EECE 2560: Fundamentals of Engineering Algorithms

C++ Programming Environment

Typical C++ Development Environment

- C++ systems generally consist of three parts:
  1. a program development environment,
  2. the language and
  3. the C++ Standard Library.
- C++ programs typically go through six phases:
  1. edit,
  2. preprocess,
  3. compile,
  4. link,
  5. load and
  6. execute.

Phase 1: Edit

- Phase 1 consists of editing a file with an *editor* program.
  - Type a C++ program (source code) using the editor.
  - Make any necessary corrections.
  - Save the program.
  - C++ source code filenames often end with the .cpp, .cxx, .cc or .C extensions, which indicate that a file contains C++ source code.
- Linux editors:
  - `vi` (Vim is improved Vi)
    - http://yannesposito.com/Scratch/en/blog/Learn-Vim-Progressively/
    - Vim Cheat Sheet: https://vim.rtorr.com/
  - `nano:` http://www.nano-editor.org/dist/v2.2/nano.html
  - `emacs:` http://www.gnu.org/software/emacs/tour/
- You can also use a simple text editor, such as Notepad or Notepad++ in Windows.
  - Notepad++ (a free source code editor for Windows. It supports several programming languages) https://notepad-plus-plus.org/download/

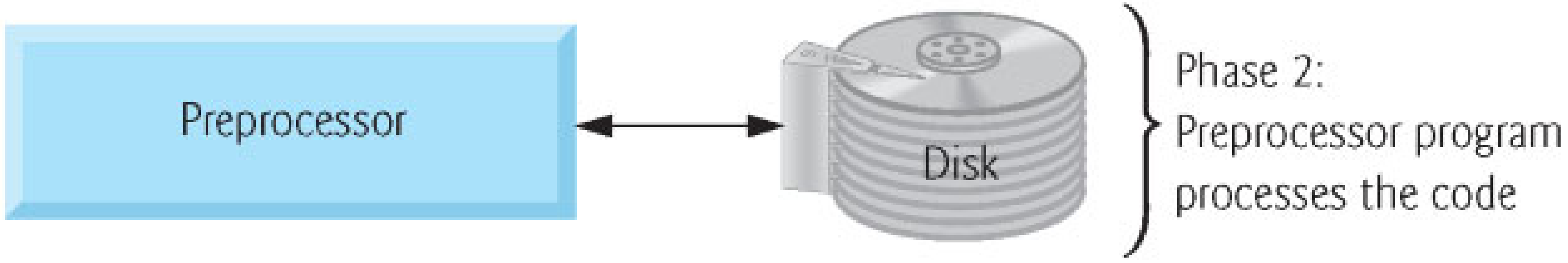
Integrated Development Environments

- IDEs provide tools that support the software-development process, including editors for writing and editing programs and debuggers for locating logic errors—errors that cause programs to execute incorrectly.

- Popular IDEs
  - Microsoft® Visual Studio
  - NetBeans
  - Eclipse
  - Apple's Xcode
  - CodeLite
  - Clion

Phase 2: Preprocess

- The C++ preprocessor obeys commands called preprocessing directives, which indicate that certain manipulations are to be performed on the program before compilation.
- These manipulations usually include (copy into the program file) other text files to be compiled, and perform various text replacements.

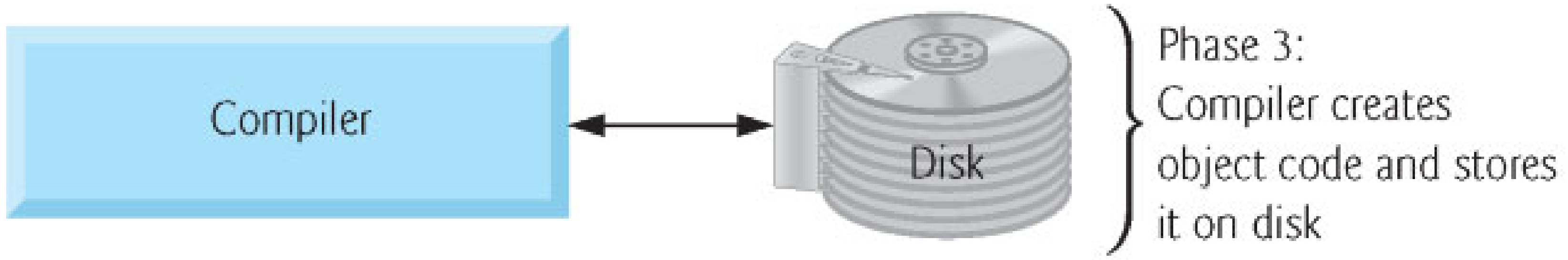


Preprocessing Directive

- A preprocessing directive is a message to the C++ preprocessor.
- Lines that begin with # are processed by the preprocessor before the program is compiled.
- `#include <iostream>` notifies the preprocessor to include in the program the contents of the input/output stream header file <iostream>.
  - This header is a file containing information used by the compiler when compiling any program that outputs data to the screen or inputs data from the keyboard.
- The std:: before cout is required when we use names that we have brought into the program by the preprocessing directive #include <iostream>.

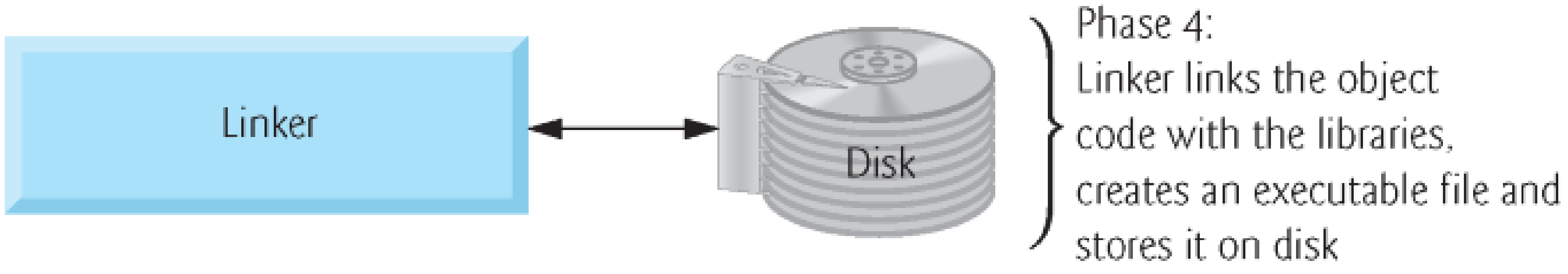
Phase 3: Compile

- The compiler translates the C++ program into machine-language code—also referred to as *object code*.



