# Text File I/O

Chapter 6

Pages 287 – 298 (Sixth Edition)

Pages 262 – 274 (Fifth Edition)

# File I/O in an Object-Oriented Language

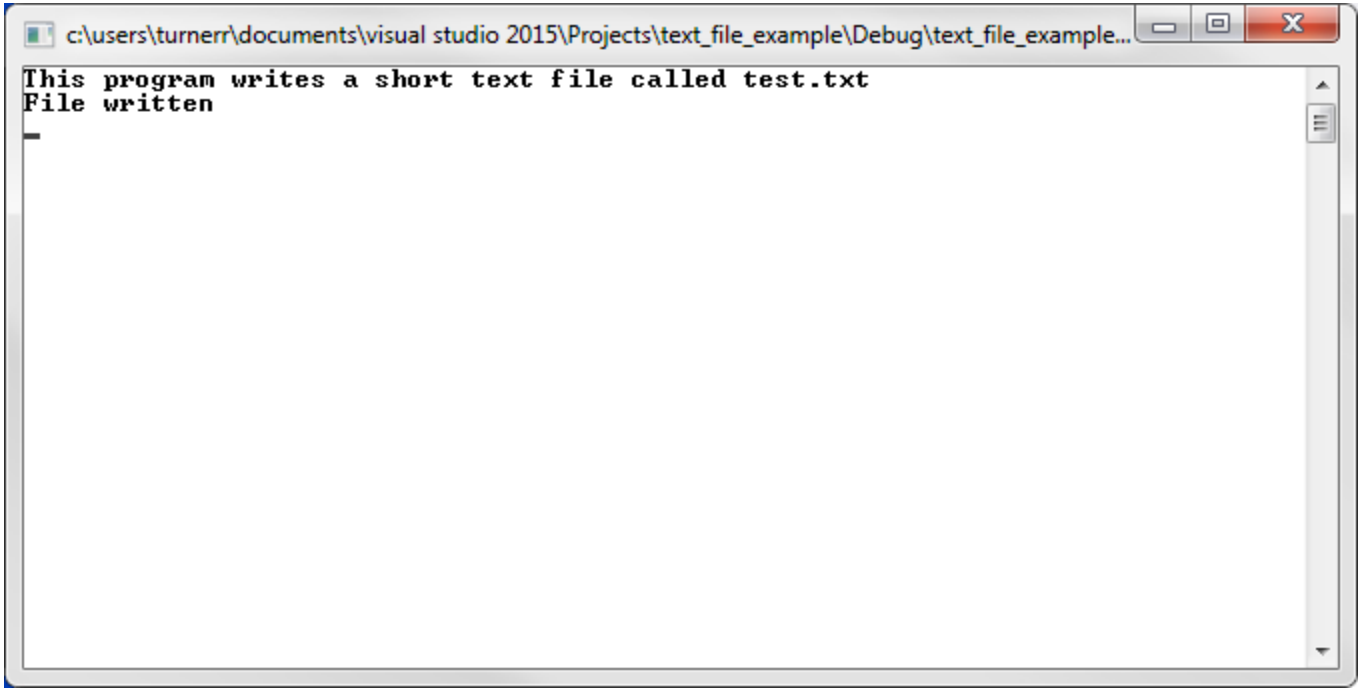
- Instantiate an **`ofstream`** object.
  - Like opening a file for output in C.
  - The object holds the state information that is in the FILE struct in C.
  - Call methods of the object to write the file.

- Must #include <fstream>

# Writing a Text File

```
#include <iostream>
#include <fstream>

int main()
{
    using namespace std;
    ofstream outfile;

    cout << "This program writes a short text file ";
    cout << "called test.txt\n";

    outfile.open("test.txt");
    outfile << "Here is the first line of text\n";
    outfile << "Here is the second line\n";
    outfile << "Here is the third and final line\n";
    outfile.close();

    cout << "File written\n";
    cin.get();  // Keep window open;
    return 0;
}
```

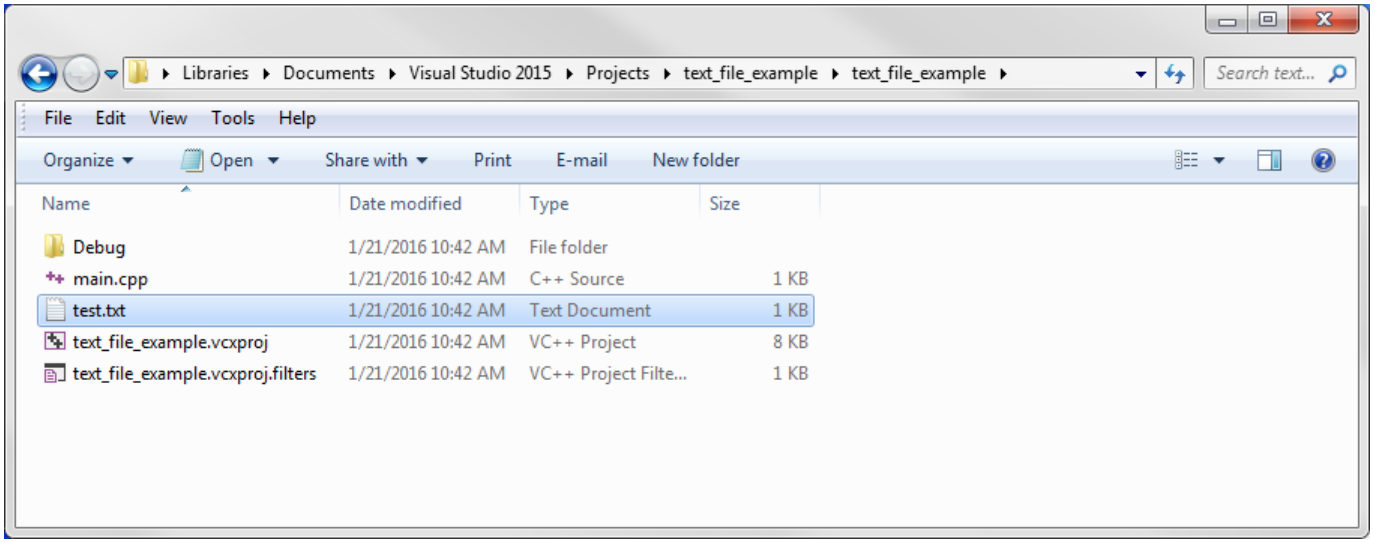
# Writing a Text File




Where is it?

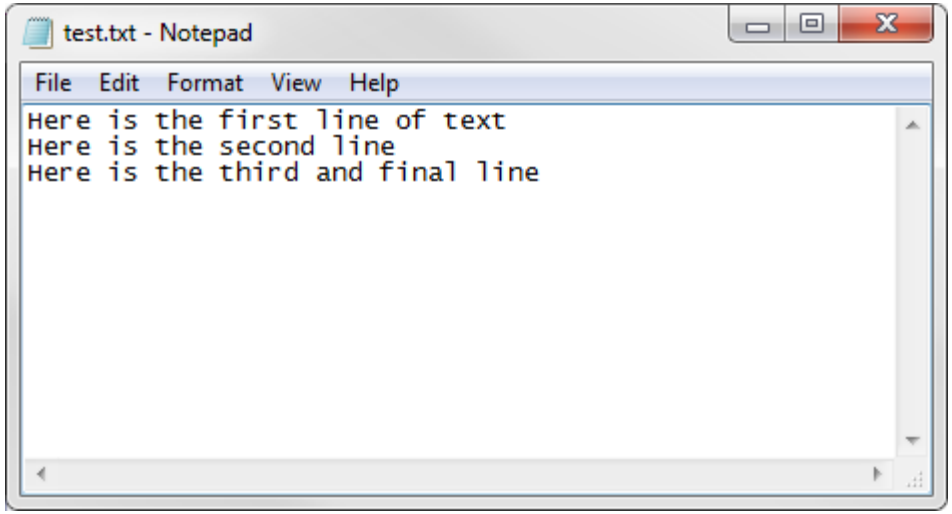
# Writing a Text File

Click on file to open in Notepad++.



