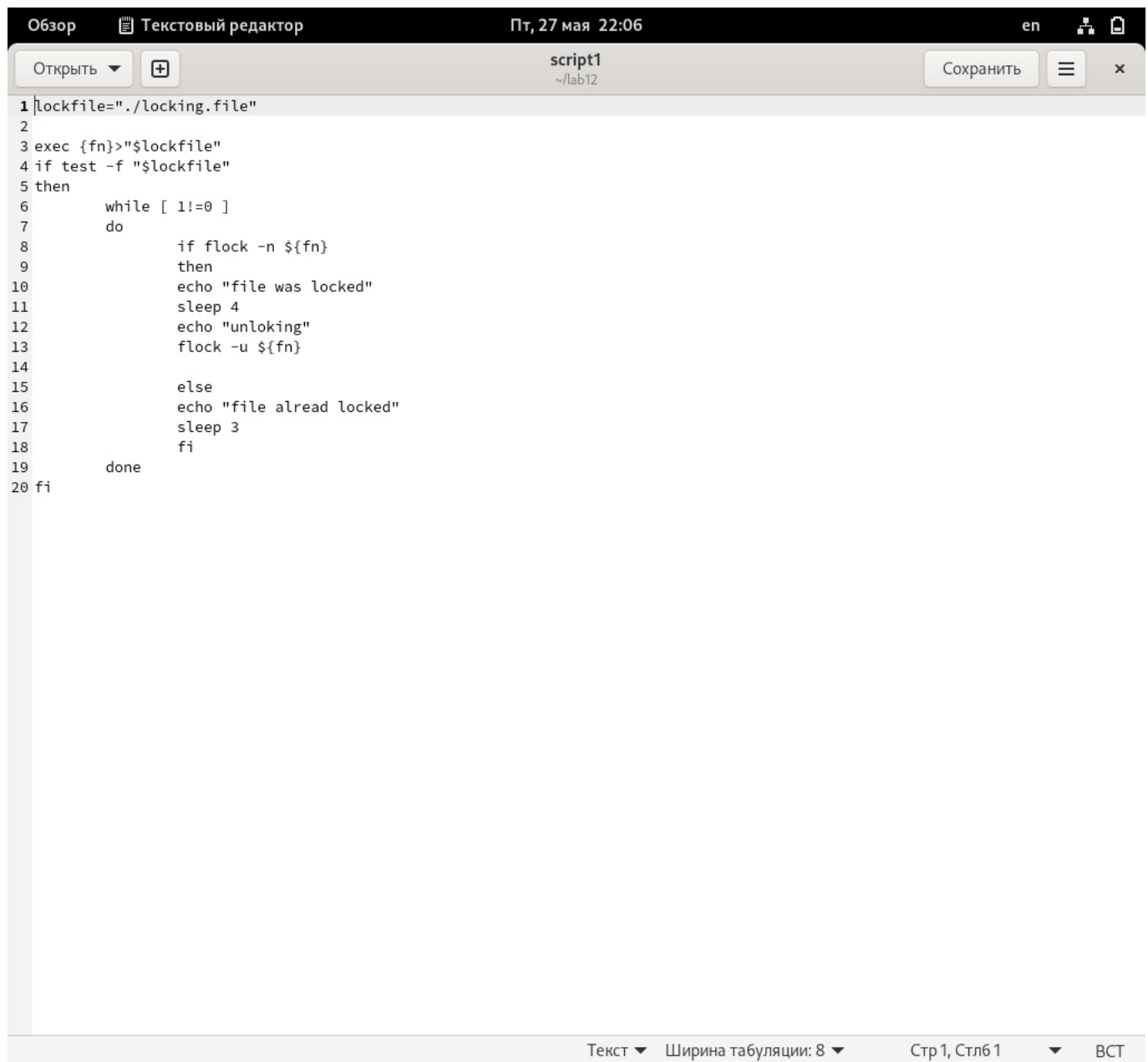
- `\usepackage{indentfirst}`
- `\usepackage{float} # keep figures where there are in the text`
- `\floatplacement{figure}{H} # keep figures where there are in the text`




Цель работы

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

Ход работы

1. Написали 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). Запустили 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. Доработали программу так, чтобы имелась возможность взаимодействия трёх и более процессов (рис.1,2).

