What you should get from today’s session

- A quick look at some details you’ll need to do our programming assignment, including:
  - The framework we’re using
  - C++ Exceptions
  - A tiny bit about inheritance
  - Makefiles
  - C/C++ tips and tricks
Working through the Demo Client
Preamble

#include "c150dgmsocket.h"
#include "c150debug.h"
#include <fstream>

using namespace std;   // for C++ std library
using namespace C150NETWORK;
Preamble

```
#include "c150dgmsocket.h"
#include "c150debug.h"
#include <fstream>

using namespace std; // for C++ std library
using namespace C150NETWORK;
```
Main pingclient Logic
Client logic

```c++
try {
    C150DgmSocket *sock = new C150DgmSocket();
    sock -> setServerName(argv[serverArg]);

    sock -> write(argv[msgArg], strlen(argv[msgArg])+1);
    readlen = sock -> read(incomingMessage, sizeof(incomingMessage));

    checkAndPrintMessage(readlen, incomingMessage, sizeof(incomingMessage));
}

catch (C150NetworkException e) {
    cerr << argv[0] << "": caught C150NetworkException: " <<
        e.formattedExplanation()
    << endl;
}
```

This is not an ordinary socket…it’s a smart wrapper around a socket

Establishes us as a client…and identifies the server…ports set based on login id
Client logic

```cpp
try {
    C150DgmSocket *sock = new C150DgmSocket();
    sock -> setServerName(argv[serverArg]);

    sock -> write(argv[msgArg], strlen(argv[msgArg])+1);
    readlen = sock -> read(incomingMessage, sizeof(incomingMessage));

    checkAndPrintMessage(readlen, incomingMessage, sizeof(incomingMessage));
}

catch (C150NetworkException e) {
    cerr << argv[0] << ": caught C150NetworkException: " <<
         e.formattedExplanation() << endl;
}
```
Demo Server
try {
    C150DgmSocket *sock = new C150DgmSocket();
    c150debug->printf(C150APPLICATION,"Ready to accept messages");
    while(1) {
        readlen = sock -> read(incomingMessage, sizeof(incomingMessage)-1);
        // ... WORK WITH MESSAGE HERE
        string response = "SOME RESPONSE HERE";
        sock -> write(response.c_str(), response.length()+1);
    }
}

catch (C150NetworkException e) {
    c150debug->printf(C150ALWAYSLOG,"Caught C150NetworkException: %s\n",
                      e.formattedExplanation().c_str());
}
try {
    C150DgmSocket *sock = new C150DgmSocket();
    c150debug->printf(C150APPLICATION,"Ready to accept messages");
    while(1) {
        readlen = sock -> read(incomingMessage, sizeof(incomingMessage)-1);
        // WORK WITH MESSAGE HERE
        string response = "SOME RESPONSE HERE";
        sock -> write(response.c_str(), response.length()+1);
    }
}

catch (C150NetworkException e) { 
    c150debug->printf(C150ALWAYSLOG,"Caught C150NetworkException: \%s\n",
        e.formattedExplanation().c_str());
}
Inferring who is a server and who is a client

try {
C150DgmSocket *sock = new C150NastyDgmSocket(nastiness);
c150debug->printf(C150APPLICATION, "Ready to accept messages");
while(1) {
readlen = sock -> read(incomingMessage, sizeof(incomingMessage)-1);
// … WORK WITH MESSAGE HERE
string response = "SOME RESPONSE HERE";
sock -> write(response.c_str(), response.length()+1);
}
}
catch (C150NetworkException e) {
c150debug->printf(C150ALWAYSLOG, "Caught C150NetworkException: %s\n", e.formattedExplanation().c_str());
}

NOTE: The socket class imposes a simple notion of client/server on UDP...

It decides whether you’re a server or client based on which methods you call first

1) client calls setServer name then writes

2) server starts by doing a read.

Note a very robust approach for production code, but handy for these simple programs.
C++ Inheritance
A super-simple look at C++ Inheritance

```cpp
class Shape {
    private:
        Point position;
    public:
        Point getPosition();
        virtual void draw() = 0;
};
```

Base class “Shape” has draw() method with no implementation (=0)
A super-simple look at C++ Inheritance

class Shape {
private:
    Point position;
public:
    Point getPosition();
    virtual void draw() = 0;
};

Class Circle : public Shape {
public:
    virtual void draw();
};

Class Square : public Shape {
public:
    virtual void draw();
};

Each subclass provides its own implementation of draw()
Class shape {
    private:
        Point position;
    public:
        Point getPosition();
        virtual void draw() = 0;
};

Class Circle : public Shape {
    public:
        virtual void draw();
};

Class Square : public Shape {
    public:
        virtual void draw();
};

Shape *shapeArray[2];
int i;

shapeArray[0] = new Circle();
shapeArray[1] = new Square();

for(i=0; i<2; i++) {
    cout << shapeArray[i] -> position;
    shapeArray[i] -> draw();
}
A super-simple look at C++ Inheritance

```cpp
class shape {
    private:
        Point position;
    public:
        Point getPosition();
        virtual void draw() = 0;
};

class Circle : public Shape {
    public:
        virtual void draw();
};

class Square : public Shape {
    public:
        virtual void draw();
};

Shape *shapeArray[2];
int i;

shapeArray[0] = new Circle();
shapeArray[1] = new Square();

for(i=0; i<2; i++) {
    cout << shapeArray[i] -> position;
    shapeArray[i] -> draw();
}
```

