
Front matter

title: "Лабораторная работа №10" subtitle: "Программирование в командном процессоре ОС UNIX. Командные файлы" author: "Данзанова Саяна Зоригтоевна"

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

l18n polyglossia

polyglossia-lang: name: russian options:

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

l18n 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:

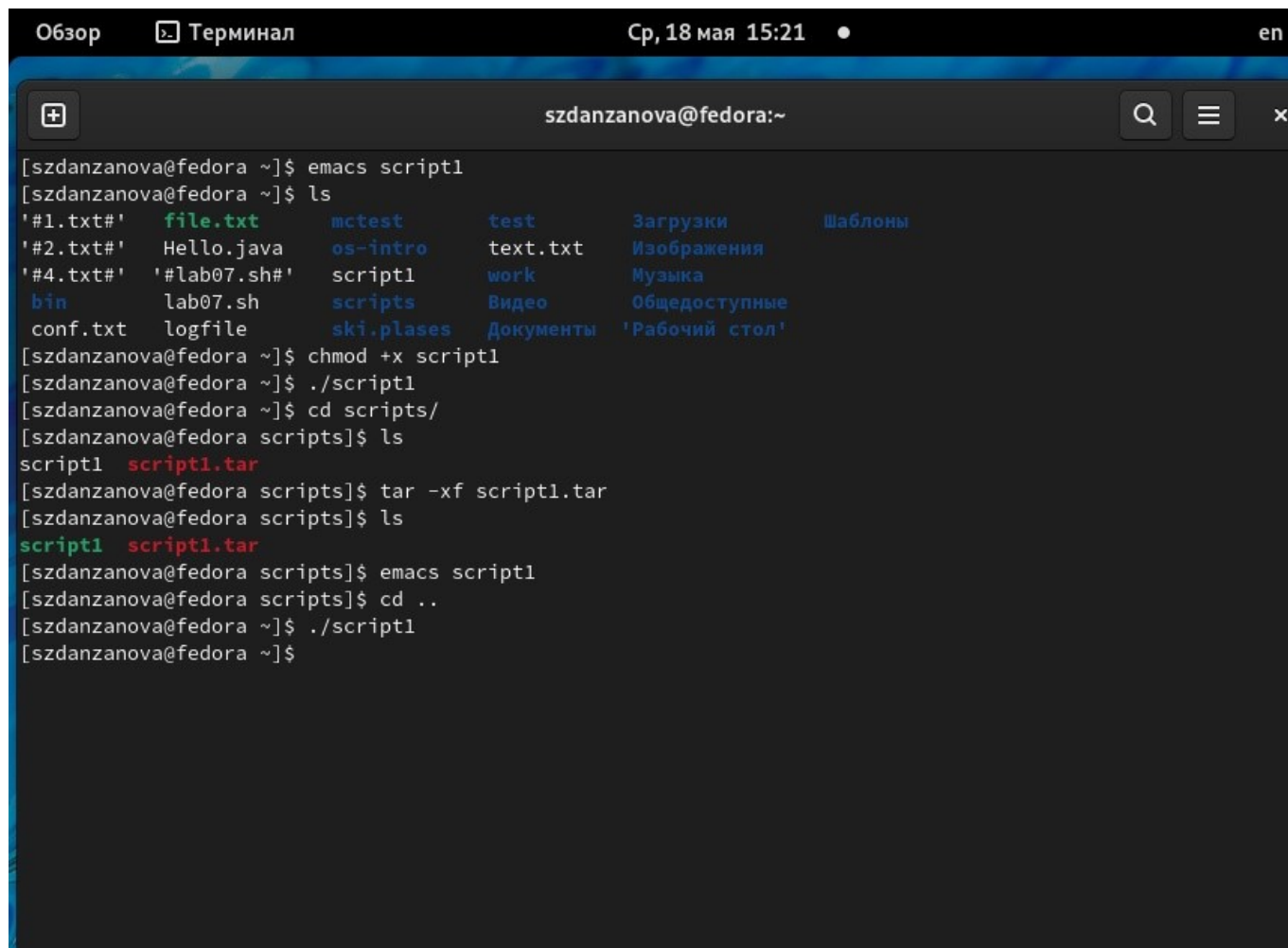
- `\usepackage{indentfirst}`
- `\usepackage{float} # keep figures where there are in the text`
- `\floatplacement{figure}{H} # keep figures where there are in the text`

Цель работы

Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

Ход работы

1. Написали скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию `scripts` в моем домашнем каталоге. При этом файл архивироваться `tar`. (рис.1,2)

