Московский авиационный институт
(национальный исследовательский университет)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»
Дисциплина «Операционные системы»

Лабораторная работа №5

Тема: Динамические библиотеки

Студент: Инютин М. А.
Группа: М8О-207Б-19
Преподаватель: Миронов Е. С.
Дата:
Оценка:

Москва, 2020

1. Постановка задачи

Требуется создать динамические библиотеки, которые реализуют определенный функционал. Далее использовать данные библиотеки двумя 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 следующее:

- Динамические библиотеки, реализующие контракты, которые заданы вариантом;
- Тестовая программа №1, которая использует одну из библиотек, используя знания, полученные на этапе компиляции;
- Тестовая программа №2, которая загружает библиотеки, используя 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 на другую (необходимо только для программы №2);
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.

Вариант 26.

№	Описание	Сигнатура	Реализация №1	Реализация №2
1	Подсчёт наибольшего общего делителя для двух натуральных чисел	`int GCD(int x, int y)`	Алгоритм Евклида	Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B.
2	Отсортировать целочисленный массив	`void Sort(int* arr, const size_t size)`	Сортировка пузырьком	Быстрая сортировка

2. Описание программы

Объявление функции расположим в заголовочном файле. Для статической реализации нам нужно компилировать основной файл вместе с динамической библиотекой. Для динамической нужно указать компилятору путь к библиотекам и открывать их с помощью системного вызова dlopen. При этом нужно хранить указатели на функции и каждый раз подгружать реализации функций из динамических библиотек с помощью dlsym. Для сортировки дополнительно нужно вводить размер массива.

3. Набор тестов

Программа обрабатывает строки по окончания ввода. На каждой строке расположена команда, как описано в постановке задачи.

```
1
12 18
2
10
3 2 1 10 9 8 7 6 5 4
0
1
24 81
2
10
5 4 3 2 1 10 9 8 7 6
0
1
8 4
2
10
9 8 7 6 5 4 3 2 1 10
0
1
25 49
2
10
10 9 8 7 6 5 4 3 2 1
0
1
800 900
2
10
3 2 1 9 8 7 6 10 5 4
```

4. Результат выполнения тестов

Программа выводит результат выполнения команды, кроме команды «0». После смены реализации программа ничего не выводит.

```
GCD(12, 18) = 6
Sorted: 1 2 3 4 5 6 7 8 9 10
GCD(24, 81) = 3
Sorted: 1 2 3 4 5 6 7 8 9 10
GCD(8, 4) = 4
Sorted: 1 2 3 4 5 6 7 8 9 10
GCD(25, 49) = 1
Sorted: 1 2 3 4 5 6 7 8 9 10
GCD(800, 900) = 100
Sorted: 1 2 3 4 5 6 7 8 9 10
```

5. Листинг программы

Программа разделена на файлы declaration.h с объявлением функций, implementation1.c и implementation2.c с соответствующими реализациями функций, static_main.c и dynamic_main.c для обработки команд пользователя.

declaration.h

```
#ifndef DECLARATION_H
#define DECLARATION_H

#include <stddef.h>

inline void Swap(int* x, int* y) {
	int tmp = *x;
	*x = *y;
	*y = tmp;
}

extern int GCD(int x, int y);
extern void Sort(int* arr, const size_t size);

#endif /* DECLARATION_H */
```

implementation1.c

```
#include "declaration.h"

int GCD(int x, int y) {
	while (y > 0) {
		if (x >= y) {
			x = x % y;
		}
		Swap(&x, &y);
	}
	return x;
}

void Sort(int* arr, const size_t size) {
	for (size_t i = 0; i < size; ++i) {
		for (size_t j = 1; j < size; ++j) {
			if (arr[j - 1] > arr[j]) {
				Swap(&arr[j - 1], &arr[j]);
			}
		}
	}
}
```

implementation2.c

```c
#include "declaration.h"

int GCD(int x, int y) {
	if (x > y) {
		Swap(&x, &y);
	}
	for (int i = x; i > 1; --i) {
		if (x % i == 0 && y % i == 0) {
			return i;
		}
	}
	return 1;
}

void QuickSort(int* arr, const size_t l, const size_t r) {
	if (l + 1 >= r) {
		return;
	}
	size_t i = l, j = r - 2;
	while (i < j) {
		if (arr[i] < arr[r - 1]) {
			++i;
		} else if (arr[j] > arr[r - 1]) {
			--j;
		} else {
			Swap(&arr[i], &arr[j]);
			++i;
			--j;
		}
	}
	if (arr[i] < arr[r - 1]) {
		++i;
	}
	Swap(&arr[i], &arr[r - 1]);
	QuickSort(arr, l, i);
	QuickSort(arr, i + 1, r);
}

void Sort(int* arr, const size_t n) {
	QuickSort(arr, 0, n);
}
```

