

Цель работы

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

Задание

1. Написать скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации необходимо узнать, изучив справку.
2. Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов.
3. Написать командный файл — аналог команды ls (без использования самой этой команды и команды dir). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.
4. Написать 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.

Выполнение лабораторной работы

1. Написать скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации необходимо узнать, изучив справку.

```
artemagarkov@fedora:~ — /usr/libexec/vi /home/artemagarkov...
artemagarkov@fedora:~ x artemagarkov@fedora:~ — /usr/li... x
#!/bin/bash

# Название самого скрипта
script_name=$(basename "$0")

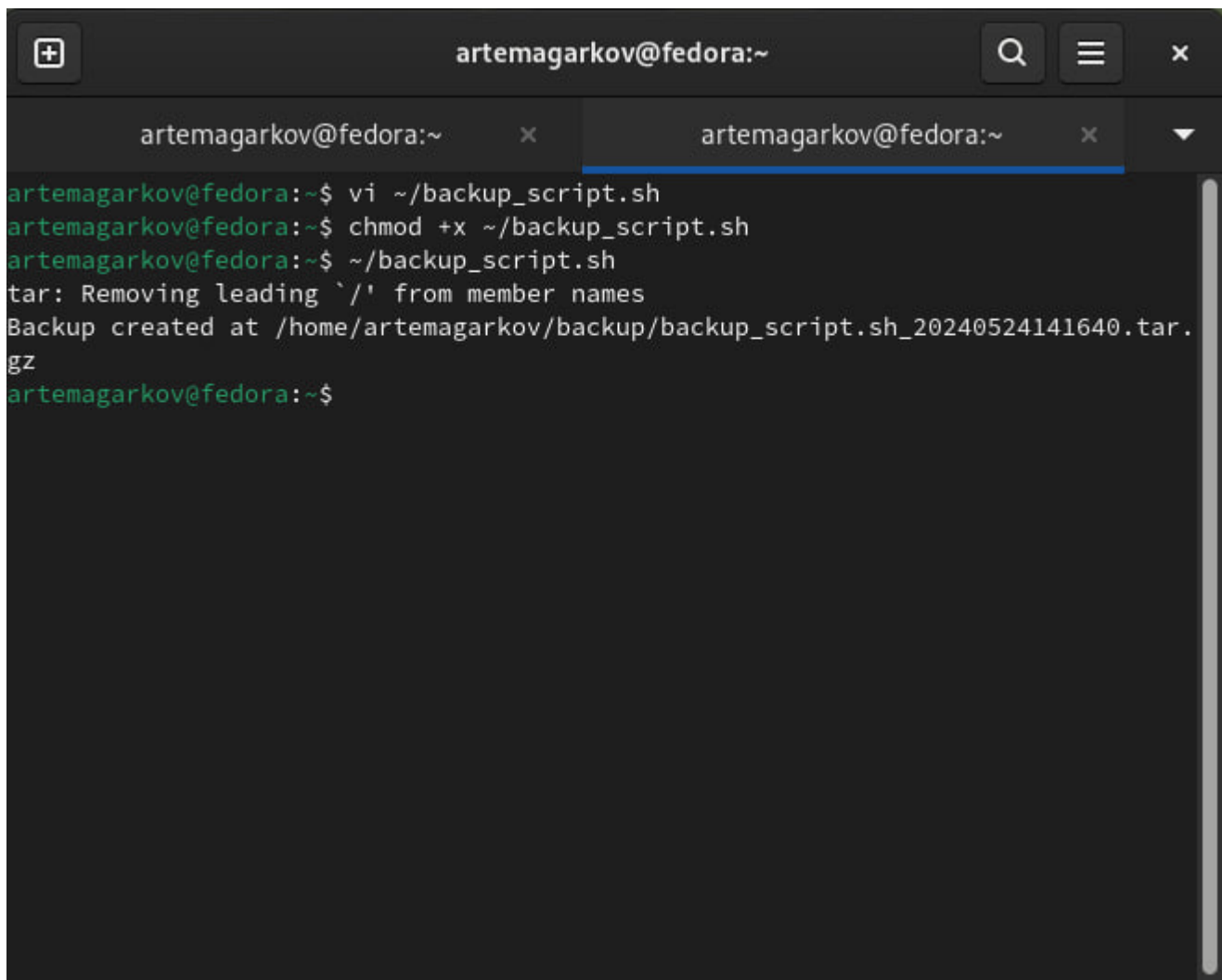
# Директория для резервной копии
backup_dir=~/.backup

# Создание директории, если её нет
mkdir -p "$backup_dir"

# Имя архива с текущей датой и временем
backup_file="$backup_dir/${script_name}_${date +%Y%m%d%H%M%S'}.tar.gz"

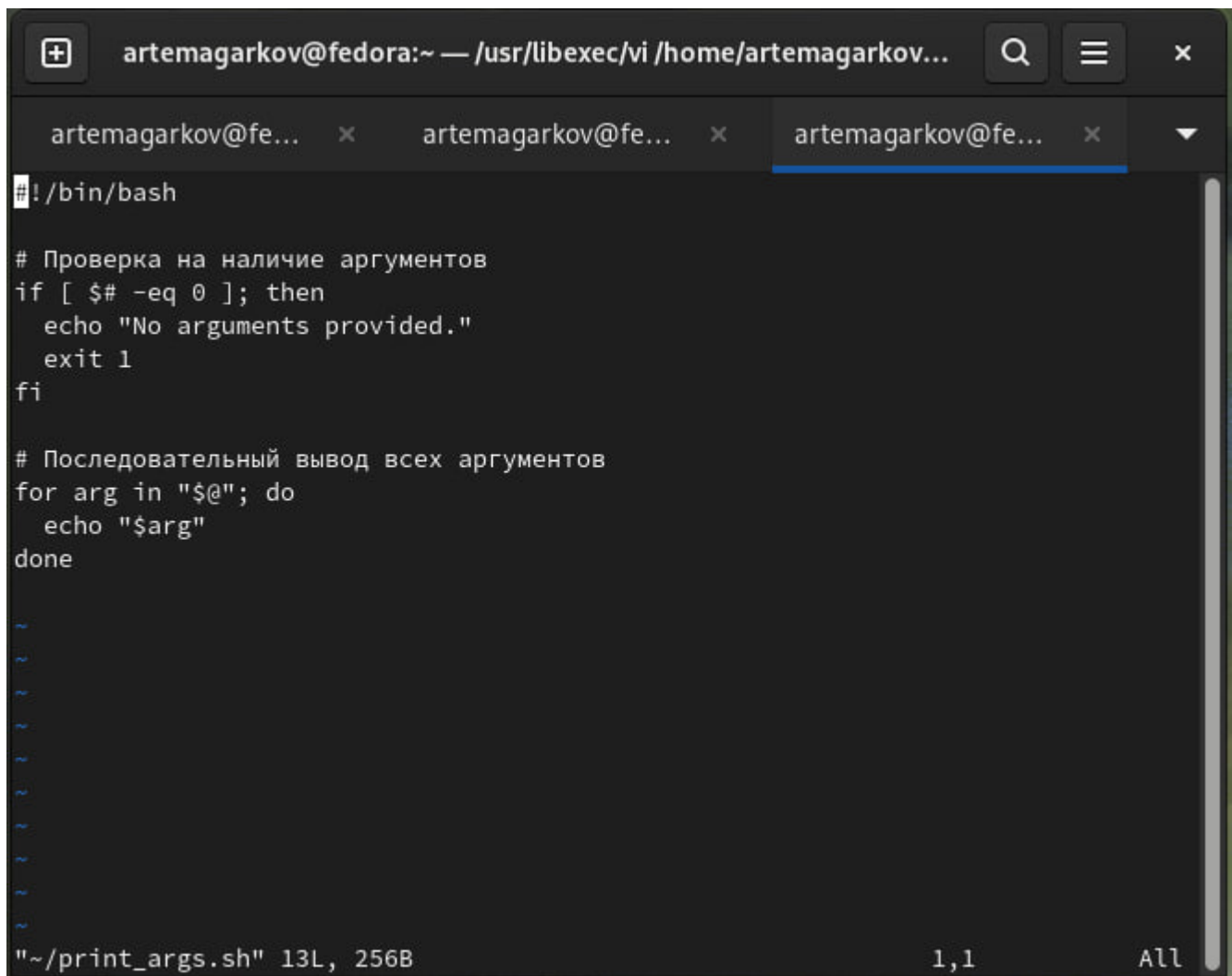
# Архивация самого скрипта
tar -czf "$backup_file" "$0"

echo "Backup created at $backup_file"
~
~
~
~
~
"/.backup_script.sh" 18L, 518B 1,1 All
```

A terminal window titled 'artemagarkov@fedora:~' with a search icon, a menu icon, and a close icon in the top right. The terminal shows the execution of a script. The prompt is 'artemagarkov@fedora:~\$'. The first command is 'vi ~/backup_script.sh'. The second command is 'chmod +x ~/backup_script.sh'. The third command is '~/backup_script.sh'. The output shows 'tar: Removing leading `/' from member names' and 'Backup created at /home/artemagarkov/backup/backup_script.sh_20240524141640.tar.gz'. The prompt returns to 'artemagarkov@fedora:~\$'.

```
artemagarkov@fedora:~$ vi ~/backup_script.sh
artemagarkov@fedora:~$ chmod +x ~/backup_script.sh
artemagarkov@fedora:~$ ~/backup_script.sh
tar: Removing leading `/' from member names
Backup created at /home/artemagarkov/backup/backup_script.sh_20240524141640.tar.gz
artemagarkov@fedora:~$
```

2. Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов.

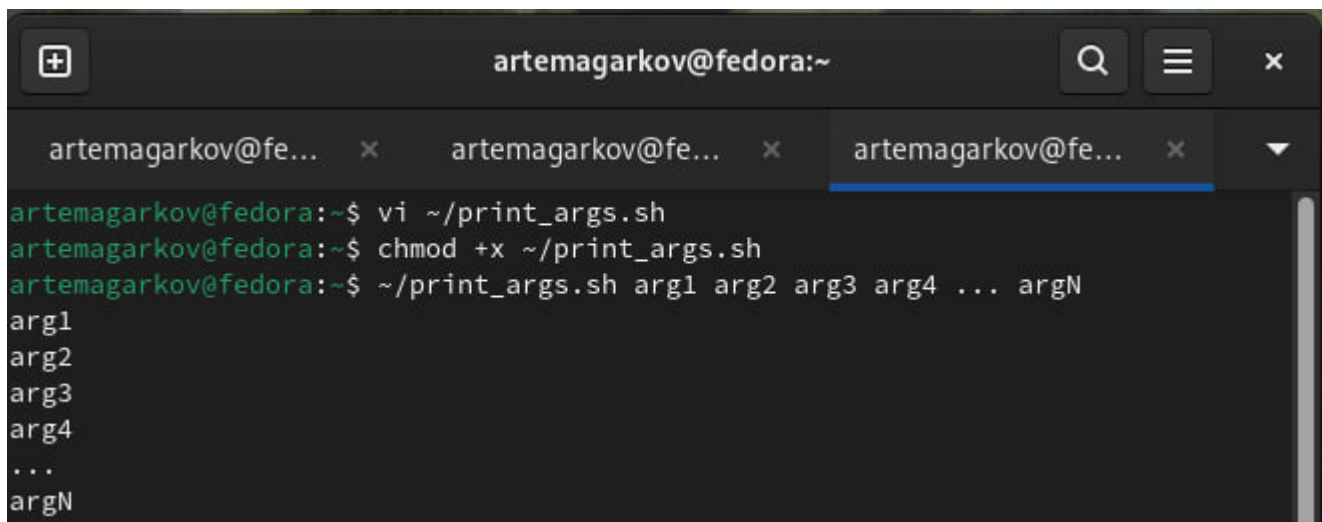


```
artemagarkov@fedora:~ — /usr/libexec/vi /home/artemagarkov...
artemagarkov@fe... x artemagarkov@fe... x artemagarkov@fe... x
#!/bin/bash

# Проверка на наличие аргументов
if [ $# -eq 0 ]; then
    echo "No arguments provided."
    exit 1
fi

# Последовательный вывод всех аргументов
for arg in "$@"; do
    echo "$arg"
done

~/print_args.sh" 13L, 256B 1,1 All
```



```
artemagarkov@fedora:~
artemagarkov@fe... x artemagarkov@fe... x artemagarkov@fe... x
artemagarkov@fedora:~$ vi ~/print_args.sh
artemagarkov@fedora:~$ chmod +x ~/print_args.sh
artemagarkov@fedora:~$ ~/print_args.sh arg1 arg2 arg3 arg4 ... argN
arg1
arg2
arg3
arg4
...
argN
```

