Test

Certainly! Let's go through how stripping and the other operations work in the context of your `libgreet.so` example.

1. Understanding the Purpose

When you compile a C program that uses a shared library like `libgreet.so`, the resulting binary includes various sections, such as:

- `.text`: The actual code (instructions) of your program.
- `.data`: Initialized data.
- `.bss`: Uninitialized data.
- `.dynamic`: Contains information about dynamic linking, like which shared libraries are needed (`DT_NEEDED` entries).
- `.dynstr`: A string table used by the dynamic linker.

The `--as-needed` and `--no-as-needed` flags affect whether a library like `libgreet.so` is included in the `.dynamic` section of the binary.

2. What Does Stripping Do?

Stripping refers to the process of removing certain sections or symbols from a binary to reduce its size or to focus on specific parts of the binary. In your example, we use the `strip` command to remove the `.dynamic` section, and we use `objcopy` to extract specific sections that are relevant to dynamic linking.

3. Detailed Steps with `libgreet.so` Example

Step 1: Compiling the Binaries

When you compile your C programs (like `hello_doesnt_need_lib.c` and `hello_needs_lib.c`), the compiler and linker create an executable binary. Depending on the flags used, `libgreet.so` might or might not be included in the `.dynamic` section of this binary.

- **With** `--no-as-needed`: The linker will include `libgreet.so` in the `.dynamic` section, even if the program doesn't actually use it.
- **With** `--as-needed`: The linker will only include `libgreet.so` if the program uses something from that library.

Step 2: Stripping Unnecessary Sections

```
strip -R .dynamic -o test_exec_no_as_needed.stripped test_exec_no_as_needed
```

- **Command Breakdown**:
  - `strip -R .dynamic`: This removes the `.dynamic` section from the binary.
  - `-o test_exec_no_as_needed.stripped`: This saves the stripped binary as `test_exec_no_as_needed.stripped`.
- **Why Strip?**:
  - Stripping the `.dynamic` section allows you to compare the binaries more directly by removing other information that might clutter the comparison. After stripping, the main difference between binaries should be related to the dynamic dependencies (like whether `libgreet.so` is needed).

Step 3: Extracting Specific Sections

```
objcopy --only-section=.dynamic --only-section=.dynstr test_exec_no_as_needed
test_exec_no_as_needed.dynamic_section
```

- **Command Breakdown**:
  - `objcopy --only-section=.dynamic --only-section=.dynstr`: This command extracts only the `.dynamic` and `.dynstr` sections from the binary.

  - `test_exec_no_as_needed.dynamic_section`: The output file containing just the extracted sections.
- **Why Extract?**:
  - By extracting just the `.dynamic` and `.dynstr` sections, you isolate the part of the binary that tells the dynamic linker which libraries are needed (`DT_NEEDED` entries). This makes it easier to focus on the exact difference caused by the `--as-needed` and `--no-as-needed` flags.

Step 4: Using `readelf` to Check the Dynamic Section

```
readelf -d test_exec_no_as_needed.dynamic_section
```

- **Command Breakdown**:
  - `readelf -d`: This command reads and displays the dynamic section of the binary.
  - `test_exec_no_as_needed.dynamic_section`: The file containing the `.dynamic` section that we extracted.
- **What You're Checking**:
  - When you run `readelf -d`, you're looking to see if `libgreet.so` is listed in the `DT_NEEDED` entries. This tells you whether the binary is linked to `libgreet.so` at runtime.

4. Practical Example

Here's how these operations affect your `libgreet.so` example:

1. **Compile** `hello_needs_lib.c` with `--no-as-needed`:
   - The `.dynamic` section of the resulting binary includes `libgreet.so` because the linker is forced to include it, even if not used.
2. **Compile** `hello_doesnt_need_lib.c` with `--as-needed`:
   - The `.dynamic` section might **not** include `libgreet.so` if the program doesn't actually use anything from `libgreet.so`.
3. **Stripping**:
   - You remove the `.dynamic` section to focus the comparison.
4. **Extracting Sections**:
   - You extract `.dynamic` and `.dynstr` to get only the relevant dynamic linking information.
5. **Reading with `readelf`**:
   - You use `readelf` to inspect the extracted sections and see whether `libgreet.so` is listed as a needed library.

5. Summary

- **Stripping** removes unnecessary sections to simplify comparison between binaries.
- **Extracting specific sections** isolates the dynamic linking information, making it easier to see how different compilation flags affect which libraries are needed.
- **Using** `readelf` on the extracted sections allows you to directly observe the impact of the `--as-needed` and `--no-as-needed` flags on the binary's dependencies.

By following these steps, you can clearly demonstrate the effect of these linker flags on whether `libgreet.so` is included in the `DT_NEEDED` section of your binaries.