A tale of ELFs and DWARFs

A glimpse into the world of linkers, loaders and binary formats

Volker Krause
v krause@kde.org
@VolkerKrause
Our Workflow

- Write code
- Run compiler
- ...
- Run application
- Profit!
Why care?

- Understanding linker errors
- Debugging weird runtime behavior
- Performance
- Surviving without CMake convenience
The Magic

- **Build time**
  - combine multiple object files into one library or executable

- **Run time**
  - find all needed libraries
  - map them into memory
  - resolve dynamic symbols
  - call the entry point functions
Linker Errors

- undefined reference to `Class::method()'
- `-Wl,--fatal-warnings -Wl,--no-undefined`
- Catch missing symbols at compile time

Causes

- Method is declared but not defined
- Signature mismatch
- Using a method that is not exported
Looking Into Libraries

- Binutils: nm, readelf, objdump
- Mac: otool
- Windows: Dependency Walker

0000000000003a344 T _ZNK4KJob11errorStringEv
00000000000039fc8 T _ZNK4KJob11isSuspendedEv
Name Mangling

- For C: just using the method name
  - extern "C" {...}
- C++: overloading, templates, namespaces, ...
- Information included:
  - argument types, cv-qualifiers
  - library name (Mach-O two-level lookup)
  - public/protected/private (MSVC)
- Demangle: `c++filt, nm -C`
Export Macros

- `–fvisibility=hidden`
- UNIX
  - `__attribute__((visibility("default")))`
- Windows
  - `__declspec(dllexport)`
  - `__declspec(dllimport)`
- Upper-case letters in `nm` output indicate exported symbols
File Formats

- ELF (Linux), Mach-O (Mac), PE (Windows)
- Simple, designed for zero-copy mapping into memory
- Essentially the same for libraries and executables
- Consist of multiple sections, custom sections possible
  - Symbol table
  - Text (executable code)
  - Data (string literals, QRC, ...)

Qt5 Plug-ins

- Implementation hidden by moc
- Plug-in meta-data in custom section
  - Contains a QJsonObject in binary format: qt_pluginMetaData
  - __attribute__((section(".qtmetadata")))
  - readelf --sections plugin.so
- Plug-in entry symbol (must be exported!)
  - nm -C plugin.so | grep qt_plugin_instance
• DWARF
• Custom section or separate file
• -g[0-3] decides level of detail
• Index for faster loading in GDB:

```bash
(gdb) save gdb-index <lib>
objcopia \
   --add-section .gdb_index=<lib>.gdb-index \
   --set-section-flags .gdb_index=readonly \
   <lib> <lib>
```

• With gold: --gdb-index
Finding Libraries

- Search paths:
  - RPATH if RUNPATH not set (`-rpath, chrpath`)
  - LD_LIBRARY_PATH
  - RUNPATH (`--enable-new-dtags`)
  - `/etc/ld.so*`

- Diagnostics:
  - Static: `ldd, readelf -d <elf>`
  - Dynamic: `LD_DEBUG=libs, strace`
Does size matter?

- mmap'ed in 3 sections:
  - Read-only
  - Read-only + executable
  - Read-write
- Read-only sections are shared
- Writable sections use copy-on-write
Writable Data

- Keep small, this cannot be shared
  
  ```c
  const char *c1 = "...";
  const char c2[] = "...";
  static const char *c3 = "...";
  char* c4;
  ```

- Verify with `nm`: "r" vs. "d" or "b"
  
  ```
  0000000000601040  D  c1
  0000000000400633  r  c2
  0000000000601048  d  c3
  0000000000601060  B  c4
  ```
const char *data = "hello world";  
int main(int, char**) {  
    const_cast<char*>(data)[0] = 'H';  
}  

- Valgrind:

Process terminating with default action of signal 11 (SIGSEGV)  
Bad permissions for mapped region at address 0x400684  
at 0x4005F0: main (main.cpp:3)
Relocation

- Procedure Linkage Table (PLT)
- Stubs for calling external functions
- Global Offset Table (GOT)
  - Filled during dynamic linking
  - Contains addresses of external symbols
- Calling another function:
  - Intra-library: relative jump (− Bs\text{ymbolic})
  - Inter-library: PLT stub + GOT
Qt5 PMF Connections

- Function pointer `connect()`
  - Compares pointer to signal with `moc` data
  - Must avoid comparison between PLT stub and actual function...
- Addressed by `-fPIC/-fPIE`
  - Safety check in `qglobal.h` enforces that
  - On ARM: possibly also `-pie` due to toolchain bug
● CMake hides most of this from us
● Nasty to debug if something goes wrong
● To learn more, read Ulrich Drepper's “How to write shared libraries”
Questions?
  http://www.iecc.com/linker/

  http://www.akkadia.org/drepper/dsohowto.pdf

● Slides & code: