Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №5 по курсу «Операционные сети»
Тема: Динамические библиотеки

Студент: Д. С. Ляшун
Преподаватель: Е. С. Миронов
Группа: М8О-207Б

Дата:
Оценка:
Подпись:

Москва, 2020

1 Постановка задачи

Цель работы: приобретение практических навыков в создании динамических библиотек и программ, которые используют их функции.

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:

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.

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 №2). 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 №30.

Контракты и реализации функций:

float Pi(int k) – расчёт значения числа Пи при заданной длине ряда (K). Реализация через ряд Лейбница и формулу Валлиса.

int* Sort(int* array) – сортировка целочисленного массива. Реализация через пузырьковую сортировку и сортировку Хоара.

2 Алгоритм решения

Для вычисления значения Пи через ряд Лейбница необходимо воспользоваться следующей формулой: $\sum_{i=0}^{\infty}((-1)^i)/(2i+1) = \pi/4$, которая может быть реализовано программно в цикле по k (длина ряда) итераций, на каждом из которых к общему значению суммы добавляется значение элемента в ряде для i. Похожим образом будет реализован расчёт числа Пи посредством формулы Валлиса, которая имеет вид: $\prod_{i=1}^{\infty}(2i)^2/((2i-1)(2i+1)) = \pi/2$.

Идея пузырьковой сортировки основывается на повторяющихся проходах по сортируемому массиву. На каждой итерации последовательно сравниваются соседние элементы, и, если порядок в паре неверный, то элементы меняют местами. За каждый проход по массиву как минимум один элемент встает на свое место, поэтому необходимо совершить не более $n-1$ проходов, где n размер массива, чтобы отсортировать массив.

Сортировка массива методом Хоара функционирует по принципу «разделяй и властвуй». Массив $a[l \dots r]$ разбивается на два (возможно пустых) подмассива $a[l \dots q]$ и $a[q+1 \dots r]$, таких, что каждый элемент $a[l \dots q]$ меньше или равен $a[q]$, который в свою очередь, не превышает любой элемент подмассива $a[q+1 \dots r]$. Индекс вычисляется в ходе процедуры разбиения. Подмассивы $a[l \dots q]$ и $a[q+1 \dots r]$ сортируются с помощью рекурсивного вызова процедуры быстрой сортировки. Поскольку подмассивы сортируются на месте, для их объединения не требуются никакие действия: весь массив $a[l \dots r]$ оказывается отсортированным.

При работе с динамическими библиотеками будут использоваться следующие системные вызовы:

1. void *dlopen(const char *filename, int flag) – загружает динамическую библиотеку, имя которой указано в строке filename, и возвращает прямой указатель на начало динамической библиотеки (нулевой указатель в случае неуспеха). Режим flag указывает компоновщику, работающему с динамическими библиотеками, когда производить перемещение данных указанной библиотеки. Возможные варианты – RTLD_NOW (сделать все необходимые перемещения в момент вызова dlopen) и RTLD_LAZY (перемещения по требованию).

2. void *dlsym(void *handle, char *symbol) – использует указатель handle на динамическую библиотеку, возвращаемую dlopen(), и оканчивающееся нулем символьное имя symbol, а затем возвращает адрес, указывающий, откуда загружается этот символ. Если символ не найден, то возвращаемым значением dlsym является NULL.

3. int dlclose(void *handle) – выгружает динамическую библиотеку (уменьшает

счётчик ссылок на неё), на которую указывает handle. В случае успеха возвращает 0, иначе номер ошибки.

4. const char *dlerror(void) – возвращает текстовое описание последней ошибки, возникшей при вызове функций, работающих с динамическими библиотеками.

3 Листинг программы

Исходный код functions.h:

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H
float Pi(int k);
int* Sort(int* array, int n);
#endif
```

Исходный код realization1.c:

```
#include "functions.h"
float Pi(int k)
{
    float value = 0;
    for (int i = 0; i <= k; ++i)
    {
        value += 1.0 * (i % 2 == 0? 1 : -1) / (2 * i + 1);
    }
    return value * 4;
}
int* Sort(int* array, int n)
{
  int oper = 1;
  while (oper)
  {
    oper = 0;
    for (int i = 1; i < n; ++i)
    {
      if (array[i-1] > array[i])
      {
        int swap = array[i];
        array[i] = array[i-1];
        array[i-1] = swap;
        oper = 1;
      }
    }
  }
  return array;
}
```

Исходный код realization2.c:

