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

Лабораторная работа №5 по курсу
«Операционные системы»

Динамические библиотеки.

Студент: Ивенкова Л.В
Группа: М80 – 208Б-19
Вариант: 24
Преподаватель: Миронов Е. С.
Дата: _____

Оценка: _____

Подпись: _____

Москва, 2021

1. Постановка задачи

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

- во время компиляции (на этапе линковки)
- во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками

В лабораторной работе необходимо получить следующие части:

- Динамические библиотеки, реализующие контракты
- Тестовая программа (программа №1), которая использует одну из библиотек, используя знания, полученные на этапе компиляции.
- Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение и контракты.

Провести анализ использования двух типов библиотек.

Пользовательский ввод должен быть организован следующим образом:

- Команда «0»: переключить одну реализацию контрактов на другую
- Команда «1 args»: вызов первой функции контрактов
- Команда «2 args»: вызов второй функции контрактов

Контракты:

- Подсчёт наибольшего общего делителя для двух натуральных чисел (основание натурального логарифма) – при помощи Алгоритма Евклида и при помощи наивного алгоритма (попытаться разделить числа на все числа, что меньше A и B).
- Подсчет площади плоской геометрической фигуры по двум сторонам – площадь прямоугольника и прямоугольного треугольника.

2. Общие сведения о программе

Программа написана на языке Си. Операционная система – Ubuntu.

Контракты описаны в заголовочном файле lib.h. Реализации контрактов описаны в файлах lib1.c и lib2.c.

Сборка программы производится с использованием makefile. Для подключения библиотеки на этапе компиляции выполняются следующие действия:

1. Получение объектного файла основной программы.
2. Компиляция файла библиотеки с ключом -shared. Получаем .so файл.
3. Линковка библиотеки и объектного файла основной программы.
4. Указание пути к библиотеке в переменной LD_LIBRARY_PATH.
   export LD_LIBRARY_PATH=/home/mosik/os_lab5:$LD_LIBRARY_PATH

Для динамической загрузки библиотек используются средства библиотеки dlfcn.h.

3. Общий метод и алгоритм решения

Программа принимает на вход неограниченное количество команд следующего вида:

- 0: переключение библиотеки. Выполняется при помощи функций dlopen, dlclose, dlsym, dlerror. Все системные ошибки обрабатываются.
- 1 x: расчет числа e по формуле $\left(1+\frac{1}{x}\right)^x$ или $\sum_{n=0}^{x} \frac{1}{n!}$.
- 2 size num_1 num_2 … num_size: сортировка массива размера size при помощи пузырьковой или быстрой сортировки.

При вводе неверной команды программа выводит соответствующее сообщения. Для завершения программы можно использовать комбинацию Ctrl + D.

4. Основные файлы программы

lib.h

```
double e(int x);
int* sort(int* arr, int size);
```

lib1.c

```
#include <stdio.h>
#include <math.h>

#include "lib.h"

// counting 'e' number
double e(int x) {
    printf("Counting 'e' number using second remarkable limit\n");
    return pow(1 + 1. / x, x);
}

// bubble sort
int* sort(int* arr, int n) {
    printf("Sorting an array usuing bubble sort\n");
    for (int i = 0; i < n - 1; ++i) {
        for (int j = i; j < n; ++j) {
            if (arr[i] > arr[j]) {
                int tmp = arr[i];
                arr[i] = arr[j];
                arr[j] = tmp;
            }
        }
    }
    return arr;
}
```

lib2.c

```
#include <stdio.h>

// counting 'e' number
double e(int x) {
    printf("Counting 'e' number using Taylor series\n");
    double ans = 0;
    long long factorial = 1;
    for (int n = 0; n <= x; ++n) {
        if (n != 0) {
            factorial *= n;
        }
        ans += 1. / factorial;
    }
    return ans;
```

```
}

// quick sort
void quick_sort(int* arr, int l, int r) {
    int m = arr[(l + r) / 2], i = l, j = r;
    while (i <= j) {
        while (arr[i] < m) {
            ++i;
        }
        while (arr[j] > m) {
            --j;
        }
        if (i <= j) {
            int tmp = arr[i];
            arr[i] = arr[j];
            arr[j] = tmp;
            ++i;
            --j;
        }
    }
    if (l < j) {
        quick_sort(arr, l, j);
    }
    if (i < r) {
        quick_sort(arr, i, r);
    }
}

int* sort(int* arr, int n) {
    printf("Sorting an array usuing quick sort\n");
    quick_sort(arr, 0, n - 1);
    return arr;
}
```

