

Цель работы

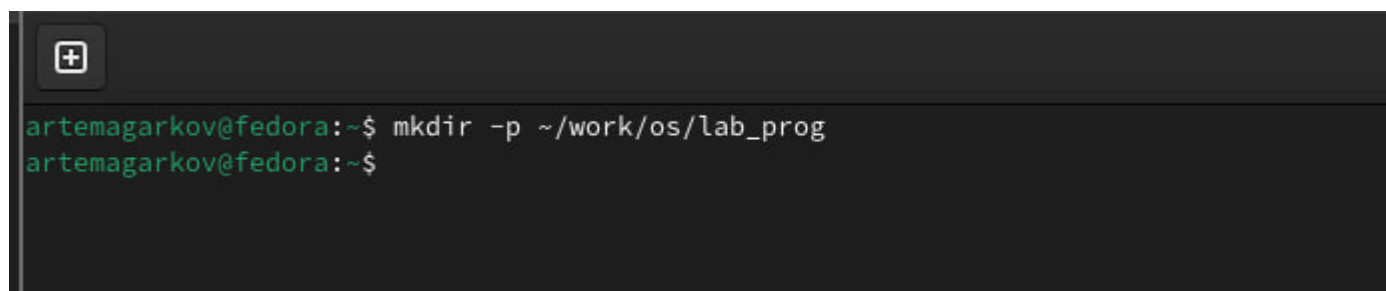
Приобрести простейшие навыки разработки, анализа, тестирования и отладки приложений в ОС типа UNIX/Linux на примере создания на языке программирования C калькулятора с простейшими функциями.

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 функции калькулятора: Основной файл `main.c`, реализующий интерфейс пользователя к калькулятору:
3. Выполните компиляцию программы посредством `gcc`:
4. При необходимости исправьте синтаксические ошибки.
5. Создайте `Makefile` со следующим содержанием:
6. С помощью `gdb` выполните отладку программы `calcul` (перед использованием `gdb` исправьте `Makefile`): – Запустите отладчик GDB, загрузив в него программу для отладки:
7. С помощью утилиты `splint` попробуйте проанализировать коды файлов `calculate.c` и `main.c`.

Выполнение лабораторной работы

1. В домашнем каталоге создайте подкаталог `~/work/os/lab_prog`.

A terminal window with a dark background. The prompt is `artemagarkov@fedora:~$`. The user enters the command `mkdir -p ~/work/os/lab_prog`. The prompt changes to `artemagarkov@fedora:~$` after the command is executed.

```
artemagarkov@fedora:~$ mkdir -p ~/work/os/lab_prog
artemagarkov@fedora:~$
```

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 функции калькулятора: Основной файл `main.c`, реализующий интерфейс пользователя к калькулятору:

```
artemagarkov@fedora:~$ mkdir -p ~/work/os/lab_prog
artemagarkov@fedora:~$ cd ~/work/os/lab_prog
artemagarkov@fedora:~/work/os/lab_prog$ touch calculate.h calculate.c main.c
artemagarkov@fedora:~/work/os/lab_prog$ vi calculate.h
artemagarkov@fedora:~/work/os/lab_prog$ vi calculate.c

[1]+  Stopped                  vi calculate.c
artemagarkov@fedora:~/work/os/lab_prog$ vi calculate.c
artemagarkov@fedora:~/work/os/lab_prog$ vi calculate.c
artemagarkov@fedora:~/work/os/lab_prog$ vi calculate.h
artemagarkov@fedora:~/work/os/lab_prog$ vi main.c
artemagarkov@fedora:~/work/os/lab_prog$
```

3. Выполните компиляцию программы посредством gcc:

```
artemagarkov@fedora:~/work/os/lab_prog$ gcc -c calculate.c
artemagarkov@fedora:~/work/os/lab_prog$ gcc -c main.c
artemagarkov@fedora:~/work/os/lab_prog$ gcc calculate.o main.o -o calcul -lm
artemagarkov@fedora:~/work/os/lab_prog$
```

4. Создайте Makefile со следующим содержанием.

```
artemagarkov@fedora:~/work/os/lab_prog$ vi Makefile
artemagarkov@fedora:~/work/os/lab_prog$
```

5. С помощью gdb выполните отладку программы calcul (перед использованием gdb исправьте Makefile): – Запустите отладчик GDB, загрузив в него программу для отладки:

```
#  
# Makefile  
#  
CC = gcc  
CFLAGS =  
LIBS = -lm  
  
calcul: calculate.o main.o  
    $(CC) calculate.o main.o -o calcul $(LIBS)  
  
calculate.o: calculate.c calculate.h  
    $(CC) -c calculate.c $(CFLAGS)  
  
main.o: main.c calculate.h  
    $(CC) -c main.c $(CFLAGS)  
  
clean:  
    -rm calcul *.o *~  
  
# End Makefile
```

```

artemagarkov@fedora:~/work/os/lab_prog$ vi Makefile
artemagarkov@fedora:~/work/os/lab_prog$ vi Makefile
artemagarkov@fedora:~/work/os/lab_prog$ make
make: 'calcul' is up to date.
artemagarkov@fedora:~/work/os/lab_prog$ gdb ./calcul
GNU gdb (Fedora Linux) 14.2-1.fc40
Copyright (C) 2023 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-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://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...

This GDB supports auto-downloading debuginfo from the following URLs:
    <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for /home/artemagarkov/work/os/lab_prog/calcul
Download failed: No route to host. Continuing without separate debug info for /home/
(No debugging symbols found in ./calcul)
(gdb)
(gdb) run
Starting program: /home/artemagarkov/work/os/lab_prog/calcul
Downloading separate debug info for system-supplied DSO at 0x7ffff7fc7000
Downloading separate debug info for /lib64/libm.so.6
Downloading separate debug info for /lib64/libc.so.6
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 2
Операция (+, -, *, /, pow, sqrt, sin, cos, tan): +
Второе слагаемое: 4
    6.00
[Inferior 1 (process 6224) exited normally]
(gdb)

