

# Цель работы

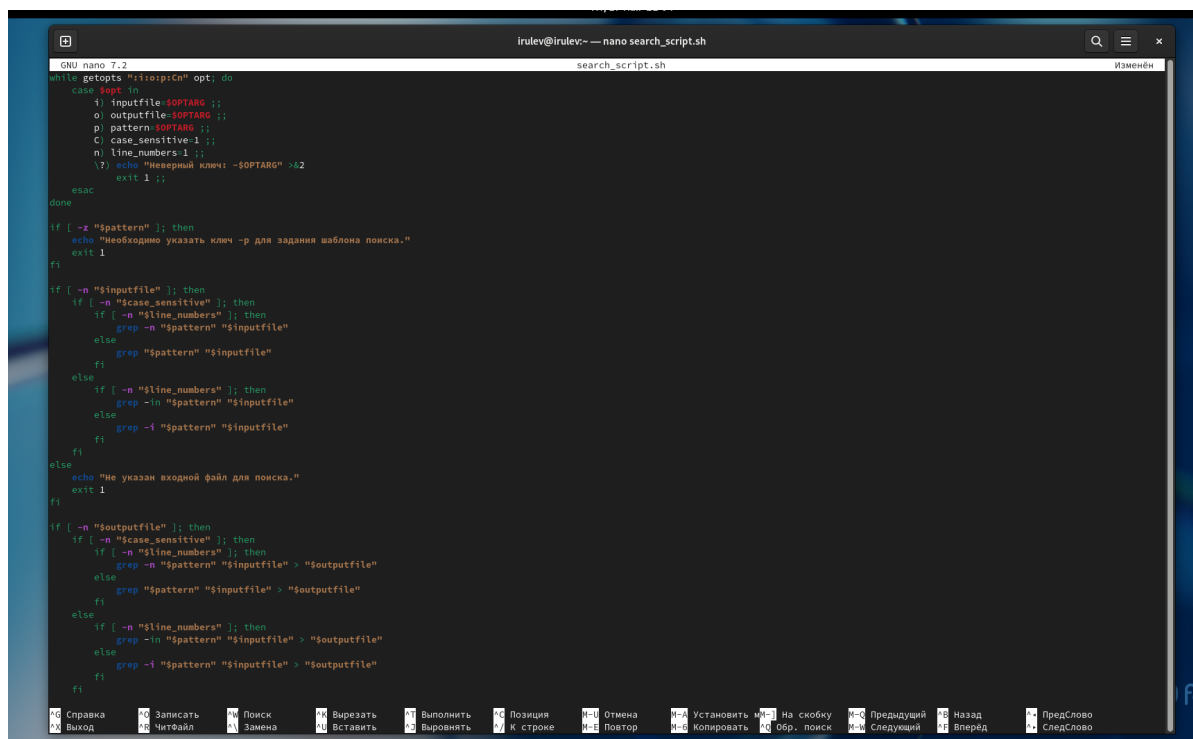
Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## Выполнение лабораторной работы

Используя команды `getopts` `grep`, написал командный файл, который анализирует командную строку с ключами:

- `-iinputfile` — прочитать данные из указанного файла;
- `-ooutputfile` — вывести данные в указанный файл;
- `-р`шаблон — указать шаблон для поиска;
- `-C` — различать большие и малые буквы;
- `-n` — выдавать номера строк.

А затем ищет в указанном файле нужные строки, определяемые ключом `-р`.