prog1.c

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

#include "lib.h"

int main() {
    printf("1 x --> count 'e' number\n");
    printf("2 size_of_array array --> sort array\n");
    int cmd;
    while (scanf("%d", &cmd) > 0) {
        if (cmd == 1) {
            int x;
            scanf("%d", &x);
            printf("%f\n", e(x));
        }
        else if (cmd == 2) {
            int size;
            scanf("%d", &size);
            int* arr = (int*)malloc(size * sizeof(int));
            for (int i = 0; i < size; ++i) {
                scanf("%d", &arr[i]);
            }
            arr = sort(arr, size);
            for (int i = 0; i < size; ++i) {
```

```
                printf("%d ", arr[i]);
            }
            printf("\n");
            free(arr);
        }
        else {
            printf("Incorrect cmd\n");
        }
    }
}
```

prog2.c

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

#include "lib.h"

int main() {
    printf("0 --> change library\n");
    printf("1 x --> count 'e' number\n");
    printf("2 size_of_array array --> sort array\n");

    int cmd, cur_lib = 1;

    // loading library
    void* handler = dlopen("lib1.so", RTLD_LAZY);
    if (!handler) {
        printf("%s\n", dlerror());
        return 1;
    }

    // resolving functions from library
    double (*e)(int);
    int* (*sort)(int*, int);
    e = dlsym(handler, "e");
    sort = dlsym(handler, "sort");
    if (dlerror() != NULL) {
        printf("%s\n", dlerror());
        return 2;
    }

    while (scanf("%d", &cmd) > 0) {
        if (cmd == 0) {
            if (dlclose(handler) < 0) {
                perror("Can't close dinamic library\n");
                return 3;
            }

            // switching libraries
            if (cur_lib == 1) {
                handler = dlopen("lib2.so", RTLD_LAZY);
                if (!handler) {
                    printf("%s\n", dlerror());
                    return 1;
                }
                cur_lib = 2;
            }
            else {
```

```
                handler = dlopen("lib1.so", RTLD_LAZY);
                if (!handler) {
                    printf("%s\n", dlerror());
                    return 1;
                }
                cur_lib = 1;
            }

            // resolving functions from library
            e = dlsym(handler, "e");
            sort = dlsym(handler, "sort");
            if (dlerror() != NULL) {
                printf("%s\n", dlerror());
                return 2;
            }

        }

        else if (cmd == 1) {
            int x;
            scanf("%d", &x);
            printf("%f\n", e(x));
        }

        else if (cmd == 2) {
            int size;
            scanf("%d", &size);
            int* arr = (int*)malloc(size * sizeof(int));
            for (int i = 0; i < size; ++i) {
                scanf("%d", &arr[i]);
            }
            arr = sort(arr, size);
            for (int i = 0; i < size; ++i) {
                printf("%d ", arr[i]);
            }
            printf("\n");
            free(arr);
        }

        else {
            printf("Incorrect cmd\n");
        }
    }
    dlclose(handler);
}
```

makefile

```
all: prog1 prog2

prog1: lib1.so
    gcc prog1.c -L/home/mosik/os_lab5 -l1 -Wall -o prog1

prog2: lib1.so lib2.so
    gcc prog2.c -ldl -o prog2

lib1.so: lib1.o
    gcc lib1.o -lm -shared -o lib1.so

lib2.so: lib2.o
```

```
    gcc lib2.o -shared -o lib2.so

lib1.o:
    gcc lib1.c -c -Wall -Werror -fpic -o lib1.o

lib2.o:
    gcc lib2.c -c -Wall -Werror -fpic -o lib2.o

clean:
    rm -rf *.so *.o prog1 prog2
```

