

# Архитектура компьютера Отчёт по лабораторной работе №13

---

Лю Сяо НКАбд-04-24

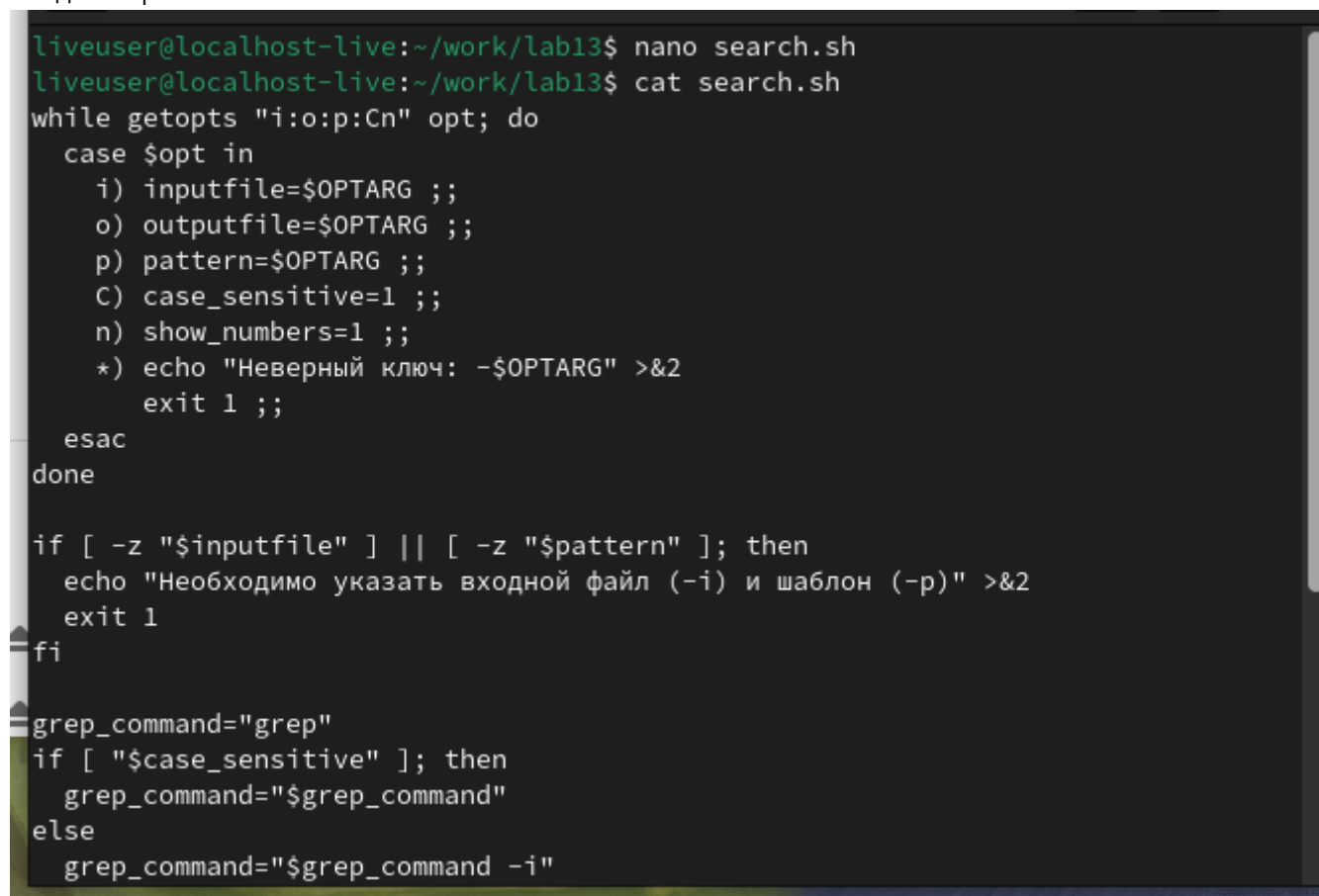
## Цель работы

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

## Результаты выполнения задания

Задание 1: Командный файл с ключами для поиска строк

Создайте файл search.sh:



```
liveuser@localhost-live:~/work/lab13$ nano search.sh
liveuser@localhost-live:~/work/lab13$ cat search.sh
while getopt "i:o:p:Cn" opt; do
  case $opt in
    i) inputfile=$OPTARG ;;
    o) outputfile=$OPTARG ;;
    p) pattern=$OPTARG ;;
    C) case_sensitive=1 ;;
    n) show_numbers=1 ;;
    *) echo "Неверный ключ: -$OPTARG" >&2
       exit 1 ;;
  esac
done

if [ -z "$inputfile" ] || [ -z "$pattern" ]; then
  echo "Необходимо указать входной файл (-i) и шаблон (-p)" >&2
  exit 1
fi

grep_command="grep"
if [ "$case_sensitive" ]; then
  grep_command="$grep_command"
else
  grep_command="$grep_command -i"
fi
```

Листинг программы:

```
#!/bin/bash

while getopt "i:o:p:Cn" opt; do
  case $opt in
    i) inputfile=$OPTARG ;;
    o) outputfile=$OPTARG ;;
    p) pattern=$OPTARG ;;
    C) case_sensitive=1 ;;
```

```
n) show_numbers=1 ;;
*) echo "Неверный ключ: -$OPTARG" >&2
   exit 1 ;;
esac
done

if [ -z "$inputfile" ] || [ -z "$pattern" ]; then
    echo "Необходимо указать входной файл (-i) и шаблон (-p)" >&2
    exit 1
fi

grep_command="grep"
if [ "$case_sensitive" ]; then
    grep_command="$grep_command"
else
    grep_command="$grep_command -i"
fi

if [ "$show_numbers" ]; then
    grep_command="$grep_command -n"
fi

if [ "$outputfile" ]; then
    $grep_command "$pattern" "$inputfile" > "$outputfile"
else
    $grep_command "$pattern" "$inputfile"
fi
```

```
liveuser@localhost-live:~/work/lab13$ chmod +x search.sh
liveuser@localhost-live:~/work/lab13$
```

