

Lecture Topics

- module design: C to C++ [continued]
 - example: netlink
 - C++ features

Administrivia

- [prewrite first set of bullets? They're review.]
- Hand out NetLink code.
- code examples (netlink and NetLink) are on the web
- [EWS accounts still not active.]

- We'll write as a module in C (netlink) [this part covered quickly last time]
 - support only TCP
 - two kinds of net links: server and connections
 - can create a server and then accept connections
 - can connect to a server by host name and port or IP address and port
 - can read/write over a connection
(we'll fix the number of bytes for simplicity)
- visible aspects of the module [oops, screwed up port types in handout]
 - prefix "NL_" or "nl_"
 - enumeration of error values that can be returned
 - enumeration of TCP port numbers (for convenience)
 - declaration of netlink_t structure (opaque)
- functions for netlink servers

```
netlink_t* nl_create_server (uint16_t port);  
netlink_t* nl_accept (netlink_t* server);
```

- functions for netlink connections

```
netlink_t* nl_connect_by_name (const char* host_name,  
                               uint16_t tcp_port);  
netlink_t* nl_connect_by_ip (uint32_t ip_addr,  
                              uint16_t tcp_port);  
nl_err_t nl_blocking_read (netlink_t* n, void* buf,  
                           uint32_t n_bytes);  
nl_err_t nl_blocking_write (netlink_t* n, void* buf,  
                             uint32_t n_bytes);
```

- functions for destroying a netlink

```
nl_err_t nl_free (netlink_t* n);
```

- functions for getting information about a netlink

```
uint32_t nl_get_remote_ip_address (netlink_t* n);  
uint16_t nl_get_tcp_port (netlink_t* n);  
int32_t nl_is_connected (netlink_t* n);  
int32_t nl_is_server (netlink_t* n);
```

- module initialization to handle SIGPIPE

```
nl_err_t nl_initialize ();
```

- that's it!
 - implementation sits in a single file (a little over 500 lines)
 - note that access functions aren't likely to be used often, so performance is not a huge concern
 - most issues are handled by module, not exposed to users, but...
 - there are some deadlock possibilities
- source code file includes
 - netlink_t structure definition
 - several internal functions
 - for closing, allocation, freeing, and connecting
 - mostly to avoid code replication

Module Design in C++

- What additional tools does C++ provide to help you develop modules?
 - a class is a module (as defined earlier); it defines
 - fields & static data
 - interface & internal functions
 - instance initialization/teardown functions (constructors/destructors)
 - module initialization/teardown
 - handled by generalizing static variable initialization
 - to allow non-constant initialization (e.g., function calls)
 - controlling inter-dependent module ordering is tricky (see Str Sec. 3.11.4)
 - usually sufficient to do a check in constructor
 - class also defines a named scope
 - symbols in the class' scope must be listed within a single block
 - but definitions can be spread out arbitrarily
 - distribution enabled by adding concept of access control for names
 - in C, scope = visibility and access rights
 - in C++, scope = visibility only, and does not imply access
 - protects against accidents, not fraud/malice
 - access granted by class to class/function (not individual objects); access cannot be taken from outside
 - access rights are per name (important for overloading)
 - keywords in a class
 - “private:” access allowed only within the class (field / impl. func.)
 - “protected:” access allowed within class and derived classes (as with private, field / impl. func.)
 - “public:” access allowed to anyone (interface func.)
 - “static” class function / variable
 - (not static) instance function with implicit pointer arg / field

- another extension: expand the single global scope
 - namespaces
 - named using arbitrarily long names
 - can create local synonyms for brevity
 - symbols need not be listed within a single block
 - can nest hierarchically
 - namespaces and classes much like a directory structure