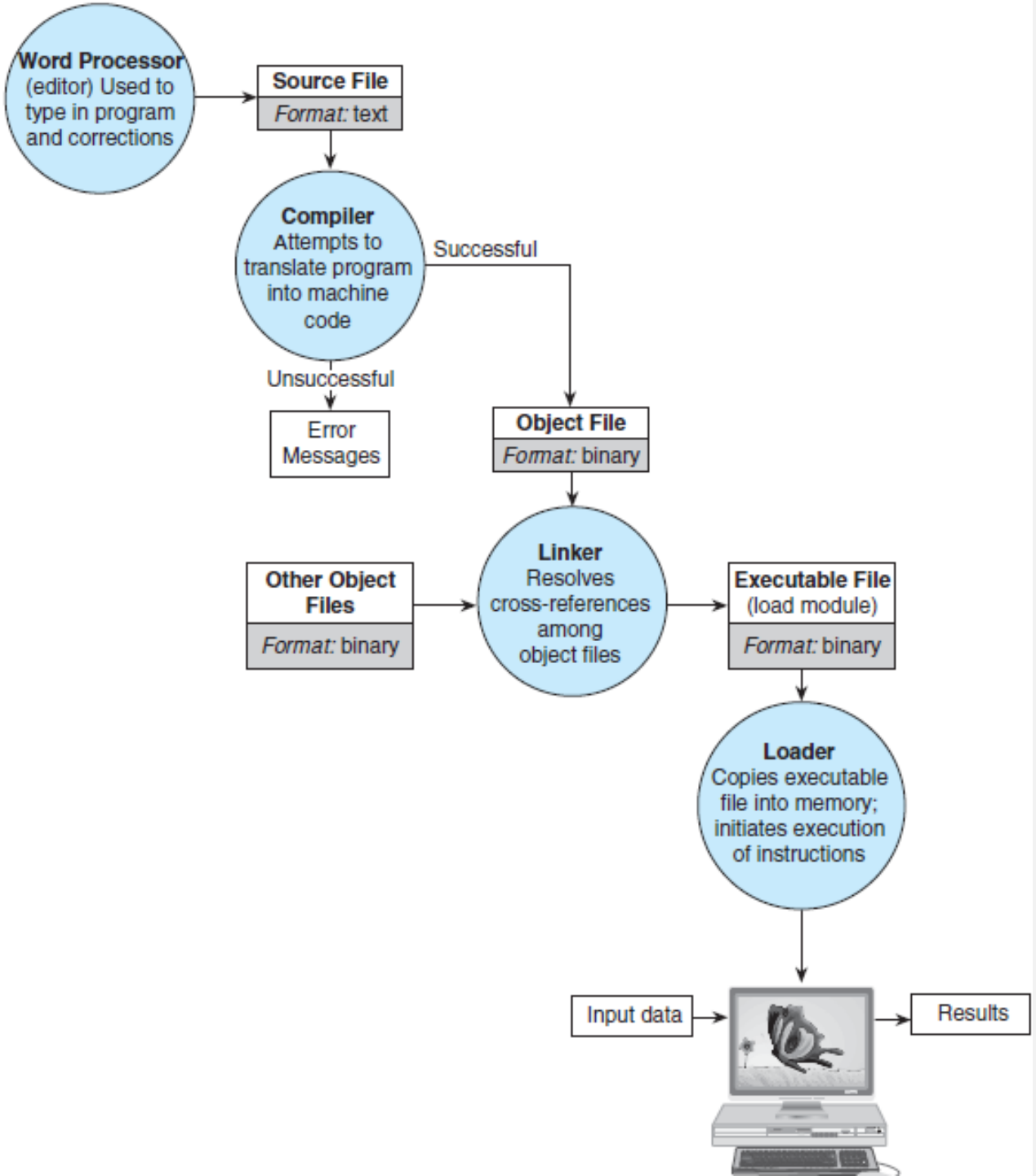
COMPILING C++ FILES

OBJECTIVES

1. Describe and review the steps in the compilation process.

2. Demonstrate how to split programs into multiple files.

3. Explore build automation tools like Makefiles and CMake.



- **Integrated Development Environment (IDE)**

Steps in the Compilation Process:

1. Preprocessing: (example: #includes)
2. Compiling: Convert source code to assembly
3. Assembling: Convert assembly code to machine code in object files .o
4. Linking: Combine object files into an executable

WORKING WITH OBJECT FILES

What is an object file?

- Contains machine code for a single source file
- Not executable until linked
- Examples: (main.o and functions.o)

Multi-file Programs

Why use Multi-file Programs?

- Improved modularity
- Allows team collaboration

Components of a Muli-File Program:

- Header files (.h or .hpp)
- Source code files (.cpp)
- The .cpp file with the main function

Examples: sayhello.h sayhello.cpp, main.cpp
functions.h functions.cpp, main.cpp

HELLO WORLD

(.h/.hpp)

```
#include <string>
//#include <iostream>
//using namespace std;
//void sayhello(string& name);


void sayhello(std::string& name);
```

(.cpp)

```
#include "sayhello.h"
#include <iostream>


using namespace std;

void sayhello(string&  name) {
    cout << "Hello " << name << '!' <<endl;
}
```

(main/ driver)

```
#include <iostream>
#include <string>
#include "sayhello.h“
using namespace std;

int main(){
     string name;
     cout << "enter name ";
     cin >> name;
     sayhello(name); //function call
     return 0;
}
```

Other code that wants to use this function only has to include sayhello.hpp.