3. Написать командный файл — аналог команды `ls` (без использования самой этой команды и команды `dir`). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.

```
artemagarkov@fedora:~ — /usr/libexec/vi /home/artemagarkov/custom_ls.sh

artemagarkov@f... x artemagarkov@f... x artemagarkov@f... x artemagarkov@f...

#!/bin/bash

# Если аргумент не передан, использовать текущий каталог
directory=${1:-.}

# Проверка, существует ли каталог
if [ ! -d "$directory" ]; then
    echo "$directory: No such directory"
    exit 1
fi

# Заголовок для вывода
printf "%-20s %-10s %-10s %-10s\n" "Name" "Size" "Permissions" "Type"

# Обработка каждого файла в каталоге
find "$directory" -maxdepth 1 | while read -r file; do
    # Имя файла
    name=$(basename "$file")

    # Размер файла
    size=$(stat -c%s "$file")

    # Права доступа
    permissions=$(stat -c%A "$file")

    # Тип файла
    if [ -d "$file" ]; then
        type="Directory"
    elif [ -f "$file" ]; then
        type="File"
    elif [ -L "$file" ]; then
        type="Symlink"
    else
        type="Other"
    fi

    # Вывод информации о файле
    printf "%-20s %-10s %-10s %-10s\n" "$name" "$size" "$permissions" "$type"
done

~
~
```

16,

```
artemagarkov@fedora:~

artemagarkov@fed... x artemagarkov@fed... x artemagarkov@fed... x artemagarkov@fed...

artemagarkov@fedora:~$ chmod +x ~/custom_ls.sh
artemagarkov@fedora:~$ ~/custom_ls.sh ~/
Name                Size      Permissions Type
artemagarkov        1204      drwx----- Directory
.mozilla            48        drwxr-xr-x  Directory
```

```

.bash_logout      18      -rw-r--r-- File
.bash_profile     144     -rw-r--r-- File
.bashrc           522     -rw-r--r-- File
.cache            362     drwx----- Directory
.config           364     drwxr-xr-x Directory
.local            20      drwx----- Directory
Desktop           0       drwxr-xr-x Directory
Downloads         0       drwxr-xr-x Directory
Templates         0       drwxr-xr-x Directory
Public            0       drwxr-xr-x Directory
Documents         0       drwxr-xr-x Directory
Music             0       drwxr-xr-x Directory
Pictures          0       drwxr-xr-x Directory
Videos            0       drwxr-xr-x Directory
.vboxclient-clipboard-tty2-control.pid 5      -rw-r----- File
.vboxclient-clipboard-tty2-service.pid 5      -rw-r----- File
.vboxclient-hostversion-tty2-control.pid 5    -rw-r----- File
.vboxclient-seamless-tty2-control.pid 5      -rw-r----- File
.vboxclient-seamless-tty2-service.pid 5      -rw-r----- File
.vboxclient-draganddrop-tty2-control.pid 5    -rw-r----- File
.vboxclient-draganddrop-tty2-service.pid 5    -rw-r----- File
.bash_history     2763    -rw----- File
.gitconfig        334     -rw-r--r-- File
.ssh              32      drwx----- Directory
.gnupg            136     drwx----- Directory
course-directory-student-template 274      drwxr-xr-x Directory
test              0       -rw-r--r-- File
ski.plases        28      drwxr-xr-x Directory
abc1              0       -rw-r--r-- File
australia         0       -rwxr--r-- File
my_os             0       -rwxr--r-- File
feathers          0       -rw-rw-r-- File
play              26      drwxr-xr-x Directory
file.txt          2807    -rw-r--r-- File
conf.txt          621     -rw-r--r-- File
logfile1          0       -rw-r--r-- File
logfile2          0       -rw-r--r-- File
logfile3          0       -rw-r--r-- File
logfile           111     -rw-r--r-- File
text.c            12      drwxr-xr-x Directory
work              4       drwxr-xr-x Directory
backup            152     drwxr-xr-x Directory
backup_script.sh  518     -rwxr-xr-x File
print_args.sh     256     -rwxr-xr-x File
custom_ls.sh      1017    -rwxr-xr-x File
artemagarkov@fedora:~$