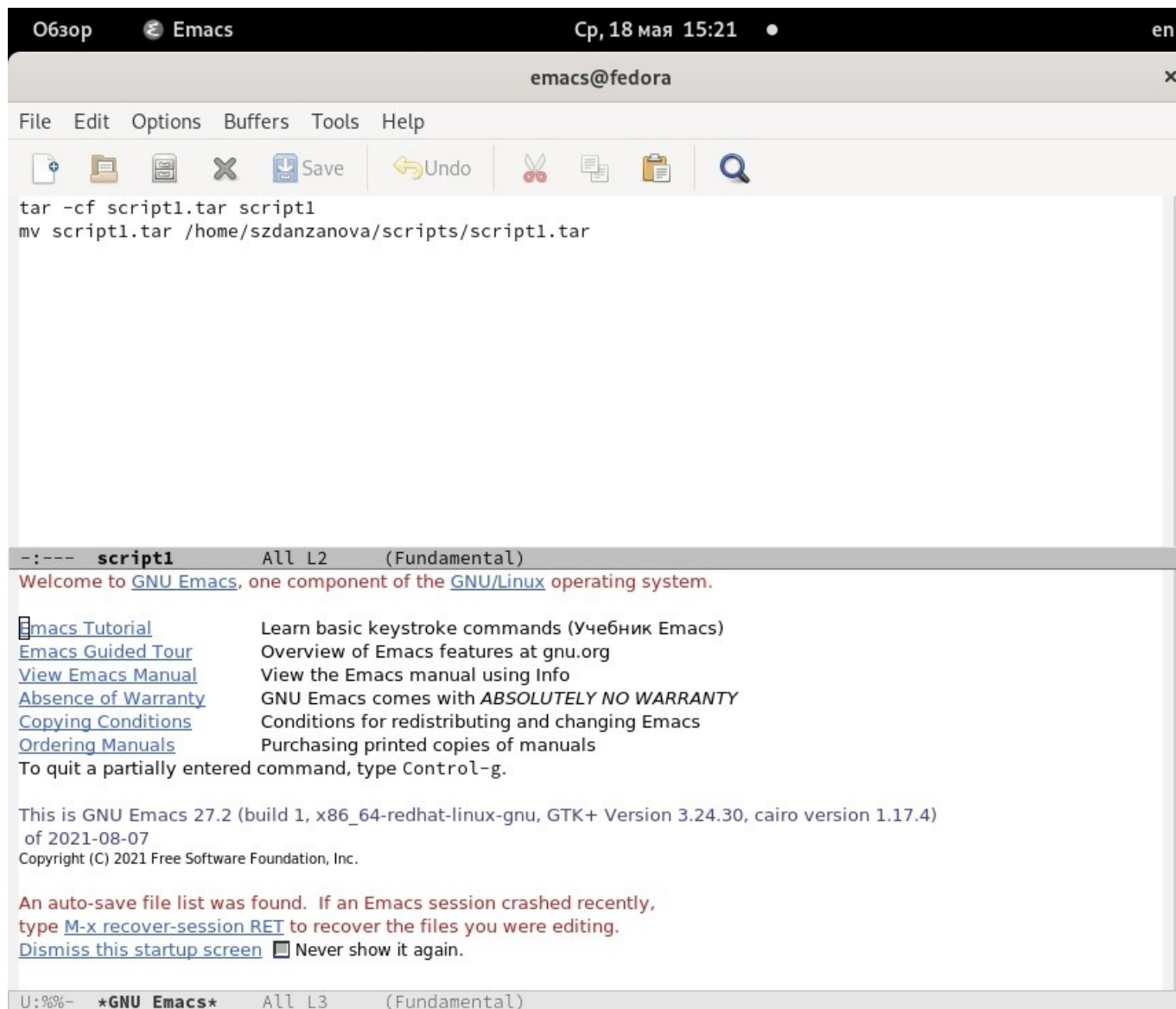


The image shows a terminal window titled "Обзор" (Overview) and "Терминал" (Terminal) with the date and time "Ср, 18 мая 15:21" and language "en". The terminal displays the following commands and output:

```
[szdanzanova@fedora ~]$ emacs script1
[szdanzanova@fedora ~]$ ls
'#1.txt#'  file.txt      mctest       test          Загрузки      Шаблоны
'#2.txt#'  Hello.java    os-intro     text.txt      Изображения
'#4.txt#'  '#lab07.sh#' script1       work          Музыка
bin        lab07.sh      scripts      Видео         Общедоступные
conf.txt   logfile      ski.places   Документы     'Рабочий стол'

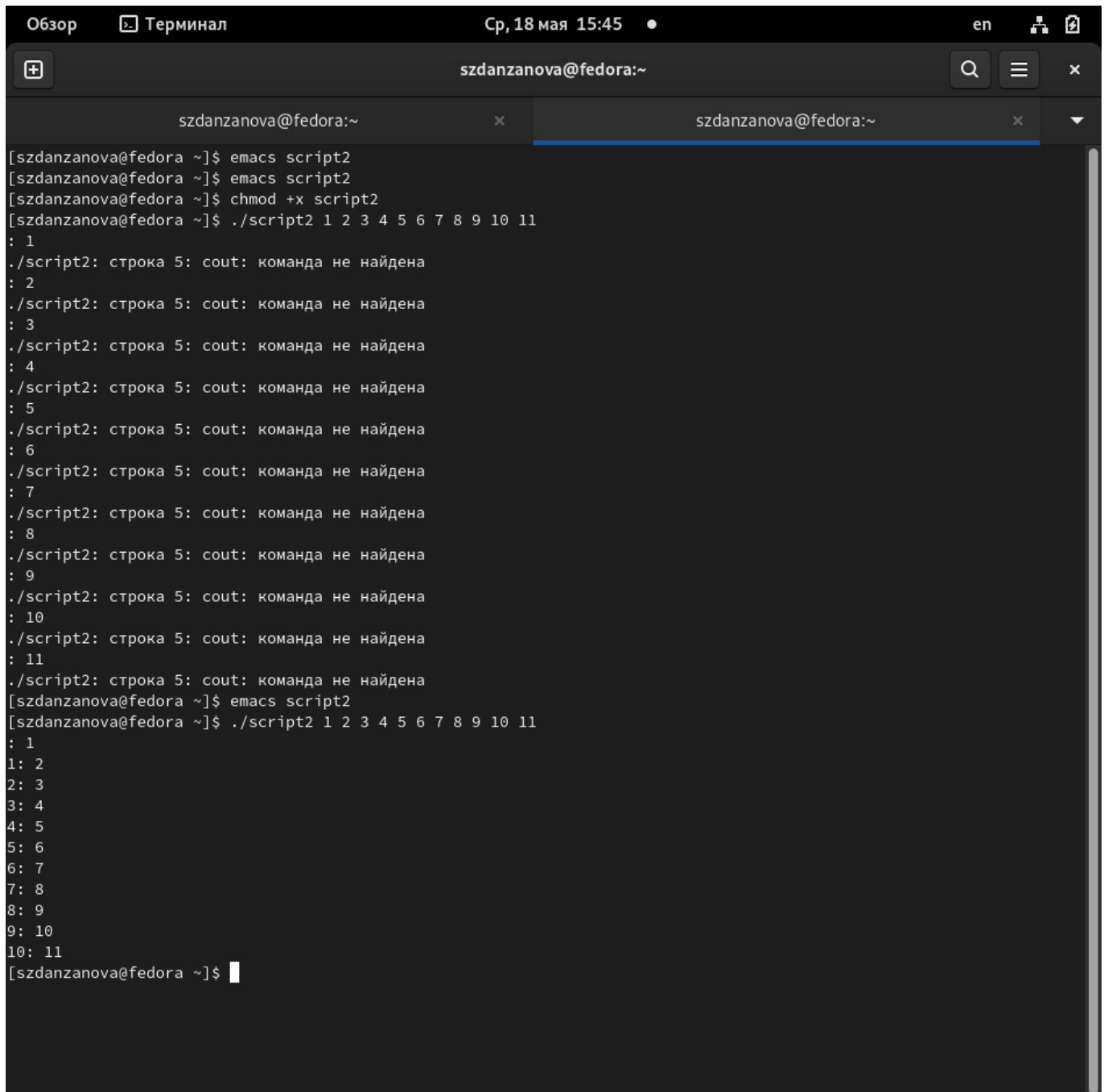
[szdanzanova@fedora ~]$ chmod +x script1
[szdanzanova@fedora ~]$ ./script1
[szdanzanova@fedora ~]$ cd scripts/
[szdanzanova@fedora scripts]$ ls
script1  script1.tar
[szdanzanova@fedora scripts]$ tar -xf script1.tar
[szdanzanova@fedora scripts]$ ls
script1  script1.tar
[szdanzanova@fedora scripts]$ emacs script1
[szdanzanova@fedora scripts]$ cd ..
[szdanzanova@fedora ~]$ ./script1
[szdanzanova@fedora ~]$
```

{ #fig:001 width=90% }



{ #fig:002 width=90% }

2. Написали пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов. (рис.3,4)



