Лабараторная работа №14

Журавлев Георгий Иванович

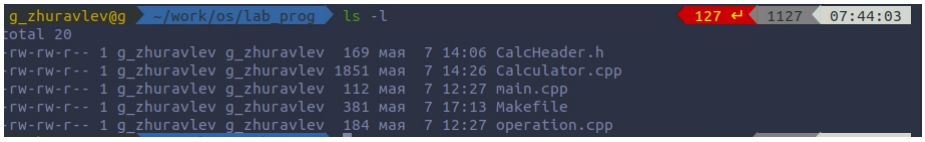
Цель работы

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.

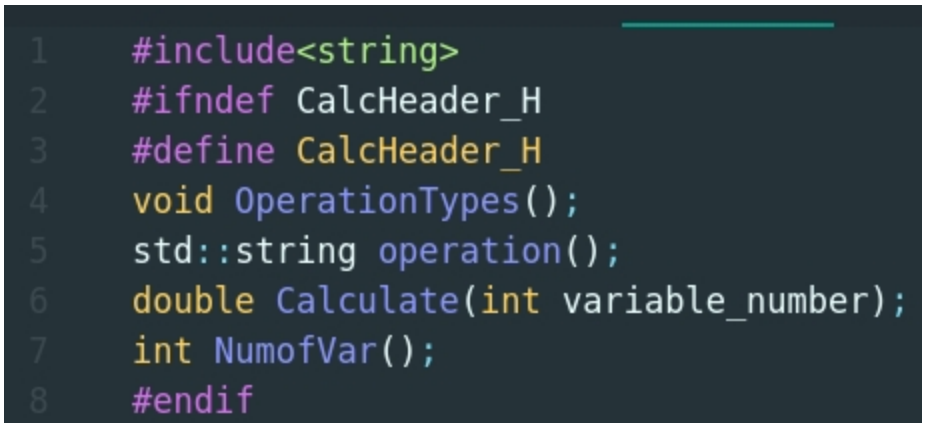
Ход работы.

1. В домашнем каталоге создал подкаталог ~/work/os/lab_prog.

2. Создал в нём файлы: CalcHeader.h, Calculator.cpp, operation.cpp, main.cpp.



2.1. Header file.



```
#include<string>
#ifndef CalcHeader_H
#define CalcHeader_H
void OperationTypes();
std::string operation();
double Calculate(int variable_number);
int NumofVar();
#endif
```

2.2. Calculator.cpp.

```cpp
#include <iostream>
#include<cmath>
#include<string>
#include "CalcHeader.h"
using namespace std;

void OperationTypes() {
    cout << "There are only few possible operations with ONE or TWO variables: \n";
    cout << "Basic: + ; -; * ; / ; pow.\nTrigonometric: sin; cos; tan.\n";
    cout << "Please, use only this operators, otherwise it won't work.\n";
}

double Calculate(int variable_number) {
    double a=0;
    double b=0;
    double result;
    string operationCalc;
    if (variable_number == 2) {
    cout << "first variable: "; cin >> a;
    operationCalc = operation();
    cout << "second variable: "; cin >> b;
        if (operationCalc == "+") {
            result = a + b;
        }
        else if (operationCalc == "-") {
            result = a - b;
        }
        else if (operationCalc == "*") {
            result = a * b;
        }
        else if (operationCalc == "/") {
            if (b == 0) {
                cout << "that can't be done(doesn't exist)\n";
                return 1;
            }
            else {
                result = a / b;
```

```
            }
            else {
                    cout << "wrong option, operator is undefined ";
                    return 1;
                }
            }

        else if(variable_number==1){
            cout << "variable: "; cin >> a;
            operationCalc = operation();
            if (operationCalc == "sin") {
                 result = sin(a);
            }
            else if (operationCalc == "cos") {
                result = cos(a);
            }
            else if (operationCalc == "tan") {
                if (cos(a) == 0) {
                    cout << "that can't be done(doesn't exist)\n";
                    return 1;
                }
                else {
                    result = tan(a);
                }
            }
            else if (operationCalc == "pow") {
                    int power;
                    cout << "to power of: "; cin >> power;
                    result = pow(a,power);

            }
            else {
                cout << "wrong option, operator is undefined ";
                return 1;
            }
```

```
        else {
            cout << "wrong option, only  ONE or TWO variables";
            return 1;
        }


        if (variable_number == 2) {
            cout << a <<" "<< operationCalc <<" "<< b << " = "<< result;
        }
        else{
            cout << operationCalc << "(" << a << ") = " << result;
        }
    return 0;
    }
    int NumofVar() {
        int VarNum;
        cout << "how many variables do you need?( 1 or 2): "; cin >> VarNum;
        return VarNum;
    }
```