```
#include "functions.h"
float Pi(int k)
{
    float value = 1;
    for (int i = 1; i <= k; ++i)
    {
        value *= 4.0 * i * i / ((2 * i - 1) * (2 * i + 1));
```

```
    }
    return value * 2;
}
void FastSort(int* a, int l, int r)
{
  if (l < r)
  {
    int v = a[(l + r) / 2];
    int i = l;
    int j = r;
    while (i <= j)
    {
      while (a[i] < v) ++i;
      while (a[j] > v) --j;
      if (i >= j) break;
      else
      {
        int swap = a[i];
        a[i] = a[j];
        a[j] = swap;
        ++i;
        --j;
      }
    }
    FastSort(a, l, j);
    FastSort(a, j + 1, r);
  }
}
int* Sort(int* array, int n)
{
  FastSort(array, 0, n - 1);
  return array;
}
```

Исходный код static_main.c:

```
#include "functions.h"
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int comand;
    while (scanf("%d", &comand) != EOF)
    {
        if (comand == 1)
        {
            int k;
            scanf("%d", &k);
            float value = Pi(k);
            printf("Pi(%d) = %f\n", k, value);
```

```
15          }
16          else if (comand == 2)
17          {
18              int n;
19              scanf("%d", &n);
20              int* array = NULL;
21              array = malloc(n * sizeof(int));
22              if (array == NULL)
23              {
24                  printf("Error allocating memory!\n");
25                  return -1;
26              }
27              for (int i = 0; i < n; ++i)
28              {
29                  scanf("%d", &array[i]);
30              }
31              array = Sort(array, n);
32              printf("Sort(array) = [ ");
33              for (int i = 0; i < n; ++i)
34              {
35                  printf("%d", array[i]);
36                  if (i != n - 1) printf(", ");
37              }
38              printf(" ]\n");
39              free(array);
40          }
41          else
42          {
43            printf("Error! Wrong input!\n");
44          }
45      }
46      return 0;
47  }
```

Исходный код dynamic_main.c:

```
1   #include <stdio.h>
2   #include <dlfcn.h>
3   #include <stdlib.h>
4   #define check_ok(VAR, VAL) if (VAR != VAL) { printf("Error: %s\n", dlerror()); return
        -1; }
5   #define check_wrong(VAR, VAL) if (VAR == VAL) { printf("Error: %s\n", dlerror());
        return -1; }
6   const char* LIBRARY1_NAME = "./realization1.so";
7   const char* LIBRARY2_NAME = "./realization2.so";
8   const char* FUNCTION1_NAME = "Pi";
9   const char* FUNCTION2_NAME = "Sort";
10  int main()
11  {
12      void* dl_handler = dlopen(LIBRARY1_NAME, RTLD_LAZY);
```

```
13      check_wrong(dl_handler, NULL);
14      int type = 1;
15      int command;
16      while (scanf("%d", &command) != EOF)
17      {
18          if (command == 0)
19          {
20            check_ok(dlclose(dl_handler), 0);
21              dl_handler = (type == 1? dlopen(LIBRARY2_NAME, RTLD_LAZY) : dlopen(
                    LIBRARY1_NAME, RTLD_LAZY));
22              check_wrong(dl_handler, NULL);
23              type = (type == 1? 2 : 1);
24              printf("Change dynamic library from %d to %d\n", (type == 1? 2 : 1), type);
25          }
26          else if (command == 1)
27          {
28              float (*Pi)(int) = dlsym(dl_handler, FUNCTION1_NAME);
29              check_wrong(Pi, NULL);
30              int k;
31              scanf("%d", &k);
32              float value = (*Pi)(k);
33              printf("Pi(%d) = %f\n", k, value);
34          }
35          else if (command == 2)
36          {
37              int* (*Sort)(int*, int) = dlsym(dl_handler, FUNCTION2_NAME);
38              check_wrong(Sort, NULL);
39              int n;
40              scanf("%d", &n);
41              int* array = NULL;
42              array = malloc(n * sizeof(int));
43              if (array == NULL)
44              {
45                  printf("Error allocating memory!\n");
46                  return -1;
47              }
48              for (int i = 0; i < n; ++i)
49              {
50                  scanf("%d", &array[i]);
51              }
52              array = (*Sort)(array, n);
53              printf("Sort(array) = [ ");
54              for (int i = 0; i < n; ++i)
55              {
56                  printf("%d", array[i]);
57                  if (i != n - 1) printf(", ");
58              }
59              printf(" ]\n");
60              free(array);
```

