Лабораторная работа №13.

Средства, применяемые при разработке программного обеспечения в ОС типа UNIX/Linux

ОЗЬЯС Стев Икнэль Дани

Содержание

# 1 Цель работы

Цель данной 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.

# 2 Задание

1. В домашнем каталоге создайте подкаталог ~/work/os/lab\_prog.
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. Реализация функций калькулятора в файле calculate.h, интерфейсный файл calculate.h, описывающий формат вызова функциикалькулятора, основной файл main.c, реализующий интерфейс пользователя к калькулятору.
3. Выполните компиляцию программы посредством gcc:
   * gcc -c calculate.c
   * gcc -c main.c
   * gcc calculate.o main.o -o calcul -lm
4. При необходимости исправьте синтаксические ошибки.
5. Создайте Makefile с заданным содержанием. Поясните в отчёте его содержание.
6. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile):
   * Запустите отладчик GDB, загрузив в него программу для отладки:
     + gdb ./calcul
   * Для запуска программы внутри отладчика введите команду run:
     + run
   * Для постраничного (по 9 строк) просмотра исходного код используйте команду list:
     + list
   * Для просмотра строк с 12 по 15 основного файла используйте list с параметрами:
     + list 12,15
   * Для просмотра определённых строк не основного файла используйте list с параметрами:
     + list calculate.c:20,29
   * Установите точку останова в файле calculate.c на строке номер 21:
     + list calculate.c:20,27
     + break 21
   * Выведите информацию об имеющихся в проекте точка останова:
     + info breakpoints
   * Запустите программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки останова:
     + run
     + 5
     + backtrace
   * Отладчик выдаст следующую информацию:
     + #0 Calculate (Numeral=5, Operation=0x7fffffffd280 “-”)
     + at calculate.c:21
     + #1 0x0000000000400b2b in main () at main.c:17
   * А команда backtrace покажет весь стек вызываемых функций от начала программы до текущего места.
   * Посмотрите, чему равно на этом этапе значение переменной Numeral, введя:
     + print Numeral
   * На экран должно быть выведено число 5.
   * Сравните с результатом вывода на экран после использования команды:
     + display Numeral
   * Уберите точки останова:
     + info breakpoints
     + delete 1
7. С помощью утилиты splint попробуйте проанализировать коды файлов calculate.c и main.c.

# 3 Теоретическое введение

* Процесс разработки программного обеспечения обычно разделяется на следующие этапы:
  + планирование, включающее сбор и анализ требований к функционалу и другим характеристикам разрабатываемого приложения;
  + проектирование, включающее в себя разработку базовых алгоритмов и спецификаций, определение языка программирования;
  + непосредственная разработка приложения:
  + кодирование — по сути создание исходного текста программы (возможно в нескольких вариантах);
  + анализ разработанного кода;
  + сборка, компиляция и разработка исполняемого модуля;
  + тестирование и отладка, сохранение произведённых изменений;
  + документирование.
* Для создания исходного текста программы разработчик может воспользоваться любым удобным для него редактором текста: vi, vim, mceditor, emacs, geany и др.
* После завершения написания исходного кода программы (возможно состоящей из нескольких файлов), необходимо её скомпилировать и получить исполняемый модуль.
* В табл. 1 приведено краткое описание полезных для выполнения работы команд и клавиш.

Таблица 1: Описание полезных для выполнения работы команд

| Команда | Описание действия |
| --- | --- |
| backtrace | вывод на экран пути к текущей точке останова (по сути вывод названий всех функций) |
| break | установить точку останова (в качестве параметра может быть указан номер строки или название функции) |
| continue | продолжить выполнение программы |
| info breakpoints | вывести на экран список используемых точек останова |
| delete | удалить точку останова |
| list | вывести на экран исходный код |
| run | запуск программы на выполнение |

* Более подробно об Unix см. в [1–6].