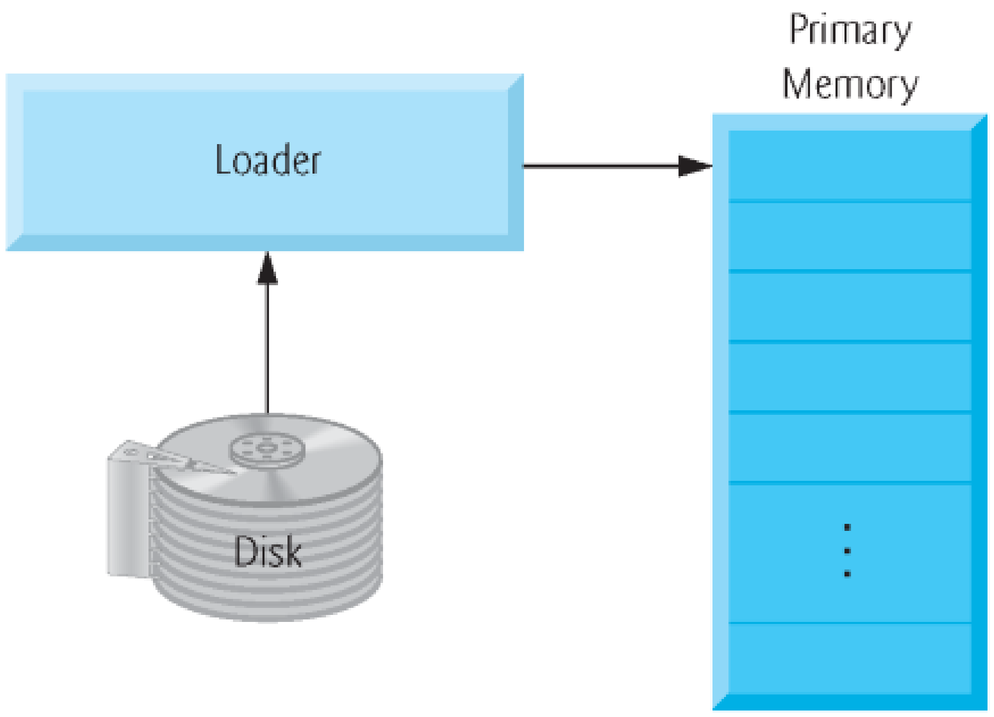
Phase 4: Link

- The object code produced by the C++ compiler typically does not include the object code of the called libraries (code written and compiled separately by you, other programmers, or provided by the language).
- A linker links your program object code with the object code of those missing libraries to produce an executable program.
- If the program compiles and links correctly, an executable file is produced.


Linker
Disk
Phase 4:
Linker links the object code with the libraries, creates an executable file and stores it on disk

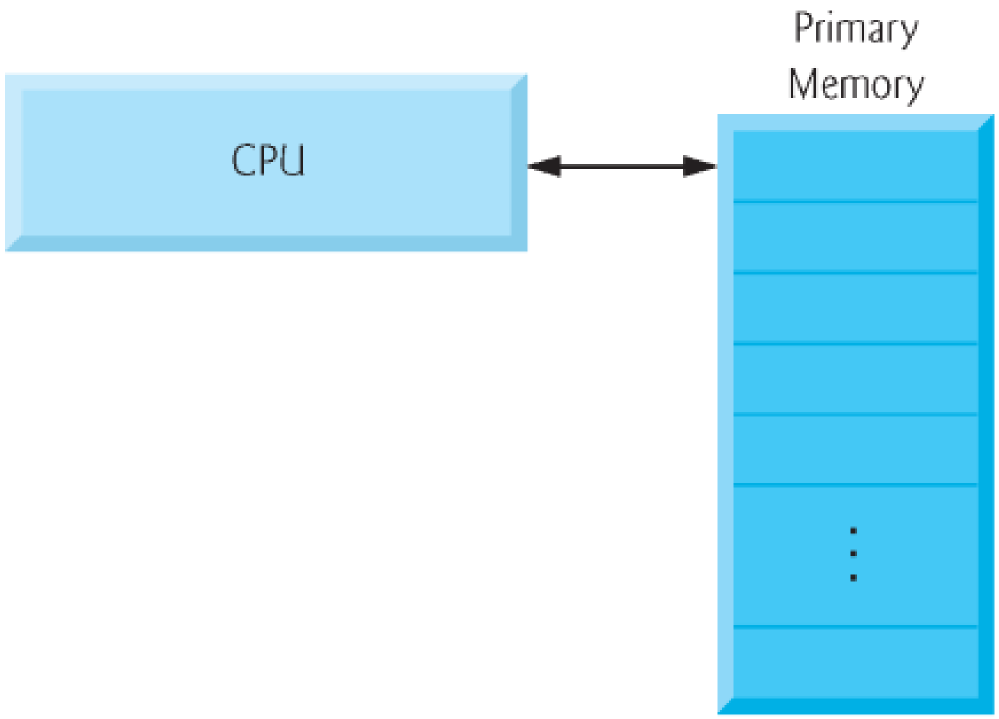
Phase 5: Load

- Before a program can be executed, it must first be placed in memory.
- This is done by the loader, which takes the executable image from disk and transfers it to memory.
- Additional components from shared libraries that support the program are also loaded.



Phase 6: Execute

- Finally, the computer, under the control of its CPU, executes the program one instruction at a time.
- Some modern computer architectures often execute several instructions
in parallel.



Source: Prof. Kimani, Northeastern University

Remote Access to the COE Lab

- SSH
  - "secure shell": a network protocol for secure data communication
  - Typically used to log into a remote machine and execute commands in command-line
- SSH tools
  - In Windows: Putty, MobaXTerm
  - In Linux/Mac: ssh is already installed, can be used directly from a terminal
- If you do not have access to a COE Linux Account, check: https://coe.northeastern.edu/computer/

Access COE Account with VPN

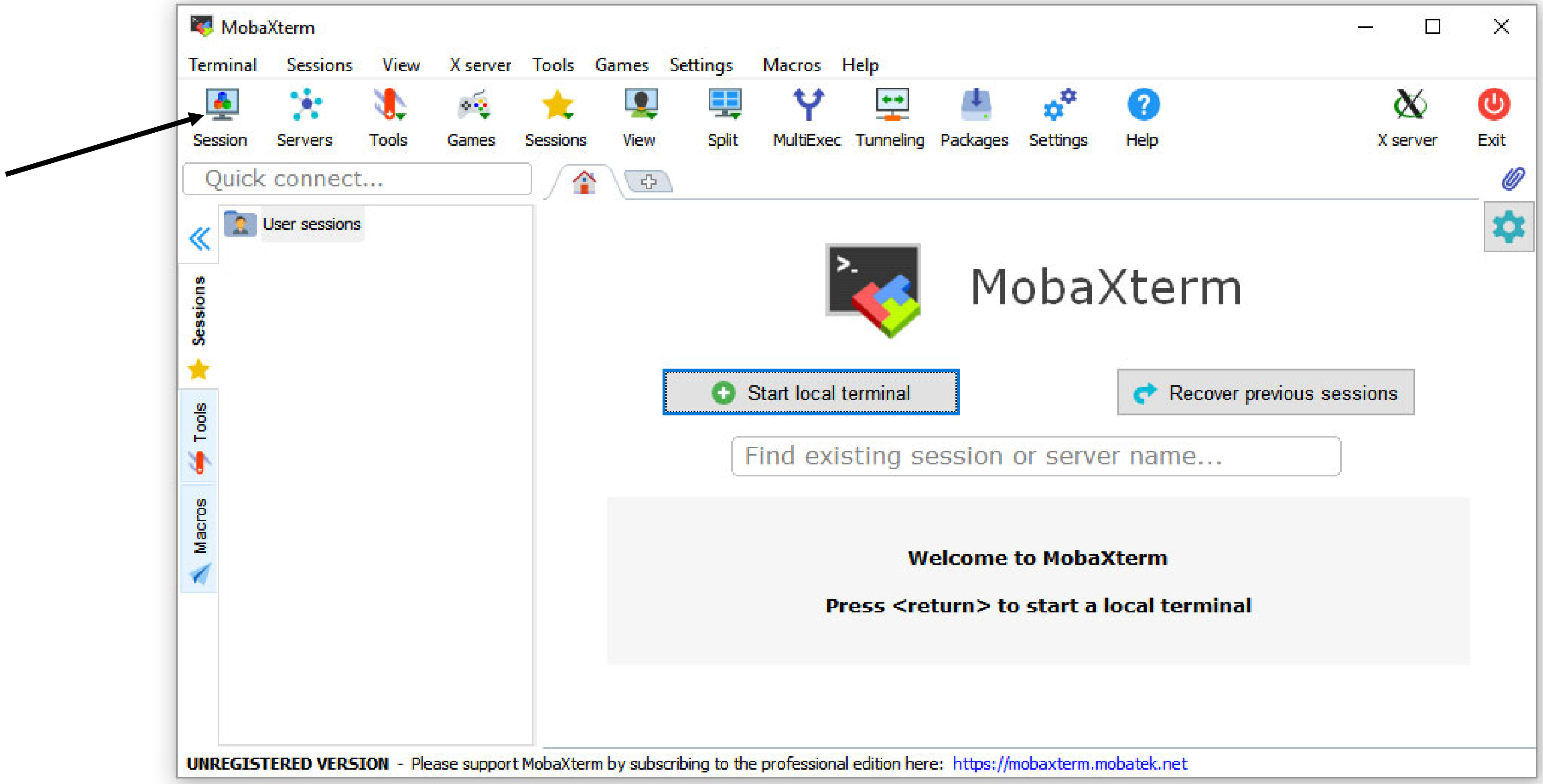
- You will need to use VPN (virtual private network) to access the COE Linux account from outside the Northeastern network.

- The following link has the resources you need to do that:

https://northeastern.service-now.com/tech?id=kb_category&kb_category=07d42f714f02cf0099c2fd511310c7b2

MobaXTerm Download

- Download MobaXTerm Home (Personal) Edition from: https://mobaxterm.mobatek.net/download-home-edition.html
- Unzip and run the executable file to start MobaXTerm.
  - Click on Session to open a new session.


MobaXterm
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help X server Exit
Quick connect...
User sessions
Sessions
Tools
Macros
MobaXterm
Start local terminal
Recover previous sessions
Find existing session or server name...

Welcome to MobaXterm

Press <return> to start a local terminal

UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

Login to COE Linux Account

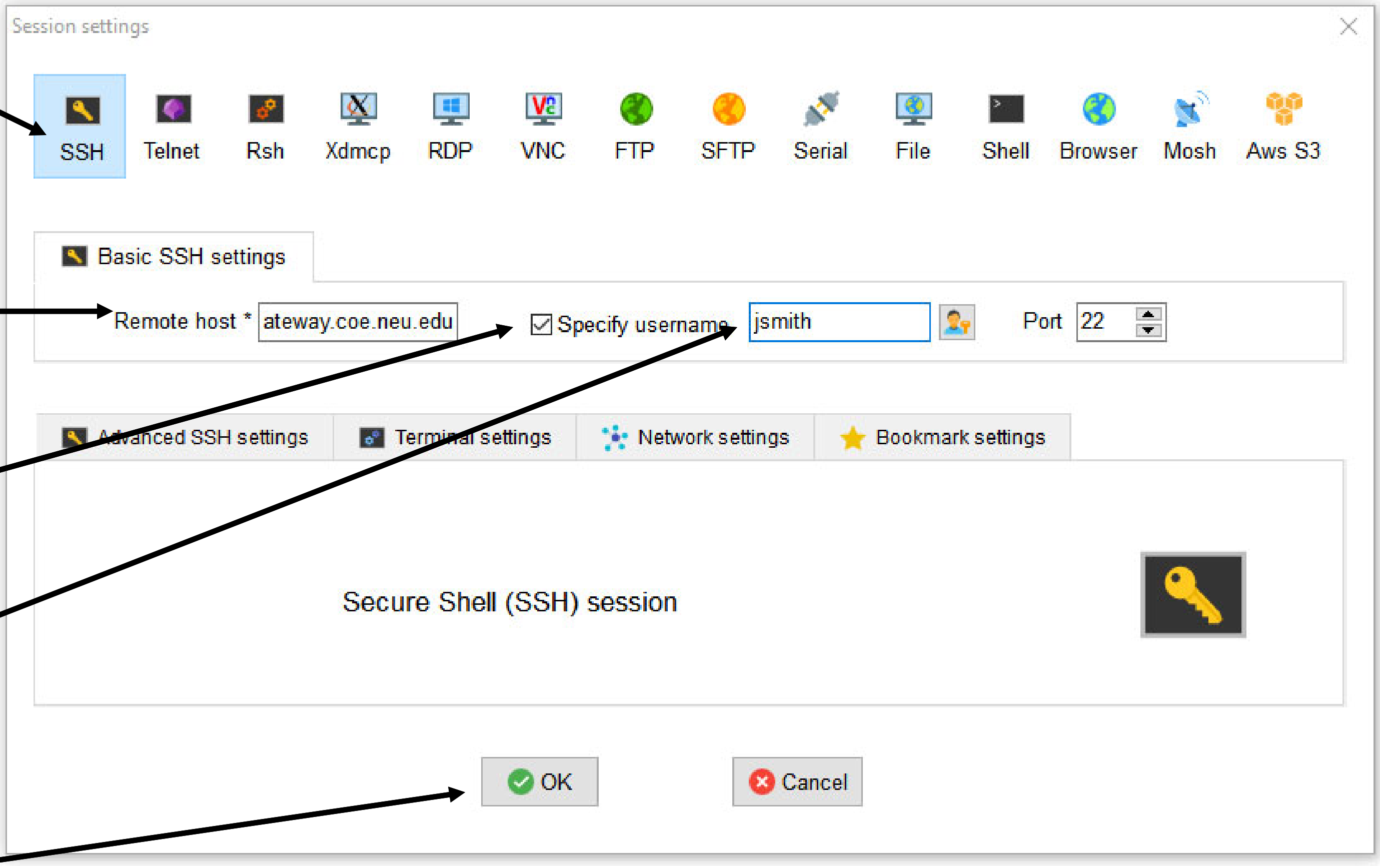
1. Select **SSH**

2. Enter "gateway.coe.neu.edu"

3. Check to specify your username

4. Enter your username

5. Click **OK**


Session settings

Remote host * ateway.coe.neu.edu
☑ Specify username jsmith
Port 22

Secure Shell (SSH) session

6. Respond with your password when requested

GCC

- We will use g++ as the GCC compiler to run the C++ programs.

