Fuchsia

Introduction to LLVM-IFS and its usages in Fuchsia build system

Haowei Wu (haowei@google.com)

Agenda

- What is LLVM-IFS
- Motivation
- Comparison with similar tools
- LLVM-IFS integration in Fuchsia
- Future work

LLVM-IFS

- IFS is InterFace Stub
- Generate human readable text representation of an ELF shared objects (IFS file)
- Generate linkable ELF shared object stub from either ELF shared object or text-based IFS file
- Generate IFS file from source header files

Motivation

- Shared object stubbing is useful
  - Generate smaller link only shared objects stubs
  - Smaller SDKs
- Link time ABI changes are annoying
  - Hard to spot in code review.
  - Causing build breakages that hard to be traced back to affecting changes.

Regular ELF Shared Object

```
$ llvm-objdump --section-headers ./libicui18n.so

./libicui18n.so:     file format elf64-x86-64

Sections:
Idx Name               Size     VMA              Type
  0                    00000000 0000000000000000
  1 .note.gnu.build-id 00000018 0000000000000270
  2 .dynsym            000344e8 0000000000000288
  3 .gnu.hash          0000e09c 0000000000034770
  4 .dynamic           00000160 0000000000042810
  5 .dynstr            00073cbd 0000000000042970
  6 .rela.dyn          0000a1d0 00000000000b6630
  7 .relr.dyn          00000170 00000000000c0800
  8 .rela.plt          000213c0 00000000000c0970
  9 .rodata            0001cb6c 00000000000e1d30 DATA
 10 .eh_frame_hdr      0000fd54 00000000000fe89c DATA
 11 .eh_frame          0003f6a4 000000000010e5f0 DATA
 12 .text              0023d48b 000000000014e000 TEXT
 13 .plt               00016290 000000000038b490 TEXT
 14 .data.rel.ro       00005e98 00000000003a2000 DATA
 15 .got               00000728 00000000003a7e98 DATA
 16 .got.plt           0000b158 00000000003a85c0 DATA
 17 .data              000004e8 00000000003b4000 DATA
 18 .bss               00000db8 00000000003b44f0 BSS
 19 .shstrtab          000000ab 0000000000000000
```

Size: 3794.6KiB

ELF Shared Object Stub

```
$ llvm-ifs --output-format=ELF --output=./libicui18n.so.stub ./libicui18n.so
$ llvm-objdump --section-headers ./libicui18n.so.stub

./libicui18n.so.stub:     file format elf64-x86-64

Sections:
Idx Name                 Size     VMA              Type
  0                      00000000 0000000000000000
  1 .note.gu.build-id    00000018 0000000000000270
  2 .dynsym              000344e8 0000000000000040
  3 .gnu.hash            0000e09c 0000000000034770
  4 .dynamic             00000070 00000000000a8168
  5 .dynstr              00073c3c 0000000000034528
  6 .rela.dyn            0000a1d0 0000000000b6630
  7 .relr.dyn            00000170 0000000000c0800
  8 .rela.plt            000213c0 0000000000c0970
  9 .rodata              0001cb6c 0000000000e1d30 DATA
 10 .eh_frame_hdr        0000fd54 0000000000fe89c DATA
 11 .eh_frame            0003f6a4 00000000010e5f0 DATA
 12 .text                0023d48b 000000000014e000 TEXT
 13 .plt                 00016290 000000000038b490 TEXT
 14 .data.rel.ro         00005e98 0000000003a2000 DATA
 15 .got                 00000728 0000000003a7e98 DATA
 16 .got.plt             0000b158 0000000003a85c0 DATA
 17 .data                000004e8 0000000003b4000 DATA
 18 .bss                 00000db8 0000000003b44f0 BSS
 19 .shstrtab            00000024 00000000000a81d8
```

Size: 672.8KiB
(209 KiB dynsym + 463 KiB dynstr)