# 4 Выполнение лабораторной работы

1. В домашнем каталоге создал подкаталог ~/work/os/lab\_prog. (рис. 1)

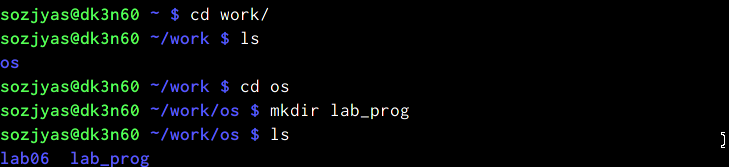


Рис. 1: Создание подкаталога ~/work/os/lab\_prog

1. Создал в нём файлы: calculate.h, calculate.c, main.c. (рис. 2)

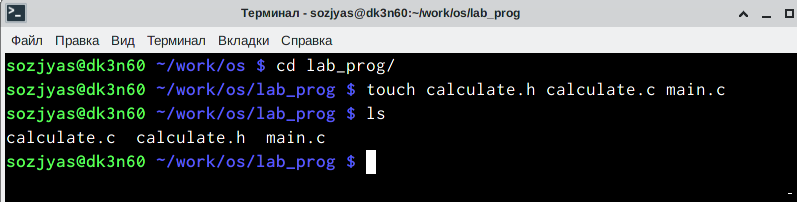


Рис. 2: Создание файлов calculate.h, calculate.c, main.c

* calculate.h, (рис. 3)

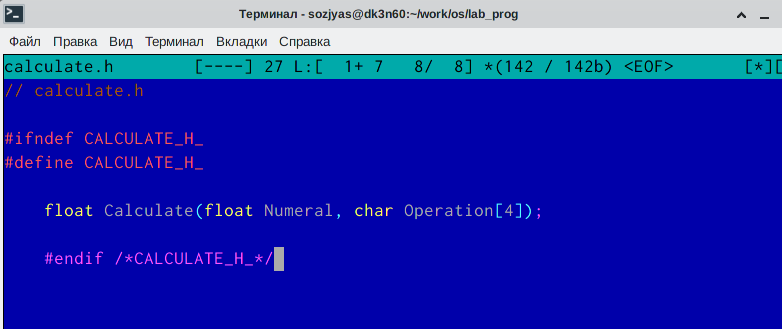


Рис. 3: calculate.h

* calculate.c, (рис. 4)

