Contribution ID: **186** Type: **not specified**

Indirect External Access

Tuesday, 21 September 2021 08:30 (30 minutes)

On systems with copy relocation:
- A copy in executable is created for the definition in a shared library at run-time by ld.so.
- The copy is referenced by executable and shared libraries.
- Executable can access the copy directly.

Issues are:
- Overhead of a copy, time and space, may be visible at run-time.
- Read-only data in the shared library becomes read-write copy in executable at run-time.
- Local access to data with the STV_PROTECTED visibility in the shared library must use GOT.

On systems without function descriptor, function pointers vary depending on where and how the functions are defined.

- If the function is defined in executable, it can be the address of function body.
- If the function, including the function with STV_PROTECTED visibility, is defined in the shared library, it can be the address of the PLT entry in executable or shared library.

Issues are:
- The address of function body may not be used as its function pointer.
- ld.so needs to search loaded shared libraries for the function pointer of the function with STV_PROTECTED visibility.

Here is a proposal to remove copy relocation and use canonical function pointer:
1. Accesses, including in PIE and non-PIE, to undefined symbols must use GOT.
2. Read-only data in the shared library remain read-only at run-time
3. Address of global data with the STV_PROTECTED visibility in the shared library is the address of data body.
4. For systems without function descriptor,
- All global function pointers of undefined functions in PIE and non-PIE must use GOT.
- Function pointer of functions with the STV_PROTECTED visibility in executable and shared library is the address of function body.

I agree to abide by the anti-harassment policy

I agree

Primary author: LU, H.J. (Intel)

Presenter: LU, H.J. (Intel)

Session Classification: GNU Tools Track

Track Classification: GNU Tools Track