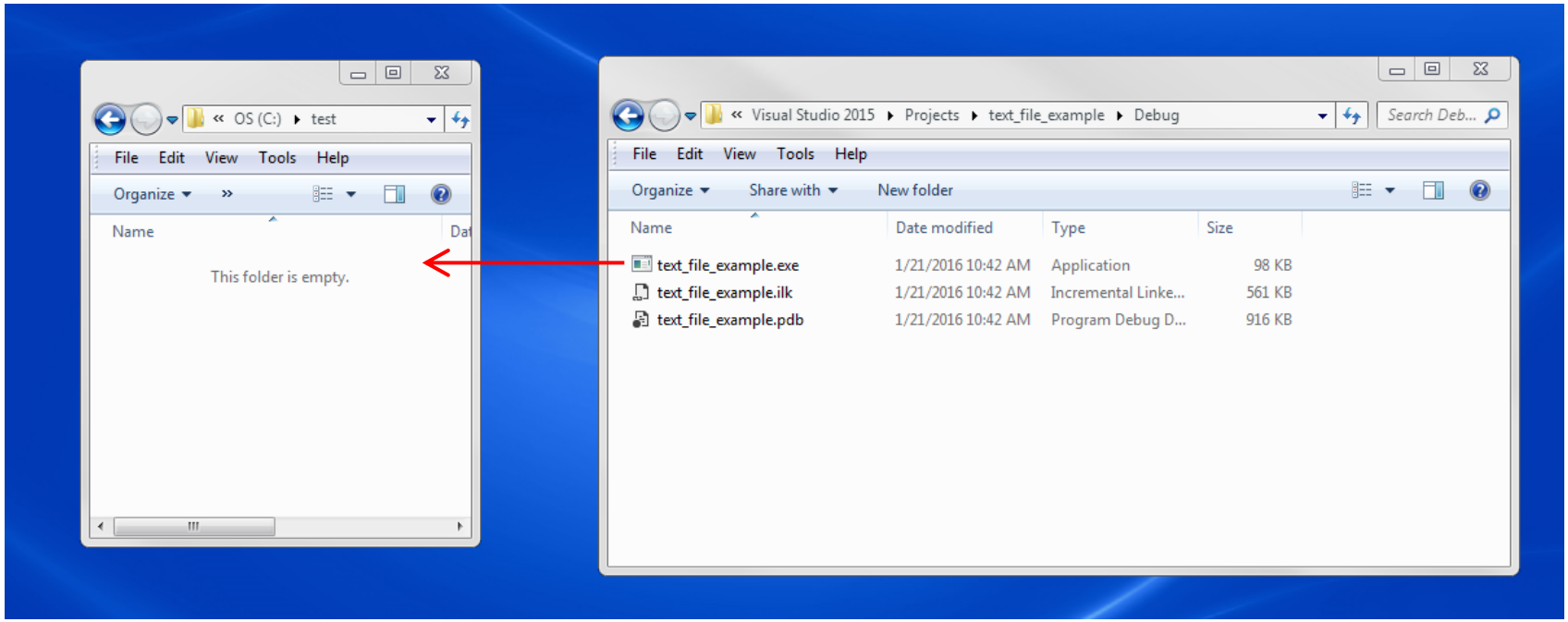

When running from Visual Studio, the default directory is the project directory.

# Here is the File in Notepad



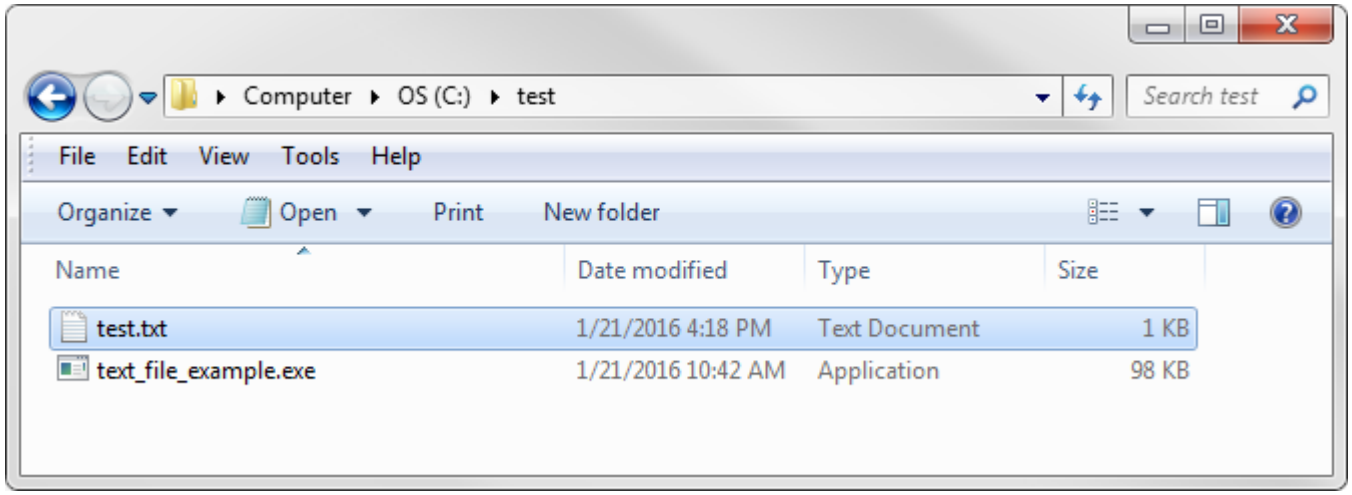
# What if we run the exe file?

- Create directory test at top level of C:
- Open directory window.
- Copy write_file.exe from the project debug directory to C:\test



# What if we run the exe file?

- Double click on the exe file in C:\test to run it.
- Look at C:\test




- Output file is in same directory as the exe file.