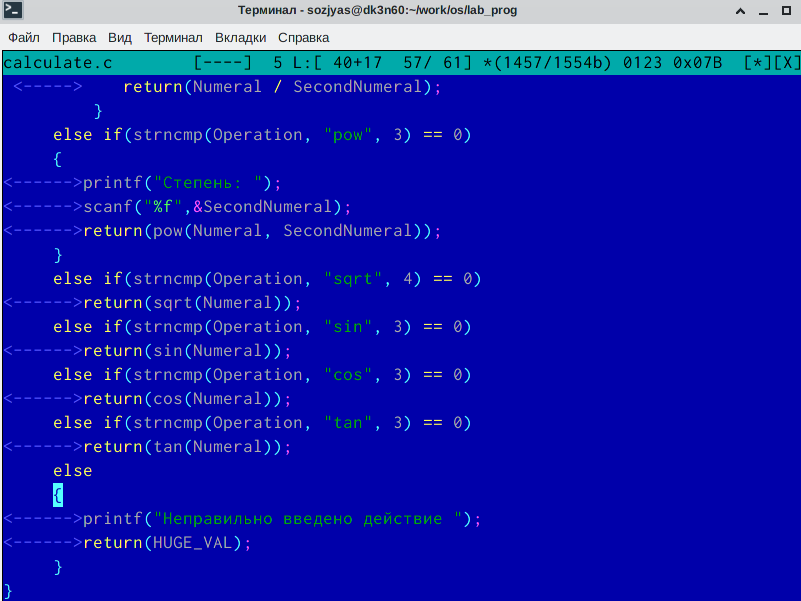


Рис. 4: calculate.c

* main.c (рис. 5

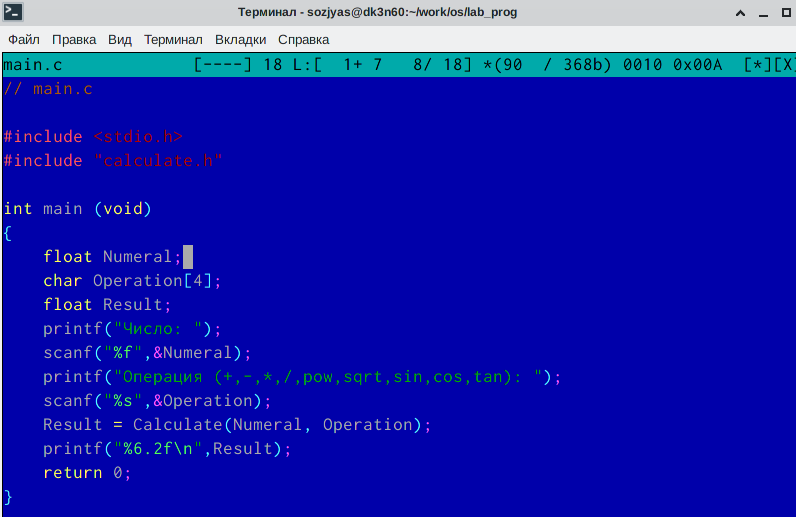


Рис. 5: main.c

1. Выполнил компиляцию программы посредством gcc (рис. 6)

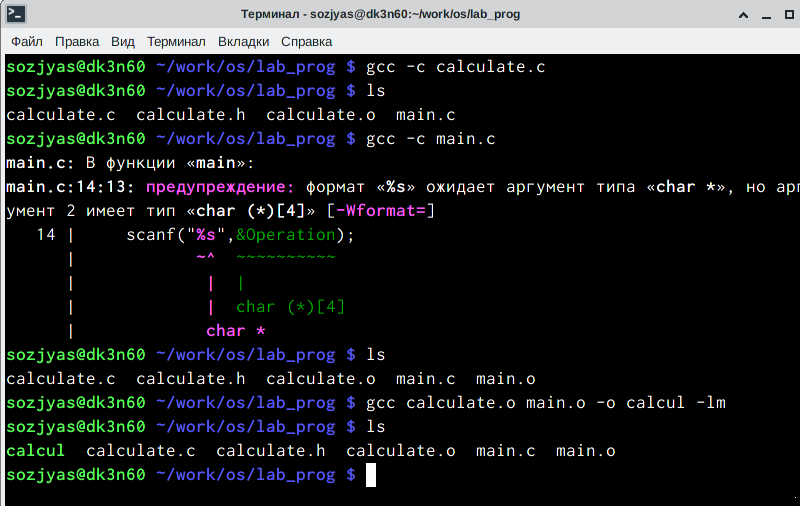


Рис. 6: Выполнение компиляции программы посредством gcc

1. Я исправил незначительную синтаксическую ошибку.
2. Создал Makefile с заданным содержанием (рис. 7)
   * CC = gcc замена слова gcc на CC
   * LIBS = -lm дополнительные опции
   * calcul: calculate.o main.o
     + gcc calculate.o main.o -o calcul $(LIBS) команда для создания исполдняемого файла calcul
   * calculate.o: calculate.c calculate.h
     + gcc -c calculate.c $(CFLAGS) команда для создания объектного файла calculate.o
   * main.o: main.c calculate.h
     + gcc -c main.c $(CFLAGS) команда для создания объектного файла main.o
   * clean:
     + -rm calcul *.o* ~ команда для удаления всех объектных файлов и файлов с знаком ~ в конец

