Search notes:

Creating a shared and static library with the gnu compiler (gcc)

2018-01-27: *Moved and adapted from* `adp-gmbh.ch`.

Here's a summary on how to create a shared and a static library with gcc. The goal is to show the basic steps. I do not want to go into the hairy details. It should be possible to use this page as a reference.

These examples were tested and run on Linux

Update *2018-12-08*: Alberto Fanjul (albfan) has contributed the necessary files to demonstrate the building of the libraries with the Meson build system.

The tq84-library sources

The library we're going to build is called *tq84*. The sources for this library are located under `./src/tq84`.
The library consists of two c-files: `add.c` and `answer.c`.

`add.c` exhibits two functions: `setSummand()` and `add()`.
`add()` returns the passed value and adds it to the value that was set with `setSummand()`.

`answer.c` has only one function: `answer()` which uses `setSummand(20)` and `add(22)` to produce the number `42` which it returns as answer.

Additionally, the library (`add.c`) has also the functions `initLibrary` and `exitLibrary` to demonstrate the influence of `__attribute__ ((constructor))` and `__attribute__ ((destructor))`. These functions are called when the library is loaded and unloaded.

add.c

The code for the library:

```
#include <stdio.h>

int gSummand;


void setSummand(int summand) {
  gSummand = summand;
}

int add(int summand) {
  return gSummand + summand;
}

void __attribute__ ((constructor)) initLibrary(void) {
 //
 // Function that is called when the library is loaded
 //
    printf("Library is initialized\n");
    gSummand = 0;
}
void __attribute__ ((destructor)) cleanUpLibrary(void) {
 //
 // Function that is called when the library is »closed«.
 //
```

```
    printf("Library is exited\n");
}
```

Github repository gcc-create-library, path: /src/tq84/add.c

add.h

The header file for the library:

```
void setSummand(int summand);
int  add(int summand);
```

Github repository gcc-create-library, path: /src/tq84/add.h

answer.c

```
#include "add.h"

int answer() {

  setSummand(20);
  return add(22);  // Will return 42 (=20+22)

}
```

Github repository gcc-create-library, path: /src/tq84/answer.c

answer.h

The header file for answer.c

```
int answer();
```

Github repository gcc-create-library, path: /src/tq84/answer.h

main.c

`./src/main.c` is the source code that uses the tq84-library

```
#include <stdio.h>
#include "tq84/add.h"
#include "tq84/answer.h"

int main(int argc, char* argv[]) {

  setSummand(5);

  printf("5 + 7 = %d\n", add(7));

  printf("And the answer is: %d\n", answer());

  return 0;
}
```

Github repository gcc-create-library, path: /src/main.c

Create the object files

First, we create the object files.

Object files for the shared library need to be compiled with the `-fPIC` flag (PIC = position independent code).
The object files for the static library don't need this flag.

So, the created object files go into different directories: `bin/shared` and `/bin/static`

```
gcc -c       src/main.c        -o bin/main.o

#
# Create the object files for the static library (without -fPIC)
#
gcc -c       src/tq84/add.c    -o bin/static/add.o
gcc -c       src/tq84/answer.c -o bin/static/answer.o

#
# object files for shared libraries need to be compiled as position independent
# code (-fPIC) because they are mapped to any position in the address space.
#
gcc -c -fPIC src/tq84/add.c    -o bin/shared/add.o
gcc -c -fPIC src/tq84/answer.c -o bin/shared/answer.o
```

Github repository gcc-create-library, path: /steps/create-object-files

Create static library

A static library is basically a set of object files that were copied into a single file with the suffix `.a`.

The static file is created with the archiver (`ar`).

```
ar rcs bin/static/libtq84.a bin/static/add.o bin/static/answer.o
```

Github repository gcc-create-library, path: /steps/create-static-library

Link statically

With the static library, we can statically link `main.o` with the library.

The `-L` flag indicates (a non standard) directory where the libraries can be found.

The `-l` flag indicates the name of the library. Note, that it assumes the library to start with `lib` and end with `.o` (so `lib` and `.o` must not be specified)

```
gcc   bin/main.o  -Lbin/static -ltq84 -o bin/statically-linked
```

Github repository gcc-create-library, path: /steps/link-statically

The created executable `bin/statically-linked` is not dependent on any other object file or library. It can be distributed without the `.a` file or the `.o` files. It can be executed on the shell like so:

```
$ ./bin/statically-linked
```

Create the shared library

Here, we create a shared library without SONAME.

