



Brief Tutorial: Programming in C/C++ for Digital Image Processing

Mahmut Karakaya,
Graduate Teaching Assistant

TA Hours: M:9am-11am, W:1pm-3pm

<http://www.ece.utk.edu/~mkarakay>

Email: mkarakay@utk.edu

Basis structure of a C++ program

```
/******  
* This program converts gallons to liters.  
* This is our first C++ program.  
* Author:  
* Date: 10/07/99  
*****/  
  
#include <iostream>  
using namespace std;  
  
int main()  
{  
    float gallons, liters;  
  
    cout << "Enter number of gallons: ";  
    cin >> gallons;    // this inputs from the user  
  
    liters = gallons * 3.7854; // converts to liters  
  
    cout << "Liters: " << liters << "\n";  
  
    return 0;  
}
```

Header comment - multi-line comment

header file is included in the system by **#include**. Here, `<iostream>` is used to support the C++ I/O system.

The **using** statement informs the compiler that you want to use the **std** namespace. This is the namespace in which the entire Standard C++ library is declared. By using the **std** namespace, you simplify access to the standard library.

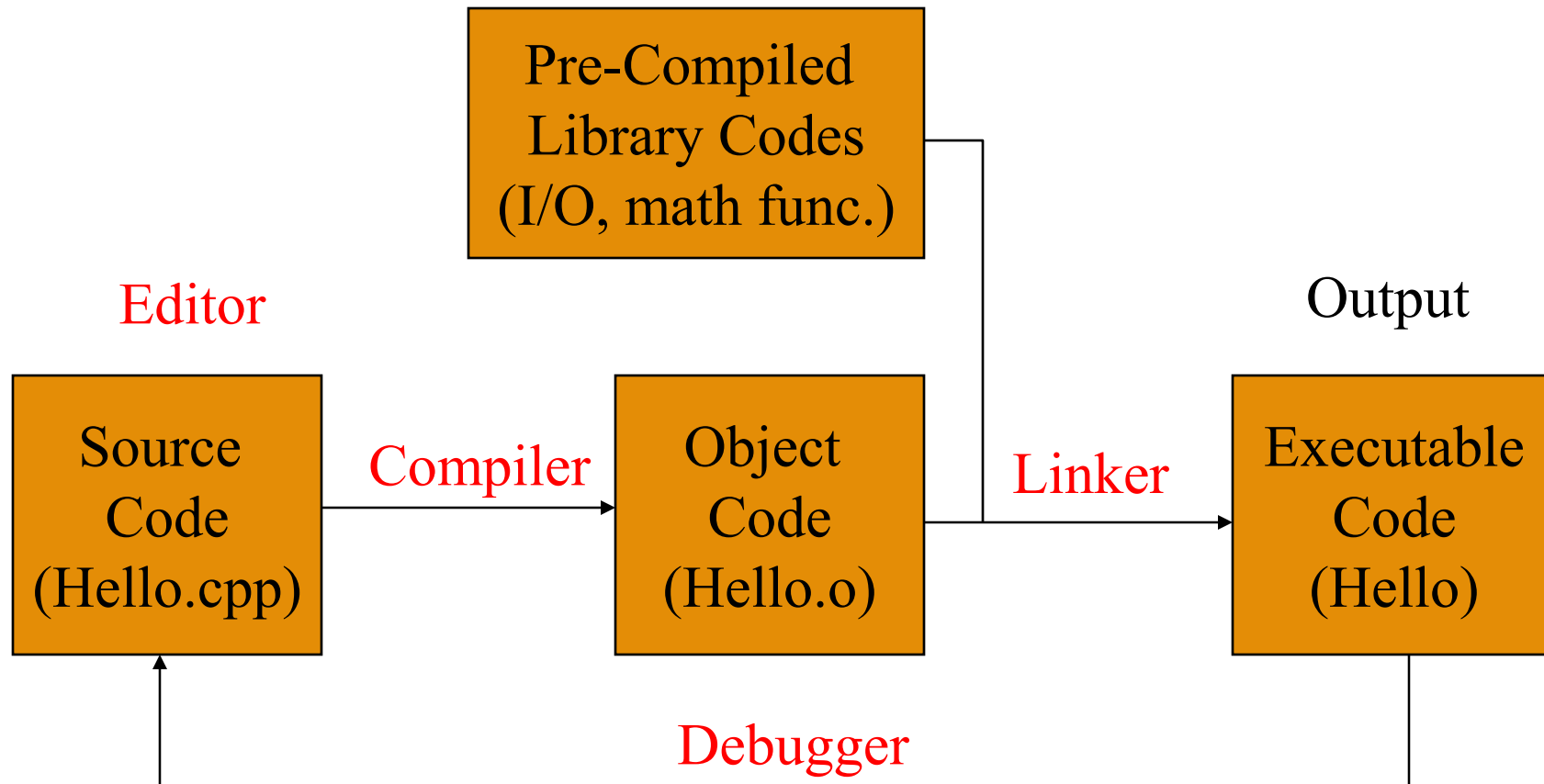
The only function that any C++ program **must** include is the **main()** function.