2.3. operation.cpp.

```cpp
#include <iostream>
#include <string>
#include "CalcHeader.h"
using namespace std;
string operation() {
    string operation;
    cout << "operator: "; cin >> operation;
    return operation;
}
```

2.4. main.cpp.

```cpp
#include <iostream>
#include "CalcHeader.h"
int main()
{
    OperationTypes();
    Calculate(NumofVar());
    return 0;
}
```

3.Выполнил компиляцию программы посредством g++:

```
g_zhuravlev@g ~/work/os/lab_prog g++ -c Calculator.cpp                    1 ↵ 1129 07:45:28
g_zhuravlev@g ~/work/os/lab_prog g++ -c main.cpp                            ✔ 1130 07:45:35
g_zhuravlev@g ~/work/os/lab_prog g++ -c operation.cpp                       ✔ 1131 07:45:42
g_zhuravlev@g ~/work/os/lab_prog g++ calculator.o main.o operation.o -o calculator
g++: error: calculator.o: No such file or directory
g_zhuravlev@g ~/work/os/lab_prog g++ Calculator.o main.o operation.o -o calculator
g_zhuravlev@g ~/work/os/lab_prog                                            ✔ 1134 07:46:43
```

4. Создал Makefile.

```
CC = g++
CFLAGS = -g -c
CompFlags= -g -o

calculate: Calculator.o main.o operation.o
	$(CC) Calculator.o main.o operation.o
	$(CompFlags) calculate

Calculator.o: Calculator.cpp CalcHeader.h
	$(CC) $(CFLAGS) Calculator.cpp

main.o: main.cpp CalcHeader.h
	$(CC) $(CFLAGS) main.cpp

operation.o: operation.cpp CalcHeader.h
	$(CC) $(CFLAGS) operation.cpp

clean:
	rm -rf *.o calculate
```

5. Собрал Makefile.

```
g_zhuravlev@g ~/work/os/lab_prog make -f Makefile ✔ 1139 08:13:05
g++ -g -c Calculator.cpp
g++ -g -c main.cpp
g++ -g -c operation.cpp
g++ Calculator.o main.o operation.o -g -o calculate
g_zhuravlev@g ~/work/os/lab_prog ✔ 1140 08:13:10
```

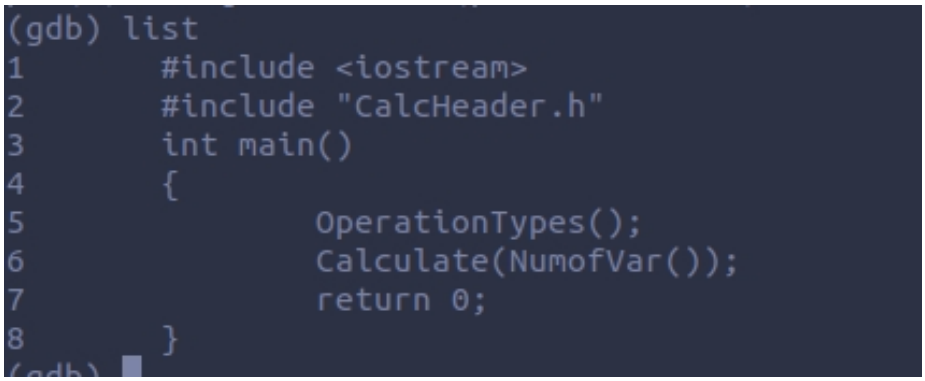
В нём указаны :
1. CC - тип компилятора.
2. CFLAGS - опции, с которыми мы компилируем основной файл.
3. CompFlags - опции, с которыми мы компилируем остальные файлы.
4. Компиляция каждого зависящего файла .o
5. Сборщик всей программы "calculate"
6. Команда clean, для быстрого удаления всех файлов.

5. Провёл отладку с помощью GNUdebbuger(gdb).

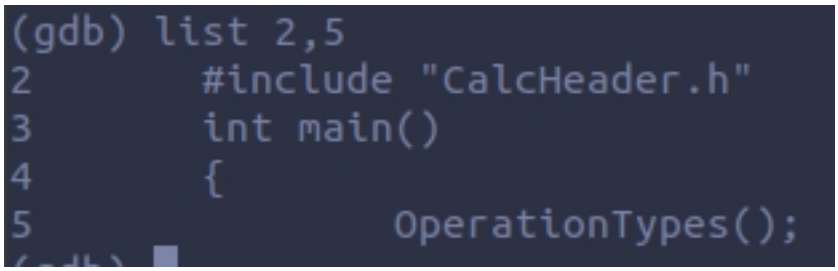
5.1. Запустил отладчик GDB, загрузив в него программу для отладки.