- Examples of using g++
  - `g++ myprogram.cc`
    (generates an `a.out` executable binary, `./a.out` to run it)
  - `g++ myprogram.cc -o myprogram.exe`
    (generates `myprogram.exe` instead of the default `a.out`)
  - `g++ -S myprogram.cc`
    (generates assembly code in a the `myprogram.s` file)

C++11 and C++14

- On the Linux server to compile a program using the C++ version 11:

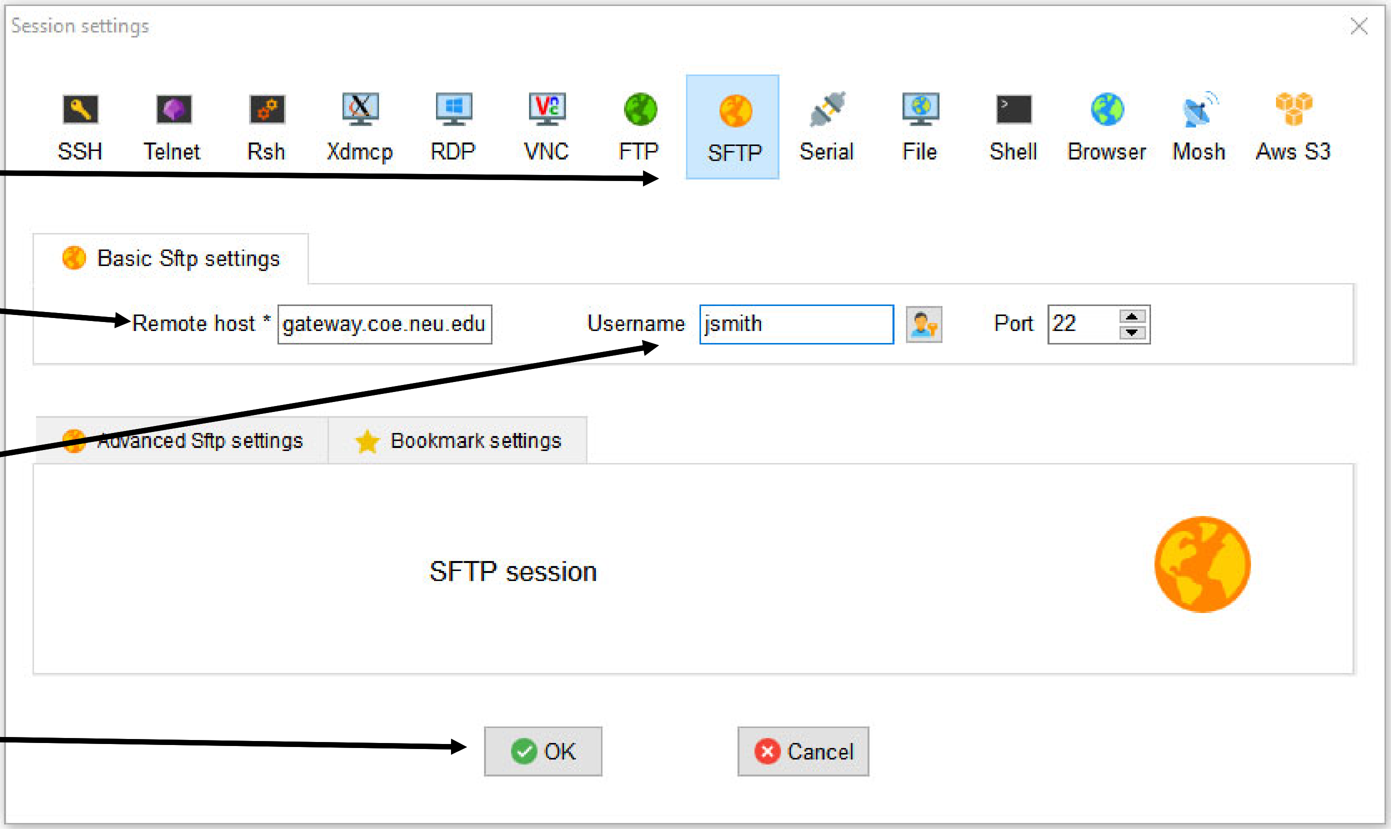
```
g++ -std=c++11 myprogram.cpp
```

- For C++ version 14 (the following first command is to enable the required version of the GNU Compiler):

```
scl enable devtoolset-7 bash
g++ -std=c++14 myprogram.cpp
```

Transferring Files

- You will need to use the SFTP service to transfer, or edit, files between your local machine and a COE lab machine. In MobaXterr



Remote host * gateway.coe.neu.edu
Username jsmith
Port 22

1. Select **SFTP**

2. Enter "gateway.coe.neu.edu"

3. Enter your username

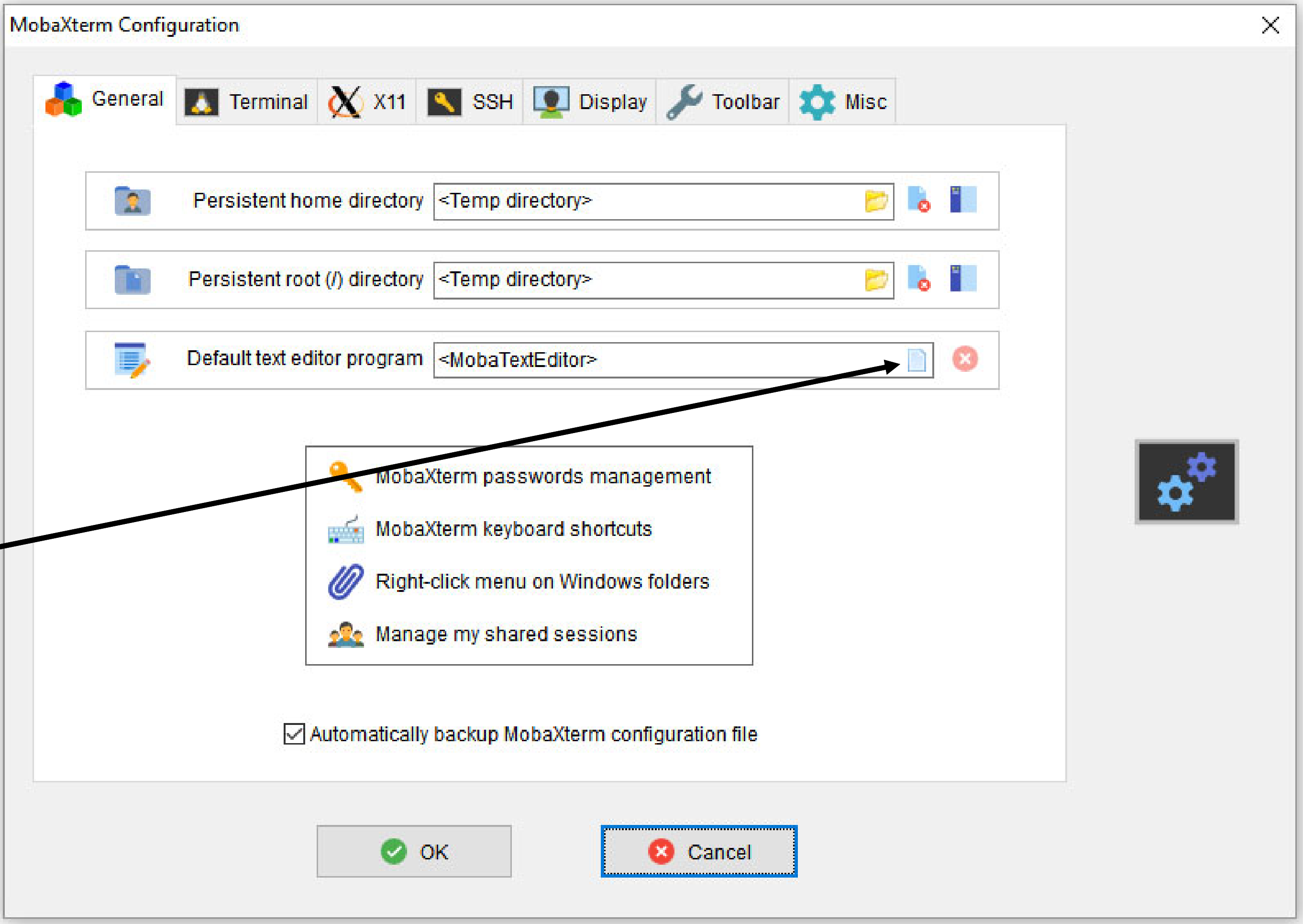
4. Click **OK**

5. Respond with your password if requested

MobaXterm Default Text Editor

- You can remotely edit text files by right clicking on the file name (on the SFTP browser) and select **Open with Moba TextEditor**.