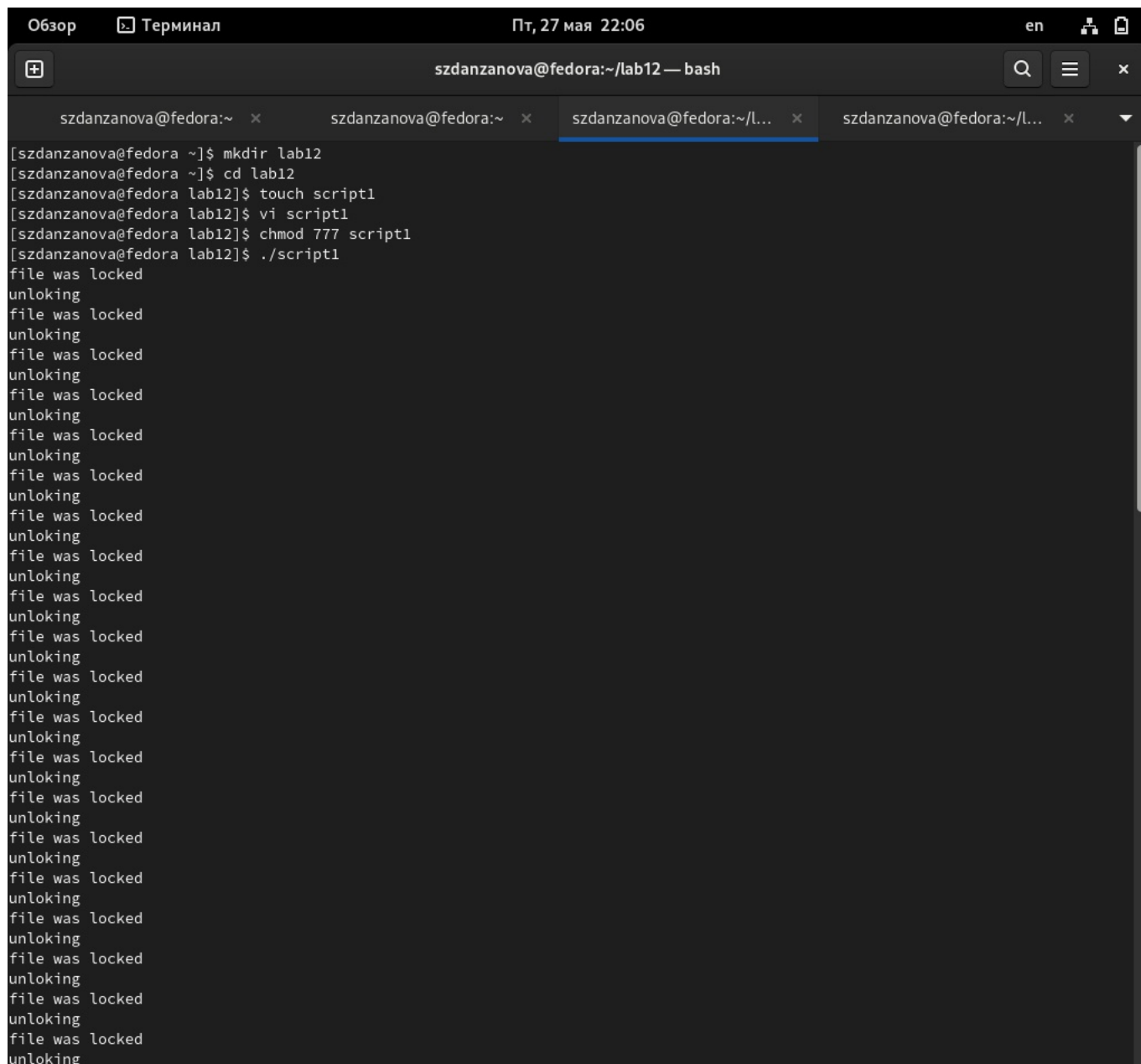


```
Обзор  Текстовый редактор  Пт, 27 мая 22:06  en  
Открыть    script1  Сохранить    x
~/lab12

1 lockfile="./locking.file"
2
3 exec {fn}>"$lockfile"
4 if test -f "$lockfile"
5 then
6     while [ 1!=0 ]
7     do
8         if flock -n ${fn}
9         then