```

4. Написать 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.

```
artemagarkov@fedora:~ — /usr/libexec/vi count_f
```

```
#!/bin/bash

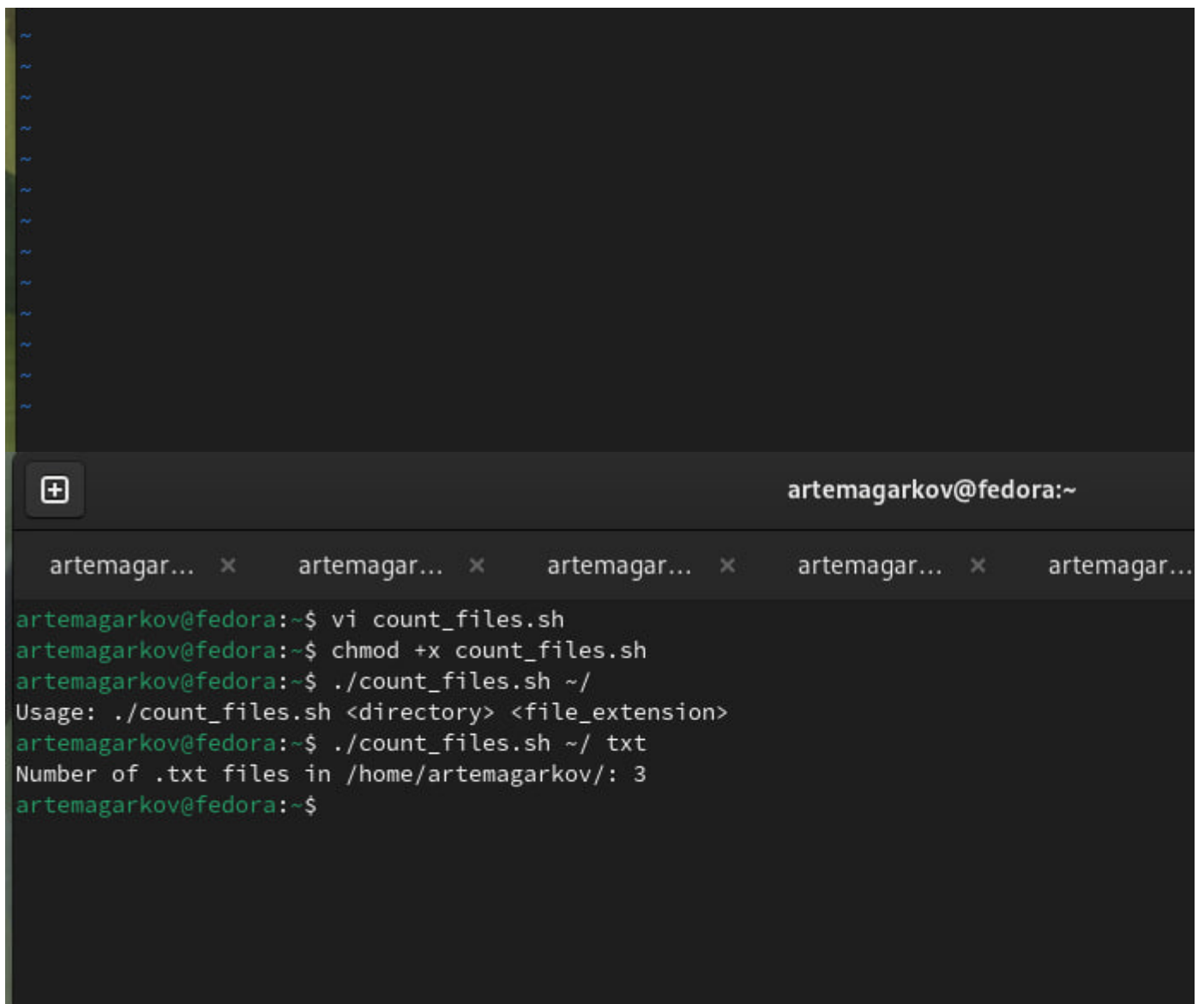
# Проверка количества аргументов
if [ "$#" -ne 2 ]; then
    echo "Usage: $0 <directory> <file_extension>"
    exit 1
fi

# Извлечение аргументов
directory="$1"
file_extension="$2"

# Проверка на существование директории
if [ ! -d "$directory" ]; then
    echo "$directory: No such directory"
    exit 1
fi

# Подсчет файлов с указанным расширением
count=$(find "$directory" -type f -name ".*$file_extension" | wc -l)

# Вывод результата
echo "Number of .$file_extension files in $directory: $count"
```

The image shows a terminal window with a dark background. At the top, there is a title bar with a '+' icon on the left and the text 'artemagarkov@fedora:~' on the right. Below the title bar, there are five tabs, each labeled 'artemagar...' followed by a close icon. The main area of the terminal displays the following commands and output:

```
artemagarkov@fedora:~$ vi count_files.sh
artemagarkov@fedora:~$ chmod +x count_files.sh
artemagarkov@fedora:~$ ./count_files.sh ~/
Usage: ./count_files.sh <directory> <file_extension>
artemagarkov@fedora:~$ ./count_files.sh ~/ txt
Number of .txt files in /home/artemagarkov/: 3
artemagarkov@fedora:~$
```

10.5. Контрольные вопросы и ответы

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются?
 - Командная оболочка - это интерактивная программа, которая позволяет пользователю взаимодействовать с операционной системой путем ввода команд. Примеры: Bash, Zsh, Fish. Они отличаются по функциональности и синтаксису.
2. Что такое POSIX?
 - POSIX (Portable Operating System Interface) - это набор стандартов, разработанных для обеспечения совместимости между различными операционными системами UNIX.
3. Как определяются переменные и массивы в языке программирования bash?

- Переменные и массивы определяются в языке программирования Bash путем присваивания значений с помощью оператора `=`. Например: `variable=value` для переменной и `array=(value1 value2 value3)` для массива.

4. Каково назначение операторов `let` и `read`?

- Оператор `let` используется для выполнения арифметических операций в контексте командного файла, а оператор `read` используется для чтения ввода пользователя и присваивания его переменной.

5. Какие арифметические операции можно применять в языке программирования `bash`?

- В языке программирования Bash можно использовать арифметические операции, такие как сложение `+`, вычитание `-`, умножение `*`, деление `/`, а также операции сравнения, такие как `>` и `<`.

6. Что означает операция `(())`?

- Операция `(())` используется для выполнения арифметических вычислений в контексте командного файла.

7. Какие стандартные имена переменных Вам известны?

- Некоторые стандартные имена переменных: `HOME` (домашний каталог пользователя), `PATH` (путь поиска для исполняемых файлов), `PWD` (текущий рабочий каталог).

8. Что такое метасимволы?

- Метасимволы - это символы, которые имеют специальное значение в командной оболочке, такое как `*`, `?`, `[` и `]`.

9. Как экранировать метасимволы?

- Метасимволы могут быть экранированы с помощью символа обратного следа `\`. Например, `\$` экранирует символ `$`.

10. Как создавать и запускать командные файлы?

- Для создания командного файла необходимо создать файл с исполняемыми правами (`chmod +x filename`) и добавить в него команды, которые вы хотите выполнить. Для запуска командного файла просто введите его имя в терминале.

11. Как определяются функции в языке программирования `bash`?

- Функции определяются в языке программирования `Bash` с помощью ключевого слова `function`, за которым следует имя функции и тело функции. Например: `function_name() { commands }`.

12. Каким образом можно выяснить, является файл каталогом или обычным файлом?

- Для определения, является ли файл каталогом или обычным файлом, можно использовать команду `test` с параметром `-d` для каталога и `-f` для файла.

13. Каково назначение команд `set`, `typeset` и `unset`?

- Команда `set` используется для установки или изменения значений параметров и настроек командной оболочки. Команда `typeset` используется для объявления переменных с заданными свойствами. Команда `unset` используется для удаления переменных или функций.

14. Как передаются параметры в командные файлы?

- Параметры передаются в командные файлы в виде аргументов командной строки. Они доступны внутри файла через специальные переменные под нумерованными индексами, такие как `$1`, `$2`, и так далее.

15. Назовите специальные переменные языка `bash` и их назначение.

- Некоторые специальные переменные `Bash`: `$0` (имя текущего исполняемого файла), `$#` (количество аргументов командной строки), `$?` (код возврата последней выполненной команды).

Выводы

В процессе выполнения работы я ознакомился с основными принципами работы в командной оболочке, научился создавать, редактировать и выполнять командные файлы. Полученные навыки позволят мне эффективно управлять операционной системой и автоматизировать

повседневные задачи.