- To change the default MobaXterm editor, select **Settings → Configuration → General →** Browse to your favorite editor here then click **Open → Ok.**



MobaXterm Configuration

General | Terminal | X11 | SSH | Display | Toolbar | Misc

Persistent home directory <Temp directory>

Persistent root (/) directory <Temp directory>

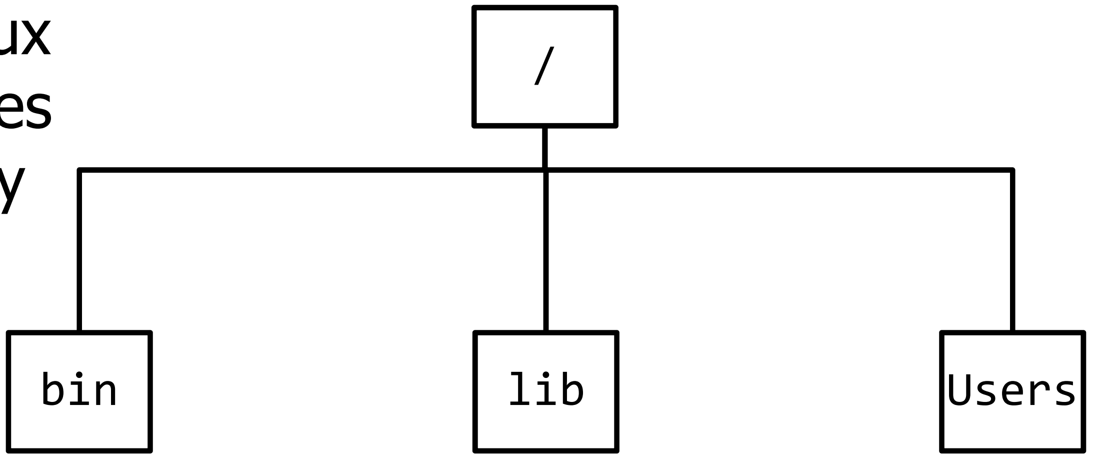
Default text editor program <MobaTextEditor>

MobaXterm passwords management
MobaXterm keyboard shortcuts
Right-click menu on Windows folders
Manage my shared sessions
OK
Cancel

☑ Automatically backup MobaXterm configuration file

Linux Commands

- After logging in, you can use the Shell to pass commands to the Linux operating system to carry out.
- Linux commands and file names are case sensitive.
- You can press the "`Tab`" key to automatically fills in partially typed commands.
- Pressing `Ctrl + c` aborts current task and regain user control.
- The file system in Linux arranges the directories and files in a hierarchy structure as shown:



General Commands

- `who` - shows who is connected to the server
- `pwd` (printing working directory) - shows the path of the current directory.
- `ps` (process status) - shows running processes on current machine along with their process identifier, PID
- `kill PID` – terminates a process. If ignored and to enforce killing of a process use `kill -s KILL PID`
- `ls` (list) - list files and directories
- `man <command>` (manual) - display the manual of the command
- `whereis <command>` - locate the commands
- `exit` - close ssh connection

Navigation Commands

- `cd` (change directory)
  - `cd ..` (change to parent directory)
  - `cd ~` (change to home directory)
  - `cd /` (change to root directory)
- `ls` (list files and directories)
  - `ls -a` (list all entries)
  - `ls -l` (list in long format)
  - `ls -al` (options can be combined)
- `find` (locate files or directories)
  - `find / -name file-name.txt`

File and Directory Commands

- `mkdir` (make directory)
  - `mkdir <directory-name>`
- `mv` (move): move file/directory, rename
  - `mv <old-file> <new-file>`
  - `mv <old-directory> <new-directory>`
  - `mv <file> <directory>`
- `touch`: create an empty file
  - `touch [option] file_name(s)`
  - `touch circle.c rectangle.cc` *(create two new, empty files)*
- `rm` (remove): delete file or directory
  - `rm <file-name>`
  - `rm -r <directory-name>` *(delete directory recursively)*
- `cp` (copy file and directories)
  - `cp hw01.txt newhw.txt`
  - `cp ./HW/* ~/HW-backup` *(copy all files to a directory)*
  - `cp -r ./HW/* ~/HW-backup` *(copy files and directories recursively)*

Source: Prof. Kimani, Northeastern University

Example Task

- To find your current path use "pwd"

```
$ pwd
 /users/Faculty/jsmith
```

- To create a directory use "mkdir"

```
$ mkdir EECE2560
```

- To change to a specific directory use "cd"

```
$ cd EECE2560
$ pwd
/users/Faculty/jsmith/EECE2560
```

- To change to your home directory use "cd"

```
$ cd
$ pwd
/users/Faculty/jsmith
```

- To move up one directory level use ".."

```
$ cd ..
$ pwd
/users/Faculty
```

Visual Studio for Windows

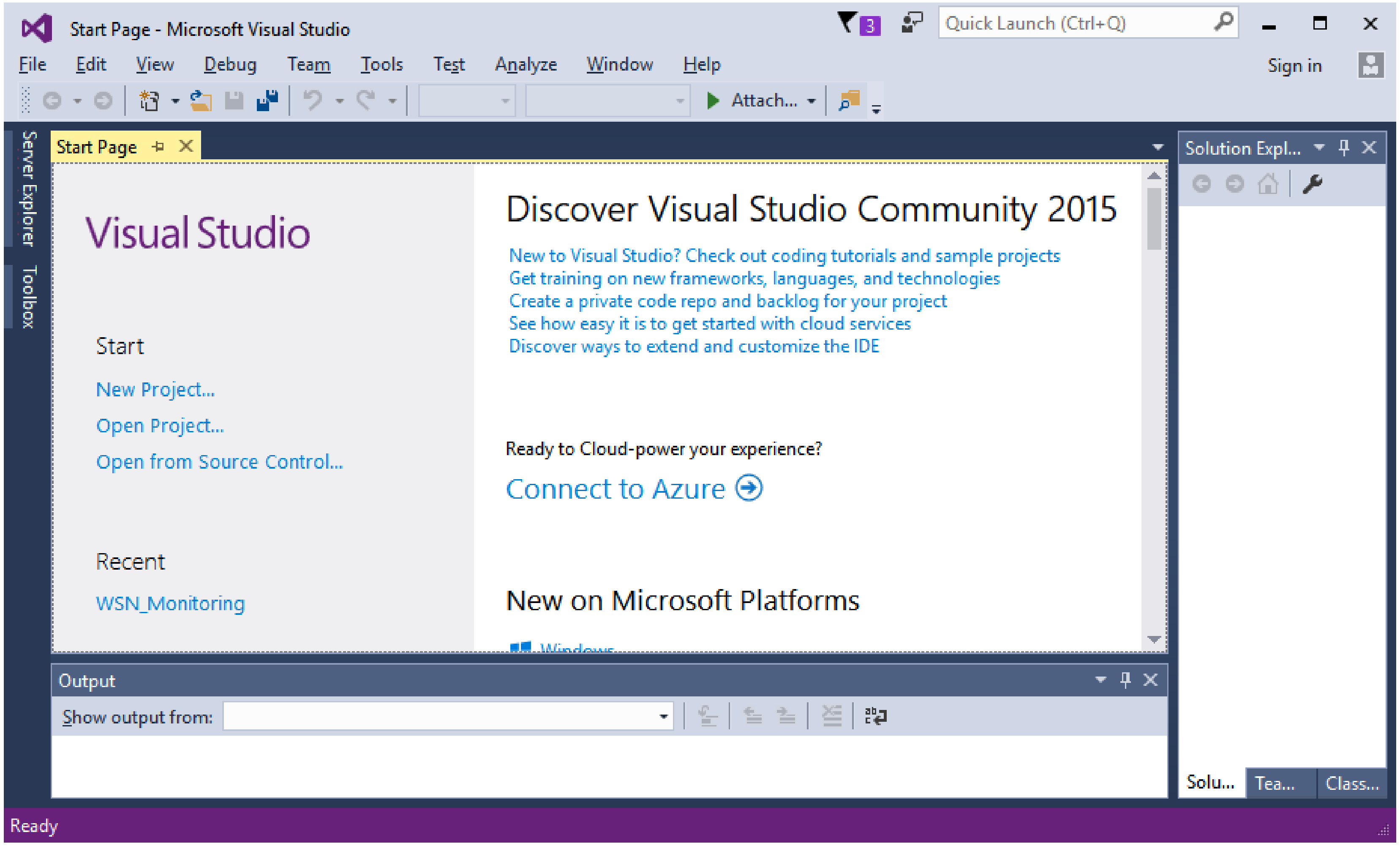
- There are several versions of Visual Studio available—on some versions, the options, menus and instructions we present might differ slightly. From this point forward, we will refer to Visual Studio Community Edition simply as “Visual Studio” or “the IDE.”