5. Демонстрация работы программы

```
mosik@LAPTOP-69S778GL:~/os_lab5$ make
gcc lib1.c -c -Wall -Werror -fpic -o lib1.o
gcc lib1.o -lm -shared -o lib1.so
gcc prog1.c -L/home/mosik/os_lab5 -l1 -Wall -o prog1
gcc lib2.c -c -Wall -Werror -fpic -o lib2.o
gcc lib2.o -shared -o lib2.so
gcc prog2.c -ldl -o prog2
mosik@LAPTOP-69S778GL:~/os_lab5$ export
LD_LIBRARY_PATH=/home/mosik/os_lab5:$LD_LIBRARY_PATH

mosik@LAPTOP-69S778GL:~/os_lab5$ ./prog1
1 x --> count 'e' number
2 size_of_array array --> sort array
1 10
Counting 'e' number using second remarkable limit
2.593742
1 100
Counting 'e' number using second remarkable limit
2.704814
2 5
5 3 1 4 2
Sorting an array usuing bubble sort
1 2 3 4 5
2 0
Sorting an array usuing bubble sort

2 10
1 2 3 4 5 6 7 8 9 10
Sorting an array usuing bubble sort
1 2 3 4 5 6 7 8 9 10
3
Incorrect cmd

mosik@LAPTOP-69S778GL:~/os_lab5$ ./prog2
0 --> change library
1 x --> count 'e' number
2 size_of_array array --> sort array
1 10000
Counting 'e' number using second remarkable limit
2.718146
2 5
123 453 12 7 54
Sorting an array usuing bubble sort
```

```
7 12 54 123 453
0
Implementation switched
1
10
Counting 'e' number using Taylor series
2.718282
2 5
2 475 12 8 43
Sorting an array usuing quick sort
2 8 12 43 475
0
Implementation switched
1
100000
Counting 'e' number using second remarkable limit
2.718268
2 3
34 1 67
Sorting an array usuing bubble sort
1 34 67

mosik@LAPTOP-69S778GL:~/os_lab5$ ltrace -o ltrace.log ./prog1
1 x --> count 'e' number
2 size_of_array array --> sort array
1 10
Counting 'e' number using second remarkable limit
2.593742
2 3 2 3 1
Sorting an array usuing bubble sort
1 2 3
mosik@LAPTOP-69S778GL:~/os_lab5$ cat ltrace.log
puts("1 x --> count 'e' number")
= 25
puts("2 size_of_array array --> sort a"...)
= 37
__isoc99_scanf(0x7fe045200bfd, 0x7fffe824be30, 0x7fe044bdd8c0,
0x7fffdf735010)           = 1
__isoc99_scanf(0x7fe045200bfd, 0x7fffe824be34, 0x7fe044bdd8d0, 16)
= 1
e(10, 1, 0x7fe044bdd8d0, 16)
= 0x4004bffc0c03023e
printf("%f\n", 2.593742)
= 9
__isoc99_scanf(0x7fe045200bfd, 0x7fffe824be30, 0, 0)
= 1
__isoc99_scanf(0x7fe045200bfd, 0x7fffe824be34, 0x7fe044bdd8d0, 16)
= 1
malloc(12)
= 0x7fffdf735a80
__isoc99_scanf(0x7fe045200bfd, 0x7fffdf735a80, 0, 0x7fffdf735a80)
= 1
__isoc99_scanf(0x7fe045200bfd, 0x7fffdf735a84, 4, 16)
= 1
__isoc99_scanf(0x7fe045200bfd, 0x7fffdf735a88, 8, 16)
= 1
sort(0x7fffdf735a80, 3, 3, 16)
= 0x7fffdf735a80
```

```
printf("%d ", 1)
= 2
printf("%d ", 2)
= 2
printf("%d ", 3)
= 2
putchar(10, 0x7fe045200c06, 0, 0)
= 10
free(0x7fffdf735a80)
= <void>
__isoc99_scanf(0x7fe045200bfd, 0x7fffe824be30, 0x7fffdf735010, 1)
= 0xffffffff
+++ exited (status 0) +++

mosik@LAPTOP-69S778GL:~/os_lab5$ ltrace -o ltrace.log ./prog2
0 --> change library
1 x --> count 'e' number
2 size_of_array array --> sort array
1
10
Counting 'e' number using second remarkable limit
