

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ  
Факультет физико-математических и естественных наук  
Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

дисциплина:Операционные системы

Студент: Мартемьянов Александр

Группа: НПМбв-02-18

МОСКВА

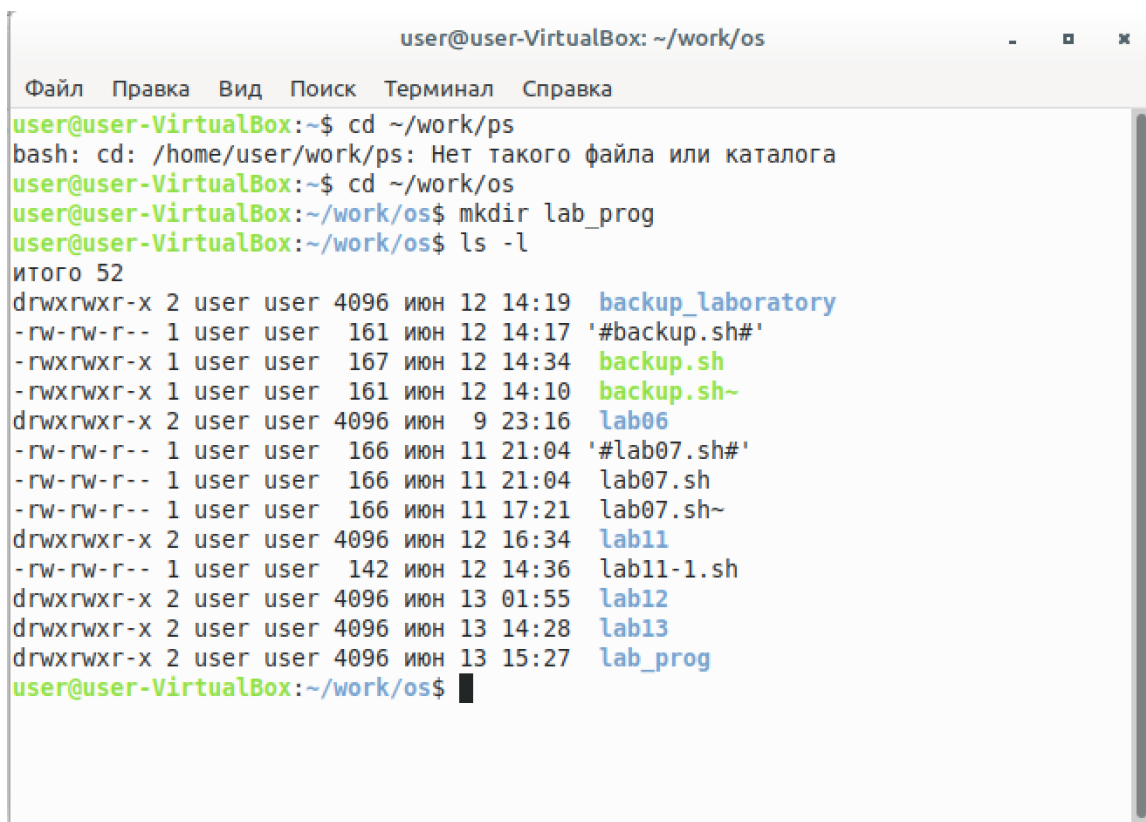
2022 г.

## 2)Задание

Приобрести простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования С калькулятора с простейшими функциями.

## 3) Последовательность выполнения работы

1. В домашнем каталоге создайте подкаталог ~/work/os/lab\_prog.



The screenshot shows a terminal window titled "user@user-VirtualBox: ~/work/os". The terminal output is as follows:

```
user@user-VirtualBox:~$ cd ~/work/ps
bash: cd: /home/user/work/ps: Нет такого файла или каталога
user@user-VirtualBox:~$ cd ~/work/os
user@user-VirtualBox:~/work/os$ mkdir lab_prog
user@user-VirtualBox:~/work/os$ ls -l
итого 52
drwxrwxr-x 2 user user 4096 июн 12 14:19 backup_laboratory
-rw-rw-r-- 1 user user 161 июн 12 14:17 '#backup.sh#'
-rwxrwxr-x 1 user user 167 июн 12 14:34 backup.sh
-rwxrwxr-x 1 user user 161 июн 12 14:10 backup.sh~
drwxrwxr-x 2 user user 4096 июн 9 23:16 lab06
-rw-rw-r-- 1 user user 166 июн 11 21:04 '#lab07.sh#'
-rw-rw-r-- 1 user user 166 июн 11 21:04 lab07.sh
-rw-rw-r-- 1 user user 166 июн 11 17:21 lab07.sh~
drwxrwxr-x 2 user user 4096 июн 12 16:34 lab11
-rw-rw-r-- 1 user user 142 июн 12 14:36 lab11-1.sh
drwxrwxr-x 2 user user 4096 июн 13 01:55 lab12
drwxrwxr-x 2 user user 4096 июн 13 14:28 lab13
drwxrwxr-x 2 user user 4096 июн 13 15:27 lab_prog
user@user-VirtualBox:~/work/os$
```

