РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 13__

дисциплина: Операционные системы

Студент: Ниемек Яи Жак

Группа: НММБд-04-24

МОСКВА

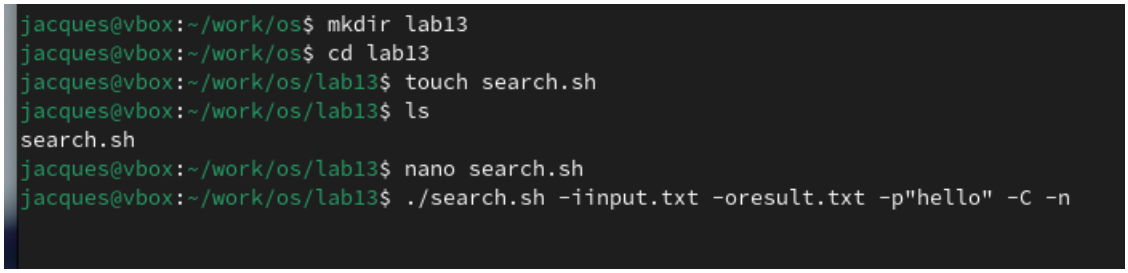
2025_ г.

Цель работы

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

Последовательность выполнения работы

1. Используя команды getopts grep, написать командный файл, который анализирует командную строку с ключами:
– -iinputfile — прочитать данные из указанного файла;
– -ooutputfile — вывести данные в указанный файл;
– -pшаблон — указать шаблон для поиска;
– -C — различать большие и малые буквы;
– -n — выдавать номера строк.
а затем ищет в указанном файле нужные строки, определяемые ключом -p.
2. Написать 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 должен вызывать эту программу и, проанализировав с помощью команды
$?, выдать сообщение о том, какое число было введено.
3. Написать 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 ко мандный файл должен уметь удалять все созданные им файлы (если они существуют).
4. Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).



jacques@vbox:~/work/os/lab13 — nano search.sh

GNU nano 7.2 search.sh

```
input_file=" "
output_file=" "
pattern=" "

while getopts "i:o:p:Cn" opt; do
        case $opt in
                i) input_file="$OPTARG" ;;
                o) output_file="$OPTARG" ;;
                p) pattern="$OPTARG" ;;
                C) case_sensitive=1 ;;
                n) show_line_numbers=1 ;;
                *) echo "Usage: $0 -i inputfile -o outputfile -p pattern [-C] [-n]" ; exit 1 ;;
        esac
done

if [[ -z "$input_file" || -z  "$output_file" || -z "$pattern" ]]; then
        echo "Ошибка не хватает аргументов"
        echo "Usage: $0 -i inputfile -o outputfile -p pattern [-C] [-n]"
        exit 1
fi

cmd="grep"
[[ $cave_sensitive -eq 0]] && cmd="$cmd -i"
[[ $show_line_numbers -eq 1]] && cmd="$cmd -n"

$cmd "$pattern" "$input_file" >  "$output_file"

echo "Поиск завешен. Результаты в $output_file"
```