5.2. Для запуска программы внутри отладчика ввел команду run.

5.3. Использовала команду list.



5.4. Для просмотра строк с 2 по 5 основного файла использовал list 2,5.



5.5. Для просмотра определённых строк не основного файла использовал list с параметрами: list Calculat

```
(gdb) list Calculator.cpp: 7,45
7       void OperationTypes() {
8               cout << "There are only few possible operations with ONE or TWO variables: \n";
9               cout << "Basic: + ; -; * ; / ; pow.\nTrigonometric: sin; cos; tan.\n";
10              cout << "Please, use only this operators, otherwise it won't work.\n";
11      }
12
13      double Calculate(int variable_number) {
14              double a=0;
15              double b=0;
16              double result;
17              string operationCalc;
18              if (variable_number == 2) {
19              cout << "first variable: "; cin >> a;
20              operationCalc = operation();
21              cout << "second variable: "; cin >> b;
22                      if (operationCalc == "+") {
23                              result = a + b;
24                      }
25                      else if (operationCalc == "-") {
26                              result = a - b;
27                      }
28                      else if (operationCalc == "*") {
29                              result = a * b;
30                      }
31                      else if (operationCalc == "/") {
32                              if (b == 0) {
33                                      cout << "that can't be done(doesn't exist)\n";
34                                      return 1;
35                              }
36                              else {
37                                      result = a / b;
38                              }
39                      }
40                      else {
41                                      cout << "wrong option, operator is undefined ";
42                                      return 1;
43                              }
44                      }
45
```

5.6. Установил точку останова в файле Calculator.cpp на строке номер 18

5.7. Вывел информацию об имеющихся в проекте точках останова: info breakpoints

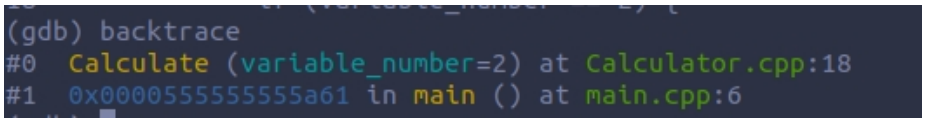
```
(gdb) break 18
Breakpoint 1 at 0x5555555553f6: file Calculator.cpp, line 18.
(gdb) info breakpoints
Num     Type           Disp Enb Address            What
1       breakpoint     keep y   0x00005555555553f6 in Calculate(int) at Calculator.cpp:18
```

5.8. Запустил программу внутри отладчика и убедился, что программа остановится в момент прохожден

```
(gdb) run
Starting program: /home/g_zhuravlev/work/os/lab_prog/calculate
There are only few possible operations with ONE or TWO variables:
Basic: + ; -; * ; / ; pow.
Trigonometric: sin; cos; tan.
Please, use only this operators, otherwise it won't work.
how many variables do you need?( 1 or 2): 2

Breakpoint 1, Calculate (variable_number=2) at Calculator.cpp:18
18              if (variable_number == 2) {
(gdb)
```

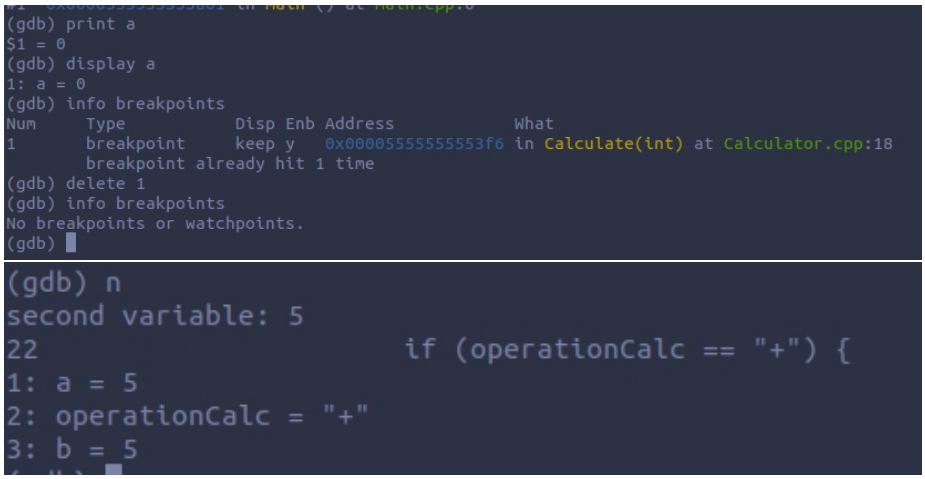
5.8.1. Использовал backtrace.



