Практикум на ЭВМ. Интерпретатор. Технические моменты

Баев А.Ж.

Казахстанский филиал МГУ

12 апреля 2022

Исходные данные

Цель: собрать бинарный файл `example` из 3 файлов:

1. `example.cpp` - пользовательский код,
2. `interpreter.cpp` и `interpreter.h` - библиотека.

```
src/
    interpreter.cpp
headers/
    interpreter.h
usr/
    example.cpp
bin/
    example
```

Обратите внимание, что в файле example.cpp заголовочный файл подключается следующим образом:

```
1 #include <interpreter.h>
2 ...
3 int main() {
4 }
```

Статическая сборка. Компиляция

Цель: собрать один бинарный файл example из 3 файлов (example.cpp, interpreter.cpp и interpreter.h).

```
src/
    interpreter.cpp
headers/
    interpreter.h
usr/
    example.cpp
bin/
    example
```

Добавляем флаг -I (от слова include) с указанием пути к заголовочным файлам.

```
1 g++ usr/example.cpp src/interpreter.cpp \
2     -I headers/ -o bin/example
```

Статическая сборка. Запуск

Достаточно иметь только один бинарный файл.

```
bin/
    example
```

Можно запускать:

```
1 bin/example
```

Динамическая сборка. Прекомпиляция

Цель:

1. собрать объектный файл `libinterpreter.so` из 2 файлов (`interpreter.cpp` и `interpreter.h`).

```
src/
    interpreter.cpp
headers/
    interpreter.h
lib/
    libinterpreter.so
usr/
    example.cpp
bin/
```

Добавляем флаги для того, чтобы получить модуль с внешним доступом (`-shared` и `-fpic`). Имя выходного файла начинается с приставки `lib`.

```
1 g++ src/interpreter.cpp -I headers/ \
2     -fpic -shared -o lib/libinterpreter.so
```

Динамическая сборка. Прекомпиляция

Теперь для компилирования пользовательских приложений достаточно иметь только следующие 2 файла:

```
headers/
    interpreter.h
lib/
    libinterpreter.so
```

То есть файл `interpreter.cpp` для дальнейшей работы не нужен.

Динамическая сборка. Компиляция

Цель:

2 собрать из example.cpp второй объектный файл example, который будет слинкован с libinterpreter.so.

```
headers/
    interpreter.h
lib/
    libinterpreter.so
usr/
    example.cpp
bin/
    example
```

Добавляем флаг -I (от слова include) с указанием пути к заголовочным файлам и флаг -L (от слова library) с указанием пути к объектным файлам *so. Также линкуем библиотеку флагом -linterpreter (не -lib а -l).

```
1 g++ usr/example.cpp -I headers/ -L lib/ \
2     -linterpreter -o bin/example
```

Динамическая сборка. Компиляция

Теперь для запуска пользовательских приложений достаточно иметь только следующие 2 файла:

```
lib/
    libinterpreter.so
bin/
    example
```

Причем первый файл может быть переиспользован несколькими пользовательскими программами.

Динамическая сборка. Запуск

Цель:

3 запустить example.

```
lib/
    libinterpreter.so
bin/
    example
```

Добавляем переменную окружения с путём до библиотеки .

```
1 export LD_LIBRARY_PATH=$pwd/lib/
2 bin/example
```

Пример с реальной библиотекой

Похожим образом собрана библиотека для математики на C, известная по флагу `-lm`.

```
/usr/include/
    math.h
/usr/lib32/
    libm.so => /lib32/libm.so.6
/lib32/
    libm.so.6
```

А посмотреть много интересного (где какие библиотеки есть в системе) можно командой

```
1 /sbin/ldconfig -p
2 /sbin/ldconfig -p | grep libm.so
```

Сборка пакета .deb под Ubuntu/Debian

https://ubuntuforums.org/showthread.php?t=910717