MANAGING DEPENDENCIES

Header files

- Declare functions (prototypes), constants, and classes
- Prevent multiple inclusions of the same header file:

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

int add(int a, int b);

#endif
```

Modern alternative: `#pragma once`

IMPLEMENTATION FILE

implementation files

- functions.cpp

```
#include "functions.h"

int add(int a, int b) {
    return a + b;
}
```

FILE WITH MAIN FUNCTION/ DRIVER

Main: driver

- main.cpp

```
#include <iostream>
#include "functions.h"

int main() {
    int result = add(10, 20);
    std::cout << "The result is: " << result << std::end
    return 0;
}
```

.CPP AND .H FILES

- Use meaningful variable names (identifiers)

Indenting

- A program should be laid out so that elements that are naturally considered a group are made to look like a group
- Use a style that shows the structure of the program, is consistent, and easy to read

Comments

- Include explanatory notes at key places in the program
- Whenever something is important and not obvious, it merits a comment
- It will take experience to get the feel for when it is best to include comments
- All programs should include a descriptive header comment

Multiple files onlinegdb

1. New File
2. Name the files
   - functions.h
   - Functions.cpp
   - main(lastname_A4.cpp)

3. Include the struct definition in functions.h

onlinegdb.com/online_c++_compiler#

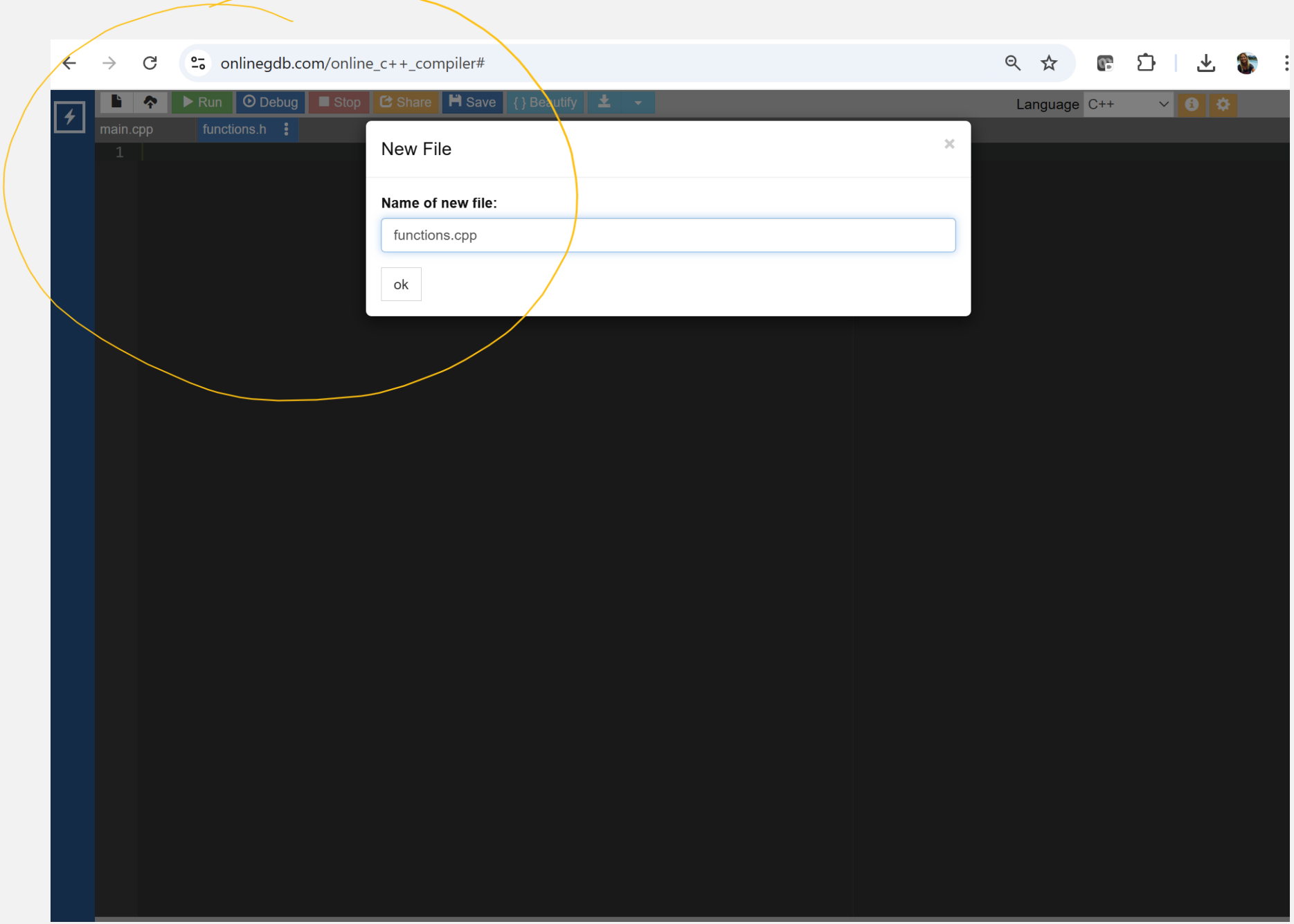
New File (Ctrl+M)

Run | Debug | Stop | Share | Save | { } Beautify

Language C++

main.cp

```
/*******************************************************************************

                                Online C++ Compiler.
                    Code, Compile, Run and Debug C++ program online.
Write your code in this editor and press "Run" button to compile and execute it.