Components of C++ program

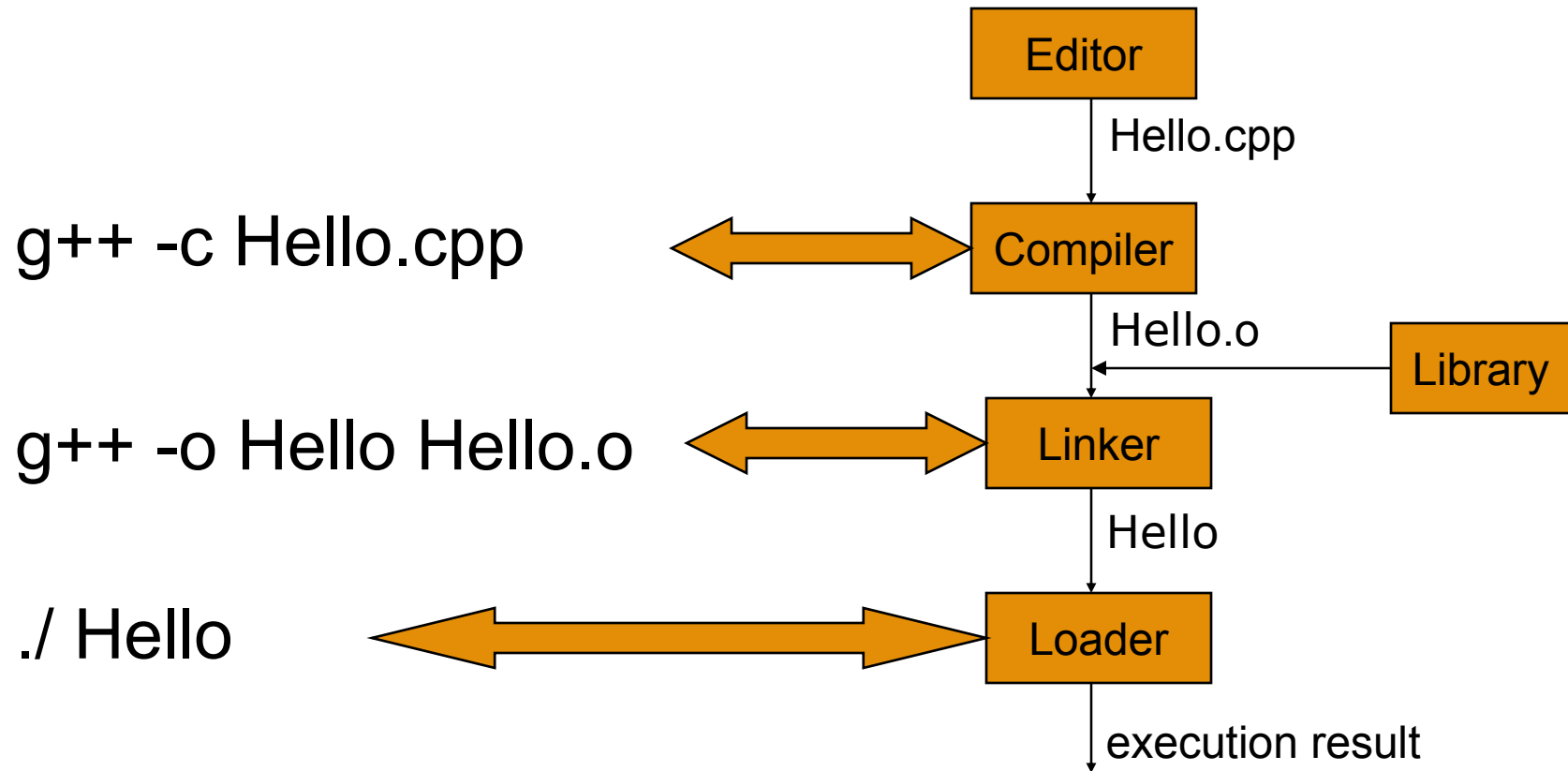
- Comments
 - // for single line or /* */ for multiple lines
- Compiler directive
 - #include
- Header file
 - <iostream>: input/output stream header
- Function block
 - int main()
- Braces
 - { begins the body of a function and } ends the body
- Statement
 - liters = gallons * 3.7854;
- Statement terminator
 - ;
- Return

```
/******  
* This program converts gallons to liters.  
* This is our first C++ program.  
* Author:  
* Date: 10/07/99  
*****/  
  
#include <iostream>  
using namespace std;  
  
int main()  
{  
    float gallons, liters;  
  
    cout << "Enter number of gallons: ";  
    cin >> gallons;    // this inputs from the user  
  
    liters = gallons * 3.7854; // converts to liters  
  
    cout << "Liters: " << liters << "\n";  
  
    return 0;  
}
```

Programming environment

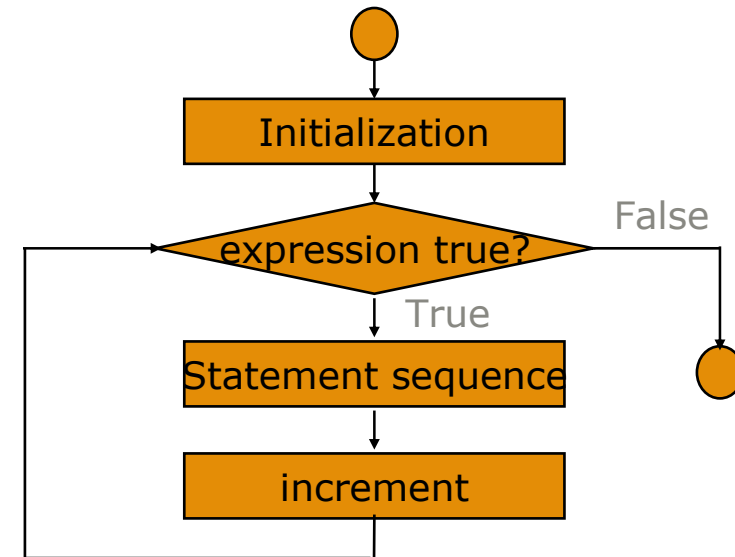


Linux commands



Control Structures

- if... / if... else... / if... else if...
- switch...
- while
- for...



```
for (initialization; expression; increment) {  
    statement sequence;  
}
```

Numbered annotations on the code: 1 points to 'initialization', 2 points to 'statement sequence', 3 points to 'increment', and 4 points to 'expression'.

Calculate the sum of all even numbers less than 1000.



```
C++:  
int total = 0;  
for ( int even=2; even<1000; even=even+2 )  
    total += even;
```

Functions

- Functions
 - C++ standard library (Pre-packaged)
 - User defined
- More manageable program - simplify the problem by decomposing it into small pieces
- Software reusability - using existing functions as building blocks to create new programs
- Avoid repeating code
- Information hiding - all variables declared in function definitions are local variables

Function format

- Function prototype

```
return-type function-name ( parameter types );
```

A diagram showing a function prototype: `return-type function-name (parameter types);`. The text is enclosed in a black rectangular box. Two yellow arrows point upwards from below the box to the words "return-type" and "parameter types".