2.593742
0
Implementation switched
1 10
Counting 'e' number using Taylor series
2.718282
0
Implementation switched
mosik@LAPTOP-69S778GL:~/os_lab5$ cat ltrace.log
puts("0 --> change library")
= 21
puts("1 x --> count 'e' number")
= 25
puts("2 size_of_array array --> sort a"...)
= 37
dlopen("lib1.so", 1)
= 0x7fffd478b6a0
dlsym(0x7fffd478b6a0, "e")
= 0x7f2c775e069a
dlsym(0x7fffd478b6a0, "sort")
= 0x7f2c775e06f0
dlerror()
= nil
__isoc99_scanf(0x7f2c78200e59, 0x7fffdc268e44, 1, 0)
= 1
__isoc99_scanf(0x7f2c78200e59, 0x7fffdc268e48, 0x7f2c77bdd8d0, 16)
= 1
printf("%f\n", 2.593742)
= 9
__isoc99_scanf(0x7f2c78200e59, 0x7fffdc268e44, 0, 0)
= 1
dlclose(0x7fffd478b6a0)
= 0
dlopen("lib2.so", 1)
= 0x7fffd478b6a0
dlsym(0x7fffd478b6a0, "e")
= 0x7f2c775e067a
```

```
dlsym(0x7fffd478b6a0, "sort")
= 0x7f2c775e0839
dlerror()
= nil
puts("Implementation switched")
= 24
__isoc99_scanf(0x7f2c78200e59, 0x7fffdc268e44, 0x7f2c77bdd8c0, 3)
= 1
__isoc99_scanf(0x7f2c78200e59, 0x7fffdc268e48, 0x7f2c77bdd8d0, 16)
= 1
printf("%f\n", 2.718282)
= 9
__isoc99_scanf(0x7f2c78200e59, 0x7fffdc268e44, 0, 0)
= 1
dlclose(0x7fffd478b6a0)
= 0
dlopen("lib1.so", 1)
= 0x7fffd478b6a0
dlsym(0x7fffd478b6a0, "e")
= 0x7f2c775e069a
dlsym(0x7fffd478b6a0, "sort")
= 0x7f2c775e06f0
dlerror()
= nil
puts("Implementation switched")
= 24
__isoc99_scanf(0x7f2c78200e59, 0x7fffdc268e44, 0x7f2c77bdd8c0, 4)
= 0xffffffff
dlclose(0x7fffd478b6a0)
= 0
+++ exited (status 0) +++
```

Примечание: красным цветом выделены системные вызовы, связанные с динамическими библиотеками.

6. Выводы

Данная лабораторная работа была направлена на изучение динамических библиотек в операционной системе Linux. Чтобы разобраться с применением таких библиотек, мной были написаны две программы: одна подключала динамическую библиотеку на стадии линковки, а вторая – непосредственно во время исполнения.

Динамические библиотеки содержат функции, которые будут загружены в оперативную память только тогда, когда они действительно понадобятся. Использование динамических библиотек существенно экономит память и ускоряет процесс сборки программы.

Операционная система Linux позволяет совершенствовать библиотеки «на ходу», когда программа уже запущена и использует какие-то библиотеки. Динамические библиотеки легче обновлять: достаточно исправить только код самой библиотеки.

Однако у динамических библиотек есть и свои недостатки. Например, это достаточно запутанная сборка программ. Также стоит отметить, что вызов функции из динамической библиотеки происходит немного медленнее.

Но преимущества использования динамических библиотек перекрывают все их недостатки. Сейчас компьютеры обладают достаточной мощностью для быстрого выполнения

системных вызовов. Важнее сэкономить объём памяти, используемой программой. Поэтому динамические библиотеки используются в большинстве современных программ.