*******************************************************************************/

#include <iostream>

int main()
{
    std::cout<<"Hello World";

    return 0;
}
```


onlinegdb.com/online_c++_compiler#
Run
Debug
Stop
Share
Save
{ } Beautify
Language
C++
main.cpp
functions.h
1
New File
Name of new file:
functions.cpp
ok

Run Debug Stop Share Save { } Beautify Language C++

main.cpp functions.h functions.cpp

```cpp
/****SAMPLE PROGRAM HEADER*********************************************************
STUDENT NAME
Due Date:
Course:  C0P3275C
Assignment: Assignment 4 FUNCTIONS STRUCT separate compilation
PART 1: Age Comparison
PART 2: Age Calculator

Professor: Sorgente

Description: (Your program description goes here -- what it does--In the program we processed....


*************************************************************/
#include <iostream> //standard library for i/o
#include <string> // always include this when you use the string class
#include "functions.h"

using namespace std;




int main()
{
    //PART 1 (age comparison)
    //declare variables for 3 first names and ages
    person p1, p2, p3;

    //ask and get the names and ages one by one
        GetNameAndAge(p1);
    GetNameAndAge(p2);
    GetNameAndAge(p3);

    //find the youngest and print their names
    FindYoungest(p1, p2, p3);

    //find the oldest and print their names
    FindOldest(p1, p2, p3);
```

main.cpp | functions.h | functions.cpp

```cpp

#ifndef FUNCTIONS_H
#define FUNCTIONS_H

using namespace std;

struct person{

    int age = 0, age2025 = 0, year = 0;
    string name, category;
    double minutes = 0.0, seconds = 0.0;
};


//reference parameter for the name and age input
void GetNameAndAge(person &p1);

//input: 3 person by constant referenc (do not want to make changes)
//finds and prints the name of the youngest age
void FindYoungest(const person &p1, const person &p2, const person &p3);

//input: 3 person by constant referenc (do not want to make changes)
//finds and prints the name of the oldest age
void FindOldest(const person &p1, const person &p2, const person &p3);

//input: 3 person by constant reference (do not want to make changes)
//prints the name and categpry for each
void Categories(const person &p1, const person &p2, const person &p3);

//reference parameter for the name and year input
void GetNameAndYear(person &p1);

//input: reference parameter
//calculatesage at the end of 2025
void Age2025(person &p1);

//input: reference parameter
//calculates the number of minutes and seconds
void AgeMinSec(person &p1);
```

Run Debug Stop Share Save { } Beautify Language C++

main.cpp functions.h functions.cpp

```cpp
#include <iostream> //standard library for i/o
#include <string> // always include this when you use the string class
#include "functions.h"

using namespace std;


//reference parameter for the name and age input
void GetNameAndAge(person &p1)
{
        cout << "Enter name: ";
    cin >> p1.name;
    cout << "Enter age: ";
    cin >> p1.age;

}

//input: 3 person by constant reference (do not want to make changes)
//finds and prints the name of the oldest age
void FindOldest(const person &p1, const person &p2, const person &p3)
{

    int old =p1.age;
    string oldN = p1.name;
    if(p2.age > old)
    {
        old = p2.age;
        oldN =p2.name;
    }
    if(p3.age > old)
    {
        old = p3.age;
        oldN = p3.name;
    }


    cout << "The oldest person is: " << oldN << endl << endl;
}

//input: 3 person by constant reference (do not want to make changes)
//finds and prints the name of the youngest age
```

STEPS TO COMPILE AND EXECUTE

Compile and link **`main.cpp`** sing the **g++ compiler** :

1. Compile .cpp file into an object file and executlable:

```
g++ main.cpp -o myprogram
```

2. *To execute:* `./myprogram`

What is Bash?

Unix **Bash** is a Unix shell and command-line interface commonly used in Unix-based operating systems,
including Linux and macOS.

"Bash" stands for **Bourne Again SHell**, which is a pun on the Bourne shell (sh), one of its predecessors.

Bash is a default shell in many Unix-like systems due to its power, flexibility, and compatibility.

Ubuntu Terminal

The **Ubuntu Terminal** is a command-line interface (CLI) application that allows users to interact with the

Ubuntu operating system using text-based commands.

It is an essential tool in Ubuntu, a popular Linux distribution, and provides access to the underlying system shell.

By default, the Ubuntu Terminal runs the **Bash shell**, although users can configure it to use other shells.

Ubuntu Terminal

Key Features on the Ubuntu Terminal

- Access to the shell (typically Bash) - execute commands to control the system, manage files, and perform various tasks
- Command-line Utilities – such as (ls) list files, (cd) change directory, (cp) copy files, (grep) search
- Script execution

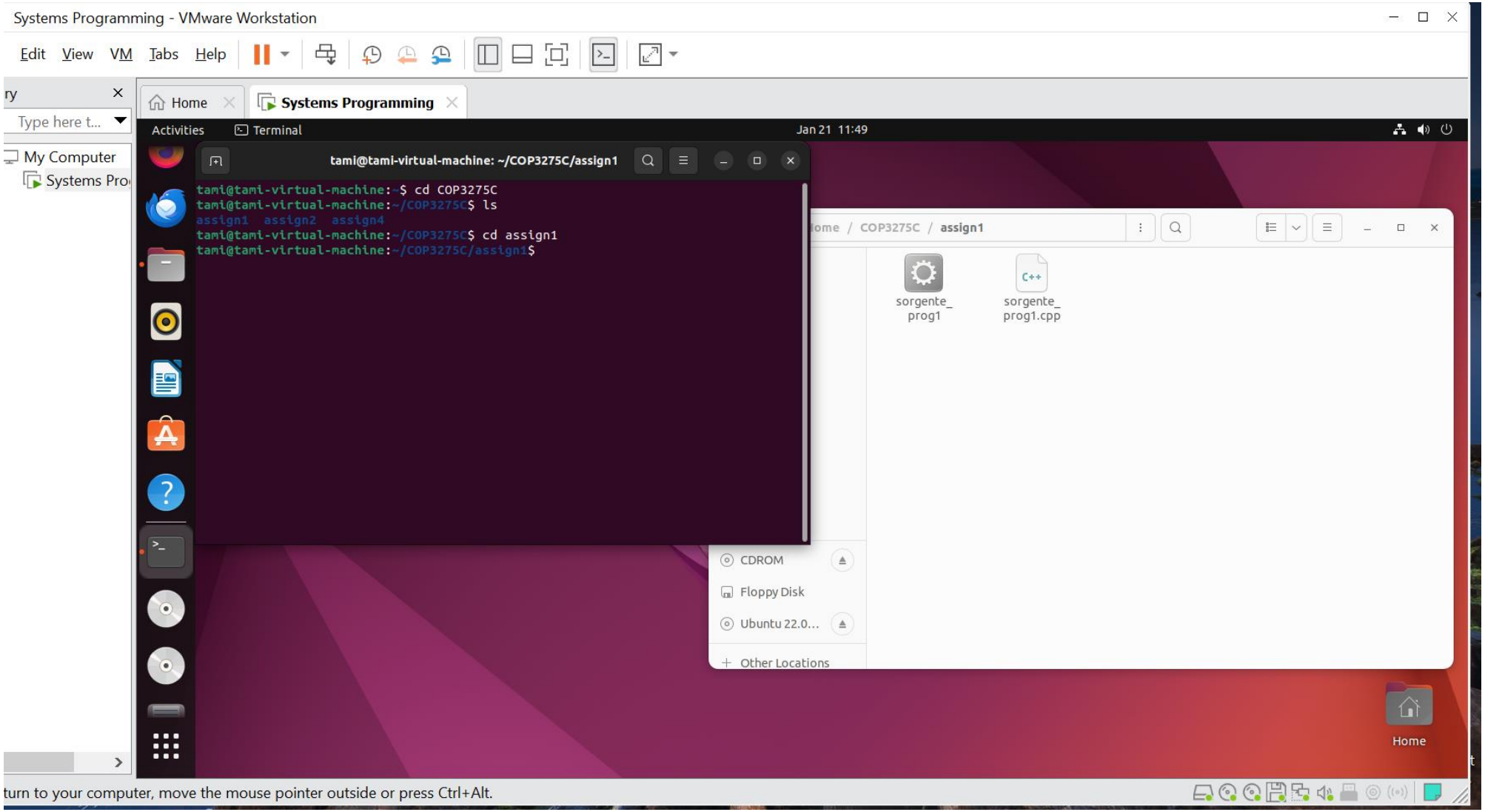
Ubuntu Terminal

Key Features on the Ubuntu Terminal

- Customization
- Networking tools
- Package management
- System administration

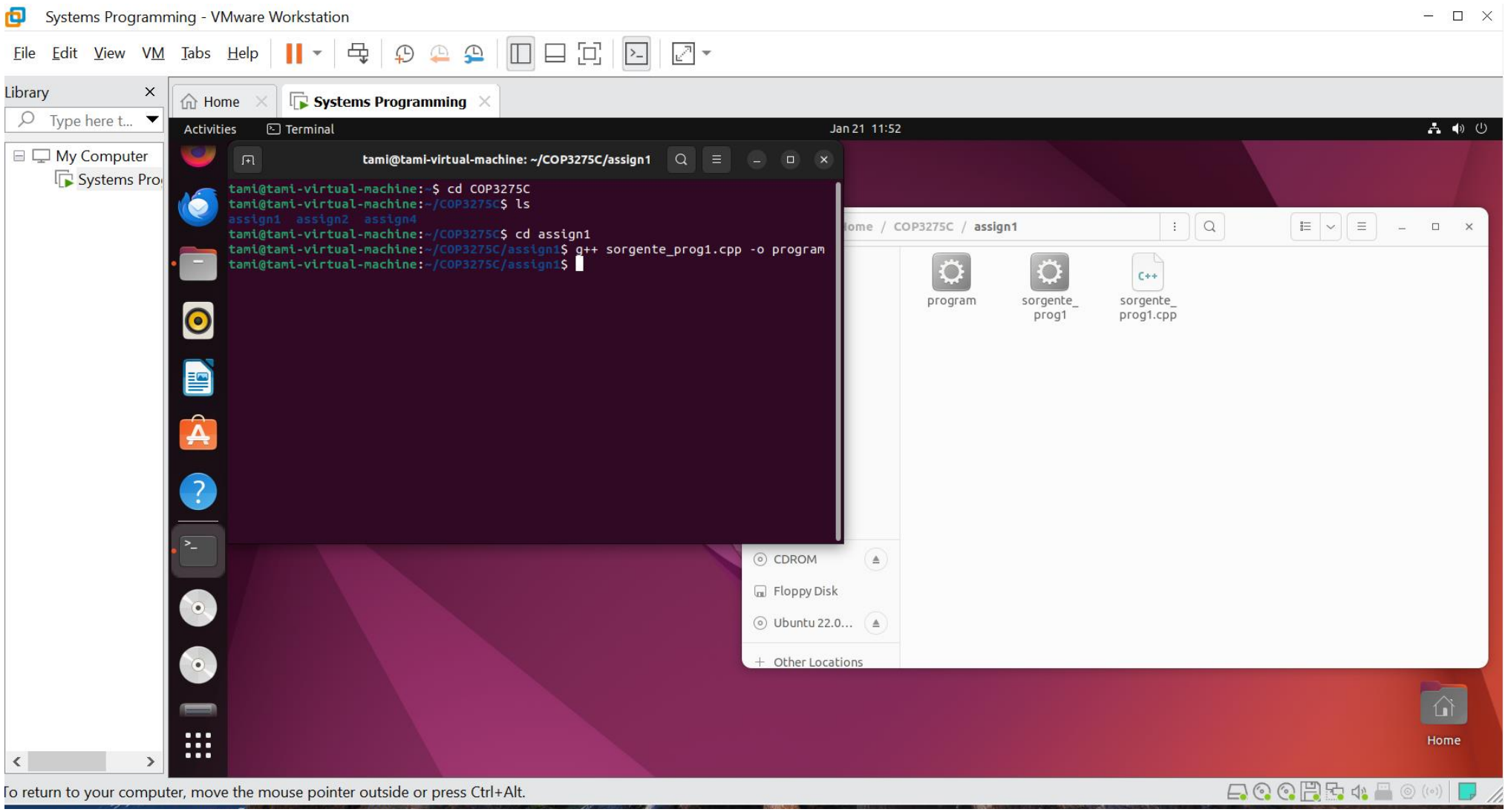
Compiling .cpp source code through Terminal on Ubuntu

Use “`cd`” to navigate to the directory, use “`ls`” to list the directories or files



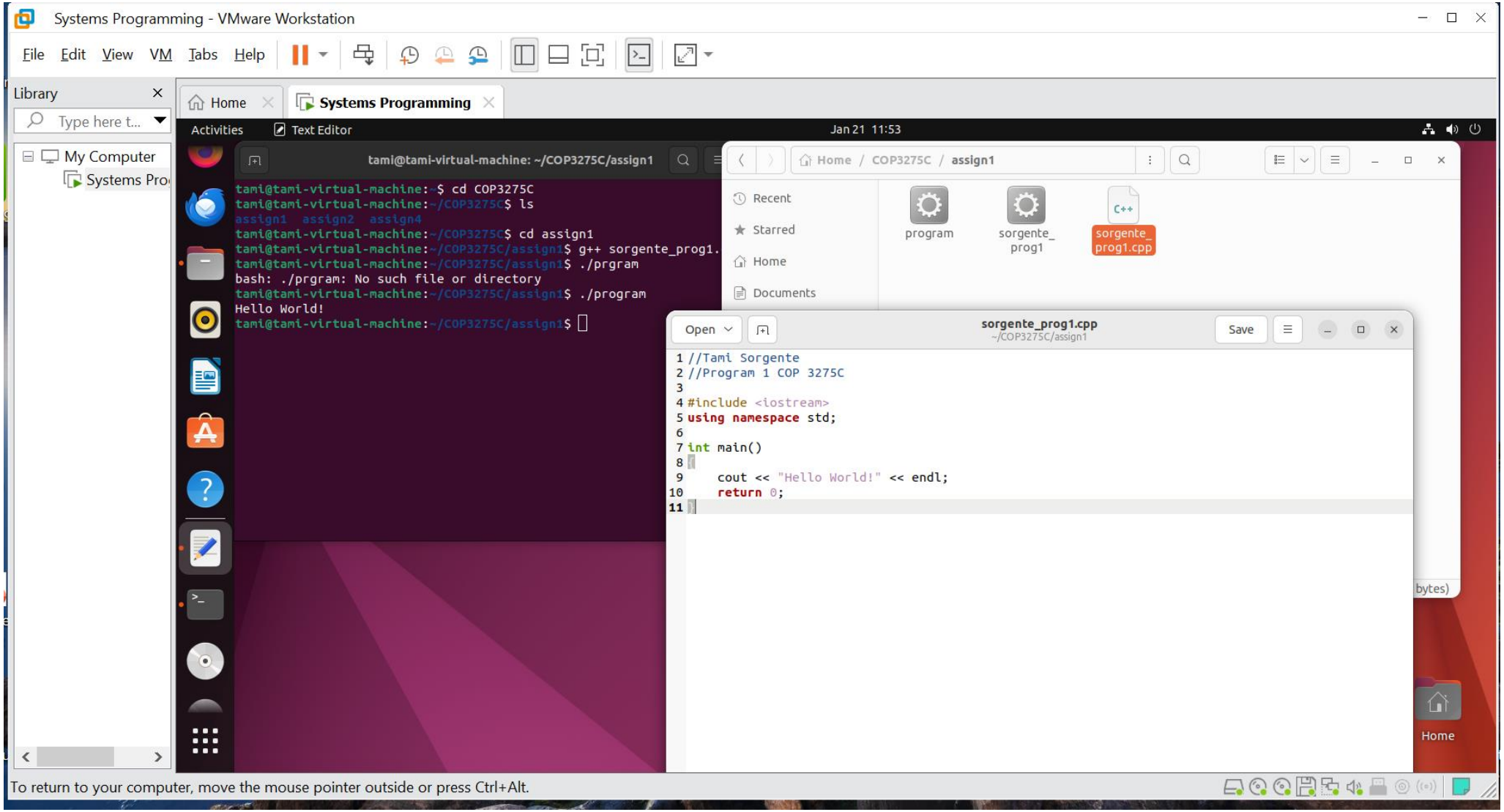
Compiling .cpp source code through Terminal on Ubuntu