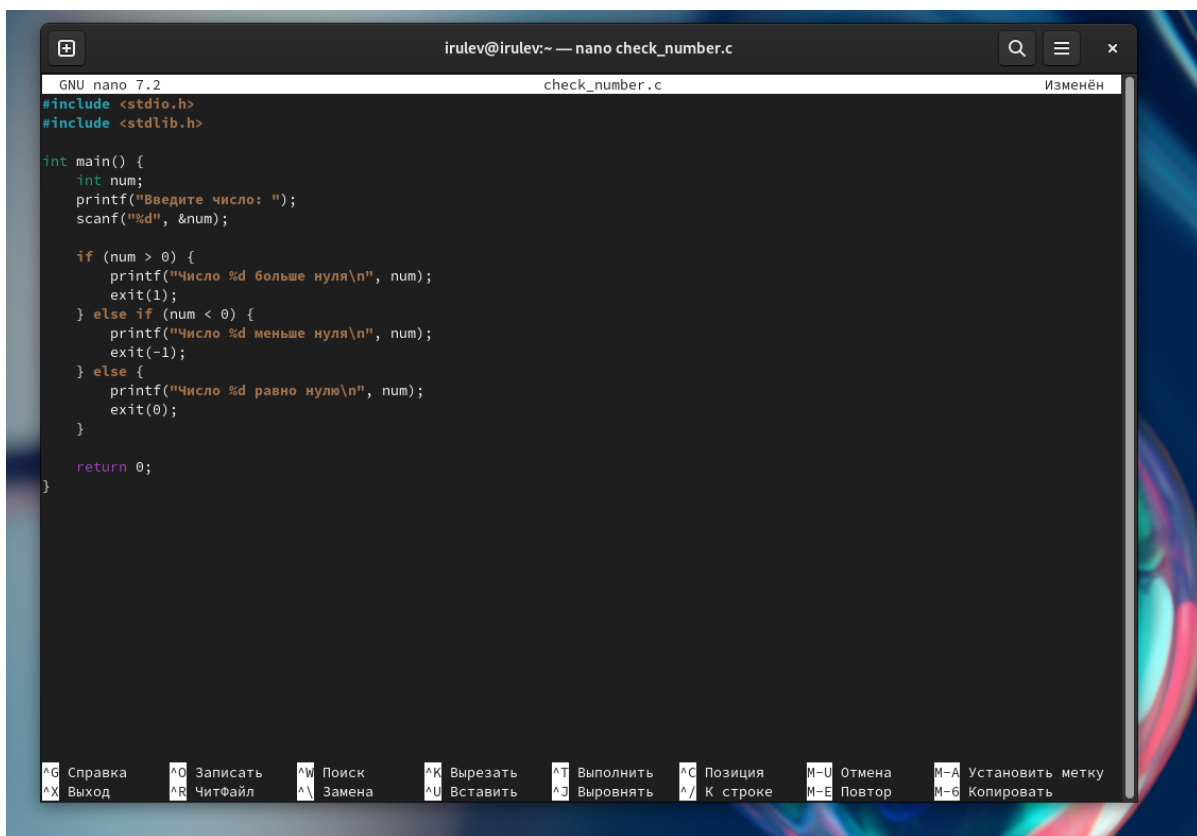
```
GNU nano 7.2 search_script.sh
#!/bin/bash
while getopts "i:op:C:n:" opt; do
  case $opt in
    i) inputfile=$OPTARG ;;
    o) outputfile=$OPTARG ;;
    p) pattern=$OPTARG ;;
    C) case_sensitive=1 ;;
    n) line_numbers=1 ;;
    ?) echo "Неверный ключ: -$OPTARG" >&2
      exit 1 ;;
  esac
done

if [ -z "$pattern" ]; then
  echo "Необходимо указать ключ -p для задания шаблона поиска."
  exit 1
fi

if [ -n "$inputfile" ]; then
  if [ -n "$case_sensitive" ]; then
    if [ -n "$line_numbers" ]; then
      grep -n "$pattern" "$inputfile"
    else
      grep "$pattern" "$inputfile"
    fi
  else
    if [ -n "$line_numbers" ]; then
      grep -in "$pattern" "$inputfile"
    else
      grep -i "$pattern" "$inputfile"
    fi
  fi
else
  echo "Не указан входной файл для поиска."
  exit 1
fi

if [ -n "$outputfile" ]; then
  if [ -n "$case_sensitive" ]; then
    if [ -n "$line_numbers" ]; then
      grep -n "$pattern" "$inputfile" > "$outputfile"
    else
      grep "$pattern" "$inputfile" > "$outputfile"
    fi
  else
    if [ -n "$line_numbers" ]; then
      grep -in "$pattern" "$inputfile" > "$outputfile"
    else
      grep -i "$pattern" "$inputfile" > "$outputfile"
    fi
  fi
fi
```