```
61          }
62          else
63          {
64            printf("Error! Wrong input!\n");
65          }
66      }
67      check_ok(dlclose(dl_handler), 0);
68      return 0;
69  }
```

Исходный код Makefile

```
1   CXX = gcc
2   CXXFLAGS = -g -O2 -Wextra -Wall -Werror -Wno-sign-compare -Wno-unused-result
3
4   all: static_main2 static_main1 solution
5
6   # Static main creating
7   static_main2: static_main realization2
8     $(CXX) $(CXXFLAGS) static_main.o realization2.o -o static_main2
9
10  static_main1: static_main realization1
11    $(CXX) $(CXXFLAGS) static_main.o realization1.o -o static_main1
12
13  static_main: static_main.c
14    $(CXX) $(CXXFLAGS) -c static_main.c
15
16  realization2: realization2.c functions.h
17    $(CXX) $(CXXFLAGS) realization2.c -c
18
19  realization1: realization1.c functions.h
20    $(CXX) $(CXXFLAGS) realization1.c -c
21
22  # Dynamic main creating
23
24  solution: dynamic_main dyn_realization2 dyn_realization1
25    $(CXX) $(CXXFLAGS) dynamic_main.o -o solution -ldl
26
27  dynamic_main: dynamic_main.c
28    $(CXX) $(CXXFLAGS) -c dynamic_main.c -ldl
29
30  dyn_realization2: realization2.c functions.h
31    $(CXX) $(CXXFLAGS) -fPIC -shared -o realization2.so realization2.c
32
33  dyn_realization1: realization1.c functions.h
34    $(CXX) $(CXXFLAGS) -fPIC -shared -o realization1.so realization1.c
35
36  clean:
37    rm -rf *.o
```

4 Тесты и протокол работы

```
dmitry@dmitry-VirtualBox:~/Work_place/OS_labs/Lab5$ make
gcc -g -O2 -Wextra -Wall -Werror -Wno-sign-compare
-Wno-unused-result  -c static_main.c
gcc -g -O2 -Wextra -Wall -Werror -Wno-sign-compare
-Wno-unused-result  realization2.c -c
gcc -g -O2 -Wextra -Wall -Werror -Wno-sign-compare
-Wno-unused-result  static_main.o realization2.o -o static_main2
gcc -g -O2 -Wextra -Wall -Werror -Wno-sign-compare
-Wno-unused-result  realization1.c -c
gcc -g -O2 -Wextra -Wall -Werror -Wno-sign-compare
-Wno-unused-result  static_main.o realization1.o -o static_main1
gcc -g -O2 -Wextra -Wall -Werror -Wno-sign-compare
-Wno-unused-result  -c dynamic_main.c -ldl
gcc -g -O2 -Wextra -Wall -Werror -Wno-sign-compare
-Wno-unused-result  -fPIC -shared -o realization2.so realization2.c
gcc -g -O2 -Wextra -Wall -Werror -Wno-sign-compare
-Wno-unused-result  -fPIC -shared -o realization1.so realization1.c
gcc -g -O2 -Wextra -Wall -Werror -Wno-sign-compare
-Wno-unused-result  dynamic_main.o -o solution -ldl
dmitry@dmitry-VirtualBox:~/Work_place/OS_labs/Lab5$ cat test1.txt
1 12
1 100
2 10 10 9 8 7 6 5 4 3 2 1
2 1 10
2 2 30 22
1 17
dmitry@dmitry-VirtualBox:~/Work_place/OS_labs/Lab5$ ./static_main1 <test1.txt
Pi(12) = 3.218403
Pi(100) = 3.151493
Sort(array) = [ 1,2,3,4,5,6,7,8,9,10 ]
Sort(array) = [ 10 ]
Sort(array) = [ 22,30 ]
Pi(17) = 3.086080
dmitry@dmitry-VirtualBox:~/Work_place/OS_labs/Lab5$ ./static_main2 <test1.txt
Pi(12) = 3.079401
Pi(100) = 3.133787
Sort(array) = [ 1,2,3,4,5,6,7,8,9,10 ]
Sort(array) = [ 10 ]
Sort(array) = [ 22,30 ]
```