```
g++ sourceFileName.cpp -o program  //program is the name of the executable
```



Run the program

```
./program // program is the name of the executable
```



STEPS TO COMPILE MULTI-FILE PROGRAMS MANUALLY

Compile and link **`main.cpp`** and **`functions.cpp`** using the **g++ compiler** (without a class):

1. Compile each .cpp file into an object file

```
g++ -c main.cpp -o main.o   (-c compile without linking)
g++ -c functions.cpp -o functions.o   (-o is the name of the output file)
```

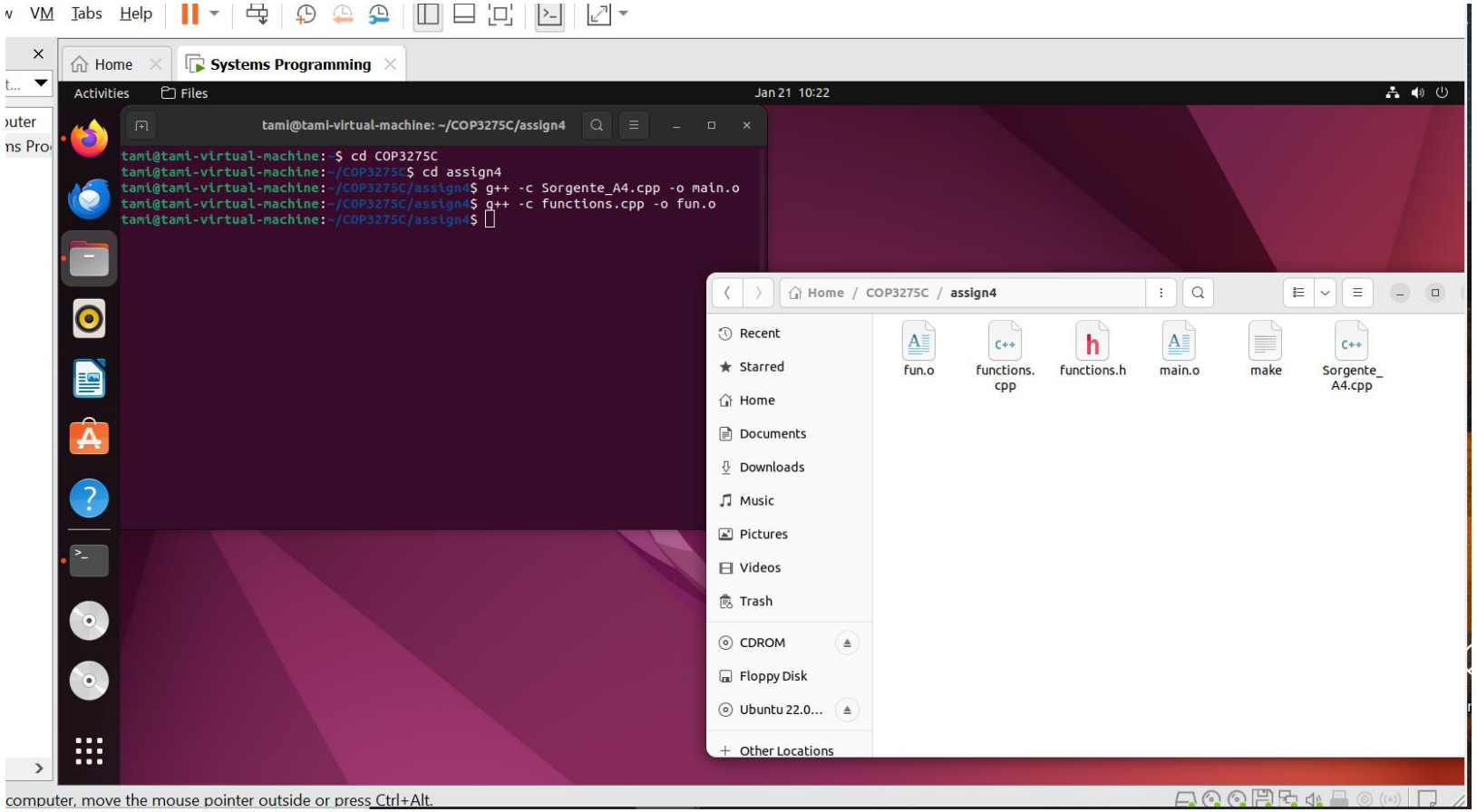
2. Link the object files into an executable:

```
g++ main.o functions.o -o program   (the executable name is program)
```

3. *To execute:* `./program`

Compiling separately – there are 2 .cpp files compiled without linking