ELF Shared Object Text based IFS

```
llvm-ifs --output-format=IFS --output=./libicui18n.so.ifs ./libicui18n.so

--- !ifs-v1
IfsVersion:      3.0
SoName:          libicui18n.so
Target:          { ObjectFormat: ELF, Arch: x86_64, Endianness: little, BitWidth: 64 }
NeededLibs:
  - libicuuc.so
  - libc.so
  - 'libc++abi.so.1'
Symbols:
  - { Name: T_CString_toLowerCase_69, Type: Func, Undefined: true }
  - { Name: _Z19uprv_decNumberClassPK9decNumberP10decContext, Type: Func }
  - { Name: _ZN6icu_6910BucketListD0Ev, Type: Func }
  - { Name: _ZN6icu_6910BucketListD1Ev, Type: Func }
  - { Name: _ZN6icu_6910BucketListD2Ev, Type: Func }
...
```

Comparison with similar tools

- Microsoft's Import Libraries
  - Stub code generated from compiler and linker
  - Import library files are not human readable
- Apple TAPI (Text API)
  - Text stub generation from header
  - Can be directly used by linker
- LLVM-IFS
  - Binary stub generation from header and shared objects
  - Binary stub can be directly used by linker
  - Text stub is human readable

LLVM-ELFABI

- Generate Text based ABI file from ELF shared object.
- Generate linkable ELF shared object stub from Text based ABI file or regular ELF shared object.

LLVM-IFS

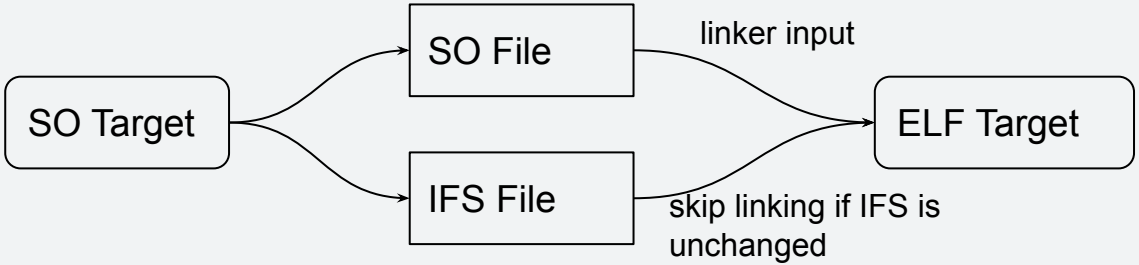
- Generate Text based IFS file from source code.
- Generate linkable ELF shared object stub from IFS file.

The current version of LLVM-IFS is merged from these two tools.

Integration with Fuchsia build system.

Incremental build optimization

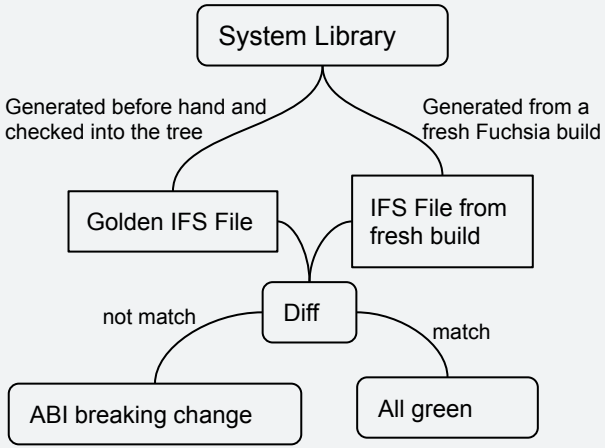
- Each shared object target will generate regular shared object and text IFS file.
- The time stamp of the text IFS will not be updated if its content is not changed (with "--write-if-changed" flag).
- If source code of depended shared object is changed but text IFS file is not changed, build system will not attempt relinking depending targets.



Integration with Fuchsia build system.

ABI breaking change detection

- For each crucial system shared library target, we generated an IFS file and check it in into Fuchsia's source tree as golden IFS file
- When Fuchsia is being build, each IFS file will be generated from fresh built system shared library
- Each generated IFS file will be compared against the golden IFS file from source tree
- If there is a difference, build system will report an error as the change breaks the ABI and require the developer to regenerate the IFS file
- Developers upload local changes for code review, which include changes to the IFS golden file so code reviewer are aware that ABI has changed
- Another benefit from this approach is that ELF targets that depends these system shared libraries can be built early by using the stubs instead of using the actual libraries, which increases build parallelism and reduces build time



Future work

- API signature change monitoring.
  - Libabigail integration
- Support more formats other than ELF.
  - COFF and Mach-O