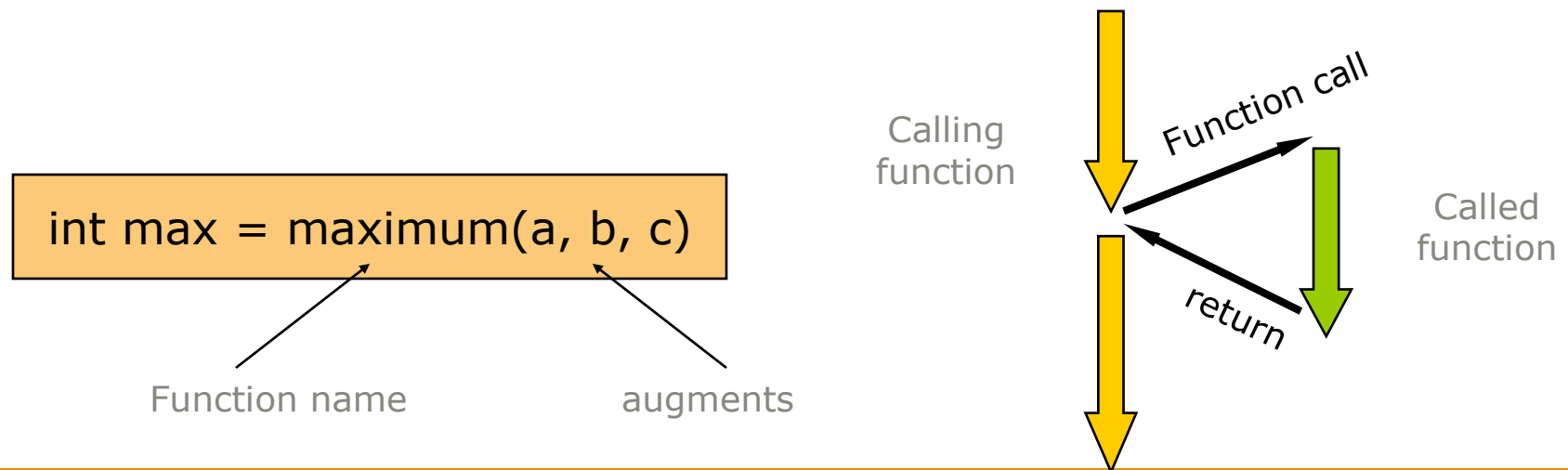
- Function definition

```
return-type function-name ( parameter-list )  
{  
    declarations and statements; // function body  
}
```

- Prototype and definition must be consistent

Function Call

- Functions can return values
- Returned value **MUST** match the type specified in the prototype
- Function arguments
 - Variables, constants, expressions



Example

```
// the main source file
// saved as tests.cpp

#include <iostream>
#include "tests.h"
using namespace std;

int main()
{
    int a, b, c;

    cout << "Enter three integers: ";
    cin >> a >> b >> c;
    cout << "Maximum is: " ;
    cout << maximum(a, b, c) << endl;
    cout << minimum(a, b, c) << endl;

    return 0;
}
```

```
// the function file
// saved as maximum.cpp
#include "tests.h"

int maximum(int x, int y, int z)
{
    int max = x;

    if (y > max) max = y;
    if (z > max) max = z;

    return max;
}

int minimum(int x, int y, int z)
{
    int min = x;

    if (y < min) min = y;
    if (z < min) min = z;

    return min;
}
```

```
// the header file
// saved as tests.h

int maximum(int, int, int);
int minimum(int, int, int);
```

Compile and link

- Compile each .cpp file individually
 - `g++ -c maximum.cpp`
 - `g++ -c tests.cpp`
- Link all object files
 - `g++ -o tests tests.o maximum.o`
- Modifications in one .cpp file DO NOT affect the compilation of other files

Using command-line arguments:

- It is possible to pass **arguments** to the **main** function from a command line by including the following two parameters in the parameter list
 - **int argc** (argc receives the number of command-line arguments)
 - **char *argv[]** (an array of strings where the actual command-line arguments are stored)

Header files:

- Standard header files are used to provide function prototypes for functions defined in the standard C++ library.
- Access standard header files
 - `#include <header-name>`
- Access user-defined header files
 - `#include "myfunctions.h"`

```

// test code for contrast stretching
#include "Image.h"
#include "Dip.h"
#include <iostream>
#include <cstdlib>
using namespace std;

#define Usage "testcs inimg outimg slope\n"

int main(int argc, char **argv)
{
    Image inimg, outimg; // the original image
    float m, b;

    // check if the number of arguments is correct
    if (argc < 5) {
        cout << Usage;
        exit(3);
    }

    // read in command-line arguments
    m = atoi(argv[3]);
    b = atoi(argv[4]);

    // read in image
    inimg = readImage(argv[1]);

    // test the contrast stretching function
    outimg = cs(inimg, m, b);

    // output the image
    writeImage(outimg, argv[2]);
    return 0;
}

```

```

// Contrast stretching.  $s = m*r + b$ 
#include "Image.h"
#include "Dip.h"
#include <iostream>
#include <cmath>

using namespace std;

Image cs(Image &inimg, float m, float b)
{
    Image outimg;
    int i, j, k;
    int nr, nc, ntype, nchan;

    // allocate memory
    nr = inimg.getRow();
    nc = inimg.getCol();
    ntype = inimg.getType();
    nchan = inimg.getChannel();

    outimg.createImage(nr, nc, ntype);

    // perform contrast stretching
    for (i=0; i<nr; i++)
        for (j=0; j<nc; j++)
            for (k=0; k<nchan; k++)
                outimg(i,j,k) = m * inimg(i,j,k) + b;

    return outimg;
}