```
Обзор Терминал Ср, 18 мая 15:45 en
szdanzanova@fedora:~
[ezdanzanova@fedora ~]$ emacs script2
[ezdanzanova@fedora ~]$ emacs script2
[ezdanzanova@fedora ~]$ chmod +x script2
[ezdanzanova@fedora ~]$ ./script2 1 2 3 4 5 6 7 8 9 10 11
: 1
./script2: строка 5: cout: команда не найдена
: 2
./script2: строка 5: cout: команда не найдена
: 3
./script2: строка 5: cout: команда не найдена
: 4
./script2: строка 5: cout: команда не найдена
: 5
./script2: строка 5: cout: команда не найдена
: 6
./script2: строка 5: cout: команда не найдена
: 7
./script2: строка 5: cout: команда не найдена
: 8
./script2: строка 5: cout: команда не найдена
: 9
./script2: строка 5: cout: команда не найдена
: 10
./script2: строка 5: cout: команда не найдена
: 11
./script2: строка 5: cout: команда не найдена
[ezdanzanova@fedora ~]$ emacs script2
[ezdanzanova@fedora ~]$ ./script2 1 2 3 4 5 6 7 8 9 10 11
: 1
1: 2
2: 3
3: 4
4: 5
5: 6
6: 7
7: 8
8: 9
9: 10
10: 11
[ezdanzanova@fedora ~]$
```

{ #fig:003 width=90% }

```

count=1
for param in "$@"
do
echo "$cout: $param"
cout=$((cout + 1))
done

```

script2 All L1 (Fundamental)

Welcome to [GNU Emacs](#), one component of the [GNU/Linux](#) operating system.