Рис 3. 1 «Создание каталога»

2. Создайте в нём файлы: calculate.h, calculate.c, main.c. Это будет примитивнейший калькулятор, способный складывать, вычитать, умножать и делить, возводить число в степень, брать квадратный корень, вычислять sin, cos, tan. При запуске он будет запрашивать первое число, операцию, второе число. После этого программа выведет результат и остановится.
3. Выполните компиляцию программы посредством gcc:

```
gcc -c calculate.c
```

```
gcc -c main.c
```

```
gcc calculate.o main.o -o calcul -lm
```

```

user@user-VirtualBox:~/work/os/lab_prog$ touch calculate.h calculate.c main.c
user@user-VirtualBox:~/work/os/lab_prog$ ls -l
итого 0
-rw-rw-r-- 1 user user 0 июн 13 15:43 calculate.c
-rw-rw-r-- 1 user user 0 июн 13 15:43 calculate.h
-rw-rw-r-- 1 user user 0 июн 13 15:43 main.c
user@user-VirtualBox:~/work/os/lab_prog$

```

Рис 3. 2 «Создание исполняемых файлов»

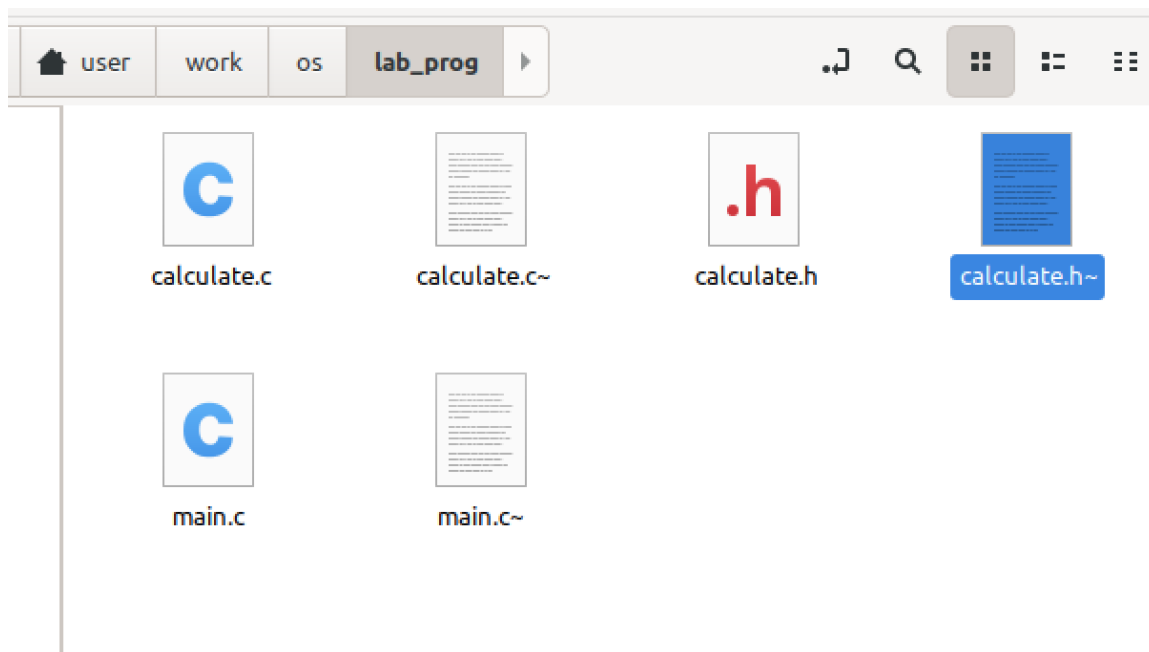


Рис 3. 3 «Запуск исполняемых файлов»

4. При необходимости исправьте синтаксические ошибки.