# How about from a console window?

- Open a command window.
- cd to C:/test
- Delete test.txt
- Type the name of the .exe file as a command
  - text_file_example

C:\Windows\system32\cmd.exe

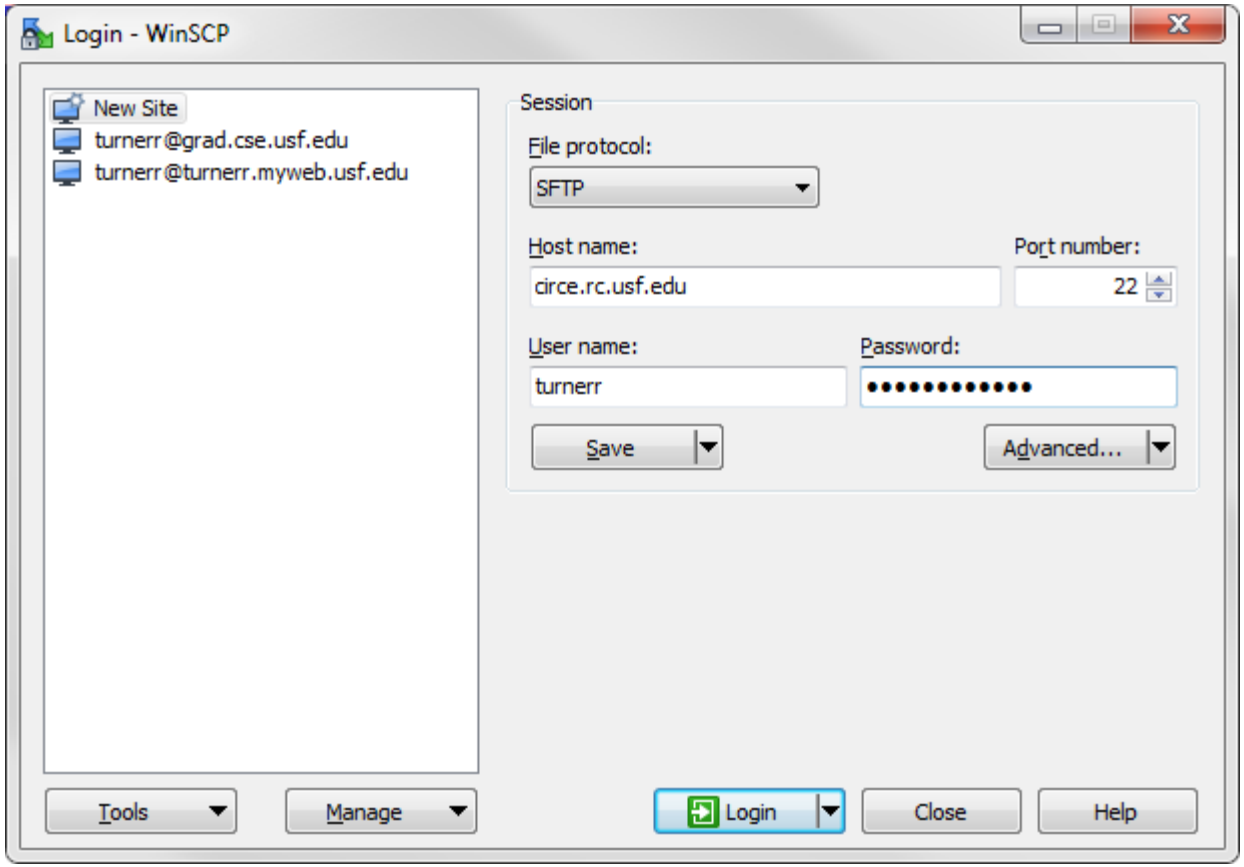
```
C:\>cd test

C:\test>dir
 Volume in drive C is OS
 Volume Serial Number is 046D-9AF1

 Directory of C:\test

01/21/2016  04:25 PM    <DIR>          .
01/21/2016  04:25 PM    <DIR>          ..
01/21/2016  04:25 PM                91 test.txt
01/21/2016  10:42 AM           100,352 text_file_example.exe
               2 File(s)        100,443 bytes
               2 Dir(s)  382,923,309,056 bytes free

C:\test>del test.txt

C:\test>dir
 Volume in drive C is OS
 Volume Serial Number is 046D-9AF1

 Directory of C:\test

01/21/2016  04:26 PM    <DIR>          .
01/21/2016  04:26 PM    <DIR>          ..
01/21/2016  10:42 AM           100,352 text_file_example.exe
               1 File(s)        100,352 bytes
               2 Dir(s)  382,923,309,056 bytes free

C:\test>text_file_example
This program writes a short text file called test.txt
File written


C:\test>dir
 Volume in drive C is OS
 Volume Serial Number is 046D-9AF1

 Directory of C:\test

01/21/2016  04:26 PM    <DIR>          .
01/21/2016  04:26 PM    <DIR>          ..
01/21/2016  04:26 PM                91 test.txt
01/21/2016  10:42 AM           100,352 text_file_example.exe
               2 File(s)        100,443 bytes
               2 Dir(s)  382,923,309,056 bytes free

C:\test>type test.txt
Here is the first line of text
Here is the second line
Here is the third and final line

C:\test>
```

Output file is in the same directory as the exe file.

# Text File Output on Unix

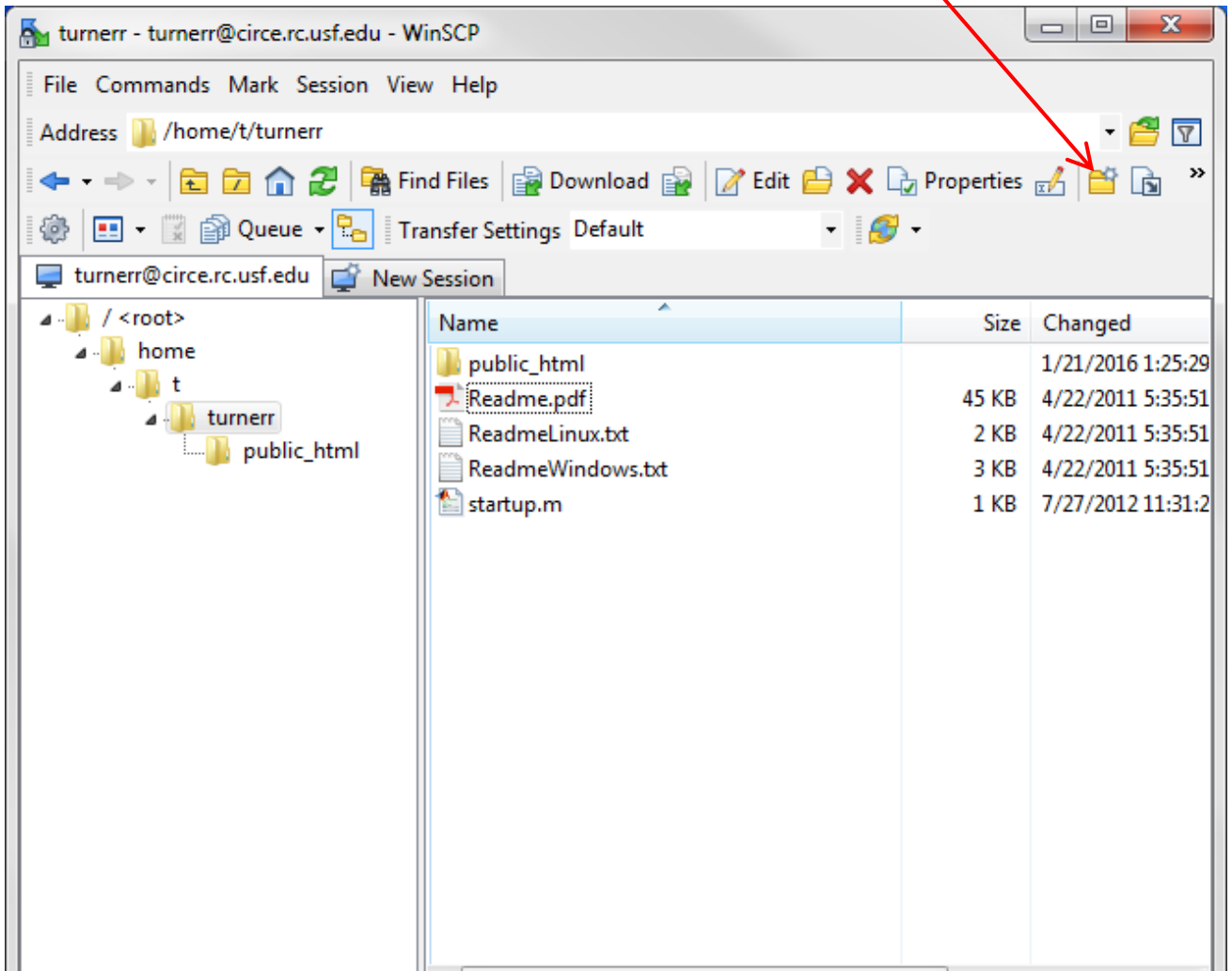
- Let’s try the same program on Unix.
- Copy the source file to a Unix directory.

