**Лекция 2**

**Пример** с тем, когда либу переместили в другую папку, и хотим указать линковщику, откуда ее брать:

Просто поменять путь с помощью команды – не пойдет, потому что в новой папке hahaha хранится не сама либа, а ссылка на либу. Тогда нужно переместить в эту папку еще и настоящую либу ( *sudo mv ../libstdc++.so.6.0.32 .*), тогда все сработает, если применить команду

*LD\_LIBRARY\_PATH=/usr/lib/x86\_64-linux-gnu/hahaha ./a.out*

*ln -s target file* – команда для создания ссылки.

*LD\_PRELOAD=/ ./a.out* – переменная, которая заставляет линковщика в приоритетном порядке загрузить либы из другого места, и только потом подгружать те либы, которые подгружаются стандартно.

**GLOBAL, LOCAL, WEAK символы**

У символов (функций и переменных) бывают внешние и внутренние связывания. Когда мы объявляем функцию глобально, она по умолчанию обладает внешним связыванием, то есть она видна из других Translation Units.

То есть, если программа состоит из нескольких файлов, и мы компилируем их вместе, то все символы, которые в них объявлены, скидываются в общую кучу. И если в этих файлах будут одинаковые функции, то это будет multiple definition.

Однако, одну из функций можно объявить как static. Static, примененное к глобальным функциям и переменным, означает «сделай эту функцию невидимой извне». То есть такой символ для линковщика будет локальным.

**WEAK символы** – такие символы, которые как бы глобальные, но если встречается такой же символ без пометки weak, то он перезаписывается. Иными словами, во время компоновки сильный символ может переопределять слабый символ с тем же именем. Напротив, при наличии двух сильных символов с одинаковыми именами компоновщик разрешает символ в пользу первого найденного. Такое поведение позволяет исполняемому файлу переопределять стандартные библиотечные функции, такие как malloc или new.

Между стадией компиляции и линковки имена претерпевают преобразование, которое называется **манглированием**. **Деманглинг** – преобразование в обратную сторону.

**c++filt** – утилита, которая проводит деманглинг (переводит имя символа в нормальное плюсовое имя)

**Задача:** хотим написать свою либу! Как?

Пишем для примера какую-нибудь простую программу, которую хотим сделать либой. Нужно так скомпилировать его, чтобы на выходе получился не executable file, a shared object file:

*g++ -shared myfunc.cpp -o libmyfunc.so*

Пусть test.cpp – программа, использующая нашу функцию. Прилинковать либу просто с флагом -lmyfunc не получится, потому что непонятно, где искать либу.

Переместить либу туда, где все либы, сработает: *g++ test.cpp -lmyfunc* компилируется.

Но можно было и без перемещения: *g++ test.cpp -lmyfunc -L.* – это хоть и скомпилируется, но не запустится, потому что динамический линковщик не знает, где искать эту либу.

Решаем проблему так: в сам бинарный файл записать адрес либы, по которому ее надо будет искать. *g++ test.cpp -lmyfunc -L. Wl,-rpath,/home/mesyarik/caos2024* (путь короче абсолютный указать).

Если хотим прилинковать либы статически, это делается с помощью флага **-static**:

*g++ -static test.cpp*

Теперь в файле нет динамических зависимостей, то есть мы как бы вшили стандартные либы в код файла, соответственно и весить он стал больше.

**nm** – перечисляет символы в файле. С флагом -C делает деманглинг.

**Дебаггеры**

**Gdb** – стандартный gnu дебаггер.

Чтобы можно было ходить построчно в коде, нужно скомпилировать код с флагом -g:

*g++ -g test.cpp*

Добавили таким образом «дебажную информацию». Получили формат файла **DWARF**.

Основные команды:

* run – запустить
* stop – остановить
* b test.cpp:6 – поставить брейкпоинт перед 7 строчкой
* n – переход на следующую строчку
* s – шаг с заходом в функцию
* bt – посмотреть backtrace (узнать, в какой именно функции возникла ошибка)

Проблема – не можем зайти таким образом в функции стандартной библиотеки. Цель – научиться это делать.

Для этого нам нужна дебажная версия библиотеки. Это можно установить так:

*sudo apt install libstdc++6-10-dbg*

Теперь эта либа находится в папке /usr/lib/x86\_64-linux-gnu/debug/

Теперь компилим код с этой либой:

*g++ -g test.cpp -Wl,-rpath,/usr/lib/x86\_64-linux-gnu/debug/*

Флаг *-nostdlib* означает буквально «забудь про стандартную либу»

Флаг *-D\_GLIBCXX\_DEBUG* активирует режим отладки в libstdc++. Это позволяет libstdc++ проводить проверки различных операций.