jacques@vbox:~/work/os/lab13 — nano check_number.c

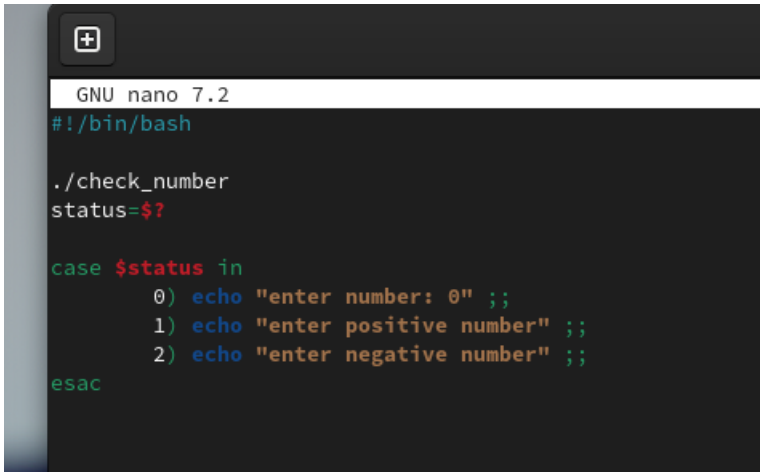
GNU nano 7.2 check_number.c

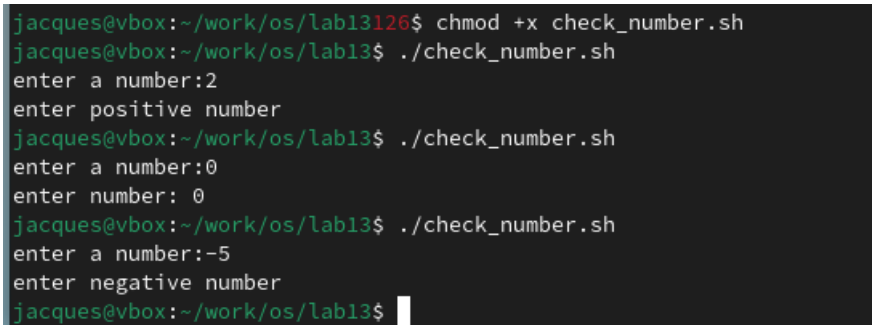
```
#include <stdio.h>
#include <stdlib.h>

int main() {
        int n;
        printf("enter a number:");
        scanf("%d", &n);

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

```
jacques@vbox:~/work/os/lab13$ touch check_number.c
jacques@vbox:~/work/os/lab13$ ls
check_number.c  search.sh
jacques@vbox:~/work/os/lab13$ nano check_number.c
jacques@vbox:~/work/os/lab13$ gcc check_number.c -o check_number
check_number.c: Dans la fonction « main »:
check_number.c:12:24: erreur: expected « ; » before « : » token
   12 |                 exit(2):
      |                        ^
      |                        ;
jacques@vbox:~/work/os/lab131$ nano check_number.c
jacques@vbox:~/work/os/lab13$ gcc check_number.c -o check_number
```





---

1. Предназначение команды getopts

* Используется для разбора аргументов командной строки с ключами (опциями).
* Позволяет удобно обрабатывать флаги -a, -b value и т.д.

```
while getopts "i:o:p:Cn" opt; do
    case $opt in
        i) inputfile=$OPTARG ;;
        o) outputfile=$OPTARG ;;
    esac
done
```

---

2. Метасимволы и генерация имён файлов

* Метасимволы (`*`, ?, `[ ]`) позволяют создавать шаблоны для поиска или подстановки имён файлов.

```
ls *.txt          # все файлы с расширением .txt
ls file?.sh     # file1.sh, file2.sh и т.д.
```

---

3. Операторы управления действиями

* ; – последовательное выполнение команд
* && – выполнить вторую команду, если первая успешна
* || – выполнить вторую команду, если первая неудачна
* & – выполнить команду в фоне

---

4. Операторы для прерывания цикла

* break – выйти из текущего цикла полностью
* continue – перейти к следующей итерации цикла

---

5. Назначение команд true и false

* true – возвращает код 0 (успех), всегда истина
* false – возвращает код 1 (ошибка), всегда ложь
* Используются для условных конструкций или заглушек:

```
while true; do
    echo "Running..."
done
```

---

6. Разбор строки if test -f man$s/$i.$s

* Проверяет существует ли файл с именем man$s/$i.$s.
* -f – проверка на обычный файл.
* Используется для условной обработки только существующих файлов.

---

7. Различия между while и until

Конструкция	Логика выполнения
while [условие]	Выполняет цикл пока условие истинно
until [условие]	Выполняет цикл пока условие ложно (т.е. противоположно `while`)

Пример:

```
# while
count=0
```

```
while [ $count -lt 3 ]; do
    echo $count
    ((count++))
done

# until
count=0
until [ $count -ge 3 ]; do
    echo $count
    ((count++))
done
```

Оба цикла выведут: 0 1 2