Emacs Tutorial	Learn basic keystroke commands (Учебник Emacs)
Emacs Guided Tour	Overview of Emacs features at gnu.org
View Emacs Manual	View the Emacs manual using Info
Absence of Warranty	GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions	Conditions for redistributing and changing Emacs
Ordering Manuals	Purchasing printed copies of manuals

To quit a partially entered command, type Control-g.

This is GNU Emacs 27.2 (build 1, x86_64-redhat-linux-gnu, GTK+ Version 3.24.30, cairo version 1.17.4) of 2021-08-07
 Copyright (C) 2021 Free Software Foundation, Inc.

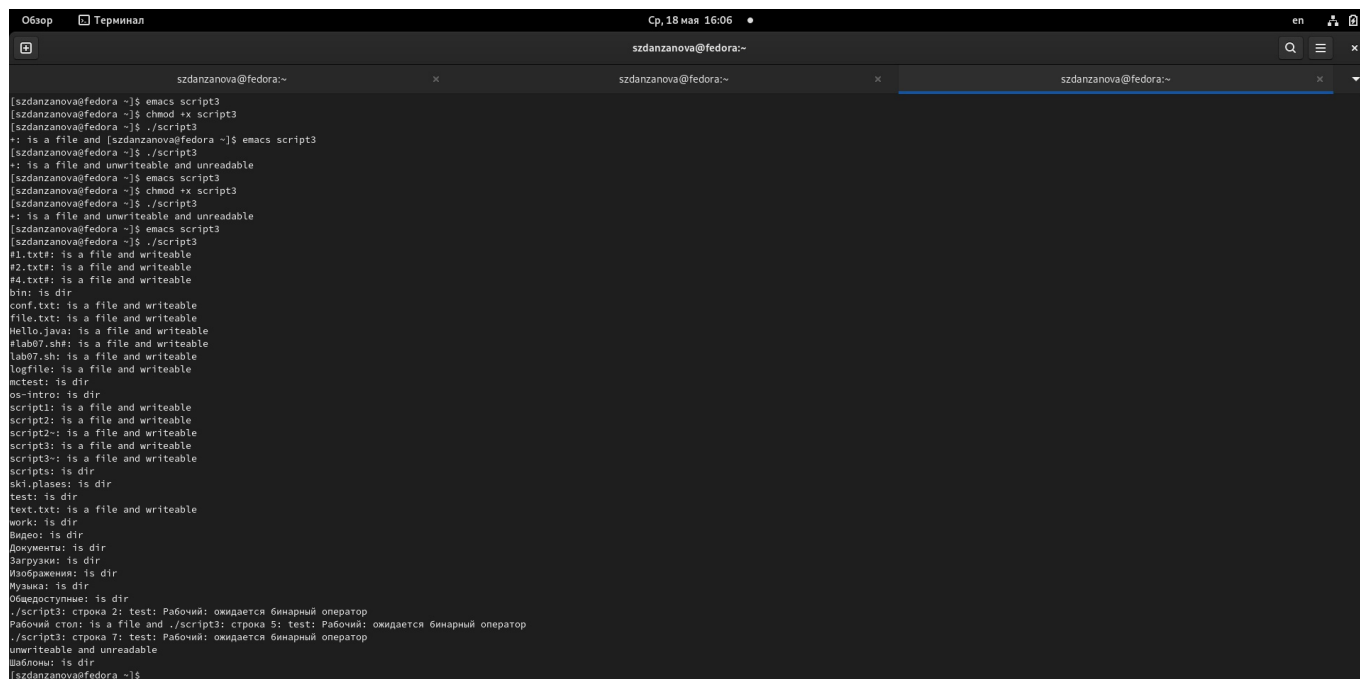
An auto-save file list was found. If an Emacs session crashed recently, type **M-x recover-session RET** to recover the files you were editing.
[Dismiss this startup screen](#) ☐ Never show it again.

U:%%- ***GNU Emacs*** All L3 (Fundamental)

For information about GNU Emacs and the GNU system, type C-h C-a.

#fig:004 width=90% }

3. Написали командный файл — аналог команды ls (без использования самой этой команды и команды dir). Требовалось, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога. (рис.5,6)



```
szdanzanova@fedora:~  
[szdanzanova@fedora ~]$ emacs script3  
[szdanzanova@fedora ~]$ chmod +x script3  
[szdanzanova@fedora ~]$ ./script3  
+1 is a file and [szdanzanova@fedora ~]$ emacs script3  
[szdanzanova@fedora ~]$ ./script3  
+1 is a file and unwriteable and unreadable  
[szdanzanova@fedora ~]$ emacs script3  
[szdanzanova@fedora ~]$ chmod +x script3  
[szdanzanova@fedora ~]$ ./script3  
+1 is a file and unwriteable and unreadable  
[szdanzanova@fedora ~]$ emacs script3  
[szdanzanova@fedora ~]$ ./script3  
#1.txt: is a file and writeable  
#2.txt: is a file and writeable  
#4.txt: is a file and writeable  
bin: is dir  
conf.txt: is a file and writeable  
file.txt: is a file and writeable  
hello.java: is a file and writeable  
lab07.sh: is a file and writeable  
lab07.sh: is a file and writeable  
logfile: is a file and writeable  
mctest: is dir  
os-intro: is dir  
script1: is a file and writeable  
script2: is a file and writeable  
script2-: is a file and writeable  
script3: is a file and writeable  
script3-: is a file and writeable  
script5: is dir  
ski.places: is dir  
test: is dir  
text.txt: is a file and writeable  
works: is dir  
Видео: is dir  
Документы: is dir  
Загрузки: is dir  
Изображения: is dir  
Музыка: is dir  
Общедоступные: is dir  
./script3: строка 2: test: Рабочий: ожидается бинарный оператор  
Рабочий стол: is a file and ./script3: строка 5: test: Рабочий: ожидается бинарный оператор  
./script3: строка 7: test: Рабочий: ожидается бинарный оператор  
unwriteable and unreadable  
шаблоны: is dir  
[szdanzanova@fedora ~]$
```

{ #fig:005 width=90% }

```

emacs@fedora
File Edit Options Buffers Tools Help
[Icons: New, Open, Save, Undo, Redo, Find, etc.]

for A in *
do if test -d $A
  then echo $A: is dir
  else echo -n $A: "is a file and "
    if test -w $A
    then echo writeable
    elif test -r $A
    then echo readable
    else echo unwriteable and unreadable
    fi
  fi
done

-:--- script3 All L1 (Fundamental)
Welcome to GNU Emacs, one component of the GNU/Linux operating system.

Emacs Tutorial      Learn basic keystroke commands (Учебник Emacs)
Emacs Guided Tour   Overview of Emacs features at gnu.org
View Emacs Manual   View the Emacs manual using Info
Absence of Warranty GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions  Conditions for redistributing and changing Emacs
Ordering Manuals    Purchasing printed copies of manuals
To quit a partially entered command, type Control-g.

This is GNU Emacs 27.2 (build 1, x86_64-redhat-linux-gnu, GTK+ Version 3.24.30, cairo version 1.17.4)
of 2021-08-07
Copyright (C) 2021 Free Software Foundation, Inc.

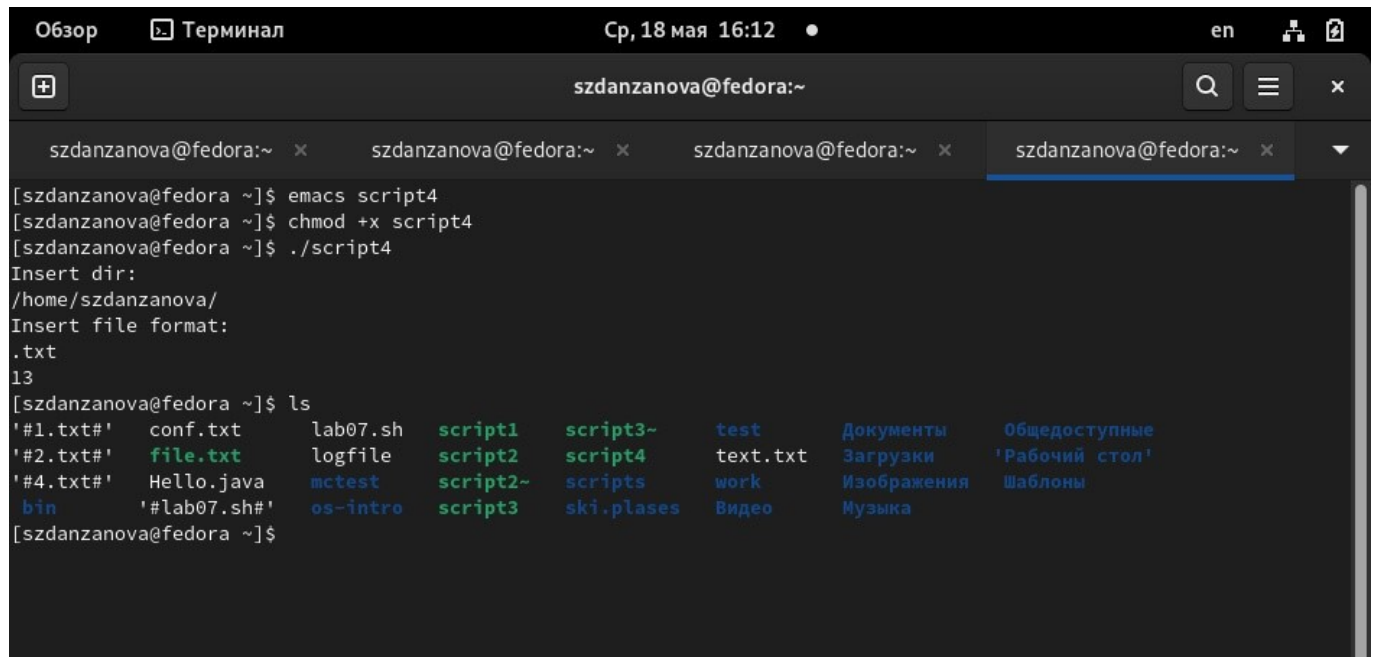
An auto-save file list was found. If an Emacs session crashed recently,
type M-x recover-session RET to recover the files you were editing.
Dismiss this startup screen [X] Never show it again.

U:%%- *GNU Emacs* All L3 (Fundamental)
For information about GNU Emacs and the GNU system. type C-h C-a.

```

#fig:006 width=90% }

4. Написали командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки.



The image shows a terminal window titled "Обзор Терминал" with the date and time "Ср, 18 мая 16:12". The user is logged in as "szdanzanova@fedora:~". The terminal shows the following commands and output:

```
[szdanzanova@fedora ~]$ emacs script4
[szdanzanova@fedora ~]$ chmod +x script4
[szdanzanova@fedora ~]$ ./script4
Insert dir:
/home/szdanzanova/
Insert file format:
.txt
13
[szdanzanova@fedora ~]$ ls
```

'#1.txt#'	conf.txt	lab07.sh	script1	script3~	test	Документы	Общедоступные
'#2.txt#'	file.txt	logfile	script2	script4	text.txt	Загрузки	'Рабочий стол'
'#4.txt#'	Hello.java	mctest	script2~	scripts	work	Изображения	Шаблоны
bin	'#lab07.sh#'	os-intro	script3	ski.places	Видео	Музыка	

```
[szdanzanova@fedora ~]$
```

{ #fig:007 width=90% }

```

emacs@fedora
File Edit Options Buffers Tools Help
[Icons: New, Open, Save, Undo, Cut, Copy, Paste, Find]
echo "Insert dir: "
read dirr
echo "Insert file format: "
read form
find $dirr -name "$form" -type f | wc -l

--:--- script4 All L1 (Fundamental)
Welcome to GNU Emacs, one component of the GNU/Linux operating system.

[Emacs Tutorial] Learn basic keystroke commands (Учебник Emacs)
[Emacs Guided Tour] Overview of Emacs features at gnu.org
[View Emacs Manual] View the Emacs manual using Info
[Absence of Warranty] GNU Emacs comes with ABSOLUTELY NO WARRANTY
[Copying Conditions] Conditions for redistributing and changing Emacs
[Ordering Manuals] Purchasing printed copies of manuals
To quit a partially entered command, type Control-g.

This is GNU Emacs 27.2 (build 1, x86_64-redhat-linux-gnu, GTK+ Version 3.24.30, cairo version 1.17.4)
of 2021-08-07
Copyright (C) 2021 Free Software Foundation, Inc.

An auto-save file list was found. If an Emacs session crashed recently,
type M-x recover-session RET to recover the files you were editing.
Dismiss this startup screen ☐ Never show it again.

U:%%~ *GNU Emacs* All L3 (Fundamental)
For information about GNU Emacs and the GNU system, type C-h C-a.

```

#fig:008 width=90% }

Вывод

Изучили основы программирования в оболочке ОС UNIX/Linux. Научились писать небольшие командные файлы.

Контрольные вопросы

1. Командные процессоры или оболочки - это программы, позволяющие пользователю взаимодействовать с компьютером. Их можно рассматривать как настоящие интерпретируемые языки, которые воспринимают команды пользователя и обрабатывают их. Поэтому командные

процессоры также называют интерпретаторами команд. На языках оболочек можно писать программы и выполнять их подобно любым другим программам. UNIX обладает большим количеством оболочек. Наиболее популярными являются следующие четыре оболочки: – оболочка Борна (Bourne) – первоначальная командная оболочка UNIX: базовый, но полный набор функций; –C-оболочка – добавка университета Беркли к коллекции оболочек: она надстраивается над оболочкой Борна, используя Сподобный синтаксис команд, и сохраняет историю выполненных команд; –оболочка Корна – 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).

