NAME

llvm-ifs - shared object stubbing tool

SYNOPSIS

llvm-ifs [*options*] *inputs*

DESCRIPTION

llvm-ifs is a tool that jointly produces human readable text-based stubs (.ifs files) for shared objects and linkable shared object stubs (.so files) from either ELF shared objects or text-based stubs. The text-based stubs is useful for monitoring ABI changes of the shared object. The linkable shared object stubs can be used to avoid unnecessary relinks when the ABI of shared libraries does not change.

IFS FORMATS

Here is an example of the text representation (IFS) of a shared object produced by the **llvm-ifs**:

```
--- !ifs-v1
IFSVersion: 3.0
SoName: libtest.so /* Optional */
Target: x86_64-unknown-linux-gnu   /* Optional, format 1, same format as llvm target triple */
Target: { Arch: x86_64, Endianness: little, Bitwidth: 64 } /* Optional, format 2 */
NeededLibs:
 - libc.so.6
Symbols:
 - { Name: sym0, Type: Notype }
 - { Name: sym1, Type: Object, Size: 0 }
 - { Name: sym2, Type: Func, Weak: false }
 - { Name: sym3, Type: TLS }
 - { Name: sym4, Type: Unknown, Warning: foo }
...
```

- **IFSVersion**: Version of the IFS file for reader compatibility.

- **SoName** (optional): Name of the shared object file that is being stubbed.

- **Target** (optional): The architecture, endianness and bitwise information of this shared object. It can be either in explicit format or in implicit LLVM triple format. It can be optional and can be overridden from command line options.

- **NeededLibs**: The list of the external shared objects that this library depends on.

- **Symbols**: A collection of all data needed to link objects for each symbol, sorted by name in ascending order.
  - **Name**: Symbol name.
  - **Type**: Whether the symbol is an object, function, no-type, thread local storage, or unknown. Symbol types not explicitly supported are mapped as unknown to improve signal-to-noise ratio.
  - **Size**: The size of the symbol in question, doesn't apply to functions, and is optional for NoType symbols.
  - **Undefined**: Whether or not the symbol is defined in this shared object file.
  - **Weak**: Whether or not the symbol should be treated as weak.
  - **Warning** (optional): Warning text to output when this symbol is linked against.

    This YAML based text format contains everything that is needed to generate a linkable ELF shared object as well as an Apple TAPI format file. The ordering of symbols is sorted, so these files can be easily compared using diff tools. If the content of the file changes, it indicates a potentially ABI breaking change.

ELF STUB FORMAT

A minimum ELF file that can be used by linker should have following sections properly populated:

- ELF header.
- Section headers.
- Dynamic symbol table (**.dynsym** section).
- Dynamic string table (**.dynstr** section).
- Dynamic table (**.dynamic** section).
  - **DT_SYMTAB** entry.
  - **DT_STRTAB** entry.
  - **DT_STRSZ** entry.

- **DT_NEEDED** entries. (optional)
- **DT_SONAME** entry. (optional)

- Section header string table (**.shstrtab** section)

This ELF file may have compatibility issues with ELF analysis tools that rely on the program headers. Linkers like LLD work fine with such a minimum ELF file without errors.

OPTIONS

--input-format=[IFS|ELF|OtherObjectFileFormats]
Specify input file format. Currently, only text IFS files and ELF shared object files are supported. This flag is optional as the input format can be inferred.

--output-elf=<output-filename>
Specify the output file for ELF shared object stub.

--output-ifs=<output-filename>
Specify the output file for text IFS.

--output-tbd=<output-filename>
Specify the output file for Apple TAPI tbd.

--arch=[x86_64|AArch64|...]
This flag is optional and it should only be used when reading an IFS file which does not define the **Arch** (architecture). This flag defines the architecture of the output file, and can be any string supported by ELF 'e_machine' field. If the value is conflicting with the IFS file, an error will be reported and the program will stop.

--endianness=[little|big]
This flag is optional and it should only be used when reading an IFS file which does not define the **Endianness**. This flag defines the endianness of the output file. If the value is conflicting with the IFS file, an error will be reported and the program will stop.

--bitwidth=[32|64]
This flag is optional and it should only be used when reading an IFS file which does not define the **BitWidth**. This flag defines the bit width of the output file. If the value is conflicting with the input IFS file, an error will be reported and the program will stop.

--target=<target triple>

This flag is optional and should only be used when reading an IFS file which does not define any target information. This flag defines architecture, endianness and bit width of the output file using llvm target triple. This flag cannot be used simultaneously with other target related flags.

--hint-ifs-target=<target triple>

This flag is optional and should only be used when reading an ELF shared object and generating an IFS file. by default, llvm-ifs will use '**Arch**, **Endianness** and **BitWidth**' fields to reflect the target information from the input object file. Using this flag will tell llvm-ifs the expected target triple in the output IFS file. If the value matches the target information from the object file, this value will be used in the 'Target:' filed in the generated IFS. If it conflicts with the input object file, an error will be reported and the program will stop.

--hint-ifs-target

This flag is optional and should only be used when outputting an IFS file. This flag strips the **Arch** field from the IFS file so it can be overridden later.

--strip-ifs-endianness

This flag is optional and should only be used when outputting an IFS file. This flag strips the **Endianness** field from the IFS file so it can be overridden later.

--strip-ifs-bitwidth

This flag is optional and should only be used when outputting an IFS file. This flag strips the **BitWidth** field from the IFS file so it can be overridden later.

--strip-ifs-target

This flag is optional and should only be used when outputting an IFS file. This flag strips the **Target** field from the IFS file so it can be overridden later.

--write-if-changed

When this flag is set, llvm-ifs will only write the output file if it does not already exist or the content will be different from the existing file.

--strip-size

When this flag is set, llvm-ifs will remove the size field from the output ifs file. This is useful for shared objects that only intend to be linked against position independent code which doesn't need copy relocations, or where the size of an object is not a useful part of the abi to track.

EXIT STATUS

If **llvm-ifs** succeeds, it will exit with 0. Otherwise, if an error occurs, it will exit with a non-zero value.

AUTHOR

Maintained by the LLVM Team (https://llvm.org/).

COPYRIGHT