Московский авиационный институт
(Национальный исследовательский университет)
Факультет "Информационные технологии и прикладная математика"
Кафедра "Вычислительная математика и программирование"

Лабораторная работа №5 по курсу “Операционные системы”

Студент: Живалев Е.А.

Группа: М8О-206Б

Преподаватель: Соколов А.А.

Вариант: 7

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2019

1 Задание

Требуется создать динамическую библиотеку, которая реализует определенный Функционал - работу с массивом, содержащим целые 32-битные числа.

2 Описание работы программы

В файле arrayAPI.c реализованы следующие функции для работы с массивом: arrayCreate, arrayInsert, arrayGet, arrayDelete, arrayResize, arrayDestroy, arrayPrint. В файле main_dynamic.c эти функции загружаются в память, выделенную для программы

3 Исходный код

main_dynamic.c

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

#include "arrayAPI.h"

int main() {
    array* (*arrayCreate)(size_t size);
    void (*arrayInsert)(array* a, size_t pos, int val);
    int (*arrayGet)(array* a, size_t pos);
    void (*arrayDelete)(array* a, size_t pos);
    void (*arrayResize)(array* a, size_t size);
    void (*arrayDestroy)(array* a);
    void (*arrayPrint)(array* a);
    char* error;
    void* libHandle;
    libHandle = dlopen("./libarr.so", RTLD_LAZY);
    if(!libHandle) {
        printf("%s\n", dlerror());
        return -1;
    }
    arrayCreate = dlsym(libHandle, "arrayCreate");
    arrayInsert = dlsym(libHandle, "arrayInsert");
    arrayGet = dlsym(libHandle, "arrayGet");
    arrayDelete = dlsym(libHandle, "arrayDelete");
    arrayResize = dlsym(libHandle, "arrayResize");
    arrayDestroy = dlsym(libHandle, "arrayDestroy");
    arrayPrint = dlsym(libHandle, "arrayPrint");
    printf("1 - create array with given size\n");
    printf("2 - insert element to array at given position\n");
    printf("3 - get element value on given position\n");
    printf("4 - delete element from given position\n");
    printf("5 - resize array to given size\n");
    printf("6 - print array\n");
    printf("0 - exit\n");
    int command = 0;
    array* a;
    int value = 0;
    int pos = 0;
    while(scanf("%d", &command) && command) {
        if(command == 1) {
            printf("Enter value\n");
            scanf("%d", &value);
            a = (*arrayCreate)(value);
        } else if(command == 2) {
            printf("Enter position\n");
            scanf("%d %d", &pos, &value);
            (*arrayInsert)(a, pos, value);
        } else if(command == 3) {
            printf("Enter position\n");
            scanf("%d", &pos);
            printf("%d\n", (*arrayGet)(a, pos));
        } else if(command == 4) {
            printf("Enter position\n");
            scanf("%d", &pos);
```

```
            (*arrayDelete)(a, pos);
        } else if(command == 5) {
            printf("Enter value\n");
            scanf("%d", &value);
            (*arrayResize)(a, value);
        } else if(command == 6) {
            (*arrayPrint)(a);
        }
    }
    (*arrayDestroy)(a);
    return 0;
}
```

main_static.c

```
#include <stdio.h>

#include "arrayAPI.h"

int main() {
    int command = 0;
    array* a;
    int value = 0;
    int pos = 0;
    printf("1 - create array with given size\n");
    printf("2 - insert element to array at given position\n");
    printf("3 - get element value on given position\n");
    printf("4 - delete element from given position\n");
    printf("5 - resize array to given size\n");
    printf("6 - print array\n");
    printf("0 - exit\n");
    while(scanf("%d", &command) && command) {
        if(command == 1) {
            printf("Enter value\n");
            scanf("%d", &value);
            a = arrayCreate(value);
        } else if(command == 2) {
            printf("Enter position and value\n");
            scanf("%d %d", &pos, &value);
            arrayInsert(a, pos, value);
        } else if(command == 3) {
            printf("Enter position\n");
            scanf("%d", &pos);
            printf("%d\n", arrayGet(a, pos));
        } else if(command == 4) {
            printf("Enter position\n");
            scanf("%d", &pos);
            arrayDelete(a, pos);
        } else if(command == 5) {
            printf("Enter value\n");
            scanf("%d", &value);
            arrayResize(a, value);
        } else if(command == 6) {
            arrayPrint(a);
        }
    }
    arrayDestroy(a);
    return 0;
}
```

arrayAPI.h

