презентация по лабораторной работе №14

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

Российский Университет Дружбы Народов

Факультет Физико-Математических и Естественных Наук

Дисциплина: Операционные системы

Студент: Алхатиб Осама

Группа: НПИбд-02-20

2021г.

1. ***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

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: /////////////////////////////////////// // calculate.c #include <stdio.h> #include <math.h> #include <string.h> #include “calculate.h” float Calculate(float Numeral, char Operation[4]) { float SecondNumeral; if(strncmp(Operation, “+”, 1) == 0) { printf(“Второе слагаемое:”); scanf(“%f”,&SecondNumeral); return(Numeral + SecondNumeral); } else if(strncmp(Operation, “-”, 1) == 0) { printf(“Вычитаемое:”); scanf(“%f”,&SecondNumeral); return(Numeral - SecondNumeral); } else if(strncmp(Operation, “”, 1) == 0) { printf(”Множитель: ”); scanf(”%f”,&SecondNumeral); return(Numeral SecondNumeral); } else if(strncmp(Operation,”/“, 1) == 0) { printf(”Делитель: “); scanf(”%f“,&SecondNumeral); if(SecondNumeral == 0) { printf(”Ошибка: деление на ноль! “); return(HUGE_VAL); } else return(Numeral / SecondNumeral); } else if(strncmp(Operation,”pow“, 3) == 0) { printf(”Степень: “); scanf(”%f“,&SecondNumeral); return(pow(Numeral, SecondNumeral)); } else if(strncmp(Operation,”sqrt“, 4) == 0) return(sqrt(Numeral)); else if(strncmp(Operation,”sin“, 3) == 0) return(sin(Numeral)); else if(strncmp(Operation,”cos“, 3) == 0) return(cos(Numeral)); else if(strncmp(Operation,”tan“, 3) == 0) return(tan(Numeral)); else { printf(”Неправильно введено действие ”); return(HUGE_VAL); } } Интерфейсный файл calculate.h, описывающий формат вызова функциикалькулятора: /////////////////////////////////////// // calculate.h #ifndef CALCULATE_H_ #define CALCULATE_H_ float Calculate(float Numeral, char Operation[4]); #endif /CALCULATE_H_/ Основной файл main.c, реализующий интерфейс пользователя к калькулятору: /////////////////////////////////////// // main.c 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 со следующим содержанием:

Makefile # CC = gcc CFLAGS = LIBS = -lm calcul: calculate.o main.o gcc

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

1. Выполнение лабораторной работы 1. В домашнем каталоге создал подкаталог ~/work/os/lab_prog. 2. **Figure 3.1: каталог** 2. Создал в нём файлы: calculate.h, calculate.c, main.c.

Figure 3.2: файлы:

Figure 3.3: calculate.h файл

Figure 3.4: calculate.c файл

Figure 3.5: main.c файл 1. Выполнил компиляцию программы посредством gcc Figure 3.6: компиляция gcc 4. Исправил синтаксические ошибки в файле main.c (удалил & перед operator в линии scanf(“%s”,&Operation);) 1. Создал Makefile со следующим содержанием

Figure 3.7: Makefile 6. С помощью gdb выполнил отладку программы calcul (перед использованием gdb исправьте Makefile) • Запустил отладчик GDB, загружил в него программу для отладки: gdb ./calcul

Figure 3.8: gdb • Для запуска программы внутри отладчика ввел команду run: run

Figure 3.9: run • Для постраничного (по 9 строк) просмотра исходного код использовал команду list: list

Figure 3.10: list • Для просмотра строк с 12 по 15 основного файла использовал list с параметрами: list 12,15

Figure 3.11: list • Для просмотра определённых строк не основного файла использовал list с параметрами: list calculate.c:20,29

Figure 3.12: list • Установил точку останова в файле calculate.c на строке номер 21: list calculate.c:20,27 break 21

Figure 3.13: list • Вывел информацию об имеющихся в проекте точка останова: info breakpoints

Figure 3.14: breakpoints • Запустил программу внутри отладчика и убедитесь, что программа остановится в момент прохождения точки останова: run 5 • backtrace

Figure 3.15: run • Посмотрил, чему равно на этом этапе значение переменной Numeral, введя: print Numeral **Figure 3.16: print** • Сравнил с результатом вывода на экран после использования команды: display Numeral

Figure 3.17: display • Убрал точки останова: info breakpoints delete 1 **Figure 3.18: delete** 1. С помощью утилиты splint попробовал проанализировать коды файлов calculate.c и main.c.

Figure 3.19: splint calculate.c Figure 3.20: splint main.c

---

4. Выводы В результате работы , я приобрёл простейшие навыки разработки, анализа, тестирования и отладки приложений в Линук