Написал на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку. Командный файл вызывает эту программу и, проанализировав с помощью команды  `$?` , выдает сообщение о том, какое число было введено.



```
GNU nano 7.2 check_number.c
#include <stdio.h>
#include <stdlib.h>

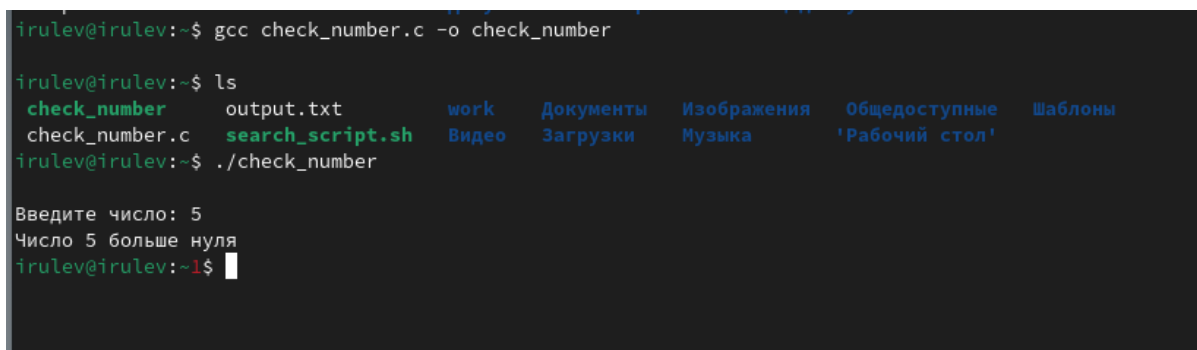
int main() {
    int num;
    printf("Введите число: ");
    scanf("%d", &num);

    if (num > 0) {
        printf("Число %d больше нуля\n", num);
        exit(1);
    } else if (num < 0) {
        printf("Число %d меньше нуля\n", num);
        exit(-1);
    } else {
        printf("Число %d равно нулю\n", num);
        exit(0);
    }

    return 0;
}
```

AG Справка AO Записать AW Поиск AK Вырезать AT Выполнить AC Позиция M-U Отмена M-A Установить метку  
AX Выход AR ЧитФайл AL Замена AU Вставить AD Выровнять A/ К строке M-E Повтор M-S Копировать

Теперь скомпилируем программу с помощью команды `gcc check_number.c -o check_number`.



```
irulev@irulev:~$ gcc check_number.c -o check_number

irulev@irulev:~$ ls
check_number  output.txt  work  Документы  Изображения  Общедоступные  Шаблоны
check_number.c  search_script.sh  Видео  Загрузки  Музыка  'Рабочий стол'

irulev@irulev:~$ ./check_number

Введите число: 5
Число 5 больше нуля
irulev@irulev:~$
```

Напишем командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до  $N$  (например 1.tmp, 2.tmp, 3.tmp, 4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).

```
GNU nano 7.2 file_manager.sh
#!/bin/bash

create_files() {
    local num_files=$1
    for ((i=1; i<=num_files; i++)); do
        touch "$i.tmp"
    done
}

delete_files() {
    local num_files=$1
    for ((i=1; i<=num_files; i++)); do
        rm -f "$i.tmp"
    done
}

if [ "$#" -ne 1 ]; then
    echo "Использование: $0 <количество_файлов>"
    exit 1
fi

num_files=$1

if ! [[ "$num_files" =~ ^[0-9]+$ ]]; then
    echo "Ошибка: аргумент должен быть числом"
    exit 1
fi

create_files "$num_files"
echo "Создано $num_files файлов"

delete_files "$num_files"
echo "Удалено $num_files файлов"

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция M-U Отмена M-A Установить метку
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^D Вывернуть ^_ К строке M-E Повтор M-B Копировать
```