A shared library is created with GCC's `-shared` flag and naming the resultant file with the suffix `.so` rather than `.a`.

```
gcc -shared bin/shared/add.o bin/shared/answer.o -o bin/shared/libtq84.so

#
#  In order to create a shared library, position independent code
#  must be generated. This can be achieved with `-fPIC` flag when
#  c-files are compiled.
#
```

```
#  If the object files are created without -fPIC (such as when the static object files are produc
#      gcc -shared bin/static/add.o bin/static/answer.o -o bin/shared/libtq84.so
#  produces this error:
#     /usr/bin/ld: bin/tq84.o: relocation R_X86_64_PC32 against symbol `gSummand' can not be used
#
```

Github repository gcc-create-library, path: /steps/create-shared-library

Link dynamically with the shared library

Note the similarity to linking with the static library. Since we specify the shared directory `bin/shared` with the `-L` flag, the linker links with `libtq84.so`.

```
# Note the order:
#   -ltq84-shared needs to be placed AFTER main.c

gcc  bin/main.o -Lbin/shared -ltq84 -o bin/use-shared-library
```

Github repository gcc-create-library, path: /steps/link-dynamically

Use the shared library with LD_LIBRARY_PATH

As long as the shared library is not installed in a default location (such as /usr/lib), we must indicate where it is found. This is possible with the `LD_LIBRARY_PATH` environment variable.

```
#  If the shared object is in a non standard location, we
#  need to tell where it is via the LD_LIBRARY_PATH
#  environment variable
#
# ./use-shared-object
#    ./use-shared-object: error while loading shared libraries: libtq84.so: cannot open shared ob

LD_LIBRARY_PATH=$(pwd)/bin/shared bin/use-shared-library
```

Github repository gcc-create-library, path: /steps/use-shared-library-LD_LIBRARY_PATH

Move the shared library to a default location

Let's move the shared library to `/usr/lib` so that we can execute `bin/use-shared-library` without explicitly setting the `LD_LIBRARY_PATH` variable.

```
#
#   Moving the shared object to a standard location
#
sudo mv bin/shared/libtq84.so /usr/lib
sudo chmod 755 /usr/lib/libtq84.so
```

Github repository gcc-create-library, path: /steps/move-shared-object

Use the shared library in a default location

Now, with the library in `/usr/lib`, the executable can be run without setting the variable:

```
#
#   Since the shared library was copied to a standard
#   location (/usr/lib), the environment variable LD_LIBRARY_PATH
#   does not need to be set to execute the executable:

bin/use-shared-library
```

Github repository gcc-create-library, path: /steps/use-shared-library-no-LD_LIBRARY_PATH

Use dlopen() etc to dynamically load the library

It's also possible to dynamically load a library from an executable. The necessary functions are `dlopen()`, `dlsym()` etc. whose definitions are found in `dlfcn.h`.

The following program tries to open the library `libtq84.so` and then to find the functions `doesNotExist`, `setSummand` and `add`.

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

// Note, add.h and answer.h
// need not be #include'd


void* getFunctionPointer(void* lib, const char* funcName) {
 //
 // Get the function pointer to the function
    void* fptr = dlsym(lib, funcName);
    if (!fptr) {
      fprintf(stderr, "Could not get function pointer for %s\n  error is: %s\n\n", funcName, dler
      return NULL;
    }
    return fptr;
}

int main(int argc, char* argv[]) {

 //
 // Declare the function pointers:
#pragma GCC diagnostic push
#pragma GCC diagnostic ignored "-Wunused-but-set-variable"
    void (*fptr_null      )(int);
#pragma GCC diagnostic pop
    void (*fptr_setSummand)(int);
    int  (*fptr_add       )(int);


 //
 // Open the dynamic library
    void* tq84_lib = dlopen("libtq84.so",  RTLD_LAZY | RTLD_GLOBAL);

    if (!tq84_lib) {
     //
     // Apparently, the library could not be opened
        fprintf(stderr, "Could not open libtq84.so\n");
        exit(1);
    }



 //
 // Get the pointers to the functions within the library:
 //
 //     Function doesNotExist does not exist, demonstrate
 //     calling dlerror()
 //
    fptr_null      =getFunctionPointer(tq84_lib, "doesNotExist");
    fptr_setSummand=getFunctionPointer(tq84_lib, "setSummand");
    fptr_add       =getFunctionPointer(tq84_lib, "add"       );