# WinSCP


Login - WinSCP
New Site
turnerr@grad.cse.usf.edu
turnerr@turnerr.myweb.usf.edu
Session
File protocol:
SFTP
Host name:
circe.rc.usf.edu
Port number:
22
User name:
turnerr
Password:
Save
Advanced...
Tools
Manage
Login
Close
Help

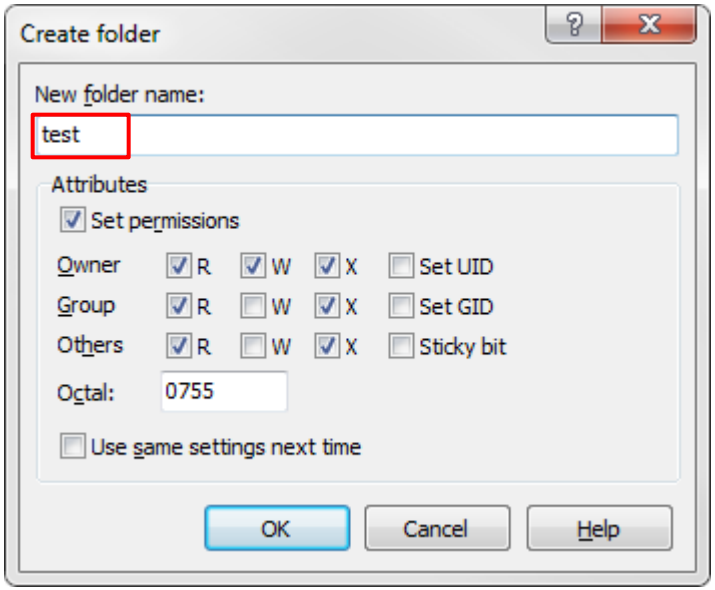
# Create Directory "test"