Синтаксических ошибок не обнаружено

5. Создайте Makefile со следующим содержанием:

```

user@user-VirtualBox:~/work/os/lab_prog$ touch make.makefile
user@user-VirtualBox:~/work/os/lab_prog$ ls -l
итого 40
-rwxrwxr-x 1 user user 17120 июн 13 16:05 calcul
-rw-rw-r-- 1 user user 1592 июн 13 15:46 calculate.c
-rw-rw-r-- 1 user user 0 июн 13 15:43 calculate.c~
-rw-rw-r-- 1 user user 165 июн 13 15:47 calculate.h
-rw-rw-r-- 1 user user 0 июн 13 15:43 calculate.h~
-rw-rw-r-- 1 user user 3992 июн 13 16:05 calculate.o
-rw-rw-r-- 1 user user 391 июн 13 15:49 main.c
-rw-rw-r-- 1 user user 0 июн 13 15:43 main.c~
-rw-rw-r-- 1 user user 2256 июн 13 16:05 main.o
-rw-rw-r-- 1 user user 0 июн 13 16:28 make.makefile
user@user-VirtualBox:~/work/os/lab_prog$ make -f make.makefile
make: «calcul» не требует обновления.

```

Рис 3. 5 «Создание makefile»

6. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile):

```
user@user-VirtualBox:~/work/os/lab_prog$ gdb ./calcul
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./calcul...
(No debugging symbols found in ./calcul)
(gdb) run
Starting program: /home/user/work/os/lab_prog/calcul
Число: 8840
Операция (+,-,*,/,pow,sqrt,sin,cos,tan):sin
-0.43
[Inferior 1 (process 5632) exited normally]
(gdb) list
1      init-first.c: Нет такого файла или каталога.
(gdb) list 12,15
12      in init-first.c
(gdb) list calculate.c:20,29
No source file named calculate.c.
(gdb) list calculate.c:20,27
No source file named calculate.c.
(gdb) break
```

Рис 3. 6 «Отладка программы»

```
No default breakpoint address now.
(gdb) break 21
Breakpoint 1 at 0x7ffff7c90f20: file init-first.c, line 44.
(gdb) info breakpoint
Num      Type      Disp Enb Address              What
1        breakpoint keep y  0x00007ffff7c90f20 in __libc_init_first
                                at init-first.c:44

(gdb) run
Starting program: /home/user/work/os/lab_prog/calcul
Число: 10
Операция (+,-,*,/,pow,sqrt,sin,cos,tan):-
Вычитаемое: 2
8.00
[Inferior 1 (process 5770) exited normally]
(gdb) backtrace
No stack.
(gdb) info breakpoints
Num      Type      Disp Enb Address              What
1        breakpoint keep y  0x00007ffff7c90f20 in __libc_init_first
                                at init-first.c:44

(gdb) delete 1
(gdb)
```

Рис 3. 7 «Отладка программы»

7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.

```
user@user-VirtualBox:~/work/os/lab_prog$ splint calculate.c
Splint 3.1.2 --- 20 Feb 2018

calculate.h:7:37: Function parameter Operation declared as manifest array (size
                    constant is meaningless)
    A formal parameter is declared as an array with size. The size of the array
    is ignored in this context, since the array formal parameter is treated as a
    pointer. (Use -fixedformalarray to inhibit warning)
calculate.c:9:36: Function parameter Operation declared as manifest array (size
                    constant is meaningless)
calculate.c: (in function Calculate)
calculate.c:13:7: Return value (type int) ignored: scanf("%f", &Sec...
    Result returned by function call is not used. If this is intended, can cast
    result to (void) to eliminate message. (Use -retvalint to inhibit warning)
calculate.c:18:7: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:23:9: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:28:9: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:29:12: Dangerous equality comparison involving float types:
                    SecondNumeral == 0
    Two real (float, double, or long double) values are compared directly using
    == or != primitive. This may produce unexpected results since floating point
    representations are inexact. Instead, compare the difference to FLT_EPSILON
    or DBL_EPSILON. (Use -realcompare to inhibit warning)
calculate.c:31:19: Return value type double does not match declared type float:
                    (HUGE_VAL)
    To allow all numeric types to match, use +relaxtypes.
calculate.c:36:9: Return value (type int) ignored: scanf("%f", &Sec...
calculate.c:37:15: Return value type double does not match declared type float:
                    (pow(Numeral, SecondNumeral))
calculate.c:39:15: Return value type double does not match declared type float:
                    (sqrt(Numeral))
calculate.c:41:15: Return value type double does not match declared type float:
                    (sin(Numeral))
calculate.c:43:15: Return value type double does not match declared type float:
                    (cos(Numeral))
calculate.c:45:15: Return value type double does not match declared type float:
                    (tan(Numeral))
calculate.c:48:15: Return value type double does not match declared type float:
                    (HUGE_VAL)

Finished checking --- 15 code warnings
user@user-VirtualBox:~/work/os/lab_prog$
```

