

Лабораторная работа 11

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Петрова Мария Евгеньевна

Содержание

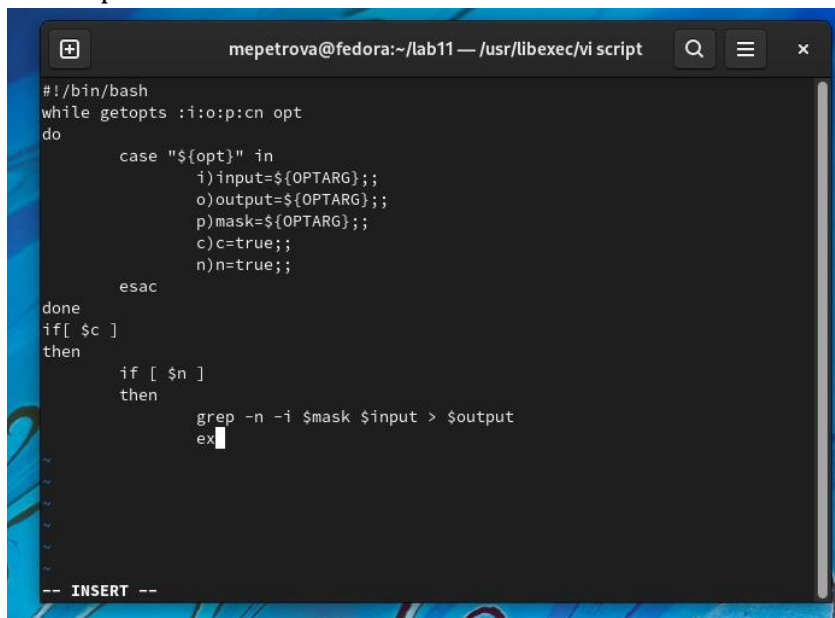
| | |
|--------------------------------------|---|
| Цель работы | 1 |
| Выполнение лабораторной работы | 1 |
| Вывод..... | 6 |
| Контрольные вопросы | 6 |

Цель работы

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

Выполнение лабораторной работы

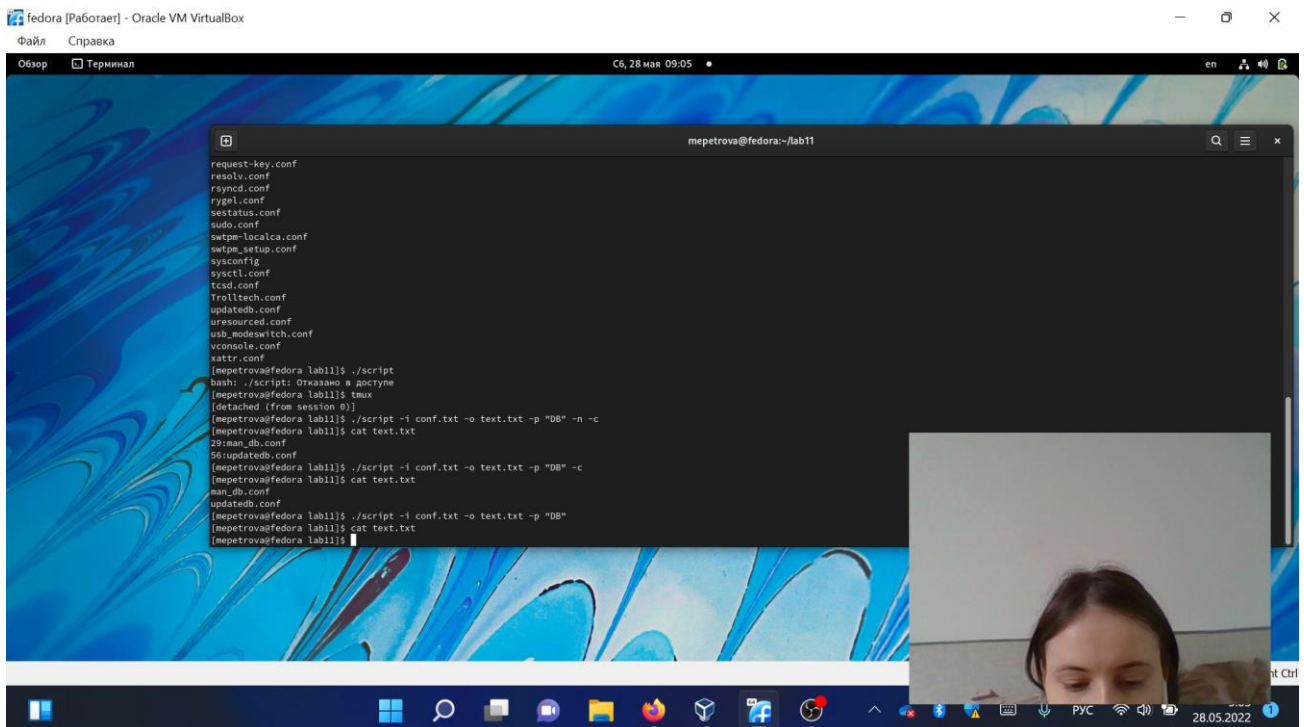
1. Скрипт 1



```
mepetrova@fedora:~/lab11 — /usr/libexec/vi script
#!/bin/bash
while getopts :i:o:p:cn opt
do
    case "${opt}" in
        i)input=${OPTARG};;
        o)output=${OPTARG};;
        p)mask=${OPTARG};;
        c)c=true;;
        n)n=true;;
    esac
done
if [ $c ]
then
    if [ $n ]
    then
        grep -n -i $mask $input > $output
    ex
-- INSERT --
```

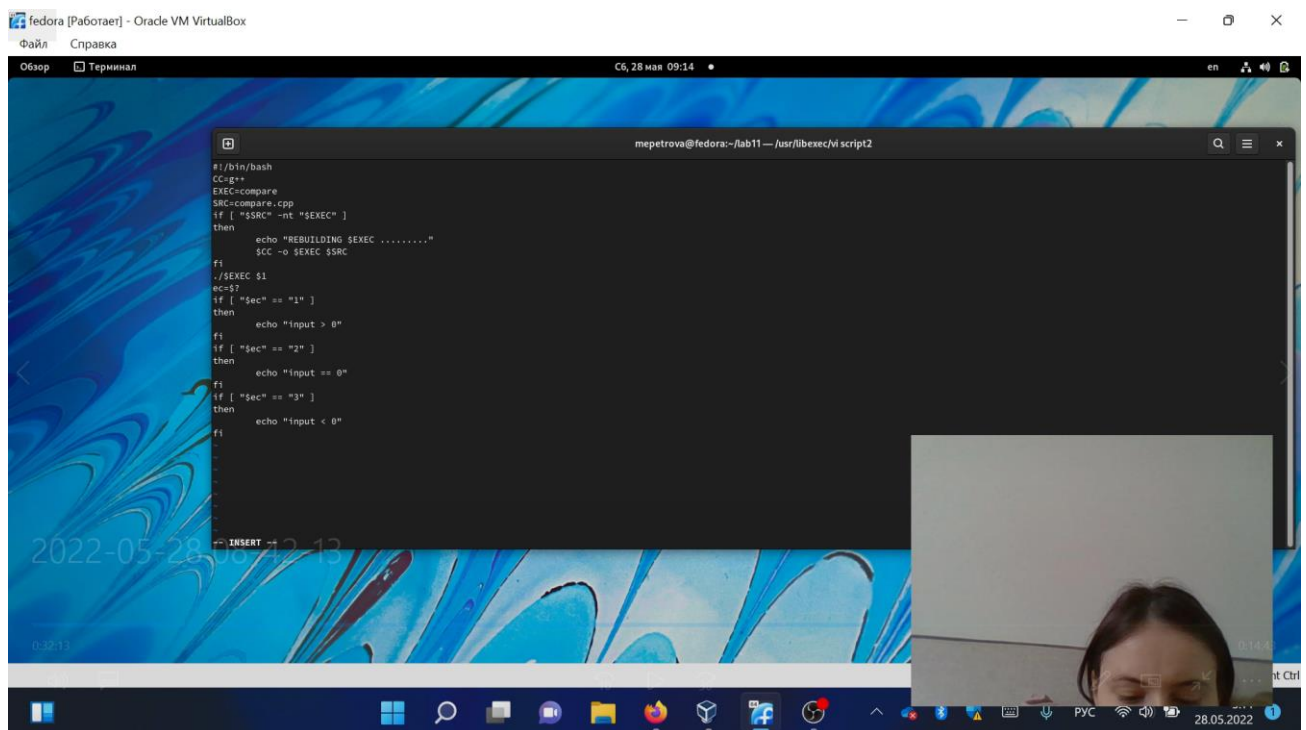
```
mepetrova@fedora:~/lab11 — /usr/libexec/vi script
i)input=${OPTARG};;
o)output=${OPTARG};;
p)mask=${OPTARG};;
c)c=true;;
n)n=true;;
    esac
done
if[ $c ]
then
    if [ $n ]
    then
        grep -n -i $mask $input > $output
        exit 0
    else
        grep -i $mask $input > $output
        exit 0
    fi
else
    grep -n $mask $input > $output
    exit 0
fi
~
-- INSERT --
```

