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: О. В. Бабин
Преподаватель: А. А. Соколов
Группа: М8О-206Б-19
Дата: 25.12.2020
Оценка:
Подпись:

Москва, 2020

1 Постановка задачи

Цель работы:
Приобретение практических навыков в:

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

Задание (вариант 26):
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, загрузив библиотеки в память с помощью системных вызовов

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.

Вариант 26

1. Подсчёт наибольшего общего делителя для двух натуральных чисел
   Реализация 1 Алгоритм Евклида
   Реализация 2 Наивный алгоритм. Пытаться разделить числа на все числа, что меньше A и B

2. Отсортировать целочисленный массив
   Реализация 1 Пузырьковая сортировка
   Реализация 2 Сортировка Хоара

2 Общие сведения о программе

Лабораторная состоит из двух программ. В первом случае, где мы подключаем библиотеку на этапе линковки, всё тривиально. Во втором требуется использовать ряд системных вызовов, таких как dlopen, dlsym и dlclose. Также требуется особым образом собирать программу.

3 Общий метод и алгоритм решения

Для реализации поставленной задачи необходимо:

1. Изучить принципы создания и использования динамических библиотек
2. Написать функции библиотеки
3. Написать первую программу
4. Написать вторую программу, используя системные вызовы
5. Написать обработку ошибок
6. Собрать проект
7. Написать тесты

4 Исходный код

Первая программа

main.c

```
1
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  #include "mylib.h"
6
7  #define GCD_INPUT_ERROR 1
8  #define SORT_SIZE_INPUT_ERROR 2
9  #define SORT_ARRAY_INPUT_ERROR 3
10
11 void gcd_cmd() {
12   int a, b;
13   if (scanf("%d%d", &a, &b) != 2) {
14     fprintf(stderr, "Func 1 input error\n");
15     exit(GCD_INPUT_ERROR);
16   }
17   printf("gcd: %d\n", gcd1(a, b));
18 }
19
20 void read_array(int* array, size_t size) {
21   for (size_t i = 0; i < size; ++i) {
22     if (scanf("%d", array + i) != 1) {
23       fprintf(stderr, "Func 2 array input error\n");
24       exit(SORT_ARRAY_INPUT_ERROR);
25     }
26   }
27 }
28
29 void print_array(int* array, size_t size) {
30   printf("sorted: ");
31   for (size_t i = 0; i < size; ++i) {
32     printf("%d ", array[i]);
33   }
34   printf("\n");
35 }
36
37 void sort_cmd() {
38   int size;
39   if (scanf("%d", &size) != 1) {
40     fprintf(stderr, "Func 2 size input error\n");
41     exit(SORT_SIZE_INPUT_ERROR);
42   }
43   int* array = malloc(size * sizeof(int));
44   read_array(array, size);
```

```
  sort1(array, size);
  print_array(array, size);
}

int main() {
  int cmd;
  while (scanf("%d", &cmd) == 1) {
    switch (cmd) {
      case 1:
        gcd_cmd();
        break;
      case 2:
        sort_cmd();
        break;
      default:
        fprintf(stderr, "Undefined command\n");
        break;
    }
  }
  return 0;
}
```

mylib.h

```

#pragma once

int gcd1(int a, int b);
int gcd2(int a, int b);

void sort1(int* array, unsigned long size);
void sort2(int* array, unsigned long size);
```

