ISA 563: Fundamentals of Systems Programming

Static and Shared Libraries

Feb. 5, 2013

Libraries

- A library is file containing one or more object files
  - Typically indexed for fast symbol lookups
- Helpful for code reuse
- Decreases compilation time
- Two types of libraries:
  - static
  - shared (dynamic)

Shared vs Static Libraries

- Static library
  - Included in the application binary
  - Can be used by multiple apps, but each one will include its own copy
  - Avoids missing library issues
  - Increases disk usage
  - Hard to update

Static vs Shared Libraries (Cont'd)

- Shared library
  - Not included in the application executable
  - Can be shared by multiple application
  - Saves space
  - Easy to update
  - Can cause library misses
  - Compiler only checks to make sure that no symbols are missing
  - Library is loaded at run time by the system loader

Static Libraries

- A collection of object files

```
$ gcc -c util_str.c -o util_str.o
$ gcc -c util_net.c -o util_net.o
$ ar rc libutil.a util_str.o util_net.o
$ ranlib libutil.a    # may not be necessary


$ gcc main.o -L. -lutil -o prog

or

$ gcc main.c -o prog libutil.a
```

Shared Libraries

```
$ gcc -fPIC -c util_str.c
$ gcc -fPIC -c util_net.c
$ gcc -shared -o libutil.so util_str.o util_net.o


$ gcc main.o -L. -lutil -o prog

$ ldd prog # list linked shared libraries
```

Loader search shared libraries in system specified directories.

LD_LIBRARY_PATH environment variable tell the loader to look into other directories.

Demo

main.c sutil_str.c, sutil_net.c ...

Using a Shared Library Dynamically

- The “dl” library
    - Load a shared library
    - Reference its symbols
    - Call its functions
    - Detach it from process
- dlopen() -- open shared library
- dlsym() -- open a reference to a symbol
    - reference can be used to call library function
- dlclose() -- detach library from process