Рис 3. 8 «Отладка программы»

```
user@user-VirtualBox:~/work/os/lab_prog$ splint main.c
Splint 3.1.2 --- 20 Feb 2018

calculate.h:7:37: Function parameter Operation declared as manifest array (size
                    constant is meaningless)
    A formal parameter is declared as an array with size. The size of the array
    is ignored in this context, since the array formal parameter is treated as a
    pointer. (Use -fixedformalarray to inhibit warning)
main.c: (in function main)
main.c:12:3: Return value (type int) ignored: scanf("%f", &Num...
    Result returned by function call is not used. If this is intended, can cast
    result to (void) to eliminate message. (Use -retvalint to inhibit warning)
main.c:14:3: Return value (type int) ignored: scanf("%s", Oper...

Finished checking --- 3 code warnings
user@user-VirtualBox:~/work/os/lab_prog$
```

Рис 3. 9 «Отладка программы»

**5) Ответы на контрольные вопросы** 1. Как получить информацию о возможностях программ gcc, make, gdb и др.?

Можно использовать стандартные команды для получения справки - `man gcc`, `man make`, `man gdb` ну или вместо утилиты `man` использовать команду – `help`

2. Назовите и дайте краткую характеристику основным этапам разработки приложений в UNIX.

Основные этапы разработки приложений в Unix: Создание исходного кода (написание в IDE) -> Сохранение промежуточных файлов или альтернативных веток разработки исходного кода -> Компиляция исходных файлов или их интерпритация в зависимости от выбранного языка программирования и/или системы сборки проектов -> Тестирование проекта который был собран -> Запись в соответствующую ветку разработки Git (main или dev, по-умолчанию)

3. Что такое суффикс в контексте языка программирования? Приведите примеры использования.

Суффикс - нужен для определения расширения в контексте файловой системы или компилятора с помощью которого будет производиться компиляция или интерпретация исходного кода в работающую программу (например `hello1.py` компилируется только `ipython`, а вот `hello2.c` компилируется только `gcc`, `Cmake`)

4. Каково основное назначение компилятора языка C в UNIX?

Компилятор Си предназначен для компиляции внутренних файлов системы без полного скачивания программ, а просто скачав исходный код системных утилит и произвести с помощью встроенного компилятора компиляцию системных утилит

5. Для чего предназначена утилита make?

Утилита `make` - предназначена для упрощения разработки приложений, путем написания файла конфигурации который описывает пути компиляции для компилятора языка программирования

6. Приведите пример структуры Makefile. Дайте характеристику основным элементам этого файла.

№№№№№№Можно использовать пример из лабараторной работы

7. Назовите основное свойство, присущее всем программам отладки. Что необходимо сделать, чтобы его можно было использовать?

Пошаговая отладка программ (трассировка) - её суть заключается в пошаговом выполнении каждой строчки кода

8. Назовите и дайте основную характеристику основным командам отладчика gdb.

Основные команды отладчика gdb:

backtrace - вывод на экран путь к текущей точке останова. break - установить точку останова (строка или функция) clear - удалить все точки останова в функции continue - продолжить выполнение программы delete (n) - удалить точку останова display - добавить выражение в список выражений, значения которых отображаются при достижении точки останова программы finish - выполнить программу до момента выхода из функции info breakpoints - вывести на экран список используемых точек останова info watchpoints - вывести на экран список используемых контрольных выражений list - вывести на экран исходный код (в качестве параметра может быть указано название файла и через двоеточие номера начальной и конечной строк) next - выполнить программу пошагово, но без выполнения вызываемых в программе функций print - вывести значение указываемого в качестве параметра выражения run - запуск программы на выполнение set[variable] - установить новое значение переменной step - пошаговое выполнение программы watch - установить контрольное выражение, при изменении значения которого программа будет остановлена

9. Опишите по шагам схему отладки программы, которую Вы использовали при выполнении лабораторной работы.

Мои действия при отладке программ: Запустил Makefile -> Начал отладку (run) -> Вывел содержимое main файла -> Установил точку останова в main файле -> Продолжил выполнение (run) -> Использовал команды print & display для вывод промежуточных данных -> Удалил точку останова -> Закончил отладку

10. Прокомментируйте реакцию компилятора на синтаксические ошибки в программе при его первом запуске.

Нейтральная реакция компилятора, т.е. программных ошибок обнаружено не было

11. Назовите основные средства, повышающие понимание исходного кода программы.

cppcheck, splint, cscope и другие

12. Каковы основные задачи, решаемые программой splint?

Проверка корректности аргументов и поиск ошибок и значений в программе которые могут быть улучшены, а также оценка всей программы