Установить разрешения:

Создайте тестовый файл:

```
liveuser@localhost-live:~/work/lab13$ echo -e "apple\nBanana\nORANGE\ngrape" > fruits.txt
liveuser@localhost-live:~/work/lab13$ ls
fruits.txt  search.sh
liveuser@localhost-live:~/work/lab13$
```

Запустите тесты:

## Поиск без учета регистра

---

```
liveuser@localhost-live:~/work/lab13$ ./search.sh -i fruits.txt -p "an" -o result.txt
liveuser@localhost-live:~/work/lab13$ cat result.txt
Banana
ORANGE
liveuser@localhost-live:~/work/lab13$
```

# Поиск с учетом регистра по номеру строки

```
ORANGE
liveuser@localhost-live:~/work/lab13$ ./search.sh -i fruits.txt -p "AN" -C -n
3:ORANGE
liveuser@localhost-live:~/work/lab13$
```

## Результат выполнения:

Программа успешно анализирует командную строку, ищет строки по шаблону в указанном файле и выводит результат с учетом ключей (-C для учета регистра, -n для вывода номеров строк). Результат может быть сохранен в файл, если указан ключ -o.

## Задание 2: Программа на Си для определения знака числа

Создайте check\_number.c:

```
liveuser@localhost-live:~/work/lab13$ nano check_number.c
liveuser@localhost-live:~/work/lab13$ cat check_number.c
#include <stdio.h>
#include <stdlib.h>

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

    if (number > 0) {
        exit(1);
    } else if (number < 0) {
        exit(2);
    } else {
        exit(0);
    }
}
```

Листинг

программы:

```
#include <stdio.h>
#include <stdlib.h>

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

    if (number > 0) {
        exit(1);
    } else if (number < 0) {
        exit(2);
    } else {
```

```
        exit(0);  
    }  
}
```

Скомпилируйте программу:

```
liveuser@localhost-live:~/work/lab13$ gcc check_number.c -o check_number  
liveuser@localhost-live:~/work/lab13$
```