First time calls Circle::draw, second time calls Square::draw
try {
    C150DgmSocket *sock = new C150NastyDgmSocket(nastiness);
    c150debug->printf(C150APPLICATION,"Ready to accept messages");
    while(1) {
        readlen = sock -> read(incomingMessage, sizeof(incomingMessage)-1);
        // ... WORK WITH MESSAGE HERE
        string response = "SOME RESPONSE HERE";
        sock -> write(response.c_str(), response.length()+1);
    }
}

catch (C150NetworkException e) {
    c150debug->printf(C150ALWAYSLOG,"Caught C150NetworkException: %s
", e.formattedExplanation().c_str());
}
C++ Exceptions
C++ Exceptions

```cpp
try {
    C150DgmSocket *sock = new C150DgmSocket();
    sock -> setServerName(argv[serverArg]);
    sock -> write(argv[msgArg], strlen(argv[msgArg]) + 1);
    readlen = sock -> read(incomingMessage, sizeof(incomingMessage));
    checkAndPrintMessage(readlen, incomingMessage, sizeof(incomingMessage));
}

catch (C150NetworkException e) {
    cerr << argv[0] << ": caught C150NetworkException: " << e.formattedExplanation() << endl;
}
```

C++ has try/catch/throw for Exceptions

- **try clause runs first**
- **Any network exception in try block or methods called by try block takes us here**
- **e is of whatever type was “thrown”**
C++ Exceptions

- Exceptions are particularly useful for network code...
- ...no need to “percolate” return codes through layers of method calls
- Standard COMP 150IDS Exception Class:

  ```cpp
  throw C150NetworkException("Client received message that was not null terminated");
  ```

When an error occurs, throw an Exception (same as “raise” in other langs):

  ```cpp
  throw C150NetworkException (or other class)
  ```

- Exception classes form a hierarchy...based on class inheritance (no need for you to worry about that if you don’t know C++ inheritance)
Makefiles
Dependency –based
Command Execution
Makefile variables

# Do all C++ compies with g++
CPP = g++
CPPFLAGS = -g -Wall -Werror -I$(C150LIB)

# Where the COMP 150 shared utilities live, including c150ids.a and userports.csv
# Note that environment variable COMP150IDS must be set for this to work!
C150LIB = $(COMP150IDS)/files/c150Utils/

[... several lines skipped...]
pingclient: pingclient.o  $(C150AR) $(INCLUDES)
   $(CPP) -o pingclient pingclient.o

Variables defined this way...

...or from environment... (this is one reason you setenv COMP150IDS /comp/150IDS)

...used this way
Targets and dependencies

# Do all C++ compies with g++
CPP = g++
CPPFLAGS = -g -Wall -Werror -I$(C150LIB)

# Where the COMP 150 shared utilities live, including c150ids.a and userports.csv
# Note that environment variable COMP150IDS must be set for this to work!
C150LIB = $(COMP150IDS)/files/c150Utils/

[… several lines skipped…]

pingclient: pingclient.o  $(C150AR) $(INCLUDES)
  $(CPP) -o pingclient pingclient.o
What gets run

# Do all C++ compies with g++
CPP = g++
CPPFLAGS = -g -Wall -Werror -I$(C150LIB)

# Where the COMP 150 shared utilities live, including c150ids.a and userports.csv
# Note that environment variable COMP150IDS must be set for this to work!

C150LIB = $(COMP150IDS)/files/c150Utils/

[.. several lines skipped...]

pingclient: pingclient.o  $(C150AR) $(INCLUDES)
  $(CPP) -o pingclient pingclient.o

When pingclient is older than pingclient.o, etc. ...

..use g++ to relink it
Fancier dependencies

Each xxx.o file

%.o: %.cpp $(INCLUDES)

$(CPP) -c $(CPPFLAGS) $<

...depends on xxx.cpp

...and is compiled from that .cpp file
C cs. C++ Strings
C vs C++ Strings – we use both!

- C++ provides automatic allocation and useful concatenation operations
- C char[] arrays needed for formatting message packets
- File and socket APIs defined in terms of C byte arrays
- Also…preference
  - For some purposes, printf/scanf are handier than C++ <<
  - Etc.
Some hints on strings

- Won’t try a full tutorial here but, remember that you can convert:

  ```
  char cstring[4] = "abc";     // remember the null!
  string newstring(cstring);  // initialize C++ string
                              // from C string
  char *fromCPlusPlus = newstring.c_str();
                              // IMPORTANT: fromCPlusPlus
                              // is stable ONLY until
                              // next change to newstring
  ```

- Our focus is on the distributed system design...performance matters some, but do what’s easy and clean
Stringstreams: useful for formatting and conversions

- C++ strings do not support `<<`, but string streams do

```cpp
#include <sstream>         // for stringstream
#include <iostream>       // for cout

stringstream ss;        // empty stringstream
int answer = 25;

ss << "The answer is " << answer << " pounds" << endl;
cout << ss.str();        // get string from stringstream
```
What I Do (mostly)

- I mostly use C++ strings: automatic allocation
- I use or convert to char[] if I'm using APIs that need them
- I use either printf or stringstream for formatting or...
- ...concatenate strings with “+” (slower, and edge cases where it doesn’t work)
- Not all the framework code makes good choices internally...you’ll find some stuff that probably should be cleaned up (e.g. excess conversions)