Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №5 по курсу
«Операционные системы»

Студент: Шатунова Юлия Викторовна
Группа: М8О-208Б-20
Вариант: 19
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

OS/lab5 at main · s0bakkaa/OS (github.com)

Постановка задачи

Цель работы

Приобретение практических навыков в:

- Создание динамических библиотек
- Создание программ, которые используют функции динамических библиотек

Задание

Требуется создать динамические библиотеки, которые реализуют определенный функционал.
Далее использовать данные библиотеки 2-мя способами:
1. Во время компиляции (на этапе «линковки»/linking)
2. Во время исполнения программы. Библиотеки загружаются в память с помощью
интерфейса ОС для работы с динамическими библиотеками
В конечном итоге, в лабораторной работе необходимо получить следующие части:

- Динамические библиотеки, реализующие контракты, которые заданы вариантом;
- Тестовая программа (программа No1), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
- Тестовая программа (программа No2), которая загружает библиотеки, используя только их местоположение и контракты.

Провести анализ двух типов использования библиотек.

Пользовательский ввод для обоих программ должен быть организован следующим образом:

1. Если пользователь вводит команду «0», то программа переключает одну реализацию контрактов на другую (необходимо только для программы No2). Можно реализовать лабораторную работу без данной функции, но максимальная оценка в этом случае будет «хорошо»;
2. «1 arg1 arg2 ... argN», где после «1» идут аргументы для первой функции, предусмотренной контрактами. После ввода команды происходит вызов первой функции, и на экране появляется результат её выполнения;
3. «2 arg1 arg2 ... argM», где после «2» идут аргументы для второй функции, предусмотренной контрактами. После ввода команды происходит вызов второй функции, и на экране появляется результат её выполнения

N	Описание	Сигнатура	Реализация 1	Реализация 2
3	Подсчёт количества простых чисел на отрезке [A, B] (A, B - натуральные)	Int PrimeCount(int A, int B)	Наивный алгоритм. Проверить делимость текущего числа на все предыдущие числа.	Решето Эратосфена
7	Подсчет площади плоской геометрической фигуры по двум сторонам	Float Square(float A, float B)	Фигура прямоугольник	Фигура прямоугольный треугольник

Общие сведения о программе

Программа компилируется в двух файлах: static_main.c и dynamic_main.c

Используемые библиотечные вызовы:

void *dlopen(const char *filename, int flag);	Загружает динамическую библиотеку, имя которой указано в строке filename и возвращает прямой указатель на начало загруженной библиотеки.
void *dlsym(void *handle, char *symbol);	Получает параметр handle, который является выходом вызова dlopen и параметр symbol, который является строкой, в которой содержится название символа, который необходимо загрузить из библиотеки. Возвращает указатль на область памяти, в которой содержится необходимый символ.
int dlclose(void *handle);	Уменьшает счетчик ссылок на указатель handle и если он равен нулю, то освобождает библиотеку.

Общий метод и алгоритм решения

Для реализации поставленной задачи необходимо:

1. Изучить работу с библиотеками.
2. Реализовать две библиотеки согласно заданию.
3. Реализовать две программы (для работы с динамическими и статическими библиотеками).

Исходный код

realization.h

```
#ifndef REALIZATION_H
#define REALIZATION_H

extern int PrimeCount(int A, int B);
extern float Square(float A, float B);

# endif // REALIZATION_H
```

realization1.c

```
#include "realization.h"

int PrimeCount(int A, int B) {
	int N = B - A - 1 + 2;
	int sieve[2][N];
	for (int index = 0; index < N; ++index) {
		sieve[0][index] = index + A;
		sieve[1][index] = 0;
	}

	for (int indexExt = 1; indexExt < N; ++indexExt) {
		for (int indexInt = 0; indexInt < indexExt; ++indexInt) {
			if (sieve[0][indexExt] % sieve[0][indexInt] == 0) {
				sieve[1][indexExt] = 1;
			}
		}
	}

	return sieve;
}
```

```
float Square(float A, float B) {
	return A * B;
}
```

realization2.c

```
#include "realization.h"

int PrimeCount(int A, int B) {
	int N = B - A - 1 + 2;
	int sieve[2][N];
	int temp[N];
	int min = 2;

	for (int index = 0; index < N; ++index) {
		sieve[0][index] = index + A;
		sieve[1][index] = 0;
		temp[index] = 0;
	}

	for (int indexExt = min; indexExt * indexExt < N; ++indexExt) {
		if (temp[indexExt] == 0) {
			for (int indexInt = indexExt * indexExt; indexInt < N; indexInt +=
indexExt) {
				temp[indexInt] = 1;
			}
		}
	}

	for (int index = A; index <= B; ++index) {
		sieve[1][index - A] = temp[index];
```

```
	}

	return sieve;
}

float Square(float A, float B) {
	return 0.5 * A * B;
}
```