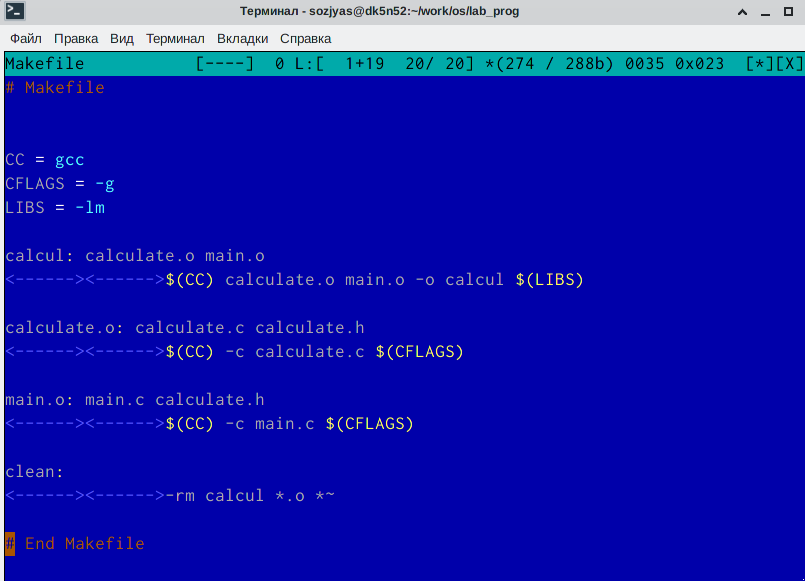


Рис. 7: Создание Makefile

1. С помощью gdb выполнил отладку программы calcul (перед использованием gdb исправил Makefile)(рис. 8

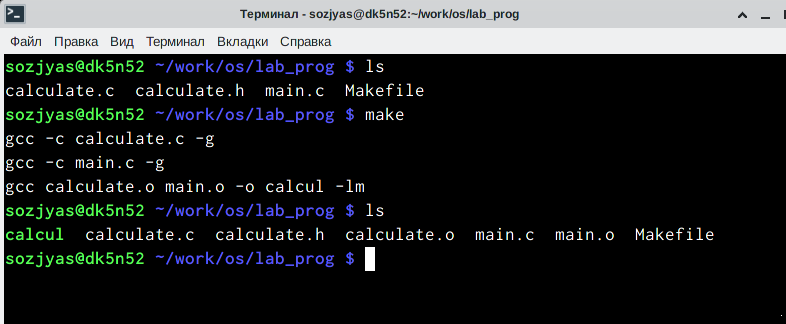


Рис. 8: Выполнение отладки программы calcul

* Запустил отладчик GDB, загрузив в него программу для отладки:(рис. 9

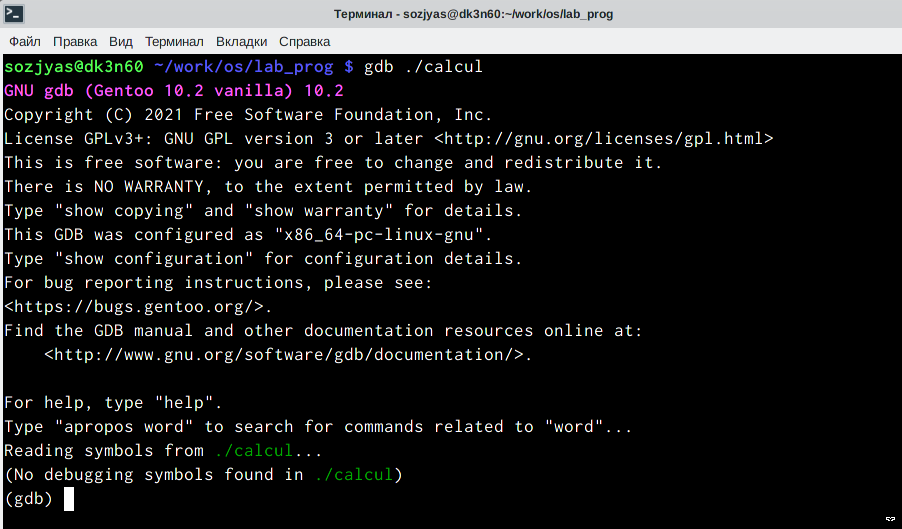


Рис. 9: Запуск отладчика GDB

* Запустил программу внутри отладчика (рис. 10)