Скринт 1

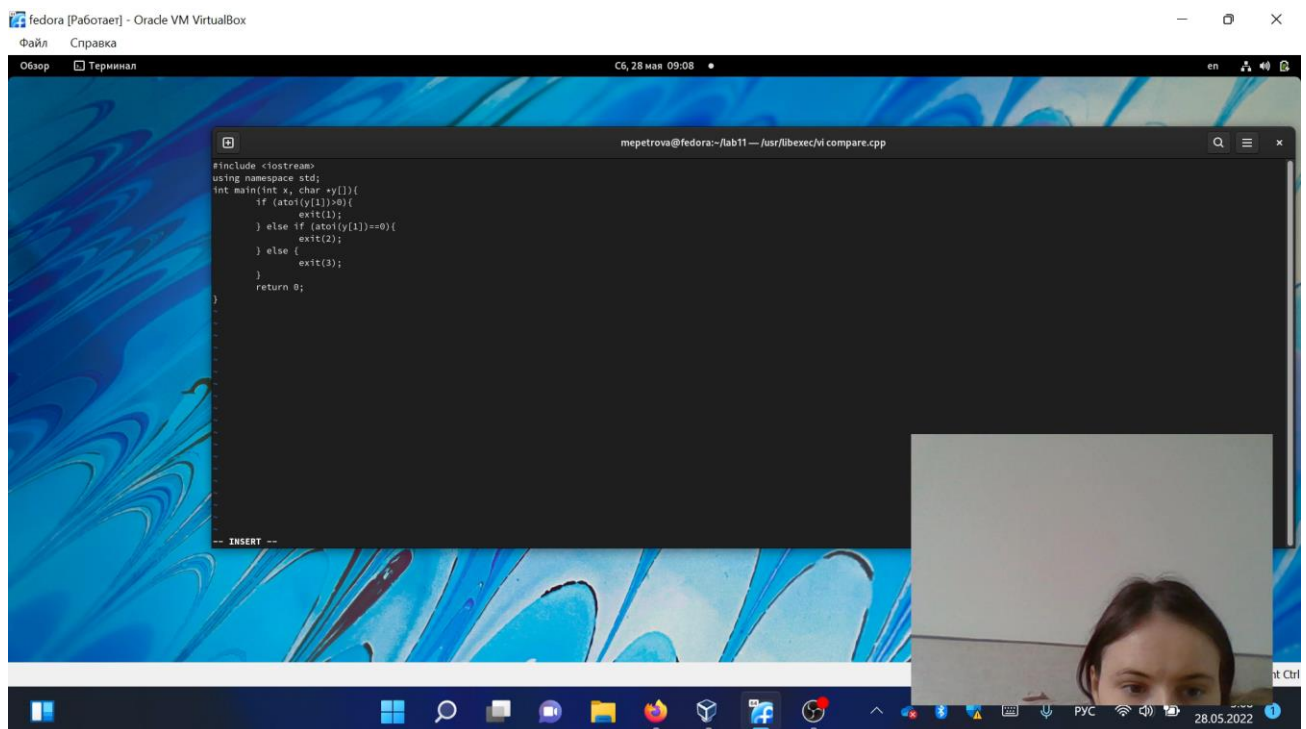


Скринт 1 запуск

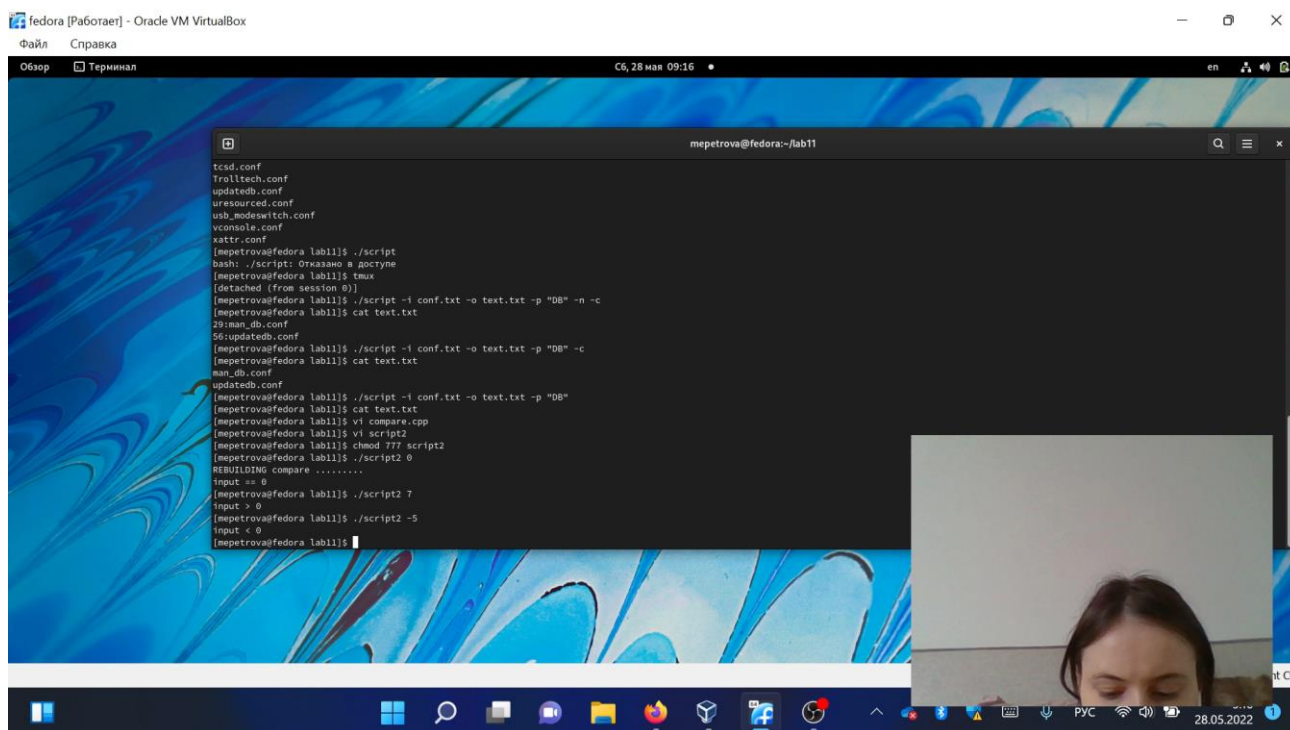