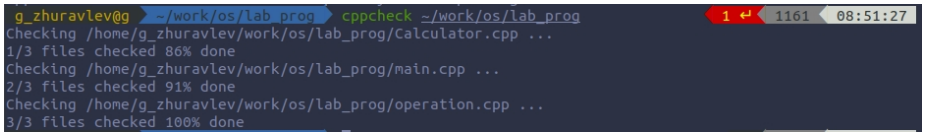
5.9. Посмотрел, чему равно на этом этапе значение переменной a, введя: print a.

5.10. Сравнил с результатом вывода на экран после использования команды: display a.

5.11. Убрал точки останова: info breakpoints -> delete 1.



6. С помощью утилиты cppcheck проанализировал коды файлов(всю директорию).



Вывод.

Благодаря этой лабараторной работе я приобрел простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования C++ калькулятора с простейшими функциями.

Контрольные вопросы.

1. Информацию об этих программах можно получить с помощью функций --help и man.

2. Разработка приложений в UNIX:

1. создание исходного кода программы;(файл с необходимым расширением(и кодом)).

2. сохранение различных вариантов исходников;

3. анализ исходников; необходимо отслеживать изменения исходного кода.

4. компиляция исходников и построение исполняемого модуля;

5. тестирование и отладка; (проверка кода на наличие ошибок)

6. сохранение всех изменений, выполняемых при тестировании и отладке.

7. Загрузка версии исходников в систему контроля версий GIT.

3. Суффикс определяет какая компиляция требуется. Суффиксы и префиксы указывают тип объекта.(п

4. Основное назначение этого компилятора заключается в компиляции всей программы и получении испо

5. Утилита make нужна для исполнения команд из Makefile(-ов). (компиляции, очистки и тп).

6. Пример:

```
CC = compiler
CFLAGS = compiler flags( like -c or -g)
.....( and so on)

target1: dependencies ( for example: report: dependentFile.o ... dependentFile.o)
 target2: dependencies
 ...
 targetn: dependencies
 <tab>(necessary) $(CC) dependentFile.o $(CFLAGS) report.
```

пример из лабараторной работы:

```
CC = g++
CFLAGS = -g -c
CompFlags= -g -o

calculate: Calculator.o main.o operation.o
    $(CC) Calculator.o main.o operation.o $(CompFlags) calculate

Calculator.o: Calculator.cpp CalcHeader.h
    $(CC) $(CFLAGS) Calculator.cpp
```

```
main.o: main.cpp CalcHeader.h
    $(CC) $(CFLAGS) main.cpp

operation.o: operation.cpp CalcHeader.h
    $(CC) $(CFLAGS) operation.cpp

clean:
    rm -rf *.o calculate
```

7. Пошаговая отладка программ заключается в том, что выполняется один оператор программы и, затем переменных, а также внутренних регистров микроконтроллера и напряжений на выводах этой микросхе

8.

1. backtrace - вывод на экран путь к текущей точке останова.

2. break - установить точку останова (в качестве параметра может быть указан номер строки или название функции)

3. clear - удалить все точки останова в функции

4. continue - продолжить выполнение программы

5. delete (n) - удалить точку останова

6. display - добавить выражение в список выражений, значения которых отображаются при достижени

7. finish - выполнить программу до момента выхода из функции

8. info breakpoints - вывести на экран список используемых точек останова

9. info watchpoints - вывести на экран список используемых контрольных выражений

10. list - вывести на экран исходный код (в качестве параметра может быть указано название файла и ч

11. next - выполнить программу пошагово, но без выполнения вызываемых в программе функций

12. print - вывести значение указываемого в качестве параметра выражения

13. run - запуск программы на выполнение

14. set[variable] - установить новое значение переменной

15. step - пошаговое выполнение программы

16. watch - установить контрольное выражение, при изменении значения которого программа будет ост

9.

1) Запустил Makefile(компиляция и сборка);

2) Начал отладку(run);

3) Вывел содержимое основного файла и Calculator.cpp;

4) Установил точку останова в Calculator.cpp.

5) Продолжил выполнение(run);

6) Посмотрел используемые функции(на данный момент времени backtrace);

7) Использовал команды print & display;

8) Удалил точку останова;

9) Закончил отладку(ошибок не было найдено);

10. Реакция была нормальной(ошибок не было)

11. cppcheck; splint; cscope;

12.

1) Проверка корректности задания аргументов всех использованных в программе функций, а также ти

2) Поиск фрагментов исходного текста, корректных с точки зрения синтаксиса языка Си, но малоэффе

3) Общая оценка мобильности пользовательской программы.