```
Pi(17) = 3.097038
dmitry@dmitry-VirtualBox:~/Work_place/OS_labs/Lab5$ cat test2.txt
1 700
0
1 10000
2 5 9 8 8 4 3
2 1 10
0
2 10 1 2 3 4 5 10 7 8 9 6
0
2 0
1 400
dmitry@dmitry-VirtualBox:~/Work_place/OS_labs/Lab5$ ./solution <test2.txt
Pi(700) = 3.143020
Change dynamic library from 1 to 2
Pi(10000) = 3.141330
Sort(array) = [ 3,4,8,8,9 ]
Sort(array) = [ 10 ]
Change dynamic library from 2 to 1
Sort(array) = [ 1,2,3,4,5,6,7,8,9,10 ]
Change dynamic library from 1 to 2
Sort(array) = [  ]
Pi(400) = 3.139632
```

5 Strace

```
dmitry@dmitry-VirtualBox:~/Work_place/OS_labs/Lab5$ strace ./solution
execve("./solution",["./solution"],0x7ffff2df0060 /* 49 vars */) = 0
brk(NULL)                                = 0x56039f75f000
arch_prctl(0x3001 /* ARCH_??? */,0x7fff72e824e0) = -1 EINVAL (Недопустимый
аргумент)
access("/etc/ld.so.preload",R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD,"/etc/ld.so.cache",O_RDONLY|O_CLOEXEC) = 3
fstat(3,st_mode=S_IFREG|0644,st_size=67205,...) = 0
mmap(NULL,67205,PROT_READ,MAP_PRIVATE,3,0) = 0x7fa8e5a22000
close(3)                                 = 0
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libdl.so.2",O_RDONLY|O_CLOEXEC) = 3
read(3,"77ELF>2"...,832) = 832
fstat(3,st_mode=S_IFREG|0644,st_size=18816,...) = 0
mmap(NULL,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0) = 0x7fa8e5a20000
mmap(NULL,20752,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7fa8e5a1a000
mmap(0x7fa8e5a1b000,8192,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE,3,0x1000) = 0x7fa8e5a1b000
mmap(0x7fa8e5a1d000,4096,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3,0x3000) = 0x7fa8e5a1d000
mmap(0x7fa8e5a1e000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE,3,0x3000) = 0x7fa8e5a1e000
close(3)                                 = 0
openat(AT_FDCWD,"/lib/x86_64-linux-gnu/libc.so.6",O_RDONLY|O_CLOEXEC) = 3
read(3,"77ELF>60q"...,832) = 832
pread64(3,"@@@"...,784,64) = 784
pread64(3,"0GNU00"
,32,848) = 32
pread64(3,"4GNU6377?320070704d45n 355Y77~34"...,68,880) = 68
fstat(3,st_mode=S_IFREG|0755,st_size=2029224,...) = 0
pread64(3,"@@@"
...,784,64) = 784
pread64(3,"0GNU00 0",32,848) = 32
pread64(3,"4GNU6377?320070704d
45n55Y77~34"...,68,880) = 68
mmap(NULL,2036952,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0)
= 0x7fa8e5828000
mprotect(0x7fa8e584d000,1847296,PROT_NONE) = 0
mmap(0x7fa8e584d000,1540096,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE,3,0x25000) = 0x7fa8e584d000
```

```
mmap(0x7fa8e59c5000,303104,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3,0x19d000) = 0x7fa8e59c5000
mmap(0x7fa8e5a10000,24576,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE,3,0x1e7000) = 0x7fa8e5a10000
mmap(0x7fa8e5a16000,13528,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS,-1,0) = 0x7fa8e5a16000
close(3)                                = 0
mmap(NULL,12288,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_ANONYMOUS,-1,0)
= 0x7fa8e5825000
arch_prctl(ARCH_SET_FS,0x7fa8e5825740) = 0
mprotect(0x7fa8e5a10000,12288,PROT_READ) = 0
mprotect(0x7fa8e5a1e000,4096,PROT_READ) = 0
mprotect(0x56039e63c000,4096,PROT_READ) = 0
mprotect(0x7fa8e5a60000,4096,PROT_READ) = 0
munmap(0x7fa8e5a22000,67205)           = 0
brk(NULL)                               = 0x56039f75f000
brk(0x56039f780000)                     = 0x56039f780000
openat(AT_FDCWD,"./realization1.so",O_RDONLY|O_CLOEXEC) = 3
read(3,"77ELF>@0"...,
832) = 832
fstat(3,st_mode=S_IFREG|0775,st_size=18296,...) = 0
getcwd("/home/dmitry/Work_place/OS_labs/Lab5",128) = 37
mmap(NULL,16424,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7fa8e5a2e000
mmap(0x7fa8e5a2f000,4096,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3,0x1000) = 0x7fa8e5a2f000
mmap(0x7fa8e5a30000,4096,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x2000)
= 0x7fa8e5a30000
mmap(0x7fa8e5a31000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3,0x2000) = 0x7fa8e5a31000
close(3)                                = 0
mprotect(0x7fa8e5a31000,4096,PROT_READ) = 0
fstat(0,st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0),...) = 0
read(0,1 20
"1 20",1024)                    = 5
fstat(1,st_mode=S_IFCHR|0620,st_rdev=makedev(0x88,0),...) = 0
write(1,"Pi(20) = 3.189185",18Pi(20) = 3.189185
)     = 18
read(0,0
"0",1024)                       = 2
munmap(0x7fa8e5a2e000,16424)           = 0
openat(AT_FDCWD,"./realization2.so",O_RDONLY|O_CLOEXEC) = 3
```