2. Скрипт 2



Скрипт 2

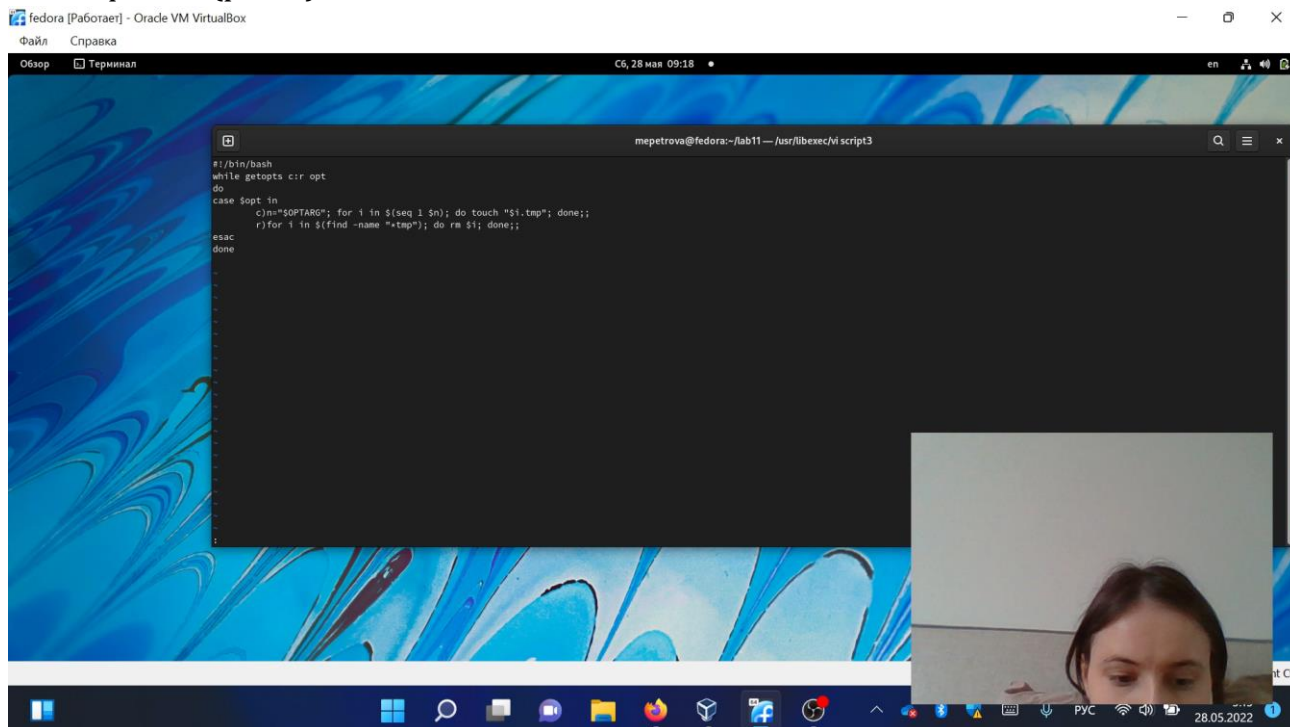


