Лабораторная работа № 5 по курсу : Операционные системы

Выполнил студент группы М8О-206Б-17 МАИ *Новиков Павел Сергеевич.*

Цель работы

Целью является приобретение практических навыков в:

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

Задание

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.

Структура данных, с которой должна обеспечивать работу библиотека:
Работа со списком
Тип данных, используемый структурой:
Строки

Информация

Динамическая библиотека - по своей сути обычный объектный файл, который может присоединяется к другой программе в два этапа. Первый этап, это естественно этап компиляции. На этом этапе линковщик встраивает в программу описания требуемых функций и переменных, которые присутствуют в библиотеке. Сами объектные файлы из библиотеки не присоединяются к программе. Присоединение этих объектных файлов осуществляет системный динамический загрузчик во время запуска программы. Загрузчик проверяет все библиотеки прилинкованные с программе на наличие требуемых объектных файлов, затем загружает их в память и присоединяет их в копии запущенной программы, находящейся в памяти. Это позволяет не забивать память лишними одинаковыми частями программ (наиболее яркий пример – *libc*, с которой пролинкованны практически все исполняемые файлы).

Линковка с динамической библиотекой:

Для использования полученной библиотеки в других программах нужно указать при линковке флаг −f%name%. Важно указать именно название библиотеки, а не её имя. То есть если файл называется “liblist.so”, то название самой библиотеки – “list” и подключать её надо с помощью ключа −llist .

Это происходит потому, что система не знает где ей искать файл с таким именем, она пытается найти её по заранее определённым системным путям (определены в файле “/etc/ld.so.conf”)

Также в дополнение к этим путям можно использовать переменную окружения:

```
export LD_LIBRARY_PATH=.
```

Загрузка библиотеки в память с помощью системных вызовов:

Использовать динамические библиотеки можно не только в начале загрузки, но и в процессе самой работы программы. Программа сама может вызывать любые функции из библиотеки, когда это необходимо. Это обеспечивает библиотека dl, которая позволяет линковать библиотеки “на лету”. Она управляет загрузкой динамических библиотек, вызовом функций из них и выгрузкой после конца работы.

Для использования функций работы с динамическими библиотеками необходимо подключить заголовочный файл: #include <dlfcn.c>

Чтобы можно было извлечь какие-либо функции из библиотеки сначала необходимо их загрузить. Этим занимается функция “dlopen”:

```
void *dlopen (const char *filename, int flag);
```

За загрузку самих функций из библиотеки отвечает функция

```
void *dlsym(void *handle, char *symbol);
```

Для этой функции требуется адрес загруженной библиотеки handle, полученный при открытии функцией dlopen(). Требуемая функция или переменная задается своим именем в переменной symbol.

После окончания работы с библиотекой её следует закрыть:

```
dlclose(void *handle);
```

Исходный код

```
#include <stdio.h>
#include <dlfcn.h>
#include "list.h"

int main(){
        void* library_handler;

        char d_create[] = "d_create";
        char d_pop[] = "d_pop_front";
        char d_push[] = "d_push_back";
        char d_print[] = "d_print";
        char d_destroy[] = "d_destroy";

        List* (*create)(void);
        void (*push)(List* list, char val[256]);
        int (*pop)(List* list);
        void (*print)(List* list);
        void (*destroy)(List* list);

        library_handler = dlopen("/home/mika/lab/os/lab05/liblist.so", RTLD_LAZY);
        if (!library_handler){
                fprintf(stderr,"dlopen() error: %s\n", dlerror());
                exit(1);
        }

        create = dlsym(library_handler, d_create);
        push = dlsym(library_handler, d_push);
        pop = dlsym(library_handler, d_pop);
        print = dlsym(library_handler, d_print);
        destroy = dlsym(library_handler, d_destroy);

        List* list = (*create)();
        (*push)(list,"awdwda");
        (*push)(list,"wad");
        (*push)(list,"351");
        (*pop)(list);
        (*print)(list);
        (*destroy)(list);

        dlclose(library_handler);
}

#include <stdio.h>
#include "list.h"

int main(){
        List* list = d_create();
        d_push_back(list,"bwa");
        d_push_back(list,"rtt");
```

```
        d_push_back(list,"gdb");
        d_pop_front(list);
        d_push_front(list,"abc");
        d_push_front(list,"awd");
        d_pop_back(list);
        d_print(list);
        d_destroy(list);
}
```

Вывод Strace

```
dynlink:
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "/usr/lib/tls/haswell/x86_64/liblist.so", O_RDONLY|O_CLOEXEC
openat(AT_FDCWD, "/usr/lib/tls/haswell/liblist.so", O_RDONLY|O_CLOEXEC) = -1
openat(AT_FDCWD, "/usr/lib/tls/x86_64/liblist.so", O_RDONLY|O_CLOEXEC) = -1 E
openat(AT_FDCWD, "/usr/lib/tls/liblist.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (
openat(AT_FDCWD, "/usr/lib/haswell/x86_64/liblist.so", O_RDONLY|O_CLOEXEC) =
openat(AT_FDCWD, "/usr/lib/haswell/liblist.so", O_RDONLY|O_CLOEXEC) = -1 ENOE
openat(AT_FDCWD, "/usr/lib/x86_64/liblist.so", O_RDONLY|O_CLOEXEC) = -1 ENOEN
openat(AT_FDCWD, "/usr/lib/liblist.so", O_RDONLY|O_CLOEXEC) = -1 ENOENT (No s
./dynlink: error while loading shared libraries: liblist.so: cannot open shar
+++ exited with 127 +++
dynload:
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "/usr/lib/libdl.so.2", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "/usr/lib/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
openat(AT_FDCWD, "/home/mika/lab/os/lab05/liblist.so", O_RDONLY|O_CLOEXEC) =
```

Выводы

Статическая линковка удобна тем, что собирает программу в один файл. После запуска программы, реализация используемых функций ищется в сборке, таким образом, гарантируется переносимость программы. Как результат – сборка несколько увеличивается в размерах. При динамической линковке мы получаем “голую” сборку без сторонних библиотек. Ее размер, безусловно, меньше, но при этом мы должны гарантировать, что на клиентской машине имеется библиотека, используемая в программе, и ее версия одинакова с той, которая была использована при сборке. В обоих способах есть минусы и плюсы, выбор зависит от необходимого результата.