 //
```

```
  // Call the function via the function pointer
  //
     fptr_setSummand(42);

     int result = fptr_add(7);

     printf("Result: %d\n", result);
}
```

Github repository gcc-create-library, path: /src/dynamic-library-loader.c

In order to use the `dl…()` functions, it must be linked against `libdl` (with the `-ldl` flag)

```
#
# Using dl functions
#
gcc src/dynamic-library-loader.c -ldl -o bin/dynamic-library-loader

#
#  As long as /usr/lib/libtq84.so exists, LD_LIBRARY_PATH
#  needs not be set
#
bin/dynamic-library-loader
```

Github repository gcc-create-library, path: /steps/create-dynamic-library-loader

Create a SONAME shared library

A shared library with a SONAME can be created with the `-soname` linker option.

```
gcc -shared  bin/shared/add.o bin/shared/answer.o -Wl,-soname,libtq84Soname.so.1 -o bin/shared/li
ln -s libtq84Soname.so.1.0.1 bin/shared/libtq84Soname.so
```

Github repository gcc-create-library, path: /steps/create-soname-library

Link with the SONAME shared library

```
gcc -Lbin/shared  bin/main.o -ltq84Soname -o bin/use-shared-library-soname

# Alterntatively
# gcc -Lbin  src/main.c -ltq84Soname -o bin/use-shared-library-soname
```

Github repository gcc-create-library, path: /steps/link-soname-library

Install the SONAME library

```
sudo cp bin/shared/libtq84Soname.so.1.0.1 /usr/lib
sudo chmod 755 /usr/lib/libtq84Soname.so.1.0.1

sudo ldconfig -v -n /usr/lib | grep tq84

# lsconfig basically created the symbolic link
# from /usr/lib/libtq84Soname.so.1 to /usr/lib/libtq84Soname.so.1.0.1:
ls -ltr /usr/lib | tail -10
```

Github repository gcc-create-library, path: /steps/install-soname-library

Using LD_DEBUG

The `LD_DEBUG` environment variable might be helpful for some debugging tasks related to shared libraries.

```
#
#  Use LD_DEBUG
#    set it to libs to display library search paths
#
LD_DEBUG=libs bin/use-shared-library

#
#  Setting LD_DEBUG to files to display progress for input files
#
LD_DEBUG=files bin/use-shared-library

#
#  Setting LD_DEBUG to reloc to display relocation processing
#
LD_DEBUG=reloc bin/use-shared-library

LD_DEBUG=symbols bin/use-shared-library
```

Github repository gcc-create-library, path: /steps/LD_DEBUG

Difference between -fPIC and without -fPIC

```
objdump --disassemble bin/shared/add.o | sed -n '/<add>/,/^$/p'
objdump --disassemble bin/static/add.o | sed -n '/<add>/,/^$/p'
```

Github repository gcc-create-library, path: /steps/show-difference-PIC

readelf reloc

```
#
#  Similar to objdump but more detailed:
#
readelf --relocs bin/shared/add.o
readelf --relocs bin/static/add.o
```

Github repository gcc-create-library, path: /steps/readelf-relocs

List symbols in object files

```
#
#  TODO
#
#  List symbols in object files
#
nm bin/static/add.o
nm bin/shared/libtq84Soname.so
nm bin/statically-linked
nm bin/dynamic-library-loader
```

Github repository gcc-create-library, path: /steps/list-symbols

Cleaning up

```
sudo rm /usr/lib/libtq84*
rm   bin/*.o
rm   bin/static/*.o
rm   bin/shared/*.o
rm   bin/statically-linked
rm   bin/use-shared-library
```

Github repository gcc-create-library, path: /steps/cleanup

Thanks

Thanks to *Donn Morrison* and *Rob Snell* who both helped me improve this page.

TODO

Linker option `-soname` (`-Wl,-soname,libtq84.so.1`)

Stripping shared libraries with `--strip-unneeded`

```
objdump -p $EXECUTABLE | grep NEEDED
```

See also

/etc/ld.so.cache
/etc/ld.so.conf
/etc/ld.so.conf.d

`ldd`

Links

https://www.cprogramming.com/tutorial/shared-libraries-linux-gcc.html

https://cygwin.com/cygwin-ug-net/dll.html

Index