Запуск (рис. 4)



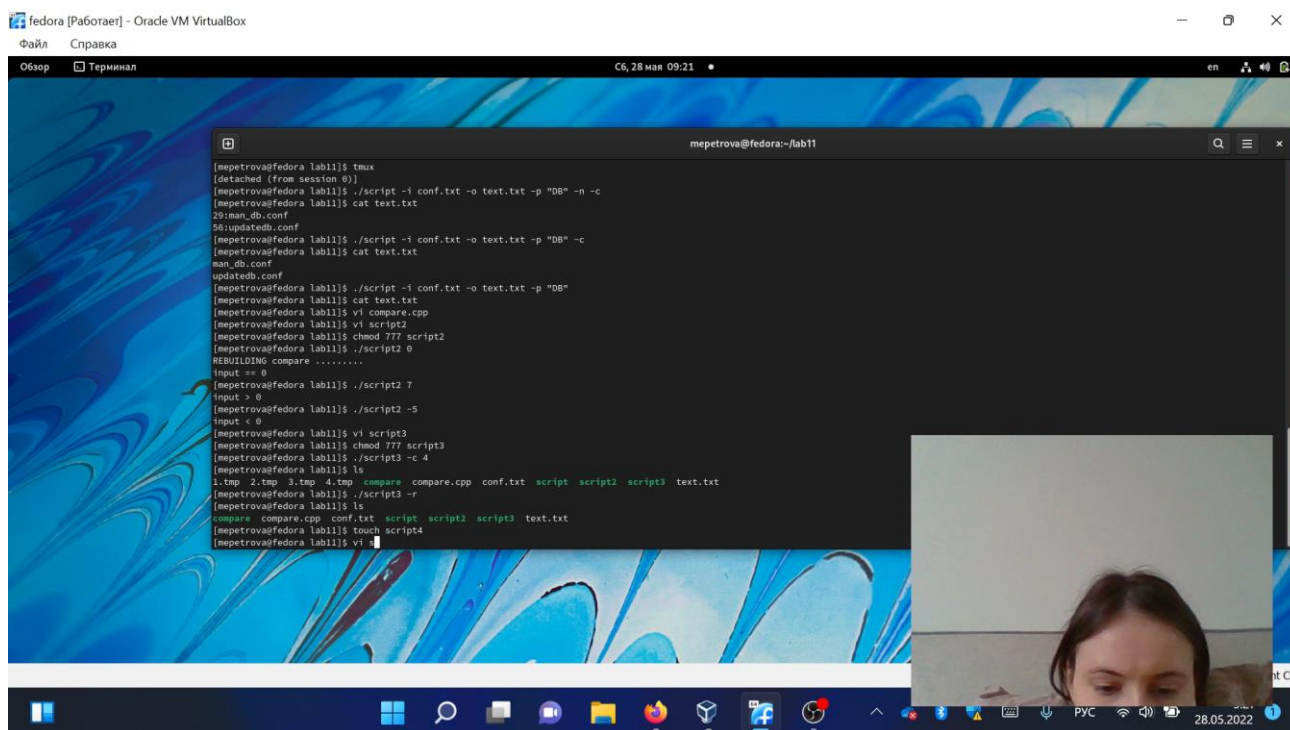
Скрипт 2 запуск

3. Скрипт 3 (рис. 5)