```h
#ifndef _ARRAY_API_H
#define _ARRAY_API_H

#include <stdlib.h>

typedef struct array array;

struct array{
    int* data;
    size_t size;
};

array* arrayCreate(size_t size);
void arrayInsert(array* a, size_t pos, int val);
int arrayGet(array* a, size_t pos);
void arrayDelete(array* a, size_t pos);
void arrayResize(array* a, size_t size);
void arrayDestroy(array* a);
void arrayPrint(array* a);
#endif // _ARRAY_API_H
```

arrayAPI.h

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

#include "arrayAPI.h"

array* arrayCreate(size_t size) {
    if(size <= 0) {
        array* a = (array*)malloc(sizeof(array));
        a->data = (int*)malloc(sizeof(int));
        a->size = 1;
    }
    array* a = (array*)malloc(sizeof(array));
    a->data = (int*)malloc(sizeof(int) * size);
    a->size = size;
    return a;
}

void arrayInsert(array* a, size_t pos, int val) {
    if(pos >= a->size) {
        printf("Out of bounds\n");
        exit(-1);
    }
    a->data[pos] = val;
}

int arrayGet(array* a, size_t pos) {
    if(pos >= a->size) {
        printf("Out of bounds\n");
        exit(-1);
    }
    return a->data[pos];
}

void arrayDelete(array* a, size_t pos) {
    if(pos >= a->size) {
        printf("Out of bounds\n");
        exit(-1);
```

```
38      }
39      int* newData = (int*)malloc(sizeof(int) * (a->size - 1));
40      for(unsigned i = 0; i < a->size; ++i) {
41          if(i < pos) {
42              newData[i] = a->data[i];
43          } else if(i > pos) {
44              newData[i - 1] = a->data[i];
45          }
46      }
47      free(a->data);
48      a->data = newData;
49  }
50
51  void arrayResize(array* a, size_t size) {
52      int* newData = (int*)malloc(sizeof(int) * size);
53      if(size <= a->size) {
54          for(unsigned i = 0; i < size; ++i) {
55              newData[i]  = a->data[i];
56          }
57      } else {
58          for(unsigned i = 0; i < a->size; ++i) {
59              newData[i] = a->data[i];
60          }
61      }
62      free(a->data);
63      a->data = newData;
64      a->size = size;
65  }
66
67  void arrayDestroy(array* a) {
68      free(a->data);
69      a->size = 0;
70      free(a);
71  }
72
73  void arrayPrint(array* a) {
74      for(unsigned i = 0; i < a->size; ++i) {
75          printf("%d ", a->data[i]);
76      }
77      printf("\n");
78  }
```

4 Консоль

```
qelderdelta@qelderdelta-UX331UA:~/Study/OS/os_lab_5/src$ ./dynamic
1 - create array with given size
2 - insert element to array at given position
3 - get element value on given position
4 - delete element from given position
5 - resize array to given size
6 - print array
0 - exit
1
Enter value
3
2
Enter position
0 1
2
Enter position
1 2
2
Enter position
2 3
6
1 2 3
3
Enter position
0
1
0
```

5 Выводы

В ходе выполнения лабораторной работы я познакомился с такой полезной вещью, как использование динамических библиотек, которые позволяют ускорить работу со сторонними библиотеками, поскольку позволяют включать в программу только необходимые функции, а не все, реализованные в библиотеке.