Click here to create a directory

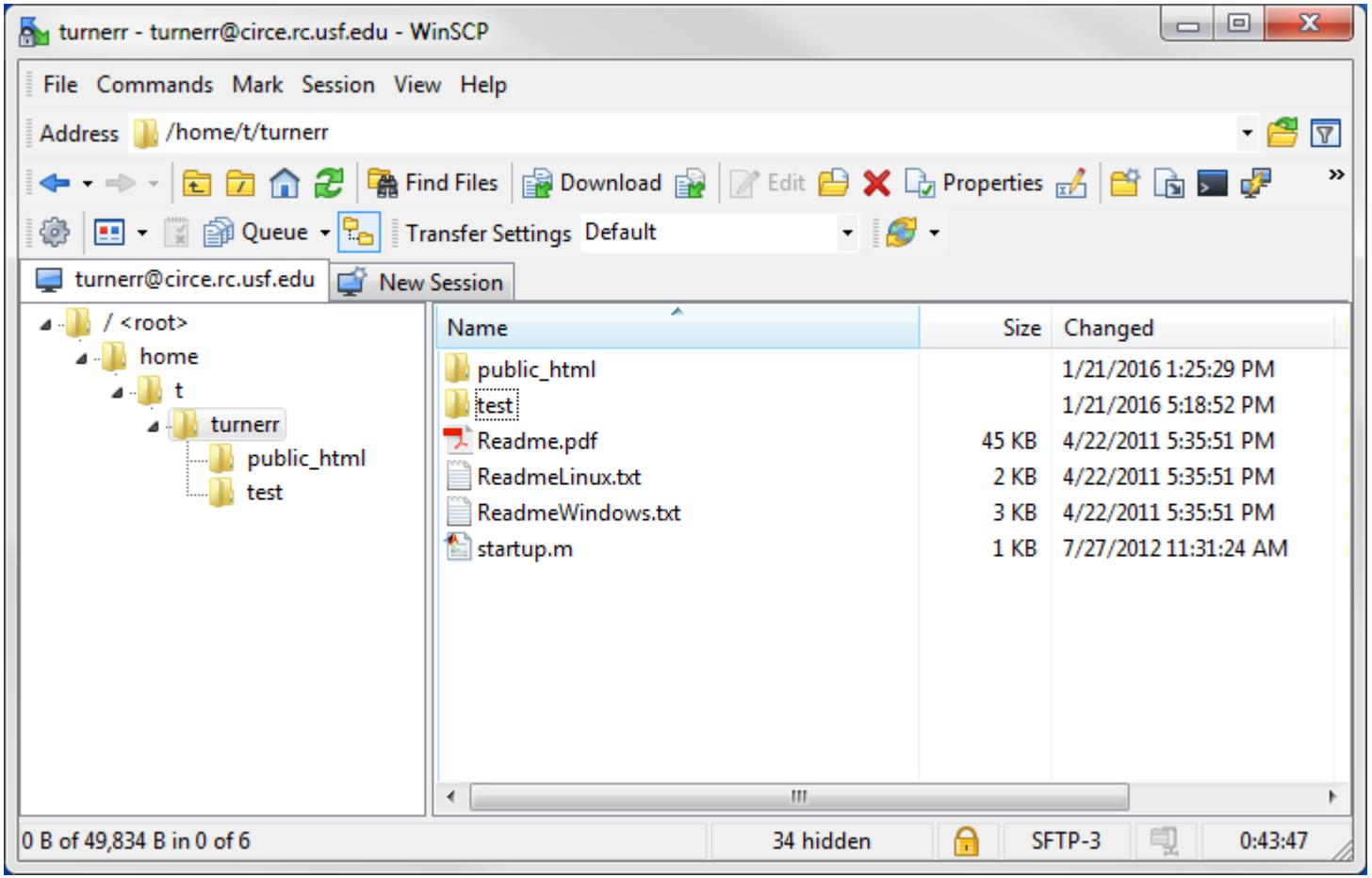


# Creating a new Directory on Circe

Note permissions



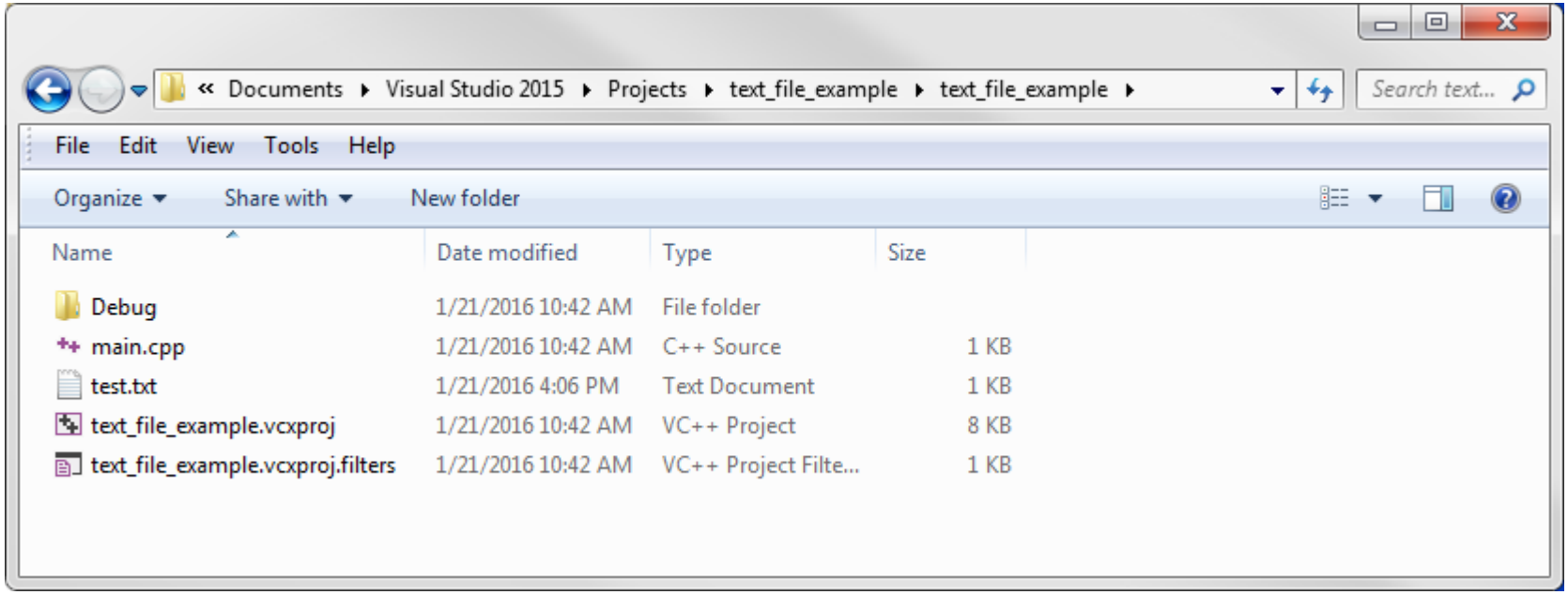
# Directory "test" is there now




Double click on test to open the directory.