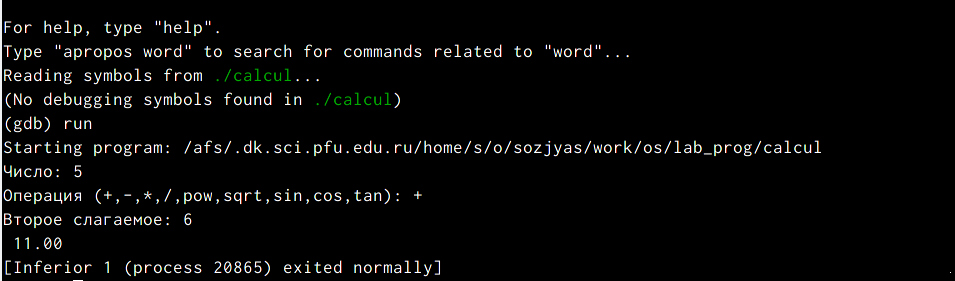


Рис. 10: Запуск программы внутри отладчика

* Просмотрел исходный код используя команду list (рис. 11)

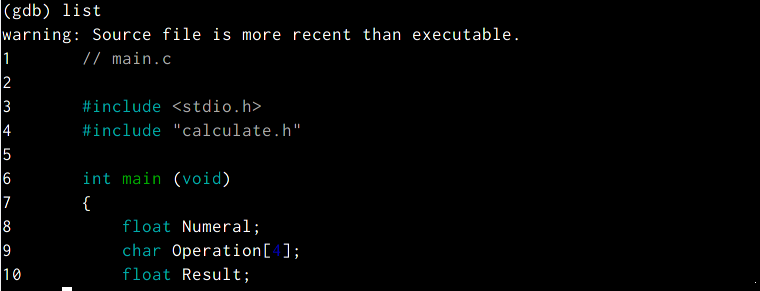


Рис. 11: Просмотр исходного кода

* Просмотрел строки с 12 по 15 основного файла (рис. 12)

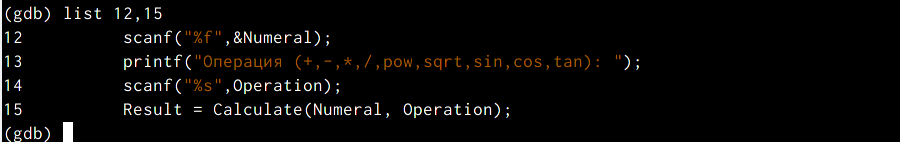


Рис. 12: Просмотр строк с 12 по 15 основного файла

* Просмотрел определённые строки не основного файла (рис. 13)

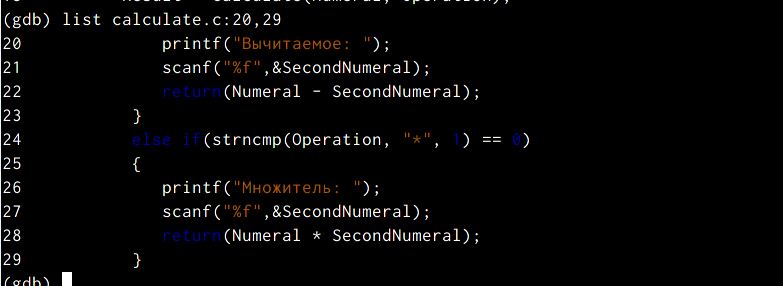


Рис. 13: Просмотр определённых строк не основного файла

* Установил точку останова в файле calculate.c на строке номер 21: (рис. 14)



Рис. 14: Установка точки останова в файле calculate.c

* Вывел информацию об имеющихся в проекте точка останова: (рис. 15)