static_main.c

```
#include "declaration.h"
#include <stdio.h>
#include <stdlib.h>

#define check(VALUE, OKVAL, MSG) if (VALUE != OKVAL) { printf("%s", MSG); return 1; }

int main() {
    int q;
    while (scanf("%d", &q) > 0) {
        if (q == 1) {
            int x, y;
            check(scanf("%d%d", &x, &y), 2, "Error reading integer!\n");
            printf("GCD(%d, %d) = %d\n", x, y, GCD(x, y));
        } else if (q == 2) {
            size_t n;
            check(scanf("%lu", &n), 1, "Error reading integer!\n");
            int* a = malloc(sizeof(int) * n);
            for (size_t i = 0; i < n; ++i) {
                check(scanf("%d", &a[i]), 1, "Error reading integer\n");
            }
            Sort(a, n);
            printf("Sorted: ");
            for (size_t i = 0; i < n; ++i) {
                printf("%d ", a[i]);
            }
            printf("\n");
            free(a);
        }
    }
}
```

dynamic_main.c

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

#define check(VALUE, OKVAL, MSG) if (VALUE != OKVAL) {
printf("%s", MSG); return 1; }

const char* DYN_LIB_1 = "./libimpl1.so";
const char* DYN_LIB_2 = "./libimpl2.so";
const char* GCD_FUNC_NAME = "GCD";
const char* SORT_FUNC_NAME = "Sort";

int main() {
     int dynLibNum = 1;
     void* handle = dlopen(DYN_LIB_1, RTLD_LAZY);
     if (handle == NULL) {
          printf("Error opening dynamic library!\n");
          return 1;
     }
     int (*GCD)(int, int);
     void (*Sort)(int*, const size_t);
     *(void**) (&GCD) = dlsym(handle, GCD_FUNC_NAME);
     *(void**) (&Sort) = dlsym(handle, SORT_FUNC_NAME);
     char* error = dlerror();
     check(error, NULL, error);
     int q;
     while (scanf("%d", &q) > 0) {
          if (q == 0) {
               check(dlclose(handle), 0, "Error closing dynamic
library!\n");
               if (dynLibNum) {
                    handle = dlopen(DYN_LIB_2, RTLD_LAZY);
               } else {
                    handle = dlopen(DYN_LIB_1, RTLD_LAZY);
               }
               if (handle == NULL) {
                    printf("Error opening dynamic library!\n");
                    return 1;
               }
               *(void**) (&GCD) = dlsym(handle, GCD_FUNC_NAME);
               *(void**) (&Sort) = dlsym(handle, SORT_FUNC_NAME);
               error = dlerror();
               check(error, NULL, error);
               dynLibNum = dynLibNum ^ 1;
          } else if (q == 1) {
               int x, y;
               check(scanf("%d%d", &x, &y), 2, "Error reading
integer!\n");
               printf("GCD(%d, %d) = %d\n", x, y, GCD(x, y));
          } else if (q == 2) {
```

```
                size_t n;
                check(scanf("%lu", &n), 1, "Error reading integer!\
n");
                int* a = malloc(sizeof(int) * n);
                for (size_t i = 0; i < n; ++i) {
                     check(scanf("%d", &a[i]), 1, "Error reading
integer\n");
                }
                Sort(a, n);
                printf("Sorted: ");
                for (size_t i = 0; i < n; ++i) {
                     printf("%d ", a[i]);
                }
                printf("\n");
                free(a);
           }
     }
     check(dlclose(handle), 0, "Error closing dynamic library!\n");
}
```

6. Выводы

Во время выполнения работы я изучил основы работы с динамическими библиотеками на операционных системах Linux, реализовал программу, которая использует созданные динамические библиотеки. Я узнал про особенности *inline* и *extern* перед функциями при линковке файлов с общим *include*. Использование библиотек добавляет модульность программе, что упрощает дальнейшую поддержку кода.

Список литературы

1. Анатомия динамических библиотек Linux — IBM
   URL: https://www.ibm.com/developerworks/ru/library/l-dynamic-libraries/
   (дата обращения 30.11.2020)
2. Shared libraries with GCC on Linux — Cprogramming.com
   URL: https://www.cprogramming.com/tutorial/shared-libraries-linux-gcc.html
   (дата обращения 30.11.2020)
3. dlopen(3) — Linux manual page — man7.org
   URL: https://www.man7.org/linux/man-pages/man3/dlopen.3.html
   (дата обращения 30.11.2020)
4. dlsym(3) — Linux manual page — man7.org
   URL: https://man7.org/linux/man-pages/man3/dlsym.3.html
   (дата обращения 30.11.2020)