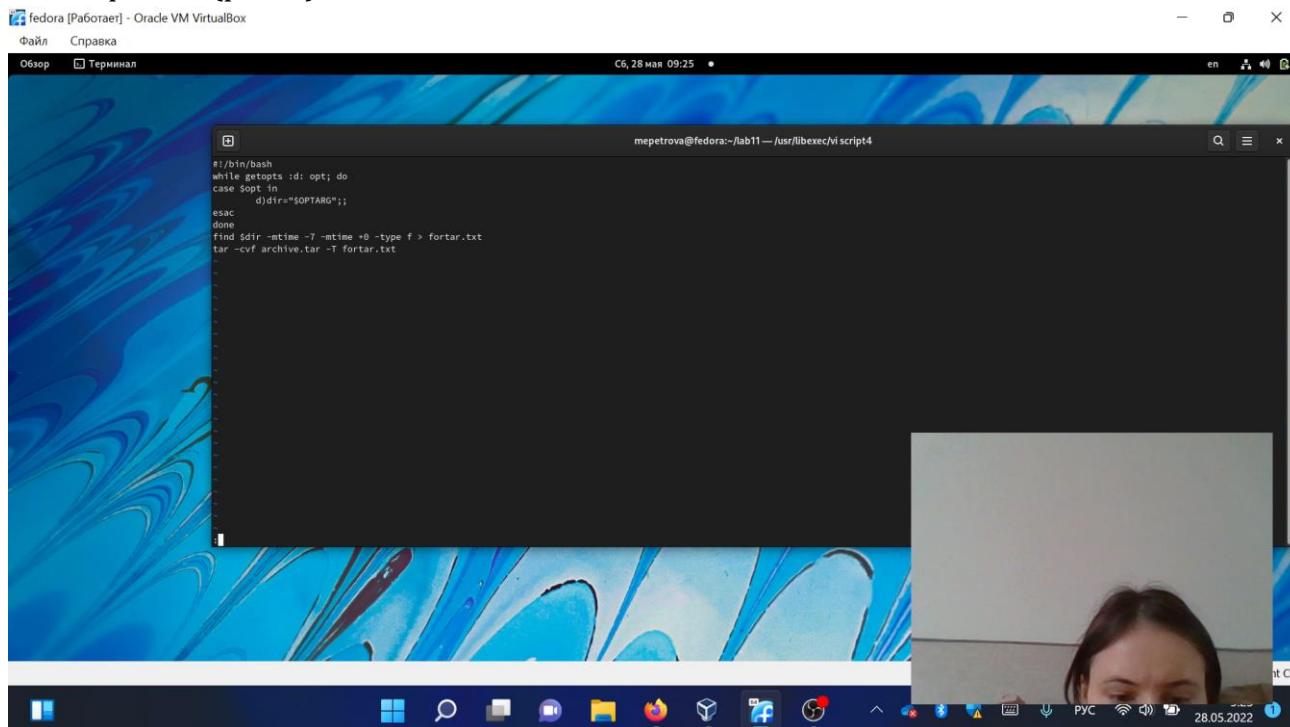
Скрипт 3

Запуск (рис. 6)



