

# ISA 563: Fundamentals of Systems Programming

Dynamically Loaded Libraries

Feb. 26, 2012

# Dynamically Loaded (DL) Libraries

- A way to load and use library functions at runtime:
  - Libraries are loaded after program startup
  - Program can start without dynamic libraries
  - Program can discover and load extra functionalities
  - Program can unload libraries when they are no longer needed

# DL Libraries (Cont'd)

- No special difference between dynamic and static libraries in file format
- The main difference is that DL libraries are loaded at runtime through system APIs:
  - Under most Unix/Linux systems:
    - `dlfcn.h`
  - Windows systems use completely different interface with DLLs

# Unix/Linux DL Interface

- Header inclusion:
  - `#include <dlfcn.h>`
- Compilation
  - `$ gcc foo.c -o prog -ldl`

# Unix/Linux DL Interface (Cont'd)

- Functions:

- `void * dlopen(const char *filename, int flag);`
  - Opens and prepares a library
- `void * dlsym(void *handle, char *symbol);`
  - Looks up symbol from opened library
- `int dlclose(void *handle);`
  - Closes opened library
- `char *dlerror(void);`
  - Returns error message from previous dl\* call

# Demo

`dynmath.c, statmath.c`

# Library Interposition

- Dynamic linker:
  - Loads and links shared libraries when a program is executed
- Environment variables that affects dynamic linker:
  - LD\_LIBRARY\_PATH
    - Lists directories to be search first (before standard library paths)
  - LD\_PRELOAD
    - Lists shared library files to be used first

# Library Interposition (Cont'd)

- No modification to application binary is necessary
- Useful scenarios:
  - Testing new libraries
  - Debugging
  - Profiling
  - Monitoring
  - Other fun stuff



# Interposing a Function Call

- Example: interpose a function: `int foo(int n);`

```
/*
 * bar.c - interposes function foo
 */
#include <stdio.h>
#include <dlfcn.h>

int foo(int n)
{
    static int (*f)();

    if(!f) {
        f = (int (*)(int)) dlsym(RTLD_NEXT, "foo");
    }
    printf("foo(%d) is called\n", n);
    return(f(n));
}
```

# Interposing a Function Call (cont'd)

- compile:
  - `gcc -shared -o bar.so bar.c -ldl`
- usage:
  - `LD_PRELOAD=bar.so app_that_uses_foo`
- The program `app_that_uses_foo` does not need be modified in any way

# Demo

imalloc.c, ifile.c