2. POSIX (Portable Operating System Interface for Computer Environments)- интерфейс переносимой операционной системы для компьютерных сред. Представляет собой набор стандартов, подготовленных институтом инженеров по электронике и радиотехнике (IEEE), который определяет различные аспекты построения операционной системы. POSIX включает такие темы, как программный интерфейс, безопасность, работа с сетями и графический интерфейс. POSIX-совместимые оболочки являются будущим поколением оболочек UNIX и других ОС. Windows NT рекламируется как система, удовлетворяющая POSIX-стандартам. POSIX-совместимые оболочки разработаны на базе оболочки Корна; фонд бесплатного программного обеспечения (Free Software Foundation) работает над тем, чтобы и оболочку BASH сделать POSIX-совместимой.
3. Командный процессор bash обеспечивает возможность использования переменных типа строка символов. Имена переменных могут быть выбраны пользователем. Пользователь имеет возможность присвоить переменной значение некоторой строки символов. Например, команда `mark=/usr/andy/bin` присваивает значение строки символов `/usr/andy/bin` переменной `mark` типа строка символов. Значение, присвоенное некоторой переменной, может быть впоследствии использовано. Для этого в соответствующем месте командной строки должно быть употреблено имя этой переменной, которому предшествует метасимвол `$`. Например, команда `mv afile $mark` переместит файл `afile` из текущего каталога в каталог с абсолютным полным именем `/usr/andy/bin`. Использование значения, присвоенного некоторой переменной, называется подстановкой. Для того, чтобы имя переменной не сливалось с символами, которые могут следовать за ним в командной строке, при подстановке в общем случае используется следующая форма записи: `${имя переменной}` например, использование команд `b=/tmp/andy-ls -l myfile > ${b}ls` приведет к переназначению стандартного вывода команды `ls` с терминала на файл `/tmp/andy-ls`, а использование команд `ls -l > $bls` приведет к подстановке в командную строку значения переменной `bls`. Если переменной `bls` не было предварительно присвоено никакого значения, то ее значением является символ пробел. Оболочка bash позволяет создание массивов. Для создания массива используется команда `set` с флагом `-A`. За флагом следует имя переменной, а затем список значений, разделенных пробелом. Например, `set -A states Delaware Michigan "New Jersey"` Далее можно сделать добавление в массив, например, `states[49]=Alaska`. Индексация массивов начинается с нулевого элемента.
4. Команда `let` является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Простейшее выражение – это единичный терм (term), обычно целочисленный. Целые числа можно записывать как последовательность цифр или в любом базовом формате. Этот формат — `radix#number`, где `radix` (основание системы счисления) – любое число не более 26. Для большинства команд основания систем счисления это – 2 (двоичная), 8 (восьмеричная) и 16 (шестнадцатеричная). Простейшими математическими

выражениями являются сложение (+), вычитание (-), умножение (*), целочисленное деление (/) и целочисленный остаток (%). Команда `let` берет два операнда и присваивает их переменной.

5. Какие арифметические операции можно применять в языке программирования `bash`?
Оператор Синтаксис Результат
`!` `expr` Если `expr` равно 0, возвращает 1; иначе 0
`!=` `expr1 != expr2` Если `expr1` не равно `expr2`, возвращает 1; иначе 0
`%` `expr1 % expr2` Возвращает остаток от деления `expr1` на `expr2`
`%=` `var %= expr` Присваивает остаток от деления `var` на `expr` переменной `var`
`&` `expr1 & expr2` Возвращает побитовое AND выражений `expr1` и `expr2`
`&&` `expr1 && expr2` Если и `expr1` и `expr2` не равны нулю, возвращает 1; иначе 0
`&=` `var &= expr` Присваивает `var` побитовое AND переменных `var` и выражения `expr`
`*` `expr1 * expr2` Умножает `expr1` на `expr2`
`*=` `var *= expr` Умножает `expr` на значение `var` и присваивает результат переменной `var`
`+` `expr1 + expr2` Складывает `expr1` и `expr2`
`+=` `var += expr` Складывает `expr` со значением `var` и результат присваивает `var`
`-` `expr` Операция отрицания `expr` (называется унарный минус)
`-` `expr1 - expr2` Вычитает `expr2` из `expr1`
`-=` `var -= expr` Вычитает `expr` из значения `var` и присваивает результат `var`
`/` `expr1 / expr2` Делит `expr1` на `expr2`
`/=` `var /= expr` Делит `var` на `expr` и присваивает результат `var`
`<` `expr1 < expr2` Если `expr1` меньше, чем `expr2`, возвращает 1, иначе возвращает 0
`<=` `expr1 <= expr2` Сдвигает `expr1` влево на `expr2` бит
`<=` `var <= expr` Побитовый сдвиг влево значения `var` на `expr`
`<=` `expr1 <= expr2` Если `expr1` меньше, или равно `expr2`, возвращает 1; иначе возвращает 0
`=` `var = expr` Присваивает значение `expr` переменной `var`
`==` `expr1 == expr2` Если `expr1` равно `expr2`. Возвращает 1; иначе возвращает 0
`>` `expr1 > expr2` 1 если `expr1` больше, чем `expr2`; иначе 0
`>=` `expr1 >= expr2` 1 если `expr1` больше, или равно `expr2`; иначе 0
`>=` `expr >= expr2` Сдвигает `expr1` вправо на `expr2` бит
`>=` `var >= expr` Побитовый сдвиг вправо значения `var` на `expr`
`^` `expr1 ^ expr2` Исключающее OR выражений `expr1` и `expr2`
`^=` `var ^= expr` Присваивает `var` побитовое исключающее OR `var` и `expr`
`|` `expr1 | expr2` Побитовое OR выражений `expr1` и `expr2`
`|=` `var |= expr` Присваивает `var` «исключающее OR» переменной `var` и выражения `expr`
`||` `expr1 || expr2` 1 если или `expr1` или `expr2` являются ненулевыми значениями; иначе 0
`~` `~expr` Побитовое дополнение до `expr`
6. Условия оболочки `bash`, в двойные скобки `--(())`.
7. Имя переменной (идентификатор) — это строка символов, которая отличает эту переменную от других объектов программы (идентифицирует переменную в программе). При задании имен переменным нужно соблюдать следующие правила:
\$ первым символом имени должна быть буква. Остальные символы — буквы и цифры (прописные и строчные буквы различаются).
Можно использовать символ «_»; \$ в имени нельзя использовать символ «.»; \$ число символов в имени не должно превышать 255; \$ имя переменной не должно совпадать с зарезервированными (служебными) словами языка. `Var1`, `PATH`, `trash`, `mon`, `day`, `PS1`, `PS2`
Другие стандартные переменные: `HOME` — имя домашнего каталога пользователя. Если команда `cd` вводится без аргументов, то происходит переход в каталог, указанный в этой переменной. `IFS` — последовательность символов, являющихся разделителями в командной строке. Это символы пробел, табуляция и перевод строки (new line). `MAIL` — командный процессор каждый раз перед выводом на экран промптера проверяет содержимое файла, имя которого указано в этой переменной, и если содержимое этого файла изменилось с момента последнего ввода из него, то перед тем как вывести на терминал промптер, командный процессор выводит на терминал сообщение `You have mail` (у Вас есть почта). `TERM` — тип используемого терминала. `LOGNAME` — содержит регистрационное имя пользователя, которое устанавливается автоматически при входе в систему. В командном процессоре `Си` имеется еще несколько стандартных переменных. Значение всех переменных можно просмотреть с помощью команды `set`.
8. Такие символы, как `'`, `<`, `>`, `*`, `?`, `|`, `\`, `"`, `&` являются метасимволами и имеют для командного процессора специальный смысл.