```
g++ -c Sorgente_A4.cpp -o main.o //(-o is the name of the output file)
g++ -c functions.cpp -o fun.o //(-o is the name of the output file)
```

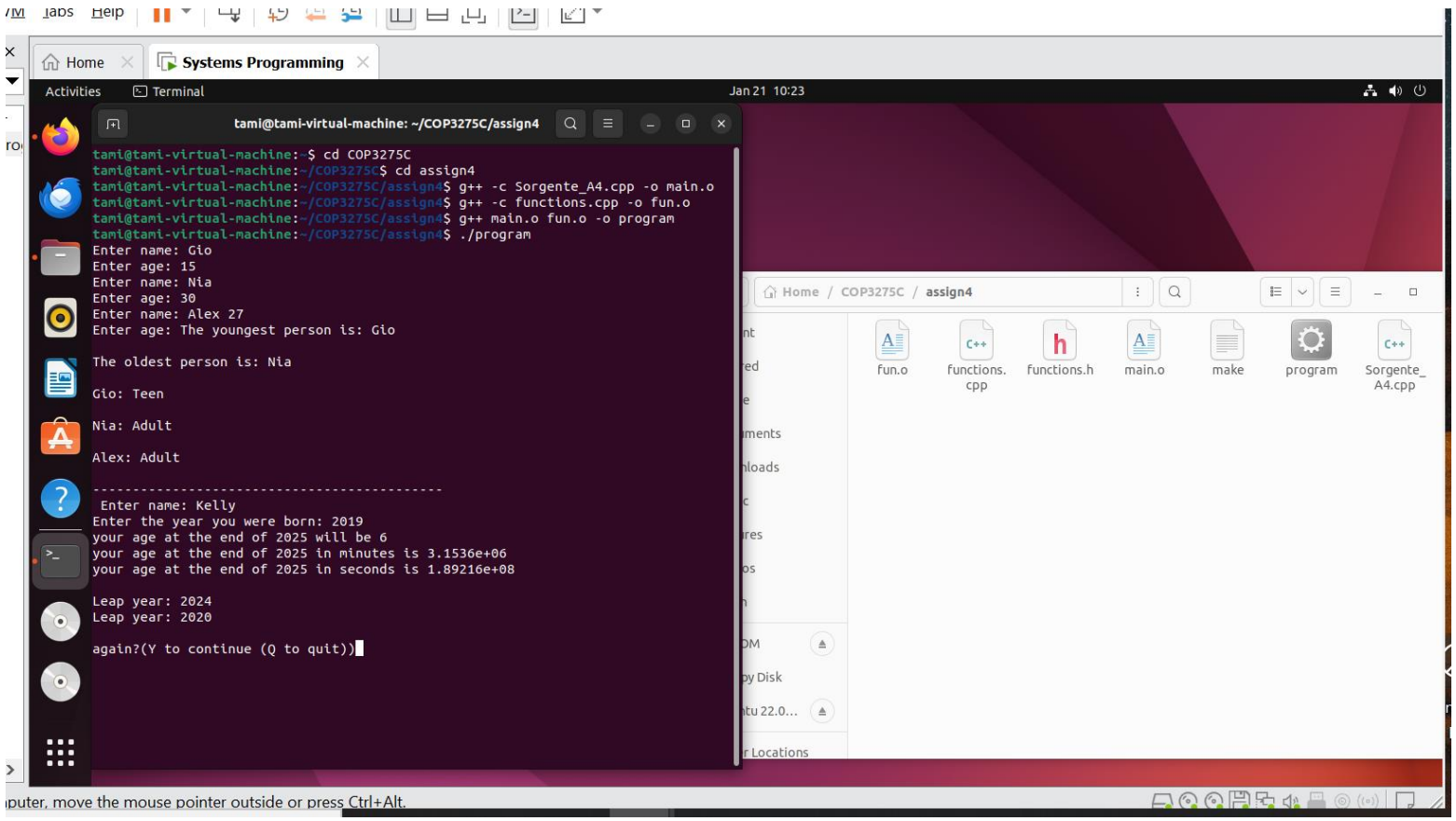


Compiling both files and linking

```
g++ *.cpp -o program//(program is the name of the executable)
```

Link the .o files and run the program

`g++ main.o fun.o program//`*(program is the name of the executable)*
`./program` *(to execute)*



Compiling both files and linking

`g++ *.cpp -o program//`*(program is the name of the executable)*

Home | Systems Programming

Activities | Terminal | Jan 21 10:24

tami@tami-virtual-machine: ~

```
tami@tami-virtual-machine:~$ cd COP3275C
tami@tami-virtual-machine:~/COP3275C$ cd assign4
tami@tami-virtual-machine:~/COP3275C/assign4$ g++ -c Sorgente_A4.cpp -o main.o
tami@tami-virtual-machine:~/COP3275C/assign4$ g++ -c functions.cpp -o fun.o
tami@tami-virtual-machine:~/COP3275C/assign4$ g++ main.o fun.o -o program
tami@tami-virtual-machine:~/COP3275C/assign4$ ./program
Enter name: Gio
Enter age: 15
Enter name: Nia
Enter age: 30
Enter name: Alex 27
Enter age: The youngest person is: Gio

The oldest person is: Nia

Gio: Teen

Nia: Adult

Alex: Adult

-----------------------------------------
 Enter name: Kelly
Enter the year you were born: 2019
your age at the end of 2025 will be 6
your age at the end of 2025 in minutes is 3.1536e+06
your age at the end of 2025 in seconds is 1.89216e+08