static_main.c

```
#include <stdio.h>
#include "realization.h"

int main() {
	int cmd = 0;
	while(scanf("%d", &cmd) != EOF) {
		switch(cmd) {
			case 1: {
				int A, B;
				if (scanf("%d %d", &A, &B) == 2) {
					if (A < 2 || B < 2) {
						printf("The number A and the number B must be 2 or larger\n");
					}
					else if (A > B) {
						printf("The number B must be larger than number A or equal\n");
					}
					else {
						printf("Hello\n");
					}
				}
```

```
                }
                break;
            case 2: {
                float numA, numB;
                if (scanf("%f %f", &numA, &numB) == 2) {
                    if (numA < 0 || numB < 0) {
                        printf("The number A and the number B must be non-negative\n");
                    }
                    else {
                        printf("The square is %.2f\n", Square(numA, numB));
                    }
                }
                }
                break;
            default: {
                printf("Wrong answer\n");
            }
        }
    }
    return 0;
}
```

dynamic_main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>

typedef enum {
    first,
    second,
```

```
} Contract;

Contract contract = first;

const char* lib1 = "libFirst.so";
const char* lib2 = "libSecond.so";

int (*PrimeCount)(int, int) = NULL;
float (*Square)(float, float) = NULL;

void *libHandle = NULL;

void libLoad(Contract cont) {
	const char* name;
	switch(cont) {
		case first:
			name = lib1;
			break;
		case second:
			name = lib2;
			break;
	}
	libHandle = dlopen(name, RTLD_LAZY);
	if (libHandle == NULL) {
		perror("dlopen");
		exit(EXIT_FAILURE);
	}
}

void contractLoad() {
	libLoad(contract);
	PrimeCount = dlsym(libHandle, "PrimeCount");
```

```
        Square = dlsym(libHandle, "Square");
}

void contractChange() {
        dlclose(libHandle);
        switch(contract) {
                case first:
                        contract = second;
                        break;
                case second:
                        contract = first;
                        break;
        }
        contractLoad;
}

int main() {
        int cmd = 0;
        contractLoad();
        while(scanf("%d", &cmd) != EOF) {
                switch(cmd) {
                        case 0: {
                                        contractChange();
                                        printf("The contract is changed\n");
                                        switch(contract) {
                                                case first:
                                                        printf("Number of contract: 1\n");
                                                        break;
                                                case second:
                                                        printf("Number of contract: 2\n");
                                                        break;
                                        }
```

```
                }
                break;
            case 1: {
                int A, B;
                if (scanf("%d %d", &A, &B) == 2) {
                    if (A < 2 || B < 2) {
                        printf("The number A and the number B
must be 2 or larger\n");
                    }
                    else if (A > B) {
                        printf("The number B must be larger than
number A or equal\n");
                    }
                    else {
                        printf("Hello\n");
                    }
                }
                }
                break;
            case 2: {
                float numA, numB;
                if (scanf("%f %f", &numA, &numB) == 2) {
                    if (numA < 0 || numB < 0) {
                        printf("The number A and the number B
must be non-negative\n");
                    }
                    else {
                        printf("The square is %.2f\n", Square(numA,
numB));
                    }
                }
                }
                break;
            default: {
```

```
                printf("Wrong answer\n");
            }
        }
    }
    return 0;
}
```

Сборка программы

```
[yulia@andromeda lab5]$ gcc -fPIC -lm -c realization1.c -o First.o
[yulia@andromeda lab5]$ gcc -fPIC -lm -c realization2.c -o Second.o
[yulia@andromeda lab5]$ gcc -shared -o libFirst.so First.o
[yulia@andromeda lab5]$ gcc -shared -o libSecond.so Second.o
[yulia@andromeda lab5]$ sudo cp libFirst.so /usr/lib
[yulia@andromeda lab5]$ sudo cp libSecond.so /usr/lib
[root@andromeda lab5]# gcc static_main.c -lFirst -lm -o static1
[root@andromeda lab5]# gcc static_main.c -lSecond -lm -o static2
[root@andromeda lab5]# gcc dynamic_main.c -ldl -lm -o dynamic
```

Демонстрация работы программы

```
[root@andromeda lab5]# ./static1
1 2 10
4
2 2 3
The square is 6.00

[root@andromeda lab5]# ./static2
1 2 12
5
2 5 7
The square is 17.50

[root@andromeda lab5]# ./dynamic
0
The contract is changed
Number of contract: 2
1 2 7
4
2 2 5
The square is 5.00
```

Выводы

В ходе лабораторной работы я познакомилась с созданием динамических библиотек в ОС Linux, а также с возможностью загружать эти библиотеки в ходе выполнения программы. Динамические библиотеки помогают уменьшить размер исполняемых файлов. Загрузка динамических библиотек во время выполнения также упрощает компиляцию. Однако также можно подключить библиотеку к программе на этапе линковки. Она все равно загрузится при выполнении, но теперь программа будет изначально знать что и где искать. Если библиотека находится не в стандартной для динамических библиотек директории, необходимо также сообщить линкеру, чтобы тот передал необходимый путь в исполняемый файл. При помощи библиотек мы можем писать более сложные вещи, которые используют простые функции, структуры и т.п., написанные ранее и сохраненные в различных библиотеках.