10            echo "file was locked"
11            sleep 4
12            echo "unlocking"
13            flock -u ${fn}
14
15        else
16            echo "file already locked"
17            sleep 3
18        fi
19    done
20 fi

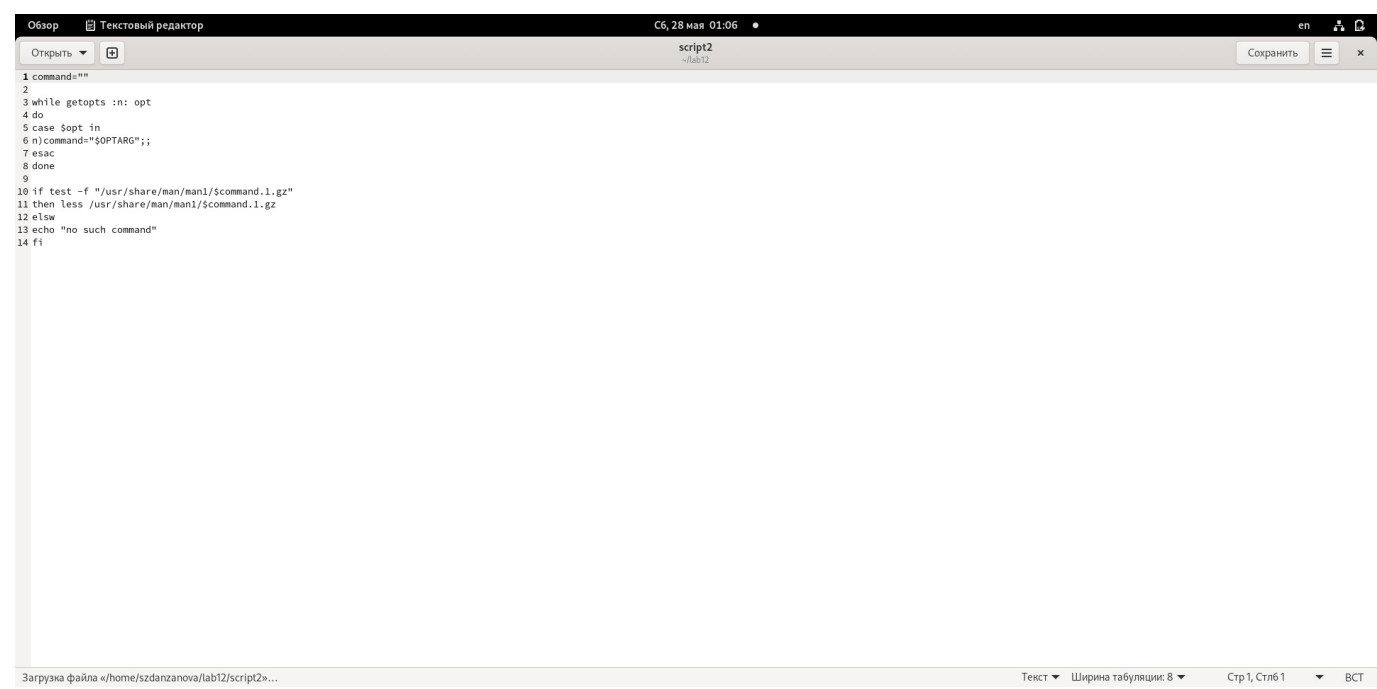
Текст  Ширина табуляции: 8  Стр 1, Стлб 1  ВСТ
```