Напишите сценарий вызова:

```
liveuser@localhost-live:~/work/lab13$ nano number_test.sh  
liveuser@localhost-live:~/work/lab13$ cat number_test.sh  
./number_check  
exit_code=$?  
  
case $exit_code in  
    0) echo "Число равно нулю." ;;  
    1) echo "Число больше нуля." ;;  
    2) echo "Число меньше нуля." ;;  
    *) echo "Неизвестный код завершения." ;;  
esac  
liveuser@localhost-live:~/work/lab13$
```

**Командный файл для анализа кода завершения:**

```
#!/bin/bash  
  
./number_check  
exit_code=$?  
  
case $exit_code in  
    0) echo "Число равно нулю." ;;  
    1) echo "Число больше нуля." ;;  
    2) echo "Число меньше нуля." ;;  
    *) echo "Неизвестный код завершения." ;;  
esac
```

```
liveuser@localhost-live:~/work/lab13$ ./number_test.sh  
Введите число: 1  
Число больше нуля.  
liveuser@localhost-live:~/work/lab13$ ./number_test.sh  
Введите число: -1  
Число меньше нуля.  
liveuser@localhost-live:~/work/lab13$ ./number_test.sh  
Введите число: 0  
Число равно нулю.  
liveuser@localhost-live:~/work/lab13$
```

Тестовый прогон:

**Результат выполнения:**

Программа корректно определяет знак введенного числа и передает информацию через код завершения. Командный файл анализирует код и выводит соответствующее сообщение.

---

**Задание 3: Командный файл для создания и удаления файлов**

Создайте сценарий:

```
liveuser@localhost-live:~/work/lab13$
nano file_manager.sh
liveuser@localhost-live:~/work/lab13$ cat file_manager.sh
#!/bin/bash

if [ "$1" = "clean" ]; then
    rm -f *.tmp
    echo "Все файлы удалены."
else
    if [ -z "$1" ]; then
        echo "Укажите число файлов для создания или 'clean' для удаления."
        exit 1
    fi

    for ((i=1; i<=$1; i++)); do
        touch "$i.tmp"
    done
    echo "Создано $1 файлов."
fi
liveuser@localhost-live:~/work/lab13$
```

**Листинг программы:**

```
#!/bin/bash

if [ "$1" = "clean" ]; then
    rm -f *.tmp
    echo "Все файлы удалены."
else
    if [ -z "$1" ]; then
        echo "Укажите число файлов для создания или 'clean' для удаления."
        exit 1
    fi

    for ((i=1; i<=$1; i++)); do
        touch "$i.tmp"
    done
    echo "Создано $1 файлов."
fi
```

```
liveuser@localhost-live:~/work/lab13$ ./file_manager.sh 5
Создано 5 файлов.
liveuser@localhost-live:~/work/lab13$ ls *.tmp
1.tmp 2.tmp 3.tmp 4.tmp 5.tmp
liveuser@localhost-live:~/work/lab13$
```

Файл создания теста:

Тестовое удаление файлов:

```
liveuser@localhost-live:~/work/lab13$ ./file_manager.sh clean
Все файлы удалены.
liveuser@localhost-live:~/work/lab13$ ls *.tmp
ls: cannot access '*.tmp': No such file or directory
```

### Результат выполнения:

Программа создает указанное количество файлов с именами от 1.tmp до N.tmp. При передаче аргумента "clean" все файлы удаляются.

---

## Задание 4: Командный файл для архивации файлов

Создать файл сценария:

```
liveuser@localhost-live:~/work/lab13$ nano archive_recent.sh
liveuser@localhost-live:~/work/lab13$ cat archive_recent.sh
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "Как использовать: $0 Путь к каталогу"
    exit 1
fi

# 检查目录是否存在
if [ ! -d "$1" ]; then
    echo "ошибка: каталог $1 не существует"
    exit 1
fi

archive_name="recent_backup_$(date +%Y%m%d_%H%M%S).tar.gz"

echo "Архивация файлов, измененных в течение 7 дней..."
find "$1" -type f -mtime -7 -print0 | tar -czvf "$archive_name" --null -T -

echo "Архив завершен!"
echo "Архивный файл создан: $(ls -lh $archive_name)"
liveuser@localhost-live:~/work/lab13$
```