mylib.c

```

#include <stdlib.h>

void swap(unsigned int* a, unsigned int* b) {
  unsigned int t = *a;
  *a = *b;
  *b = t;
}

int gcd1(int a_inp, int b_inp) {
  unsigned int a = a_inp, b = b_inp;
  while (b) {
    a %= b;
    swap(&a, &b);
  }
  return a;
```

```
}

unsigned int min(unsigned int a, unsigned int b) {
  if (a < b) {
    return a;
  }
  return b;
}

int gcd2(int a_inp, int b_inp) {
  unsigned int a = a_inp, b = b_inp;
  for (int i = min(a, b); i > 1; --i) {
    if (a % i == 0 && b % i == 0) {
      return i;
    }
  }
  return 1;
}

void sort1(int* array, unsigned long size) {
  for (unsigned long i = 0; i < size; ++i) {
    for (unsigned long j = 0; j < size - 1; ++j) {
      if (array[j] > array[j + 1]) {
        swap(array + j, array + j + 1);
      }
    }
  }
}

int compare(const void* a, const void* b) {
  return (*(int*)a - *(int*)b);
}

void sort2(int* array, unsigned long size) {
  qsort(array, size, sizeof(int), compare);
}
```

Вторая программа

main.c

```
1
2  #include <dlfcn.h>
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  #define GCD_INPUT_ERROR 1
7  #define SORT_SIZE_INPUT_ERROR 2
8  #define SORT_ARRAY_INPUT_ERROR 3
9  #define CANNOT_OPEN_LIBRARY_ERROR 4
10 #define CANNOT_FIND_FUNC_ERROR 5
11
12 void* lib;
13 int (*gcd)(int, int);
14 void (*sort)(int*, unsigned long);
15
16 void gcd_cmd() {
17   int a, b;
18   if (scanf("%d%d", &a, &b) != 2) {
19     fprintf(stderr, "Func 1 input error\n");
20     exit(GCD_INPUT_ERROR);
21   }
22   printf("gcd: %d\n", gcd(a, b));
23 }
24
25 void read_array(int* array, size_t size) {
26   for (size_t i = 0; i < size; ++i) {
27     if (scanf("%d", array + i) != 1) {
28       fprintf(stderr, "Func 2 array input error\n");
29       exit(SORT_ARRAY_INPUT_ERROR);
30     }
31   }
32 }
33
34 void print_array(int* array, size_t size) {
35   printf("sorted: ");
36   for (size_t i = 0; i < size; ++i) {
37     printf("%d ", array[i]);
38   }
39   printf("\n");
40 }
41
42 void sort_cmd() {
43   int size;
44   if (scanf("%d", &size) != 1) {
45     fprintf(stderr, "Func 2 size input error\n");
46     exit(SORT_SIZE_INPUT_ERROR);
47   }
```

```
48    int* array = malloc(size * sizeof(int));
49    read_array(array, size);
50    sort(array, size);
51    print_array(array, size);
52  }
53
54  int is_first_implementation = 1;
55  void swap_functions() {
56    if (is_first_implementation) {
57      gcd = dlsym(lib, "gcd1");
58      sort = dlsym(lib, "sort1");
59    } else {
60      gcd = dlsym(lib, "gcd2");
61      sort = dlsym(lib, "sort2");
62    }
63    is_first_implementation = !is_first_implementation;
64    if (gcd == NULL || sort == NULL) {
65      fprintf(stderr, "Cannot find functions\n");
66      exit(CANNOT_FIND_FUNC_ERROR);
67    }
68  }
69
70  void load_lib() {
71    lib = dlopen("libmy.so", RTLD_LAZY);
72    if (!lib) {
73      fprintf(stderr, "Cannot open library libmy.so\n");
74      exit(CANNOT_OPEN_LIBRARY_ERROR);
75    }
76    swap_functions();
77  }
78
79  void close_lib() {
80    dlclose(lib);
81  }
82
83  int main() {
84    load_lib();
85    int cmd;
86    while (scanf("%d", &cmd) == 1) {
87      switch (cmd) {
88        case 0:
89          swap_functions();
90          break;
91        case 1:
92          gcd_cmd();
93          break;
94        case 2:
95          sort_cmd();
96          break;
```

```
97        default:
98          fprintf(stderr, "Undefined command\n");
99          break;
100     }
101   }
102   close_lib();
103   return 0;
104 }
```

