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: Бирюков В. В.
Группа: М80-207Б-19
Преподаватель: Миронов Е. С.
Дата:
Оценка:

Москва, 2020

Постановка задачи

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:

1. Динамические библиотеки, реализующие контракты, которые заданы вариантом;
2. Тестовая программа (программа №1), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
3. Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение и контракты.

Провести анализ двух типов использования библиотек.

Вариант 18.

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
6	Расчет значения числа e (основание натурального логарифма)	float E(int x)	(1 + 1/x) ^ x	Сумма ряда по n от 0 до x, где элементы ряда равны: (1/(n!))

Алгоритм решения задачи

Объявления функций разместим в отдельном заголовочном файле.

Для «статического» использования библиотеки необходимо при компиляции указать ее имя и путь, по которому она находится. Также необходимо, чтобы сам исполняемый файл знал где искать эту библиотеку, так как она находится не в стандартном месте.

Для динамического использования необходимо знать полные имена библиотек, поэтому поместим их в массив. При необходимости изменить версию функций, загрузим следующую библиотеку. Для выполнения функции необходимо загружать её из библиотеки в соответствующий указатель.

Листинг программы

```c
// functions.h
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

extern int PrimeCount(int, int);

extern float E(int);

#endif

// functions1.c
#include "functions.h"

#include <math.h>

float E(int x) {
     return powf(1 + 1.0 / x, x);
}

#define max(a, b) ((a) > (b)) ? (a) : (b)

int PrimeCount(int A, int B) {
     int n = 0, prime;
     for (int num = max(A, 2); num <= B; ++num) {
          prime = 1;
          for (int i = 2; i < num; ++i) {
               if (num % i == 0) {
                    prime = 0;
                    break;
               }
          }
          if (prime == 1) {
               ++n;
          }
     }
     return n;
}

// functions2.c
#include "functions.h"

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

int factorial(int x) {
     int result = 1;
     for (int i = 2; i <= x; ++i) {
          result *= i;
     }
     return result;
}
```

```c
float E(int x) {
     float result = 0;
     for (int n = 0; n <= x; ++n) {
          result += 1.0 / factorial(n);
     }
     return result;
}

int PrimeCount(int A, int B) {
     int n = 0;
     int *sieve = (int *)malloc(sizeof(int) * (B+1));
     if (sieve == NULL) {
          perror("malloc error");
          exit(1);
     }
     for (int i = 0; i <= B; ++i) sieve[i] = 0;
     for (int i = 2; i <= B; ++i) {
          if (sieve[i] != 0) {
               continue;
          }
          for (int j = i + i; j <= B; j += i) {
               sieve[j] = 1;
          }
     }
     for (int i = A; i <= B; ++i) {
          if (sieve[i] == 0) {
               ++n;
          }
     }
     return n;
}

// static.c
#include "functions.h"

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

int main() {
     int f;
     int a, b, x;
     while (scanf("%d", &f) > 0) {
          if (f == 1) {
               if (scanf("%d %d", &a, &b) != 2) {
                    perror("invalid input");
                    exit(1);
               }
               printf("Prime numbers: %d\n", PrimeCount(a, b));

          } else if (f == 2) {
               if (scanf("%d", &x) != 1) {
                    perror("invalid input");
```

```
                    exit(1);
               }
               printf("e = %f\n", E(x));
          }
     }
}

// dynamic.c
#include "functions.h"

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

#define check(VALUE, MSG, BADVAL) if (VALUE == BADVAL)
{ perror(MSG); exit(1); }

int main() {
     char* libnames[] = {"./libfunctions1.so",
                         "./libfunctions2.so"};
     int lib = 0;
     int (*PrimeCount)(int, int) = NULL;
     float (*E)(int) = NULL;

     void *handle;
     handle = dlopen(libnames[lib], RTLD_NOW);
     check(handle, dlerror(), NULL)

     int f;
     int a, b, x;
     while (scanf("%d", &f) > 0) {
          if (f == 0) {
               if (dlclose(handle) != 0) {
                    perror(dlerror());
                    exit(1);
               }
               lib = (lib + 1) % 2;
               handle = dlopen(libnames[lib], RTLD_NOW);
               check(handle, dlerror(), NULL);

          } else if (f == 1) {
               if (scanf("%d %d", &a, &b) != 2) {
                    perror("invalid input");
                    exit(1);
               }
               PrimeCount = (int (*)(int, int))dlsym(handle,
                                                     "PrimeCount");
               check(PrimeCount, dlerror(), NULL);
               printf("Prime numbers: %d\n", (*PrimeCount)(a, b));

          } else if (f == 2) {
               if (scanf("%d", &x) != 1) {
                    perror("invalid input");
```

```
                    exit(1);
               }
               E = (float (*)(int))dlsym(handle, "E");
               check(E, dlerror(), NULL);
               printf("e = %f\n", (*E)(x));
          }
     }
     if (dlclose(handle) != 0) {
          perror(dlerror());
          exit(1);
     }
}

# makefile
all: static dynamic libfunctions

static: libfunctions
     gcc -o static static.c -L./ -lfunctions2 -Wl,-rpath=./

dynamic: libfunctions
     gcc -rdynamic -o dynamic dynamic.c -ldl

libfunctions: functions
     gcc -shared -o libfunctions1.so functions1.o -lm
     gcc -shared -o libfunctions2.so functions2.o

functions: functions.h functions1.c functions2.c
     gcc -fPIC -c functions1.c
     gcc -fPIC -c functions2.c

clean:
     rm -f *.so *.o static dynamic
```

Тесты и протокол исполнения

```
./static
1 100 199
Prime numbers: 21
2 7
e = 2.718254
2 4
e = 2.708333
2 20
e = 2.718282

./dynamic
1 100 199
Prime numbers: 21
2 20
e = 2.653295
2 4
e = 2.441406
2 1000
```

```
e = 2.717051
0
1 100 199
Prime numbers: 21
2 20
e = 2.718282
```

Выводы

В ходе лабораторной работы я познакомился 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.

Список литературы

1. Динамическая загрузка библиотек в Linux – URL: https://webhamster.ru/mytetrashare/index/mtb0/938
2. Таненбаум Э., Бос Х. *Современные операционные системы. – 4-е изд.* – СПб.: Издательский дом «Питер», 2018.