{ #fig:001 width=90% }

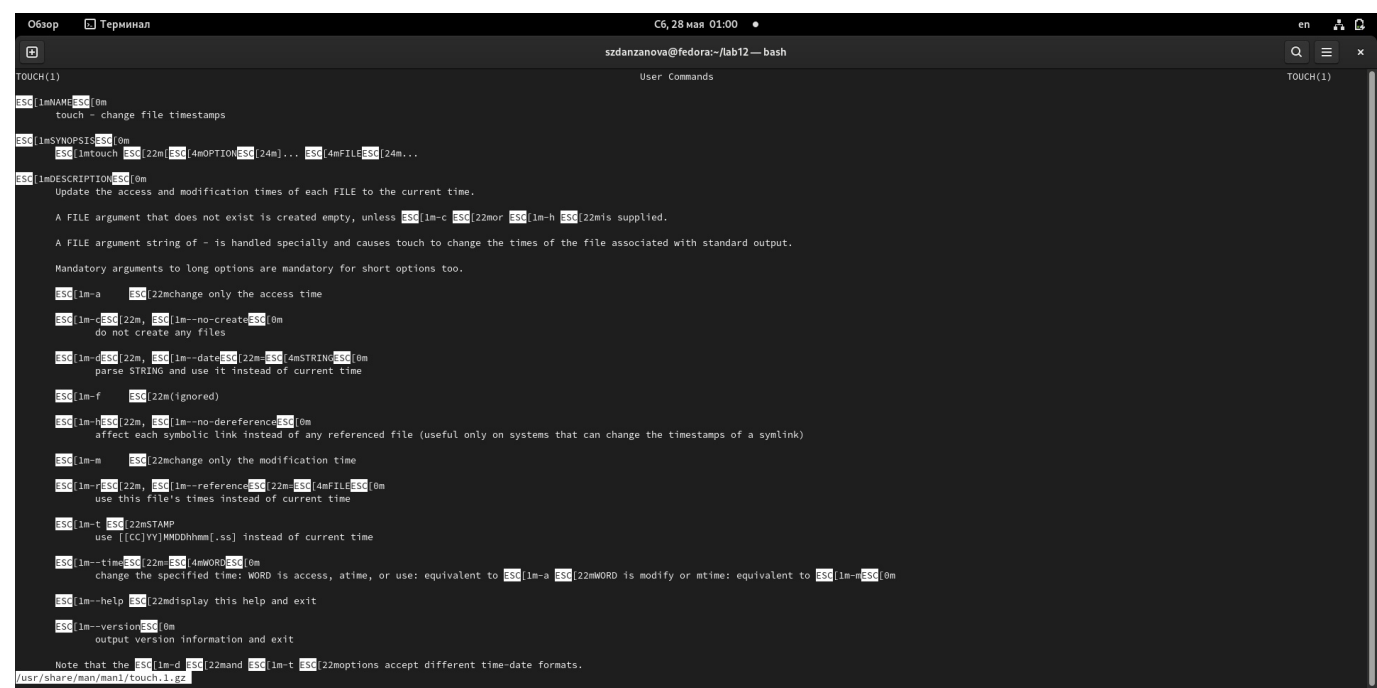


```
{ #fig:002 width=90% }
```

2. Реализовали команду `man` с помощью командного файла. Изучили 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.3,4).