```

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

```

artemagarkov@fedora:~/work/os/lab_prog$ vi Makefile
artemagarkov@fedora:~/work/os/lab_prog$ vi Makefile
artemagarkov@fedora:~/work/os/lab_prog$ make
make: 'calcul' is up to date.
artemagarkov@fedora:~/work/os/lab_prog$ gdb ./calcul
GNU gdb (Fedora Linux) 14.2-1.fc40
Copyright (C) 2023 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-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://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...

This GDB supports auto-downloading debuginfo from the following URLs:
    <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for /home/artemagarkov/work/os/lab_prog/calcul
Download failed: No route to host. Continuing without separate debug info for /home/
(No debugging symbols found in ./calcul)
(gdb)
(gdb) run
Starting program: /home/artemagarkov/work/os/lab_prog/calcul
Downloading separate debug info for system-supplied DSO at 0x7ffff7fc7000
Downloading separate debug info for /lib64/libm.so.6
Downloading separate debug info for /lib64/libc.so.6
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
Число: 2
Операция (+, -, *, /, pow, sqrt, sin, cos, tan): +
Второе слагаемое: 4
    6.00
[Inferior 1 (process 6224) exited normally]
(gdb)

```

13.5. Контрольные вопросы и ответы

1. Как получить информацию о возможностях программ gcc, make, gdb и др.?
 - Информацию о возможностях программ gcc, make, gdb и других утилит можно получить из

их документации, которая часто включена в состав системы UNIX. Кроме того, можно использовать команды `--help` или `-h`, чтобы получить краткую справку о доступных опциях и ключах.

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

- Написание исходного кода на языке программирования (например, C, C++, Python).
- Компиляция исходного кода в исполняемый файл с помощью компилятора (например, gcc).
- Создание Makefile для автоматизации сборки программы.
- Отладка программы с использованием отладчика (например, gdb).
- Тестирование программы на различных входных данных.
- Улучшение и оптимизация кода.

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

- В контексте языка программирования суффикс - это часть имени файла, которая указывает на тип или формат файла. Например, в файлах исходного кода на C обычно используются суффиксы `.c`, `.h`, а в исполняемых файлах - суффикс `.exe` (для Windows) или отсутствует (для UNIX).

4. Каково основное назначение компилятора языка C в UNIX?

- Основное назначение компилятора языка C в UNIX - это преобразование исходного кода на языке C в машинный код, который может быть выполнен на конкретной архитектуре процессора.

5. Для чего предназначена утилита make?

- Утилита make предназначена для автоматизации процесса сборки программы из исходных файлов. Она читает файл Makefile, содержащий инструкции о том, какие файлы нужно компилировать и как их компилировать, и выполняет соответствующие действия.

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

- `target: dependencies command1 command2 ...`

7. Назовите основное свойство, присущее всем программам отладки. Что необходимо

сделать, чтобы его можно было использовать?

- Основное свойство, присущее всем программам отладки, - это возможность пошагового выполнения программы и анализа её состояния во время выполнения. Для использования этой возможности необходимо скомпилировать программу с отладочной информацией и запустить её в отладчике.

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

- run: запуск программы.
- break: установка точки останова.
- list: просмотр исходного кода.
- print: вывод значения переменной.
- next: выполнение следующей строки.
- step: выполнение следующей строки с заходом в функции.
- continue: продолжение выполнения программы до следующей точки останова.

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

- Схема отладки программы включает в себя компиляцию программы с отладочной информацией, запуск программы в отладчике, установку точек останова, пошаговое выполнение программы, анализ переменных и структуры программы для выявления ошибок.

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

- При первом запуске компилятора gcc программа выводит сообщения об ошибках и предупреждениях в коде программы, если они есть. Это могут быть синтаксические ошибки, неправильное использование функций или другие проблемы, которые необходимо исправить.

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

- Основные средства, повышающие понимание исходного кода программы, включают в себя комментарии в коде, разбиение кода на логические блоки, использование осмысленных имен переменных и функций, а также документацию к программе.

12. Каковы основные задачи, решаемые программой splint?

- Программа splint предназначена для анализа исходного кода на языке программирования C и выявления потенциальных проблем с его безопасностью и надежностью. Основные задачи, решаемые программой splint, включают обнаружение утечек памяти, неинициализированных переменных, ошибок в использовании указателей, а также других типов ошибок, которые могут привести к непредсказуемому поведению программы или уязвимостям безопасности.

Выводы

В результате выполнения лабораторной работы были приобретены простейшие навыки разработки, анализа, тестирования и отладки приложений в операционных системах типа UNIX/Linux. Процесс создания калькулятора на языке программирования C включал в себя создание файлов с кодом (calculate.c, main.c, calculate.h), компиляцию и сборку программы с использованием утилит gcc и make, а также отладку с помощью отладчика gdb. В ходе работы были освоены основные этапы разработки приложений в UNIX, включая написание и отладку кода, использование инструментов сборки и отладки, а также анализ кода с использованием инструментов статического анализа, таких как splint. Полученные навыки позволяют создавать простые программы и эффективно их отлаживать и анализировать.