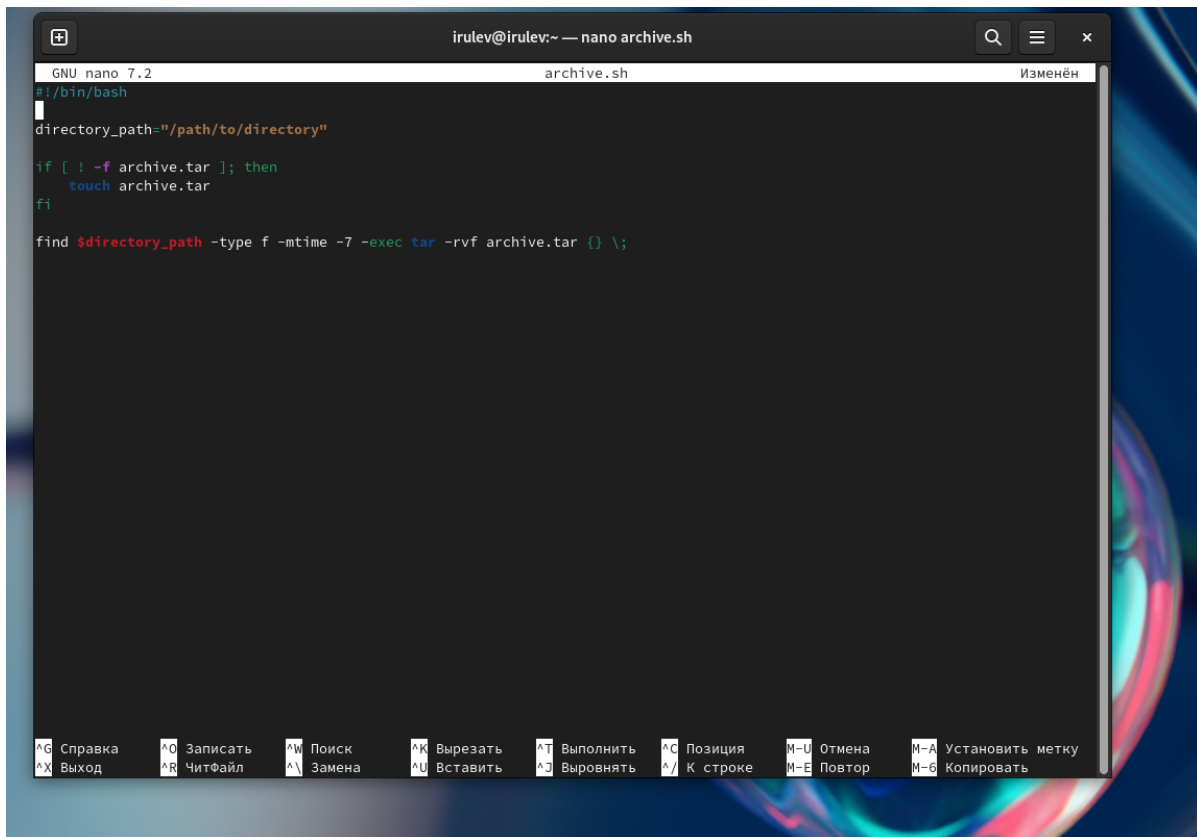
Запустим

```
irulev@irulev:~$ nano file_manager.sh
irulev@irulev:~$ chmod +x file_manager.sh

irulev@irulev:~$ ./file_manager.sh 5

Создано 5 файлов
Удалено 5 файлов
irulev@irulev:~$
```

Напишем командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицируем его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).



```
GNU nano 7.2 archive.sh
#!/bin/bash

directory_path="/path/to/directory"

if [ ! -f archive.tar ]; then
    touch archive.tar
fi

find $directory_path -type f -mtime -7 -exec tar -rvf archive.tar {} \;
```

## Выводы

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

## Ответы на контрольные вопросы

1. Команда `getopts` предназначена для парсинга параметров командной строки в shell-скриптах. Она позволяет извлекать и обрабатывать опции, передаваемые в скрипт, и их аргументы.
2. Метасимволы (`*`, `?`, `[`, `]`, `(`, `)`, `{`, `}`, `~`, `.`) используются в Linux для генерации имён файлов, известной как globbing. Они позволяют указать шаблоны для поиска файлов, что может быть полезно при выполнении различных операций с файлами, таких как поиск, удаление, копирование и т.д.
3. Операторы управления действиями в Linux включают в себя:
  - `&&` (логическое И) - выполняет вторую команду только если первая команда выполнена успешно.
  - `||` (логическое ИЛИ) - выполняет вторую команду только если первая команда не выполнена успешно.
  - `;` (точка с запятой) - разделяет команды, которые выполняются последовательно.
  - `&` (амперсанд) - запускает команду в фоне.
  - `|` (вертикальная черта) - перенаправляет вывод одной команды на вход другой.
4. Операторы, используемые для прерывания цикла, включают в себя:
  - `break` - прерывает выполнение цикла и продолжает выполнение скрипта после цикла.

- `continue` - прерывает текущую итерацию цикла и продолжает выполнение с следующей итерации.

5. Команды `false` и `true` используются для возвращения определенного статуса выполнения. `true` всегда возвращает 0 (успешное выполнение), а `false` всегда возвращает 1 (неуспешное выполнение). Они могут быть полезны в скриптах для управления потоком выполнения в зависимости от результатов предыдущих операций.
6. Строка `if test -f man$s/$i.$s` встреченная в командном файле, проверяет, существует ли файл с именем `$i.$s` в директории `man$s`. Если файл существует, то условие `if` будет истинным, иначе - ложным.
7. Конструкции `while` и `until` используются для создания циклов в shell-скриптах.
- `while` - цикл будет продолжаться, пока условие является истинным.
  - `until` - цикл будет продолжаться, пока условие является ложным.