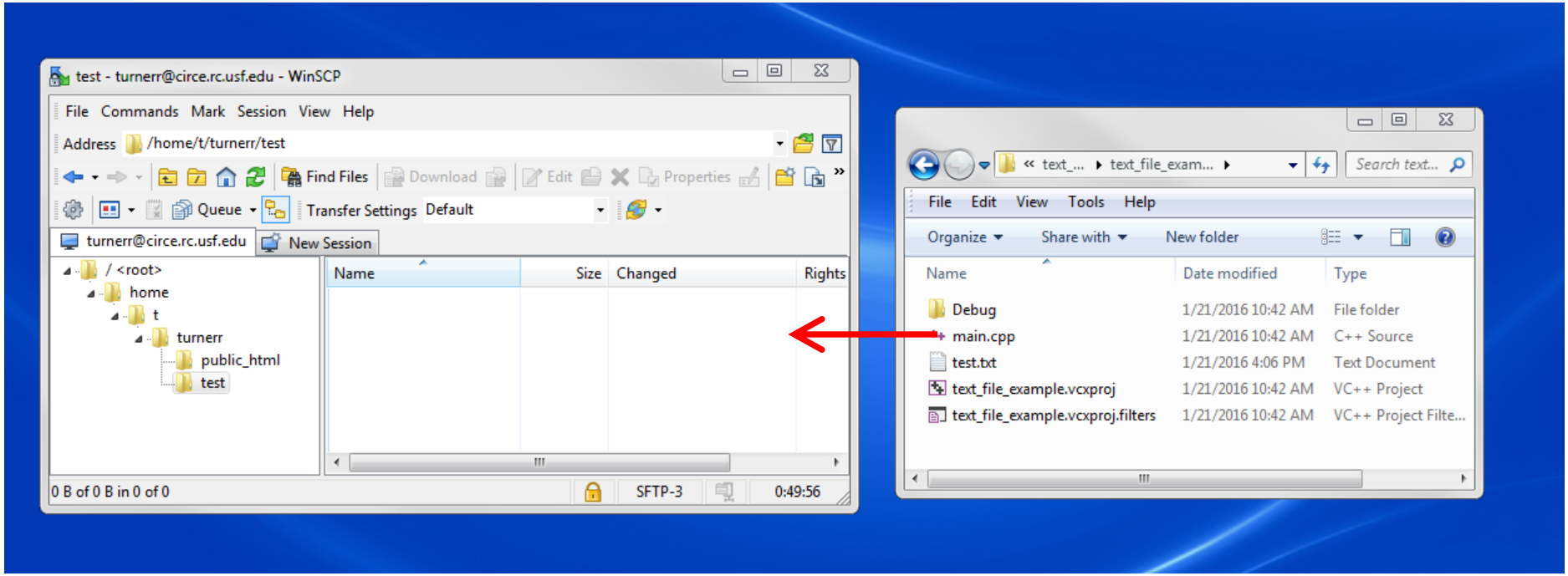
# Project Folder

In your project folder, navigate to main.cpp

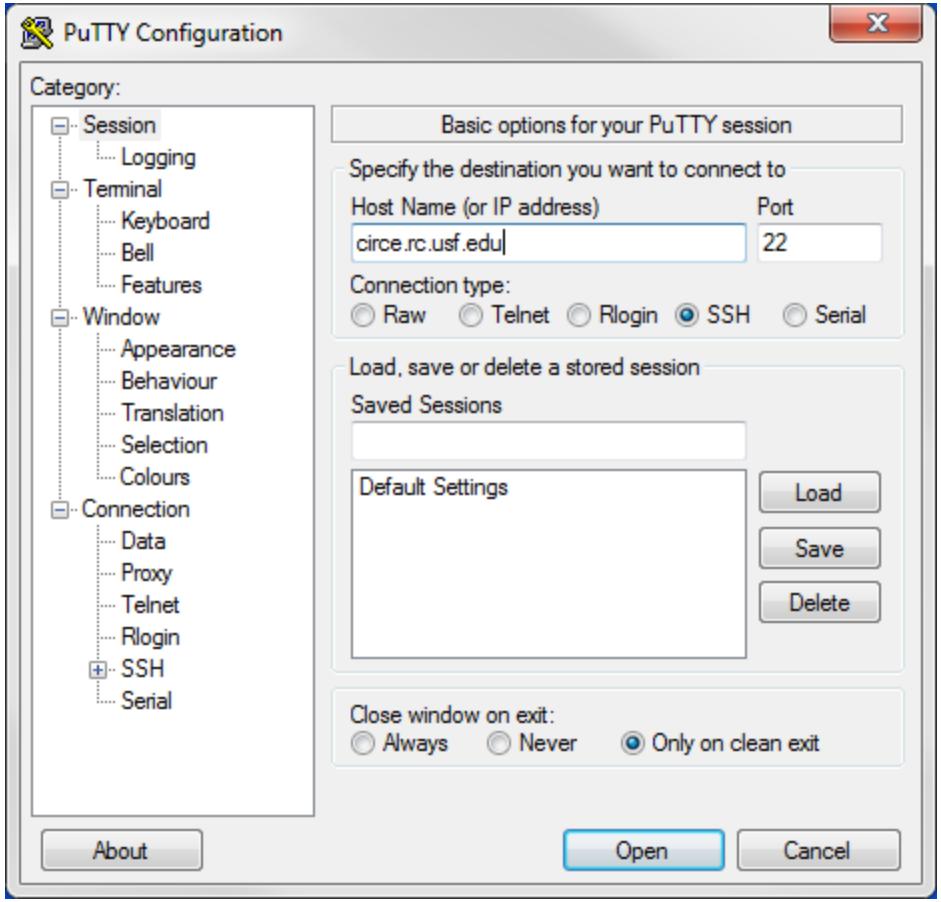


# Copy main.cpp to Circe

Drag main.cpp from your project folder
to the test directory in WinSCP



# Connect to Circe in PuTTY



# Connect to Circe in PuTTY

turnerr@login9:~

```
login as: turnerr

      University of South Florida Central Instructional and Research
                     Computing Environment (CIRCE)

     WARNING: UNAUTHORIZED ACCESS TO THIS SYSTEM IS PROHIBITED BY LAW

 By accessing the CIRCE cluster, you are agreeing to comply with the Terms of
 Service, USF's Privacy Policy, and any other guidelines posted on our website:
 https://cwa.rc.usf.edu/.

turnerr@circe.rc.usf.edu's password:
Last login: Mon Jan 11 12:13:44 2016 from 10.247.16.128

   Documentation:  https://cwa.rc.usf.edu/projects/research-computing/wiki
         Support:  help@usf.edu
           Phone:  813-974-1222

    To change your password or to recover a lost password, please visit
                           https://netid.usf.edu

    Please report any and all problems to the Helpdesk at help@usf.edu so
    we can track your incident!
```

# On Circe

turnerr@login9:~/test

```
[turnerr@login9 ~]$
[turnerr@login9 ~]$ cd test
[turnerr@login9 test]$ ls
main.cpp
[turnerr@login9 test]$
[turnerr@login9 test]$
[turnerr@login9 test]$
[turnerr@login9 test]$
[turnerr@login9 test]$
```

# Running on Circe

turnerr@login9:~/test

```
[turnerr@login9 test]$
[turnerr@login9 test]$ g++ main.cpp
[turnerr@login9 test]$ a.out
-bash: a.out: command not found
[turnerr@login9 test]$ ./a.out
This program writes a short text file called test.txt
File written

[turnerr@login9 test]$ cat test.txt
Here is the first line of text
Here is the second line
Here is the third and final line
[turnerr@login9 test]$
```

# Get Filename from Keyboard

- We often want the user to specify the file name.
- Get a file name from the keyboard
- Let's go back to Visual Studio on Windows.

```
#include <iostream>
#include <fstream>
#include <string>
int main()
{
    using namespace std;
    string outfilename;
    ofstream outfile;

    cout << "Name for output file? ";
    getline(cin, outfilename);
    outfile.open(outfilename);

    outfile << "Here is the first line of text\n";
    outfile << "Here is the second line\n";
    outfile << "Here is the third and final line\n";
    outfile.close();

    cout << "File " << outfilename << " written\n";
    cin.get();  // Keep window open;
    return 0;
}
```

# Program Running on Windows

C:\Users\turnerr\Documents\Visual Studio 2015\Projects\text_file_example\Debug\text_file_examp...

```
Name for output file? myfile.txt
File myfile.txt written
```

As before, the output file is in the project directory.

« text_file_example ▸ text_file_example ▸ Search text...

File Edit View Tools Help

Organize ▾ Open ▾ Share with ▾ Print E-mail »

| Name | Date modified | Type | Size |
|---|---|---|---|
| Debug | 1/21/2016 5:47 PM | File folder | |
| main.cpp | 1/21/2016 5:47 PM | C++ Source | 1 KB |
| myfile.txt | 1/21/2016 5:47 PM | Text Document | 1 KB |
| test.txt | 1/21/2016 4:06 PM | Text Document | 1 KB |
| text_file_example.vcxproj | 1/21/2016 10:42 AM | VC++ Project | 8 KB |
| text_file_example.vcxproj.filters | 1/21/2016 10:42 AM | VC++ Project Filte... | 1 KB |