mylib.c

```
1
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  void swap(unsigned int* a, unsigned int* b) {
6    unsigned int t = *a;
7    *a = *b;
8    *b = t;
9  }
10
11 int gcd1(int a_inp, int b_inp) {
12   fprintf(stderr, "gcd first implementation\n");
13   unsigned int a = a_inp, b = b_inp;
14   while (b) {
15     a %= b;
16     swap(&a, &b);
17   }
18   return a;
19 }
20
21 unsigned int min(unsigned int a, unsigned int b) {
22   if (a < b) {
23     return a;
24   }
25   return b;
26 }
27
28 int gcd2(int a_inp, int b_inp) {
29   fprintf(stderr, "gcd second implementation\n");
30   unsigned int a = a_inp, b = b_inp;
31   for (int i = min(a, b); i > 1; --i) {
32     if (a % i == 0 && b % i == 0) {
33       return i;
34     }
35   }
36   return 1;
37 }
38
39 void sort1(int* array, unsigned long size) {
```

```
40    fprintf(stderr, "sort first implementation\n");
41    for (unsigned long i = 0; i < size; ++i) {
42      for (unsigned long j = 0; j < size - 1; ++j) {
43        if (array[j] > array[j + 1]) {
44          swap(array + j, array + j + 1);
45        }
46      }
47    }
48  }
49
50  int compare(const void* a, const void* b) {
51    return (*(int*)a - *(int*)b);
52  }
53
54  void sort2(int* array, unsigned long size) {
55    fprintf(stderr, "sort second implementation\n");
56    qsort(array, size, sizeof(int), compare);
57  }
```

5 Пример работы

Продемонстрирую процесс сборки второй программы:

```
windicor@Lina-HP:~$ gcc -fPIC -c mylib.c
windicor@Lina-HP:~$ gcc -shared -o libmy.so mylib.o
windicor@Lina-HP:~$ gcc -c main.c
windicor@Lina-HP:~$ gcc main.o -Wl,-rpath,. -ldl
```

Первый тест - проверка на обработку корректных входных данных:

```
windicor@Lina-HP:~$ cat test1
1 4 6
1 10 10
1 1000000 2000000
2 5 1 2 3 4 5
2 5 5 4 3 2 1
2 5 3 4 2 5 1
0
1 4 6
1 10 10
1 1000000 2000000
2 5 1 2 3 4 5
2 5 5 4 3 2 1
2 5 3 4 2 5 1
windicor@Lina-HP:~$ ./a.out <test1
gcd first implementation
gcd: 2
gcd first implementation
gcd: 10
gcd first implementation
gcd: 1000000
sort first implementation
sorted: 1 2 3 4 5
sort first implementation
sorted: 1 2 3 4 5
sort first implementation
sorted: 1 2 3 4 5
```

```
gcd second implementation
gcd: 2
gcd second implementation
gcd: 10
gcd second implementation
gcd: 1000000
sort second implementation
sorted: 1 2 3 4 5
sort second implementation
sorted: 1 2 3 4 5
sort second implementation
sorted: 1 2 3 4 5
```

Вторым тестом продемонстрируем обработку некорректного ввода:

```
windicor@Lina-HP:~$ cat test2
5 4 6
9 4 2
2 5
windicor@Lina-HP:~$ ./a.out <test2
Undefined command
Undefined command
Undefined command
Undefined command
Undefined command
Func 2 array input error
```

Третьим тестом продемонстрируем работу программы при отсутствии библиотеки libmy.so:

```
windicor@Lina-HP:~$ ls
USING.txt  a.out  main.c  main.o  mylib.c  mylib.o  not_a_libmy.so
windicor@Lina-HP:~$ ./a.out
Cannot open library libmy.so
```

6 Вывод

В процессе работы над лабораторной я научился основам работы с динамическими библиотеками в Си. Это важный и мощный инструмент в разработке ПО, поскольку позволяет экономить память, а также подгружать библиотеки динамически. Трудностью оказалась некоторая запутанность документации по теме, а также слишком неинформативные статьи об этом на русском.