9. Снятие специального смысла с метасимвола называется экранированием метасимвола. Экранирование может быть осуществлено с помощью предшествующего метасимволу символа, который, в свою очередь, является метасимволом. Для экранирования группы метасимволов, ее нужно заключить в одинарные кавычки. Строка, заключенная в двойные кавычки, экранирует все метасимволы, кроме \$, ' , , ". Например, `echo выведет на экран символ, echo ab'|'cd` выдаст строку `ab*|*cd`.
10. Последовательность команд может быть помещена в текстовый файл. Такой файл называется командным. Далее этот файл можно выполнить по команде `bash командный_файл [аргументы]`. Чтобы не вводить каждый раз последовательности символов `bash`, необходимо изменить код защиты этого командного файла, обеспечив доступ к этому файлу по выполнению. Это может быть сделано с помощью команды `chmod +x имя_файла`. Теперь можно вызывать свой командный файл на выполнение просто, вводя его имя с терминала так, как будто он является выполняемой программой. Командный процессор распознает, что в Вашем файле на самом деле хранится не выполняемая программа, а программа, написанная на языке программирования оболочки, и осуществит ее интерпретацию.
11. Группу команд можно объединить в функцию. Для этого существует ключевое слово `function`, после которого следует имя функции и список команд, заключенных в фигурные скобки. Удалить функцию можно с помощью команды `unset` с флагом `-f`. Команда `typeset` имеет четыре опции для работы с функциями: `-f` — перечисляет определенные на текущий момент функции; `--ft` — при последующем вызове функции иницирует ее трассировку; `--fx` — экспортирует все перечисленные функции в любые дочерние программы оболочки; `--fu` — обозначает указанные функции как автоматически загружаемые. Автоматически загружаемые функции хранятся в командных файлах, а при их вызове оболочка просматривает переменную `FPATH`, отыскивая файл с одноименными именами функций, загружает его и вызывает эти функции.
12. `ls -lrt` Если есть `d`, то является файл каталогом
13. Используется команда `set` с флагом `-A`. За флагом следует имя переменной, а затем список значений, разделенных пробелом. Например, `set -A states Delaware Michigan "New Jersey"`. Далее можно сделать добавление в массив, например, `states[49]=Alaska`. Индексация массивов начинается с нулевого элемента. В командном процессоре `Си` имеется еще несколько стандартных переменных. Значение всех переменных можно просмотреть с помощью команды `set`. Наиболее распространенным является сокращение, избавляющееся от слова `let` в программах оболочки. Если объявить переменные целыми значениями, любое присвоение автоматически трактуется как арифметическое. Используйте `typeset -i` для объявления и присвоения переменной, и при последующем использовании она становится целой. Или можете использовать ключевое слово `integer` (псевдоним для `typeset -i`) и объявлять переменные целыми. Таким образом, выражения типа `x=y+z` воспринимаются как арифметические. Группу команд можно объединить в функцию. Для этого существует ключевое слово `function`, после которого следует имя функции и список команд, заключенных в фигурные скобки. Удалить функцию можно с помощью команды `unset` с флагом `-f`. Команда `typeset` имеет четыре опции для работы с функциями: `-f` — перечисляет определенные на текущий момент функции; `--ft` — при последующем вызове функции иницирует ее трассировку; `--fx` — экспортирует все перечисленные функции в любые дочерние программы оболочки; `--fu` — обозначает указанные функции как автоматически загружаемые. Автоматически загружаемые функции хранятся в командных файлах, а при их вызове оболочка просматривает переменную `FPATH`, отыскивая файл с одноименными именами функций, загружает его и вызывает эти функции. В переменные `mon` и `day` будут считаны соответствующие значения, введенные с клавиатуры, а переменная

trash нужна для того, чтобы отобрать всю избыточно введенную информацию и игнорировать ее. Изъять переменную из программы можно с помощью команды unset.