Leap year: 2024
Leap year: 2020

again?(Y to continue (Q to quit))q
tami@tami-virtual-machine:~/COP3275C/assign4$ cd
tami@tami-virtual-machine:~$ S
```

Home / COP3275C / assign4

fun.o | functions.cpp | functions.h | main.o | make | program | Sorgente_A4.cpp

nputer, move the mouse pointer outside or press Ctrl+Alt.

BUILD AUTOMATION

Makefiles: Automate compilation and linking

- Save time and avoid manual errors

A Makefile is a script containing rules on how to build targets (example: executables)

CMake: an open-source, cross-platform tool designed to manage the build process of software projects.

Instead of writing build scripts (like Makefiles) manually for each platform and compiler, you write a single CMakeLists.txt file that describes your project. CMake then generates the appropriate build files for your specific environment (e.g., Makefiles for Linux, Visual Studio project files for Windows)

Compiling multiple files in VS Code Terminal

Components of a Muli-File Program:

- Header files (.h or .hpp)
- Source code files (.cpp)
- The .cpp file with the main function

Systems Programming - VMware Workstation

File Edit View VM Tabs Help

Library

Type here t...

My Computer

Systems Pro

Home | Systems Programming

Activities | Visual Studio Code | Jan 21 11:59

Sorgente_A4.cpp - assign4VS - Visual Studio Code

File Edit Selection View Go Run Terminal Help

EXPLORER

ASSIGN4VS

functions.cpp

functions.h

Sorgente_A4.cpp

OUTLINE

TIMELINE Sorgente_A4.cpp

Local History will track recent changes as you save them unless the file has been excluded or is too large. Source Control has not been configured.

Sorgente_A4.cpp > ...

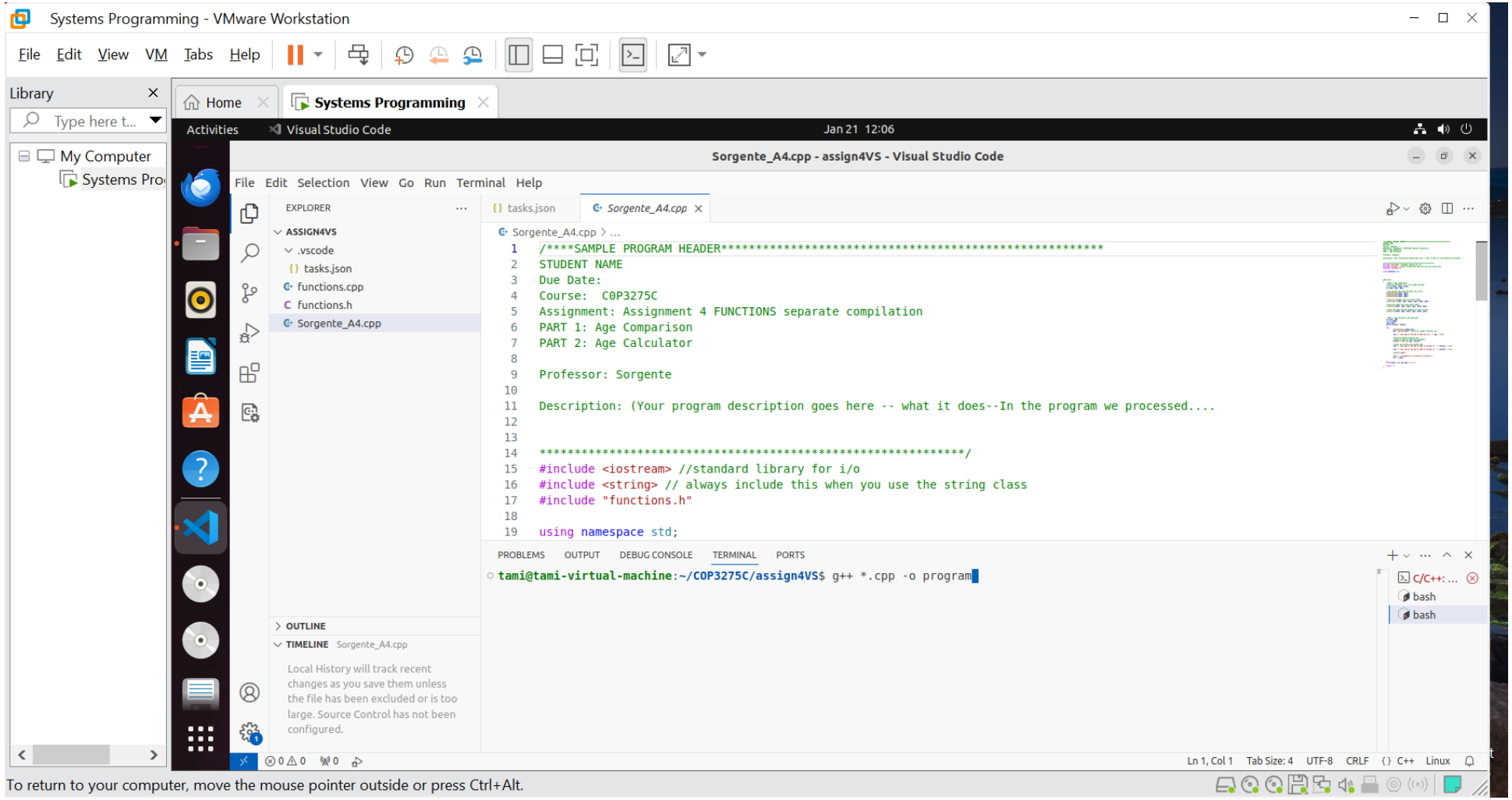
```cpp
/****SAMPLE PROGRAM HEADER*************************************************************
STUDENT NAME
Due Date:
Course:  C0P3275C
Assignment: Assignment 4 FUNCTIONS separate compilation
PART 1: Age Comparison
PART 2: Age Calculator

Professor: Sorgente

Description: (Your program description goes here -- what it does--In the program we processed....


*************************************************************/
#include <iostream> //standard library for i/o
#include <string> // always include this when you use the string class
#include "functions.h"

using namespace std;




int main()
{
    //PART 1 (age comparison)
    //declare variables for 3 first names and ages
    string name1, name2, name3;
    int age1, age2, age3;

    //ask and get the names and ages one by one
    GetNameAndAge(name1, age1);
```

Ln 1, Col 1 Tab Size: 4 UTF-8 CRLF {} C++ Linux

To return to your computer, move the mouse pointer outside or press Ctrl+Alt.

Compiling both files and linking

```
g++ *.cpp -o program//(program is the name of the executable)
```



./program *(to execute)*