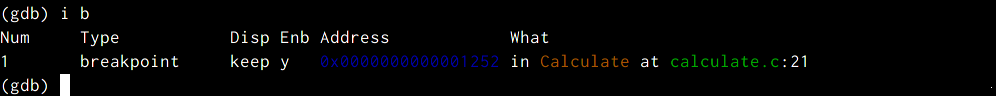


Рис. 15: Вывод информации о точке останова

* Запустил программу внутри отладчика и убедился, что программа остановился в момент прохождения точки останова: (рис. 16)

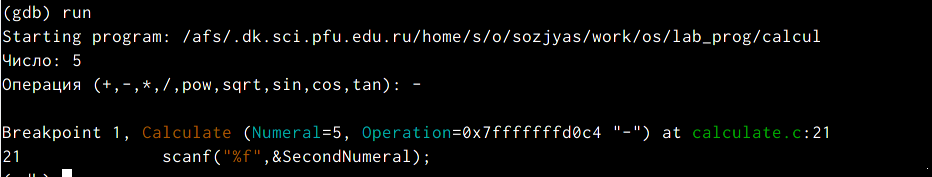


Рис. 16: Запуск программы внутри отладчика при наличии точки останова

* backtrace (рис. 17)

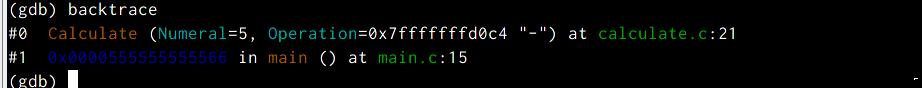


Рис. 17: backtrace

* Посмотрел, чему равно на этом этапе значение переменной Numeral (рис. 18)

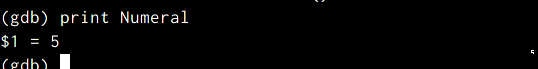


Рис. 18: Значение переменной Numeral на этом этапе

* Сравнил с результатом вывода на экран (рис. 19)

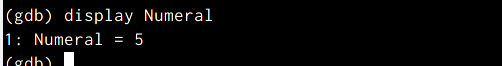


Рис. 19: Сравнение с результатом вывода на экран после исползования команды display

* Убрал точку останова: (рис. 20)

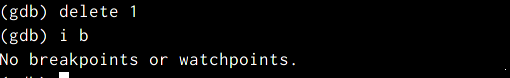


Рис. 20: Удаление точки останова

1. С помощью утилиты splint попробуйте проанализировал коды файлов:
   * calculate.c (рис. 21)

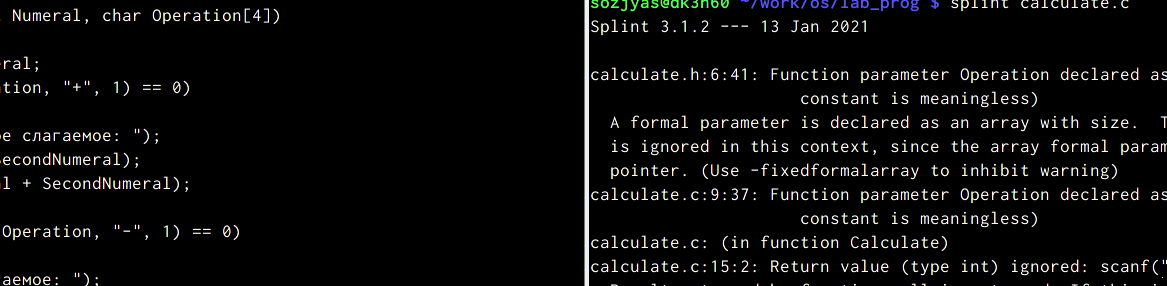


Рис. 21: Анализ кода файла calculate.c

* и main.c. (рис. 22)