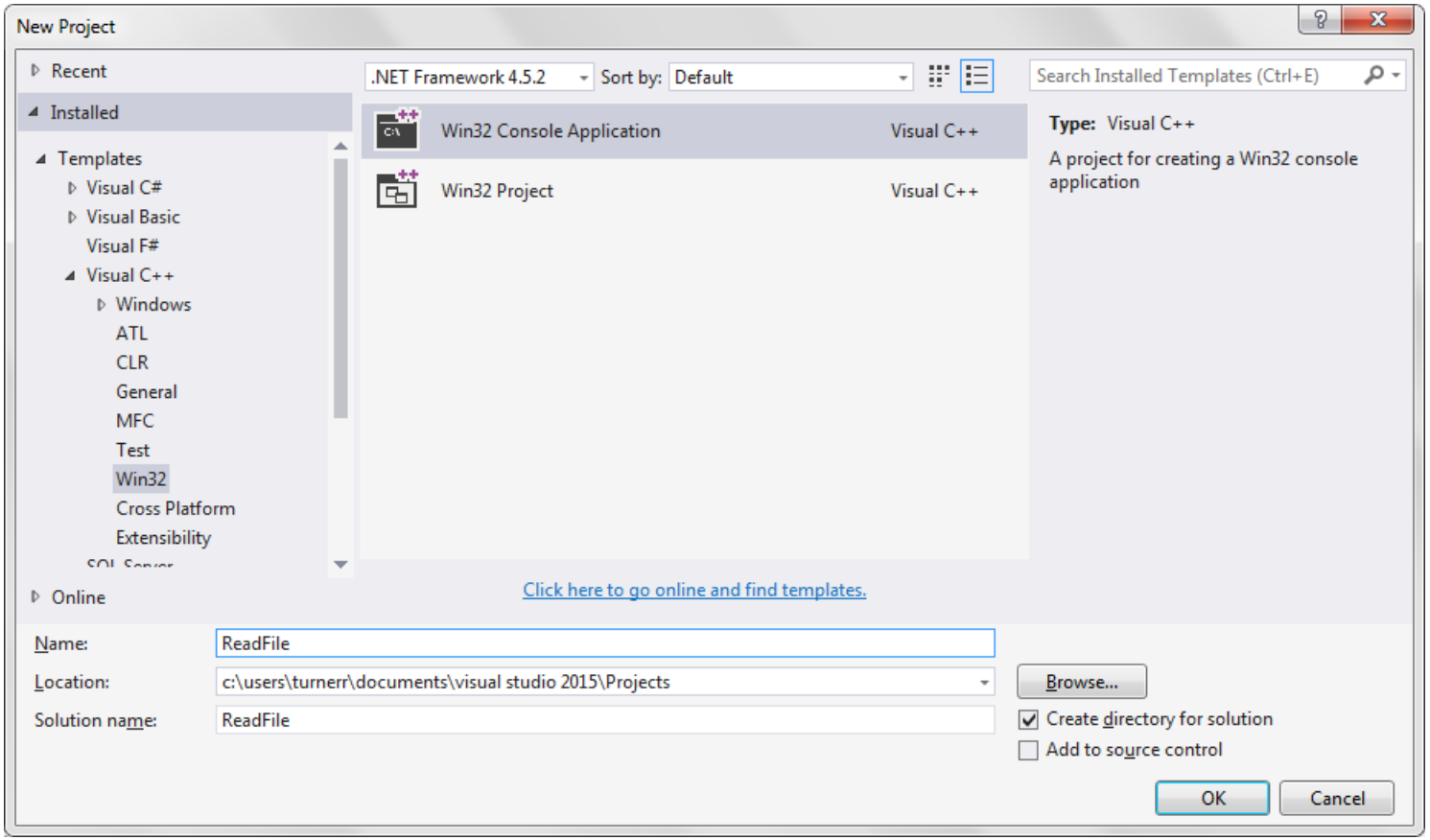
End of Section

# Reading a Text File

- #include <fstream>
- Instantiate an **`ifstream`** object
- Call methods of the object to do input.

- Create a new project for this example.

# Text File Input Example



# Reading a Text File

```
#include <iostream>
#include <fstream>  // for file I/O
#include <cstdlib>  // for exit()
#include <string>

int main()
{
    using namespace std;
    string infilename;
    ifstream infile;
    string input_line;

    cout << "File Input Example\n";
    cout << "Name of file to read: ";
    getline(cin, infilename);

    cout << "Opening file " << infilename << endl;
    infile.open(infilename);
```

# Reading a Text File

Always check for success of fopen

```
if (!infile.is_open())
{
    cout << "Failed to open file\n";
    cin.get();  // Keep window open.
    cin.get();
    exit (EXIT_FAILURE);
}

// Input file is open
while (infile.good())
{
    infile >> input_line;
    cout << "Next line is: " << input_line << endl;
}
```

Read the file.

Continue here when input fails (EOF or error)

# Reading a Text File

Check cause of input failure

```
    if (infile.eof())
    {
        cout << endl << "End of file \n";
    }
    else
    {
        cout << endl  << "Error reading file\n";
    }

    infile.close();
    cout << "Normal termination\n";
    cin.get();       // Keep window open.
    cin.get();
    return 0;
}
```

Good!

Bad!

# We need a file to read.

- Copy myfile.txt into project directory
  - Beside main.cpp

# Input Example Running

c:\users\turnerr\documents\visual studio 2015\Projects\ReadFile\Debug\ReadFile.exe

```
File Input Example
Name of file to read: myfile.txt
Opening file myfile.txt
Next line is: Here
Next line is: is
Next line is: the
Next line is: first
Next line is: line
Next line is: of
Next line is: text
Next line is: Here
Next line is: is
Next line is: the
Next line is: second
Next line is: line
Next line is: Here
Next line is: is
Next line is: the
Next line is: third
Next line is: and
Next line is: final
Next line is: line
Next line is: line

End of file
Normal termination
```

# Observations

- >> for file input works just like it does for keyboard input.
  - Read the next word.
  - Terminate input at whitespace.
- Probably not what was intended
- To read an entire line, use **`getline()`**

# Changes to Read Entire Line

```
...

// Input file is open
while (infile.good())
{
    //infile >> input_line;
    getline(infile, input_line);
    cout << "Next line is: " << input_line << endl;
}
```

# File Input Example Running

Now using getline() method

c:\users\turnerr\documents\visual studio 2015\Projects\ReadFile\Debug\ReadFile.exe

```
File Input Example
Name of file to read: myfile.txt
Opening file myfile.txt
Next line is: Here is the first line of text
Next line is: Here is the second line
Next line is: Here is the third and final line
Next line is:

End of file
Normal termination
```

Note final blank line

End of Section

# Reading Numbers from a File

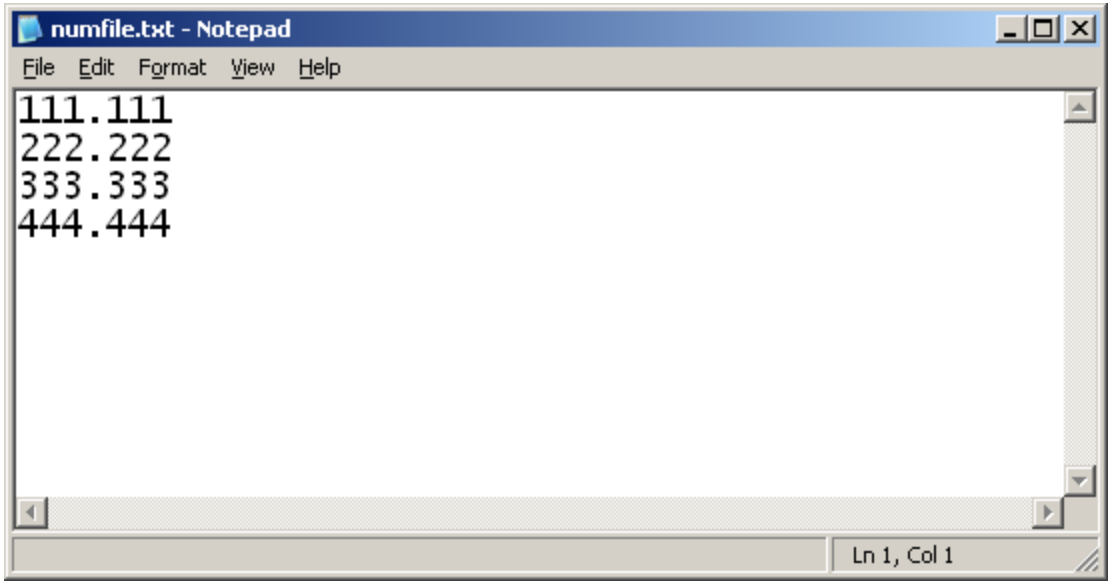
- >> works as expected
  - More or less
  - So long as file format is what was expected.

# Reading Numbers from a File

## Change for numeric input

```
// Input file is open
while (infile.good())
{
    double value;
    infile >> value;
    cout << "Next input is: " << value << endl;
}
```

# The Input File



# Program read_file_numeric Running

c:\users\turnerr\documents\visual studio 2015\Projects\ReadFile\Debug\R...

```
File Input Example
Name of file to read: numfile.txt
Opening file numfile.txt
Next input is: 111.111
Next input is: 222.222
Next input is: 333.333
Next input is: 444.444
Next input is: 444.444

End of file
Normal termination
```

What is this?

# Reading Numbers from a File

```
// Input file is open
while (infile.good())
{
    double value;
    infile >> value;
    cout << "Next input is: " << value << endl;
}
```

value is unchanged when EOF is reached.