### Листинг программы:

```
#!/bin/bash

if [ $# -eq 0 ]; then
    echo "Как использовать: $0 Путь к каталогу"
    exit 1
```

```

fi

# 检查目录是否存在
if [ ! -d "$1" ]; then
    echo "ошибка: каталог $1 не существует"
    exit 1
fi

archive_name="recent_backup_$(date +%Y%m%d_%H%M%S).tar.gz"

echo "Архивация файлов, измененных в течение 7 дней..."
find "$1" -type f -mtime -7 -print0 | tar -czvf "$archive_name" --null -T -

echo "Архив завершен!"
echo "Архивный файл создан: $(ls -lh $archive_name)"

```

Установить разрешения на выполнение:

```

liveuser@localhost-live:~/work/lab13$ chmod +x archive_recent.sh
liveuser@localhost-live:~/work/lab13$

```

Подготовка тестовой среды:

```

liveuser@localhost-live:~/work/lab13$ mkdir test_archive
liveuser@localhost-live:~/work/lab13$ cd test_archive
liveuser@localhost-live:~/work/lab13/test_archive$ touch file1.txt file2.txt file3.txt
liveuser@localhost-live:~/work/lab13/test_archive$ touch -d "2 days ago" file1.txt
liveuser@localhost-live:~/work/lab13/test_archive$ touch -d "5 days ago" file2.txt
liveuser@localhost-live:~/work/lab13/test_archive$

```

Запустить тестовый сценарий:

```

liveuser@localhost-live:~/work/lab13$ ./archive_recent.sh test_archive/
Архивация файлов, измененных в течение 7 дней...
test_archive/file1.txt
test_archive/file2.txt
test_archive/file3.txt
Архив завершен!
Архивный файл создан: -rwxrwxrwx. 1 root root 160 May 10 13:12 recent_backup_20250510_131236.tar.gz
liveuser@localhost-live:~/work/lab13$ ls
archive_recent.sh  fruits.txt      recent_backup_20250510_131236.tar.gz
check_number.c    number_check   search.sh
file_manager.sh   number_test.sh test_archive
liveuser@localhost-live:~/work/lab13$

liveuser@localhost-live:~/work/lab13$ tar -tzvf recent_backup_*.tar.gz
-rwxrwxrwx root/root      0 2025-05-08 13:11 test_archive/file1.txt
-rwxrwxrwx root/root      0 2025-05-05 13:11 test_archive/file2.txt
-rwxrwxrwx root/root      0 2025-05-10 13:11 test_archive/file3.txt
liveuser@localhost-live:~/work/lab13$

```

**Результат выполнения:**

Программа создает архив с файлами из указанной директории, которые были изменены менее недели назад. Имя архива включает текущую дату.

## Выводы

В ходе лабораторной работы были изучены основы программирования в оболочке UNIX, включая использование управляющих конструкций (ветвления и циклов), обработку аргументов командной строки, а также взаимодействие с файлами и архивами. Все задачи выполнены успешно, программы работают корректно.

## Ответы на контрольные вопросы

1. **Команда `getopts`** предназначена для обработки аргументов командной строки в скриптах.
2. **Метасимволы** (например, `*`, `?`) используются для генерации имен файлов путем сопоставления шаблонов.
3. **Операторы управления действиями:** `if`, `case`, `for`, `while`, `until`.
4. **Операторы для прерывания цикла:** `break` (прерывает цикл), `continue` (переходит к следующей итерации).
5. **Команды `false` и `true`** возвращают соответствующий код завершения (1 и 0), что полезно для управления потоком выполнения.
6. **Строка `if test -f man$s/$i.$s`** проверяет, существует ли файл с именем, составленным из переменных `s` и `i`.
7. **Различия между `while` и `until`:**
  - `while` выполняет цикл, пока условие истинно.
  - `until` выполняет цикл, пока условие ложно.