- Get the free version “*Visual Studio Community*” from: https://visualstudio.microsoft.com/vs/community/

Starting Visual Studio

- Open Visual Studio from the **Start** menu.
- The IDE displays the Start Page, which provides links for creating new programs, opening existing programs and learning about the IDE and various programming topics.
- Close this window for now by pressing the **X** in its tab—you can access this window any time by selecting **View → Start**

Visual Studio Start Page


Start Page - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Debug Team Tools Test Analyze Window Help
Sign in
Attach...
Start Page
Server Explorer
Toolbox
Solution Expl...
Visual Studio
Start
New Project...
Open Project...
Open from Source Control...
Recent
WSN_Monitoring
Discover Visual Studio Community 2015
New to Visual Studio? Check out coding tutorials and sample projects
Get training on new frameworks, languages, and technologies
Create a private code repo and backlog for your project
See how easy it is to get started with cloud services
Discover ways to extend and customize the IDE
Ready to Cloud-power your experience?
Connect to Azure
New on Microsoft Platforms
Output
Show output from:
Solu... Tea... Class...
Ready

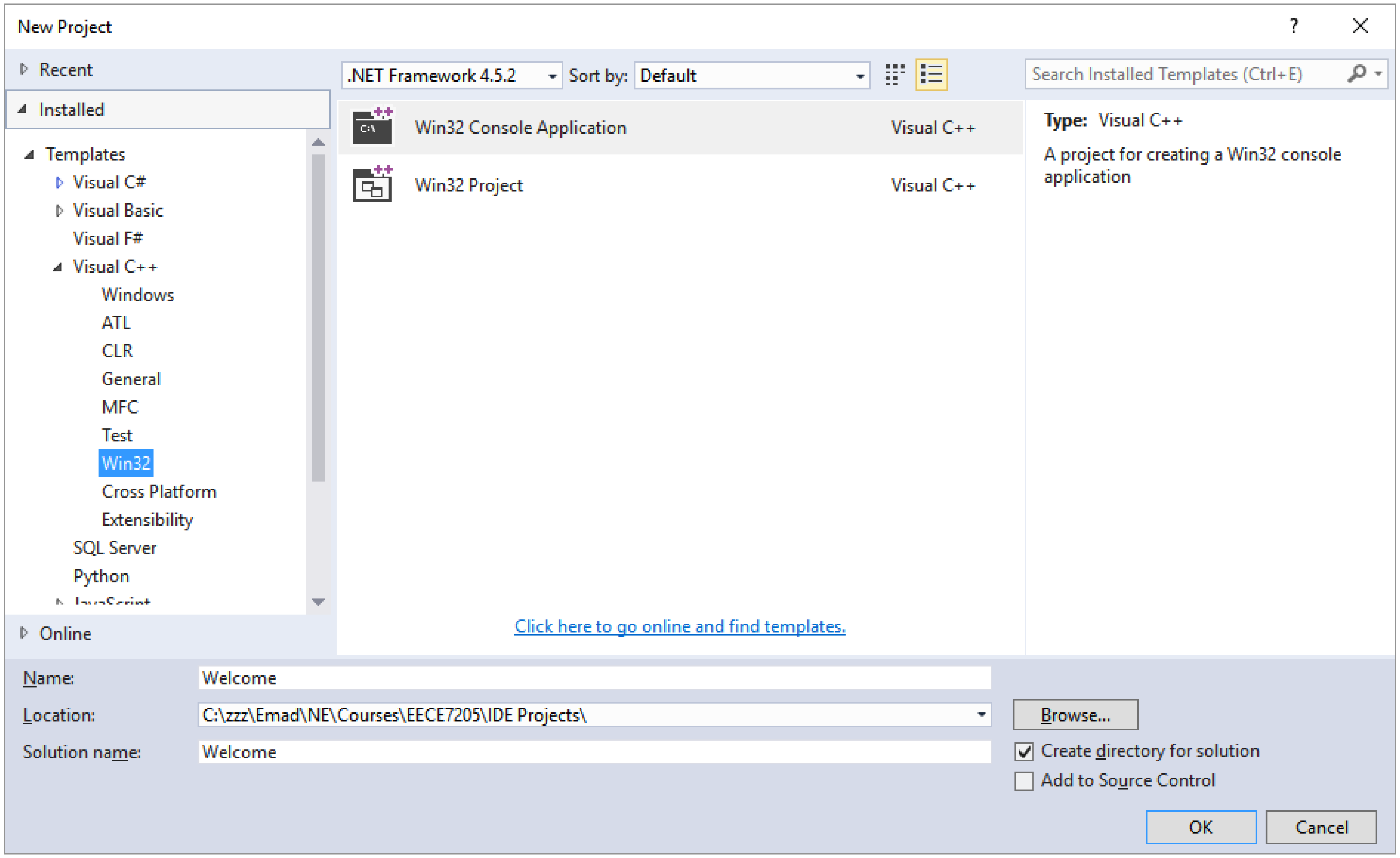
Visual Studio Projects

- A project is a group of related files, such as the C++ source-code files that compose an application.
- Visual Studio organizes applications into projects and solutions, which contain one or more projects.
- Multiple-project solutions are used to create large-scale applications.
- Each application we will create will be a solution containing a single project.
- Our applications are of type **Win32 Console Application** projects that you will execute from the IDE.

Creating a Project (1 of 5)

- Select **File → New → Project…**.
- At the **New Project** dialog's left side, select the category **Installed → Templates → Visual C++ → Win32 →** In the **New Project** dialog's middle section, select **Win32 Console Application**.
- Provide a name for your project in the **Name** field, we specified **Welcome** → click **OK** to display the **Win32 Application Wizard** window → click **Next >** to display the **Application Settings** step.