# Avoiding Bogus Input at EOF

```
while (infile.good())
{
    double value;
    infile >> value;
    if (!infile.good())
    {
        break;
    }
    cout << "Next input is: " << value << endl;
}
```

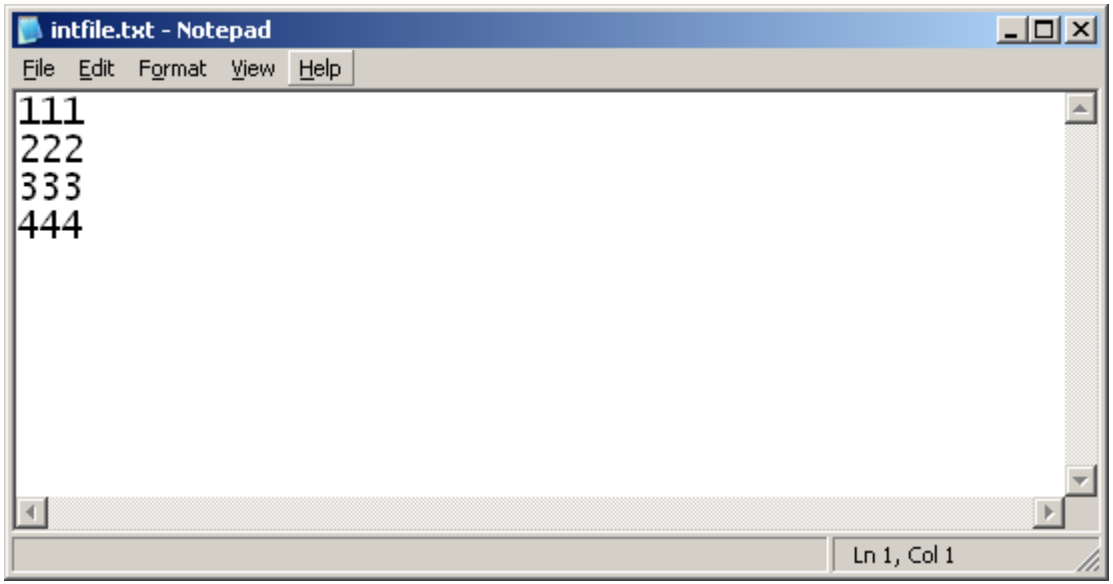
Check for error or EOF *after* attempting input and *before* using the result

# Revised Program Running

c:\users\turnerr\documents\visual studio 2015\Projects\ReadFile\Debug\ReadFile.exe

```
File Input Example
Name of file to read: numfile.txt
Opening file numfile.txt
Next input is: 111.111
Next input is: 222.222
Next input is: 333.333
Next input is: 444.444

End of file
Normal termination
```

# Reading Integers from a Text File




Integer Input File

# Reading Integers from a Text File

```
// Input file is open
while (infile.good())
{
    int value;
    infile >> value;
    if (!infile.good())
    {
        break;
    }
    cout << "Next input is: " << value << endl;
}
```

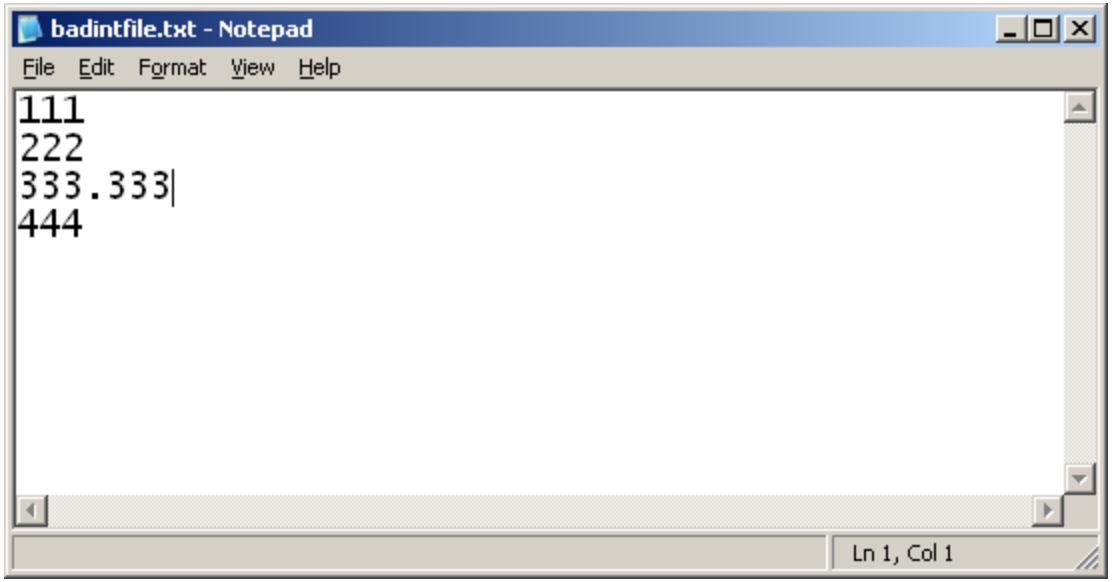
Only change

# Integer File Input Example Running

c:\users\turnerr\documents\visual studio 2015\Projects\ReadFile\Debug\ReadFile.exe

```
File Input Example
Name of file to read: intfile.txt
Opening file intfile.txt
Next input is: 111
Next input is: 222
Next input is: 333
Next input is: 444

End of file
Normal termination
```

# What If Input Is Not As Expected?



# Attempting to Read Bad Integer Input

c:\users\turnerr\documents\visual studio 2015\Projects\ReadFile\Debug\ReadFile.exe

```
File Input Example
Name of file to read: badintfile.txt
Opening file badintfile.txt
Next input is: 111
Next input is: 222
Next input is: 333

Error reading file
Normal termination
```

# Input Failure

- Invalid input puts the stream in the *fail* state.
- No further input will succeed unless we reset the state.
  - Can hang in an endless loop if we just try to continue.
- Typically the best policy is to abort the program.
- Recovery is possible when it makes sense.

```
// Input file is open
while (infile.good())
{
    int value;
    infile >> value;
    if (infile.eof())
    {
        break;
    }
    if (infile.fail())
    {
        cout << "Invalid data in input file\n";
        string input_line;
        infile.clear();
        getline(infile,input_line);
        cout << input_line << endl;
        cout << "Continuing to read \n\n";
        continue;
    }
    cout << "Next input is: " << value << endl;
}
```

Check for EOF first

Check for other errors

Clear the "fail" state

Skip over rest of line

Continue reading

# Recovering from Input Failure

c:\users\turnerr\documents\visual studio 2015\Projects\ReadFile\Debug\ReadFile.exe

```
File Input Example
Name of file to read: badintfile.txt
Opening file badintfile.txt
Next input is: 111
Next input is: 222
Next input is: 333
Invalid data in input file
.333
Continuing to read

Next input is: 444

End of file
Normal termination
```

# Summary

- File I/O in C++ is similar to console I/O
- Instantiate object to read or write.
  - Call its methods to do I/O
  - << and >> work as with cout and cin.
- Check for EOF and errors on input
  - Error handling is tricky!

# Assignment

- Read about file I/O in the text
  - Chapter 6, page 287 – 298
- Try the examples from this presentation for yourself.