```

```

// Dip.h - header file
#include "Image.h"

Image cs(Image &, float, float);

#endif

```

Makefile

```
OBJ = Image.o imageIO.o cs.o
AR = ar
INCLUDE = -I../include
all:
    ${MAKE} libimage.a

libimage.a: $(OBJ)
    $(AR) rvc $@ $(OBJ)
    ranlib $@

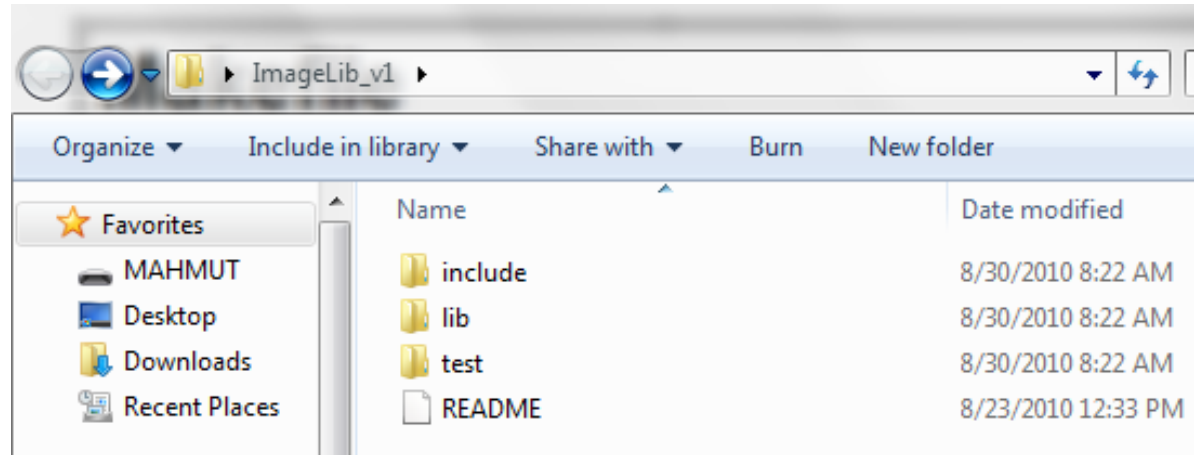
cs.o: cs.cpp
    g++ -c cs.cpp $(INCLUDE)

Image.o: Image.cpp
    g++ -c Image.cpp $(INCLUDE)

clean:
    -rm *.o *~
```

```
To generate the image library
cd lib
make
```

```
To run the test codes
cd test
make
```



```
EXES = testcs

all:
    ${MAKE} ${EXES}

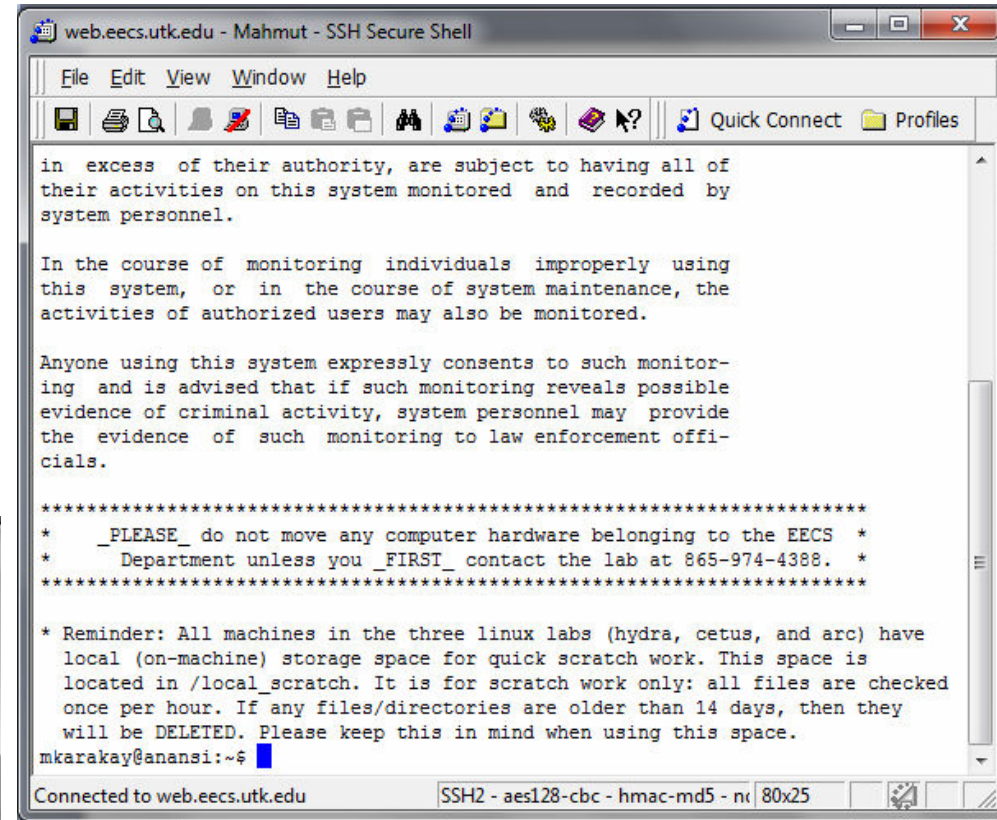
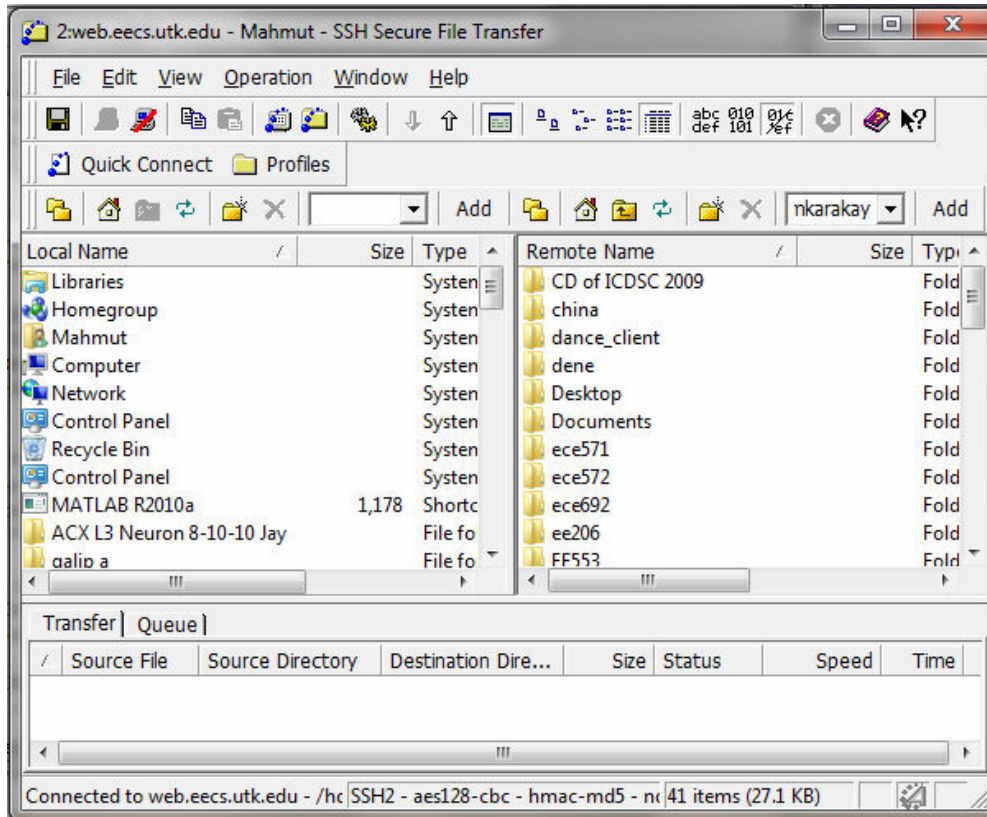
INCLUDE = -I../include
LIB = -L../lib

testcs: testcs.o
    g++ -o testcs testcs.o $(LIB) -limage
testcs.o: testcs.cpp
    g++ -c testcs.cpp $(INCLUDE)

clean:
    -rm -rf *.o
```

SSH-Secure Shell

- If you want to use linux system, you can download and install the "SSH secure shell" into your computer and do your projects at home instead of doing in the lab.



- To download the program from http://www.ece.utk.edu/~mkarakay/courses_files/dip.html.
- To make a new connection, click "Quick Connect"
 - Host Name: arc1.eecs.utk.edu
 - Username: your webID
 - Enter your password

Programming in Image Processing by using Dev C++

- If you want to use windows system, you can download and install the free software "Dev-C++" provided by Bloodshed Software into your computer.
- To download the program http://www.ece.utk.edu/~mkarakay/courses_files/devcpp-4.9.9.2_setup.exe.

