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

Лабораторная работа №5 по курсу
«Операционные системы»

Создание и использование динамических библиотек

Студент: Артемьев Дмитрий Иванович
Группа: М8О-206Б-18
Вариант: 23
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2019

Условие

Требуется создать динамическую библиотеку, которая реализует определенный функционал.

Далее использовать данную библиотеку 2-мя способами:

1. Во время компиляции (на этапе «линковки»/linking)

2. Во время исполнения программы, подгрузив библиотеку в память с помощью системных вызовов

В конечном итоге, программа должна состоять из следующих частей:

• Динамическая библиотека, реализующая заданных вариантом интерфейс;

• Тестовая программа, которая используют библиотеку, используя знания полученные на этапе компиляции;

• Тестовая программа, которая использует библиотеку, используя только местоположение динамической библиотеки и ее интерфейс.

Провести анализ между обоими типами использования библиотеки.

3. Работа с деком, целочисленный 32-битный.

Описание программы

Код программы находится в файлах before_main.c, at_time_main.c, deque.c, deque.h.

Анализ типов динамических библиотек

Динамические библиотеки, хоть и замедляют загрузку программы, обладают существенными преимуществами перед статическими: нет необходимости копировать библиотеку для каждой отдельной программы, также в одном варианте возможно применять изменения библиотеки в уже запущенной программе. Существует два типа динамических библиотек. Первый из них предполагает линковку во время компиляции. При этом становится невозможно применять изменения, происходящие в библиотеке для запущенной программы. Эту проблему решает второй тип: библиотека, подгружаемая программой во время исполнения с помощью системных вызовов. Таким образом использование динамических библиотек, подключаемых на этапе выполнения программы является более гибким решением. Однако при этом нужно больше задумываться о безопасности.

Выводы

Я научился создавать и использовать динамические библиотеки в операционной системе Linux.

Исходный код

before_main.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdint.h>
#include "deque.h"

extern void CreateDeque(Deque* dq);
extern void DeleteDeque(Deque* dq);
extern void push_back(Deque* dq, int32_t val);
extern void push_front(Deque* dq, int32_t val);
extern void pop_back(Deque* dq);
extern void pop_front(Deque* dq);
extern int32_t peek_back(Deque* dq);
extern int32_t peek_front(Deque* dq);

int main() {
    Deque dq;
    CreateDeque(&dq);
    while (1) {
        char cmd[256];
        scanf("%s", cmd);
        if (!strcmp(cmd, "push_back")) {
            int32_t val;
            scanf("%d", &val);
            push_back(&dq, val);
            printf("OK\n");
        }
        else if (!strcmp(cmd, "push_front")) {
            int32_t val;
            scanf("%d", &val);
            push_front(&dq, val);
            printf("OK\n");
        }
        else if (!strcmp(cmd, "pop_back")) {
            pop_back(&dq);
            printf("OK\n");
        }
        else if (!strcmp(cmd, "pop_front")) {
            pop_front(&dq);
```