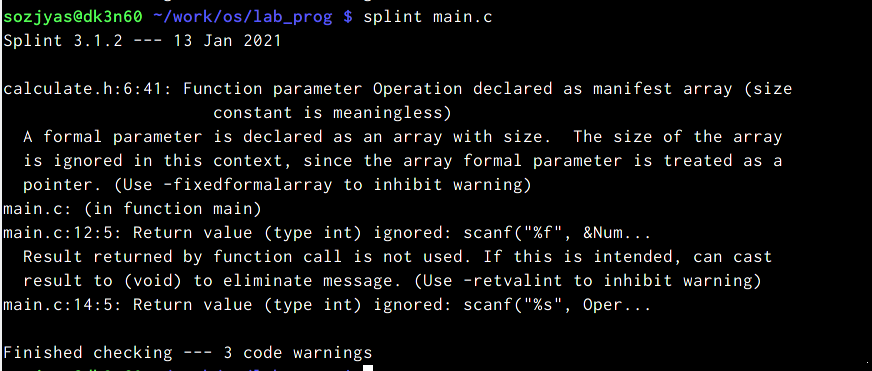


Рис. 22: Анализ кода файла main.c

# 5 Выводы

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

# 6 Контрольные вопросы

1. Получить информацию о возможностях программ gcc, make, gdb и др. можно с помощью команд man.
2. Процесс разработки программного обеспечения обычно разделяется на следующие этапы:
   * планирование, включающее сбор и анализ требований к функционалу и другим характеристикам разрабатываемого приложения;
   * проектирование, включающее в себя разработку базовых алгоритмов и спецификаций, определение языка программирования;
   * непосредственная разработка приложения:
   * кодирование — по сути создание исходного текста программы (возможно в нескольких вариантах);
   * анализ разработанного кода;
   * сборка, компиляция и разработка исполняемого модуля;
   * тестирование и отладка, сохранение произведённых изменений;
   * документирование.
   * Для создания исходного текста программы разработчик может воспользоваться любым удобным для него редактором текста: vi, vim, mceditor, emacs, geany и др.
   * После завершения написания исходного кода программы (возможно состоящей из нескольких файлов), необходимо её скомпилировать и получить исполняемый модуль.
3. Суффикс в контексте языка программирования означает расширение файла программы.
   * Пример исползования - gcc -c main.c
4. Компилятор языка С в UNIX предназначен для сборки разрабатываемой программы, написанной на языке С.
5. Она позволяет автоматизировать процесс преобразования файлов программы из одной формы в другую, отслеживает взаимосвязи между файлами.
6. Пример структуры Makefile
   * hello: main.c
   * gcc -o hello main.c
7. Любой отладчик имеет способность поискать и устранять ошибки в программе. Для его использования необходимо скомпилировать анализируемый код программы таким образом, чтобы отладочная информация содержалась в результирующем бинарном файле.
8. Основные команды GDB

| Команда | Описание действия |
| --- | --- |
| backtrace | вывод на экран пути к текущей точке останова (по сути вывод названий всех функций) |
| break | установить точку останова (в качестве параметра может быть указан номер строки или название функции) |
| continue | продолжить выполнение программы |
| info breakpoints | вывести на экран список используемых точек останова |
| delete | удалить точку останова |
| list | вывести на экран исходный код |
| run | запуск программы на выполнение |

1. Опысание схемы отладки
   * calcul: calculate.o main.o
     + gcc calculate.o main.o -o calcul $(LIBS) команда для создания исполдняемого файла calcul
   * calculate.o: calculate.c calculate.h
     + gcc -c calculate.c $(CFLAGS) команда для создания объектного файла calculate.o
   * main.o: main.c calculate.h
     + gcc -c main.c $(CFLAGS) команда для создания объектного файла main.o
2. Он обнаруживает синтаксические и семантические ошибки.
3. Основные средства, повышающие понимание исходного кода программы - GDB и splint.
4. Splint анализирует программный код, проверяет корректность задания аргументов использованных в программе функций и типов возвращаемых значений, обнаруживает синтаксические и семантические ошибки.

# Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: <https://www.gnu.org/software/bash/manual/>.

2. Newham C. Learning the bash Shell: Unix Shell Programming. O’Reilly Media, 2005. 354 с.

3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.

4. Robbins A. Bash Pocket Reference. O’Reilly Media, 2016. 156 с.

5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.

6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.