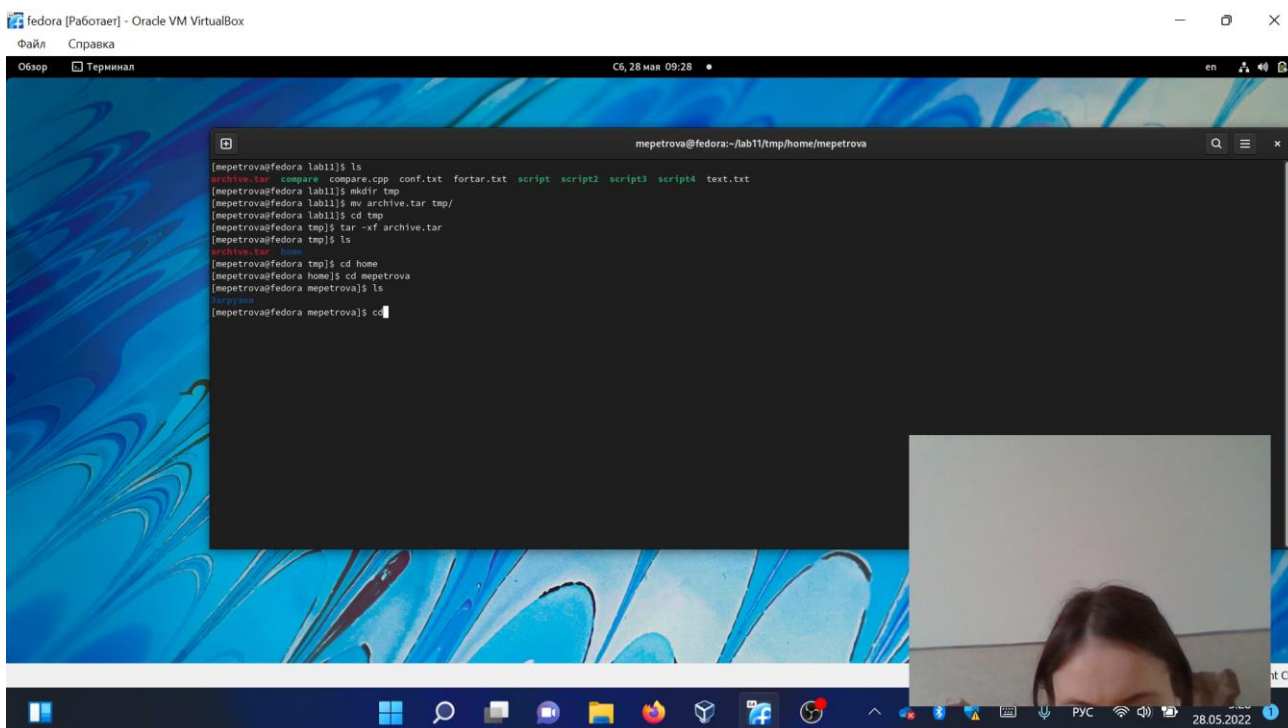
Скрипт 3 запуск

4. Скрипт 4 (рис. 7)



Скрипт 4

Запуск (рис. 8)



Скрипт 4 запуск

Вывод

Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Контрольные вопросы

1. Весьма необходимой при программировании является команда `getopts`, которая осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ...]` Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Предположим, необходимо распознать командную строку следующего формата: `testprog -ifile_in.txt -ofile_out.doc -L -t -r` Вот как выглядит использование оператора `getopts` в этом случае: `while getopts o:i:Ltr optletter do case $optletter in o) iflag=1; oval=$OPTARG;; i) iflag=1; ival=$OPTARG;; L) Lflag=1;; t) tflag=1;; r) rflag=1;; *) echo Illegal option $optletter esac done` Функция `getopts`

включает две специальные переменные среды – OPTARG и OPTIND. Если ожидается дополнительное значение, то OPTARG устанавливается в значение этого аргумента (будет равна file_in.txt для опции i и file_out.doc для опции o). OPTIND является числовым индексом на упомянутый аргумент. Функция getopt также понимает переменные типа массив, следовательно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

2. При перечислении имен файлов текущего каталога можно использовать следующие символы: `· *` — соответствует произвольной, в том числе и пустой строке; `· ?` — соответствует любому одному символу; `· [c1-c1]` — соответствует любому символу, лексикографически находящемуся между символами c1 и c2. `· echo *` — выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; `· ls .c` — выведет все файлы с последними двумя символами, равными `.c`; `· echo prog.?` — выдаст все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog.`; `· [a-z]` — соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.
3. Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости от результатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет Вам возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути дела являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда.
4. Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестает быть правильным. Пример бесконечного цикла `while`, с прерыванием в момент, когда файл перестает существовать: `while true do if [! -f $file] then break fi sleep 10 done`
5. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.
6. Введенная строка означает условие существования файла `mans/i.$s`
7. Если речь идет о 2-х параллельных действиях, то это `while`. когда мы показываем, что сначала делается 1-е действие. потом оно заканчивается при наступлении 2-го действия, применяем `until`.