```
            printf("OK\n");
        }
        else if (!strcmp(cmd, "peek_back")) {
            printf("Back element: %d\n", peek_back(&dq));
        }
        else if (!strcmp(cmd, "peek_front")) {
            printf("Front element: %d\n", peek_front(&dq));
        }
        else if (!strcmp(cmd, "quit") ||
                 !strcmp(cmd, "exit") ||
                 !strcmp(cmd, "q") ||
                 !strcmp(cmd, "C-x_C-c")) {
            break;
        }
        else {
            printf("Wrong command!\n");
        }
    }
    DeleteDeque(&dq);
    return 0;
}
```

at_time_main.c

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdint.h>
#include <dlfcn.h>
#include "deque.h"

int main() {
    void *library_handler;
    library_handler = dlopen("libdeque.so",RTLD_LAZY);
    if (!library_handler){
        fprintf(stderr, "dlopen() error: %s\n", dlerror());
        exit(1);
    }

    void (*CreateDeque)(Deque* dq);
    void (*DeleteDeque)(Deque* dq);
    void (*push_back)(Deque* dq, int32_t val);
    void (*push_front)(Deque* dq, int32_t val);
    void (*pop_back)(Deque* dq);
    void (*pop_front)(Deque* dq);
    int32_t (*peek_back)(Deque* dq);
    int32_t (*peek_front)(Deque* dq);

    CreateDeque = dlsym(library_handler, "CreateDeque");
    DeleteDeque = dlsym(library_handler, "DeleteDeque");
    push_back   = dlsym(library_handler, "push_back");
    push_front  = dlsym(library_handler, "push_front");
    pop_back    = dlsym(library_handler, "pop_back");
    pop_front   = dlsym(library_handler, "pop_front");
    peek_back   = dlsym(library_handler, "peek_back");
    peek_front  = dlsym(library_handler, "peek_front");

    Deque dq;
    (*CreateDeque)(&dq);
    while (1) {
        char cmd[256];
        scanf("%s", cmd);
        if (!strcmp(cmd, "push_back")) {
            int32_t val;
            scanf("%d", &val);
```

```
            (*push_back)(&dq, val);
            printf("OK\n");
        }
        else if (!strcmp(cmd, "push_front")) {
            int32_t val;
            scanf("%d", &val);
            (*push_front)(&dq, val);
            printf("OK\n");
        }
        else if (!strcmp(cmd, "pop_back")) {
            (*pop_back)(&dq);
            printf("OK\n");
        }
        else if (!strcmp(cmd, "pop_front")) {
            (*pop_front)(&dq);
            printf("OK\n");
        }
        else if (!strcmp(cmd, "peek_back")) {
            printf("Back element: %d\n", (*peek_back)(&dq));
        }
        else if (!strcmp(cmd, "peek_front")) {
            printf("Front element: %d\n", (*peek_front)(&dq));
        }
        else if (!strcmp(cmd, "quit") ||
                 !strcmp(cmd, "exit") ||
                 !strcmp(cmd, "q") ||
                 !strcmp(cmd, "C-x_C-c")) {
            break;
        }
        else {
            printf("Wrong command!\n");
        }
        printf("size = %d\n", dq.size);
    }
    (*DeleteDeque)(&dq);

    dlclose(library_handler);
    return 0;
}
```

deque.c

```c
#include <stdlib.h>
#include "deque.h"

void push_back(Deque* dq, int32_t val);
void push_front(Deque* dq, int32_t val);
void pop_back(Deque* dq);
void pop_front(Deque* dq);
int32_t peek_back(Deque* dq);
int32_t peek_front(Deque* dq);

void CreateDeque(Deque* dq) {
    dq->size = 0;
    dq->front = NULL;
    dq->back = NULL;
}

void DeleteDeque(Deque* dq) {
    while (dq->size > 0) {
        pop_back(dq);
    }
}

void push_back(Deque* dq, int32_t newVal) {
    Node* newNode = malloc(sizeof(Node));
    newNode->val = newVal;
    newNode->next = NULL;
    newNode->prev = dq->back;
    if (dq->back == NULL) {
        dq->front = newNode;
    }
    else {
        dq->back->next = newNode;
    }
    dq->back = newNode;
    ++dq->size;
}

void push_front(Deque* dq, int32_t newVal) {
    Node* newNode = malloc(sizeof(Node));
    newNode->val  = newVal;
    newNode->next = dq->front;
```

```
    newNode->prev = NULL;
    if (dq->front == NULL) {
        dq->back = newNode;
    }
    else {
        dq->front->prev = newNode;
    }
    dq->front = newNode;
    ++dq->size;
}

void pop_back(Deque* dq) {
    Node* delNode = dq->back;

    if (dq->back == dq->front) {
        dq->front = dq->back = NULL;
    }
    else {
        dq->back = dq->back->prev;
    }

    if (delNode != NULL)
        --dq->size;
    free(delNode);
}

void pop_front(Deque* dq) {
    Node* delNode = dq->front;

    if (dq->back == dq->front) {
        dq->back = dq->front = NULL;
    }
    else {
        dq->front = dq->back->next;
    }

    if (delNode != NULL)
        --dq->size;
    free(delNode);
}

int32_t peek_back(Deque* dq) {
```

```
    //if (dq->back == NULL)
    //    return NULL;
    return dq->back->val;
}

int32_t peek_front(Deque* dq) {
    //if (dq->front == NULL)
    //    return NULL;
    return dq->front->val;
}
```

main.h

```
#ifndef DEQUE_H
#define DEQUE_H

#include <stdint.h>

typedef struct Node_s {
    int32_t val;
    struct Node_s* next;
    struct Node_s* prev;
} Node;

typedef struct Deque_s {
    Node* front;
    Node* back;
    size_t size;
} Deque;

#endif //DEQUE_H
```