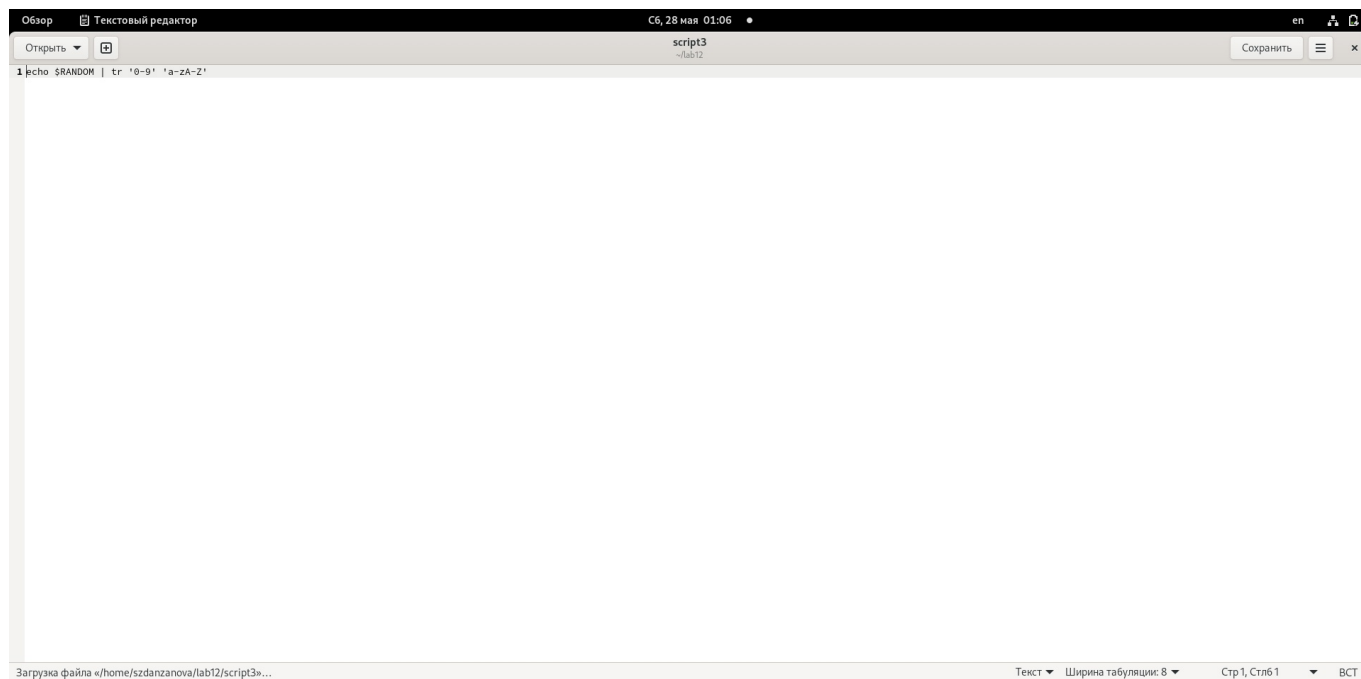


{ #fig:003 width=90% }

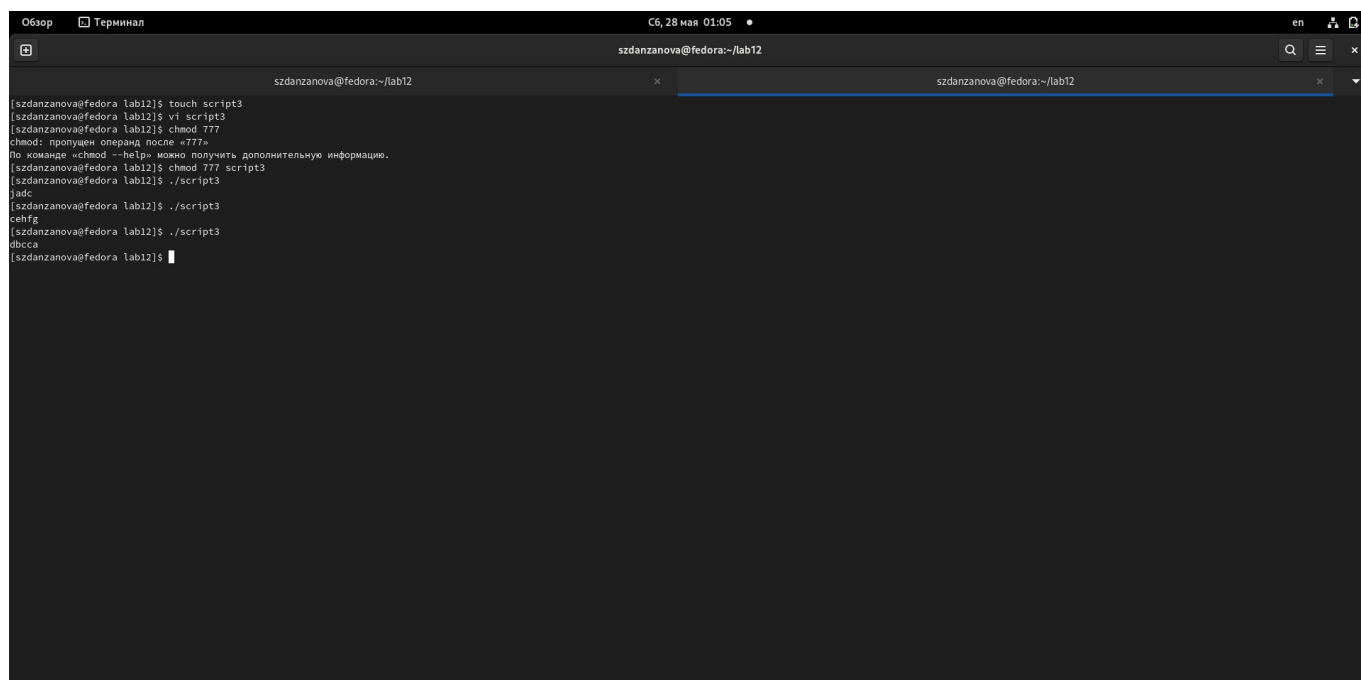


{ #fig:004 width=90% }

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.



{ #fig:005 width=90% }



{ #fig:006 width=90% }

Вывод

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

Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке: `while [$1 != "exit"] $1`. Так же между скобками должны быть пробелы. В противном случае скобки и рядом стоящие символы будут восприниматься как одно целое

2. Как объединить (конкатенация) несколько строк в одну? `cat file.txt | xargs | sed -e 's/. /\n/g'`
3. Найдите информацию об утилите `seq`. Какими иными способами можно реализовать её функционал при программировании на `bash`? `seq` - выдает последовательность чисел. Реализовать ее функционал можно командой `for n in {1..5} do <КОМАНДА> done`
4. Какой результат даст вычисление выражения `$((10/3))`? 3
5. Укажите кратко основные отличия командной оболочки `zsh` от `bash`. `Zsh` очень сильно упрощает работу. Но существуют различия. Например, в `zsh` после `for` обязательно вставлять пробел, нумерация массивов в `zsh` начинается с 1 (что не особо удобно на самом деле). Если вы собираетесь писать скрипт, который легко будет запускать множество разработчиков, то я рекомендую `Bash`. Если скрипты вам не нужны - `Zsh` (более простая работа с файлами, например)
6. Проверьте, верен ли синтаксис данной конструкции `for ((a=1; a <= LIMIT; a++))` Верен
7. Сравните язык `bash` с какими-либо языками программирования. Какие преимущества у `bash` по сравнению с ними? Какие недостатки? `Bash` позволяет очень легко работать с файловой системой без лишних конструкций (в отличие от обычного языка программирования). Но относительно обычных языков программирования `bash` очень сжат. Тот же `C` имеет гораздо более широкие возможности для разработчика.