Creating a Project (2 of 5)



.NET Framework 4.5.2 Sort by: Default

Search Installed Templates (Ctrl+E)

Win32 Console Application Visual C++

Win32 Project Visual C++

Type: Visual C++

A project for creating a Win32 console application

Click here to go online and find templates.

Name: Welcome

Location: C:\zzz\Emad\NE\Courses\EECE7205\IDE Projects\

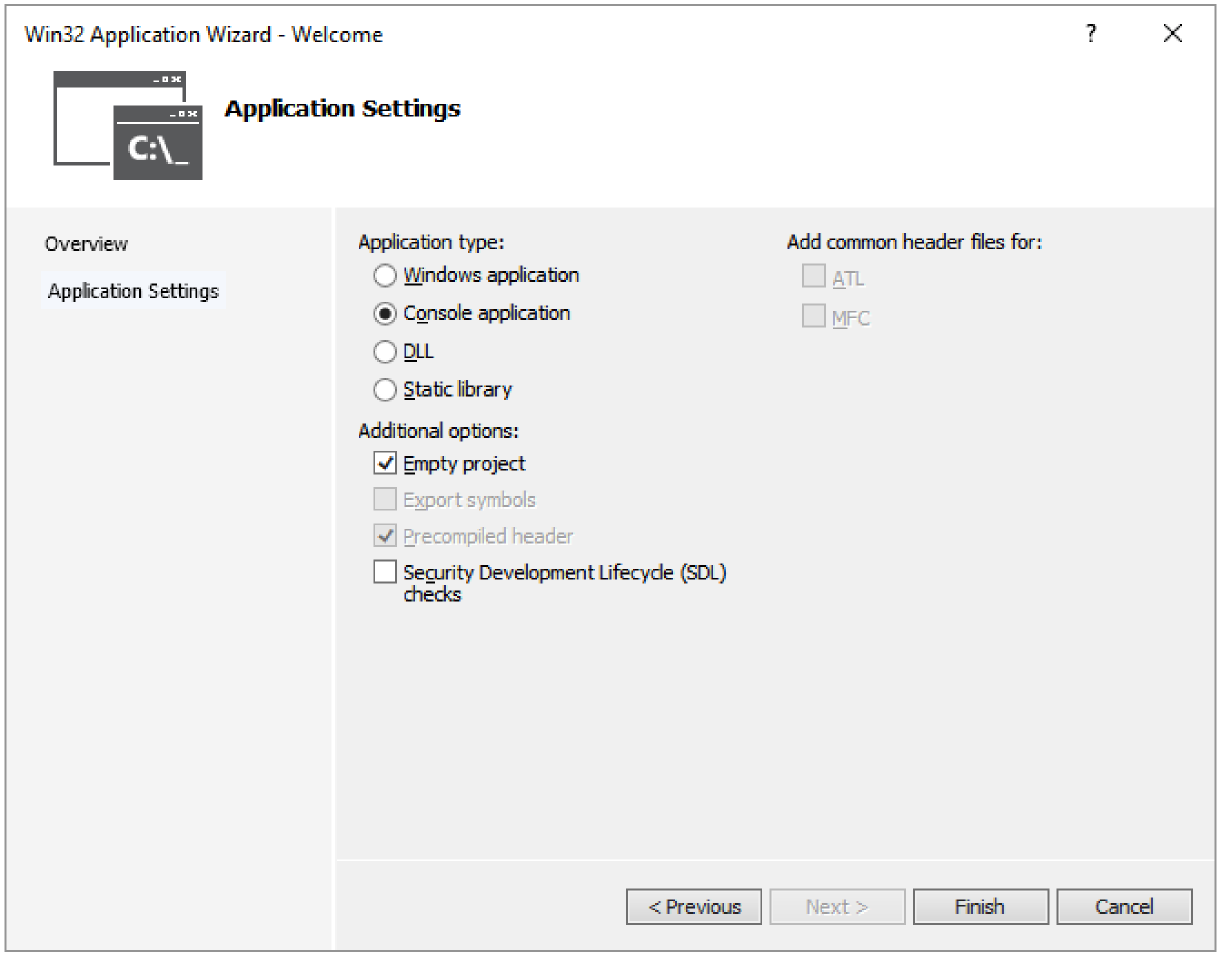
Solution name: Welcome

[✓] Create directory for solution

[ ] Add to Source Control

Creating a Project (3 of 5)

- Configure the settings as shown in the figure to create a solution containing an empty project, then click **Finish**.



Win32 Application Wizard - Welcome

Application Settings

Overview

Application Settings

Application type:

- ( ) Windows application
- (●) Console application
- ( ) DLL
- ( ) Static library

Additional options:

- [✓] Empty project
- [ ] Export symbols
- [✓] Precompiled header
- [ ] Security Development Lifecycle (SDL) checks

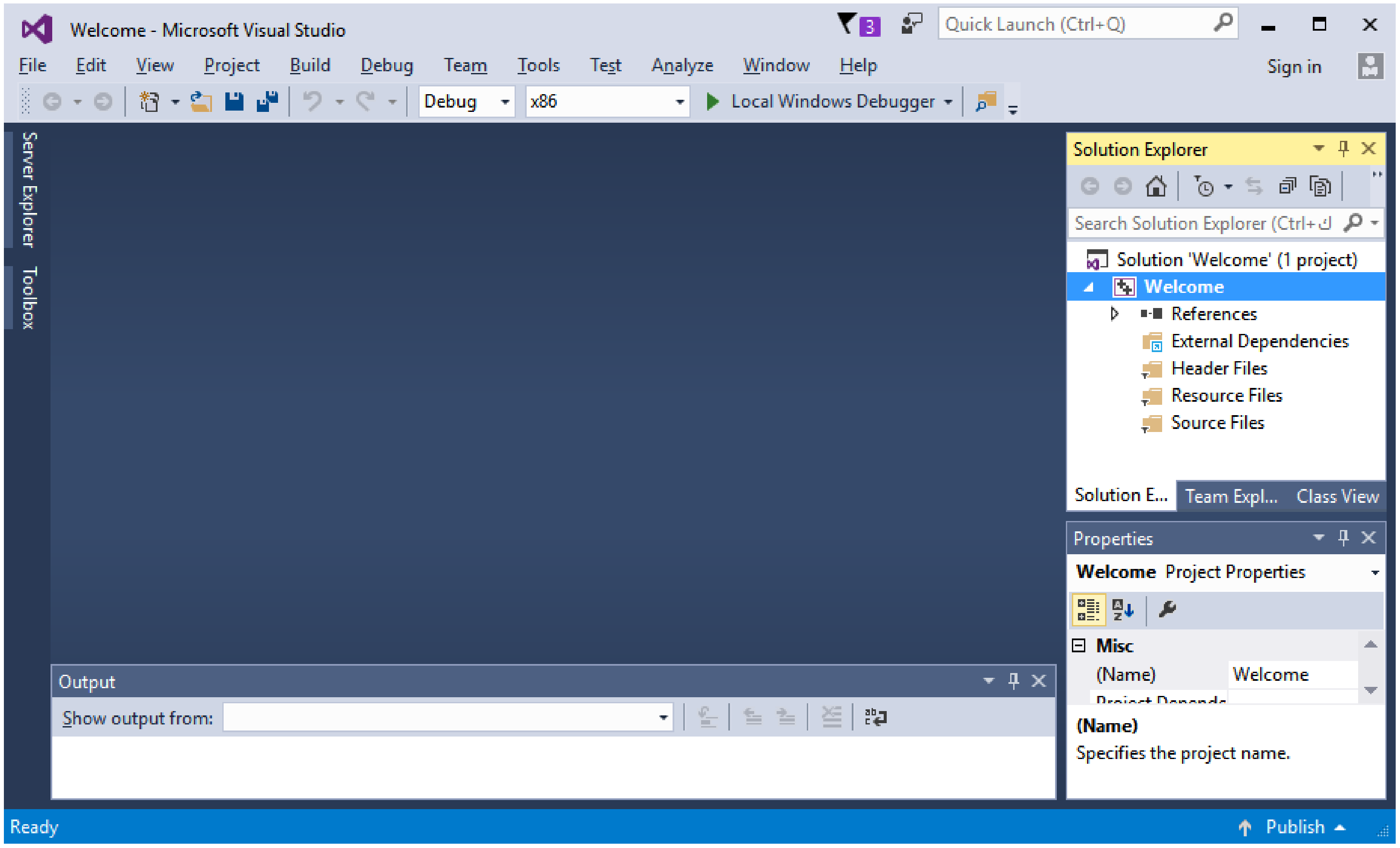
Add common header files for:

- [ ] ATL
- [ ] MFC

Creating a Project (4 of 5)

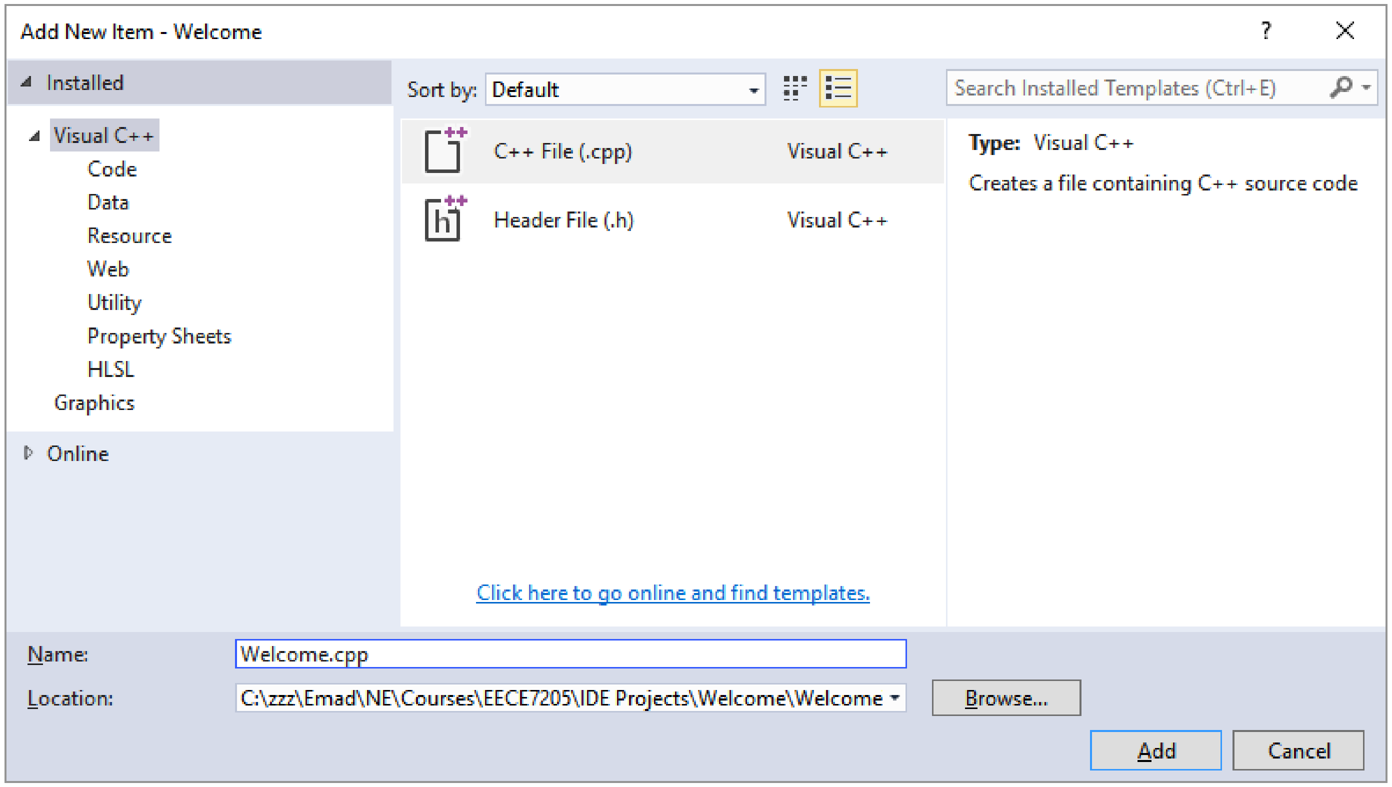
- At this point, the IDE opens the following window that displays editors as tabbed windows (one for each file) when you're editing code.
- Also displayed is the **Solution Explorer** in which you can view and manage your application's files. You'll typically place each program's code files in the **Source Files** folder.
- If the **Solution Explorer** is not displayed, you can display it by selecting **View → Solution Explorer**.

Creating a Project (5 of 5)



Adding a C++ Source File

- While the **Solution Explorer of** your **Welcome** project is open, **Right Click** on the **Source Files** folder → Select **Add** → **New item**.
- As **C++ File (.cpp)** is selected → write the name of your file, we used **Welcome.cpp** as shown → Click **Add**.



Type: Visual C++

Creates a file containing C++ source code

Name: Welcome.cpp

Location: C:\zzz\Emad\NE\Courses\EECE7205\IDE Projects\Welcome\Welcome

Adding the C++ Code

- In the editing area of **Welcome.cpp**, insert the following code and save the file (**Ctrl+S**):

```
#include <iostream> // enables program to output data to the screen
// function main begins program execution
 int main() {
// declaring of variables
 char c1;
 std::cout << "Welcome to C++!\n"; // display first message
 std::cout << "Please enter a character:"; // display first message
 std::cin >> c1; // read a character from the keyboard

 return 0; // indicate that program ended successfully
} // end function main
```

Debugging the Application

- You can debug the Welcome application by pressing **F5** (or selecting the **Debug** menu → **Start Debugging**).
  - F5 will build (compile, link, load) and execute your program in the debug mode.
- Once the program is executed, the console window (a separate window that looks like a command prompt) appears with the welcome messages and then asks you to enter a character.
- You can add a breakpoint at any line of your code where the debugger will stop to test your code.
  - Add a breakpoint at the second **cout** line by pressing **F9** (or selecting the **Debug** menu → **Toggle Breakpoint**)
  - When you start debugging the program, by pressing **F5**, the debugger will stop at that line. To continue, either single step by pressing **F10** or step into called functions by pressing **F11** or complete the rest of the program by pressing **F5**.

Build a Release Version

- Now that you have verified that everything works, you can prepare a release build of the application.
- To clean the solution files and build a release version
  1. To delete intermediate files and output files that were created during previous builds: From the menus bar → **Build** → **Clean Solution**.
  2. Under the top menus bar, locate the drop down menu where you can select for the solution configuration either **Debug** or **Release**. Make sure Release is selected.
  3. Build the solution by selecting **Build** menu → **Build Solution**.
- You can locate the executable file in the *Release folder* inside your solution folder.
  - Compare the size of the debug and release executable files.