```
read(3,"77ELF>‘0"...,832) = 832
fstat(3,st_mode=S_IFREG|0775,st_size=19432,...) = 0
getcwd("/home/dmitry/Work_place/OS_labs/Lab5",128) = 37
mmap(NULL,16432,PROT_READ,MAP_PRIVATE|MAP_DENYWRITE,3,0) = 0x7fa8e5a2e000
mmap(0x7fa8e5a2f000,4096,PROT_READ|PROT_EXEC,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3,0x1000) = 0x7fa8e5a2f000
mmap(0x7fa8e5a30000,4096,PROT_READ,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,3,0x2000)
= 0x7fa8e5a30000
mmap(0x7fa8e5a31000,8192,PROT_READ|PROT_WRITE,MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3,0x2000) = 0x7fa8e5a31000
close(3)                                = 0
mprotect(0x7fa8e5a31000,4096,PROT_READ) = 0
write(1,"Change dynamic library from 1 to"...,35Change dynamic library from
1 to 2
) = 35
read(0,2 4 8 12 30 18
"2 4 8 12 30 18",1024)       = 15
write(1,"Sort(array) = [ 8,12,18,30 ]",32Sort(array) = [ 8,12,18,30 ]
) = 32
read(0,1 40
"1 40",1024)                 = 5
write(1,"Pi(40) = 3.122260",18Pi(40) = 3.122260
)     = 18
read(0,^Cstrace: Process 4276 detached
<detached ...>
```

Как видно, сперва вызывается функция execv, которая запускает файл программы на исполнение. Печать и вывод в консоль с использованием функций scanf и printf реализуется соответственно с помощью системных вызовов read и write, где в аргументах указывается буфер, откуда читаются или записываются данные, их размер в байтах, а также дескриптор файла, откуда читаются или записываются данные (0 - стандартный поток ввода, 1 - стандартный поток вывода).

В ходе смены динамической библиотеки видно, что это производится путем вызова системных функций: openat, где указывается имя открываемой библиотеки (с добавлением информации о том, что она находится в текущем рабочем каталоге – ключ AT_FDCWD) на чтение (ключ O_RDONLY), и возвращается дескриптор открытого файла-библиотеки; read - производит чтение данных из библиотеки (возможно, служебных); fstat, использующийся для получения информации по открытому файлу-библиотеке по дескриптору, которая затем используется в mmap; mmap осуществляет отображение содержимого файла в переменную программы для доступа к его символам.

6 Выводы

В результате выполнения данной лабораторной работы я познакомился с использованием динамических библиотек при написании программ в ОС Linux. Основную трудность для меня составило написание файла сборки для статически создаваемой программы, поскольку возникает много зависимостей, чего нет при создании программы, использующей динамические библиотеки. Хочется сказать, что работать с динамическими библиотеками было довольно удобно, применяемые для этого системные вызовы лично для меня были предельно понятны.

Я уверен, что использование динамических библиотек является хорошим подходом при написании программы, поскольку в ходи их работы подгружаются только те символы, которые нужно использовать в данный момент, что более оптимально по с точки зрения выделения памяти в ОЗУ. Также при динамической линковке появляется возможность использовать, например, функции с одинаковой сигнатурой, но разной реализацией, что может быть довольно полезным. Что касается минусов таких программ, то они заключаются в большем времени работы в связи с подгрузкой данных в программу, а также возникающей необходимости обработки всех ошибок при работе с динамическими библиотеками.

Список литературы

[1] Таненбаум Э., Босх Х. *Современные операционные системы. – 4-е изд.* – СПб.: Издательский дом «Питер», 2018. – С. 301-380

[2] *Linux Man Pages*
URL: `http://ru.manpages.org` (дата обращения: 27.11.2020).