Lab7

Makefile: Separate Compilation and Namespaces (CH12)

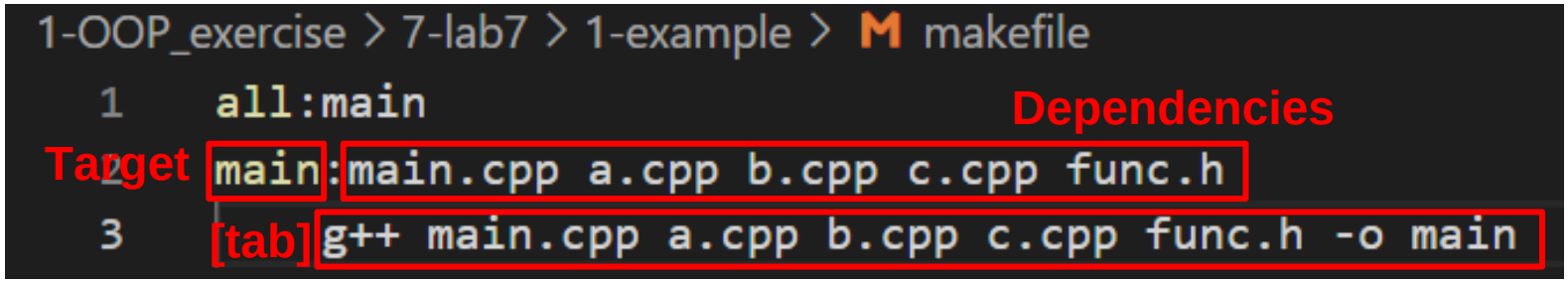
Department of Electronics Engineering
National Chiao Tung University

Makefile

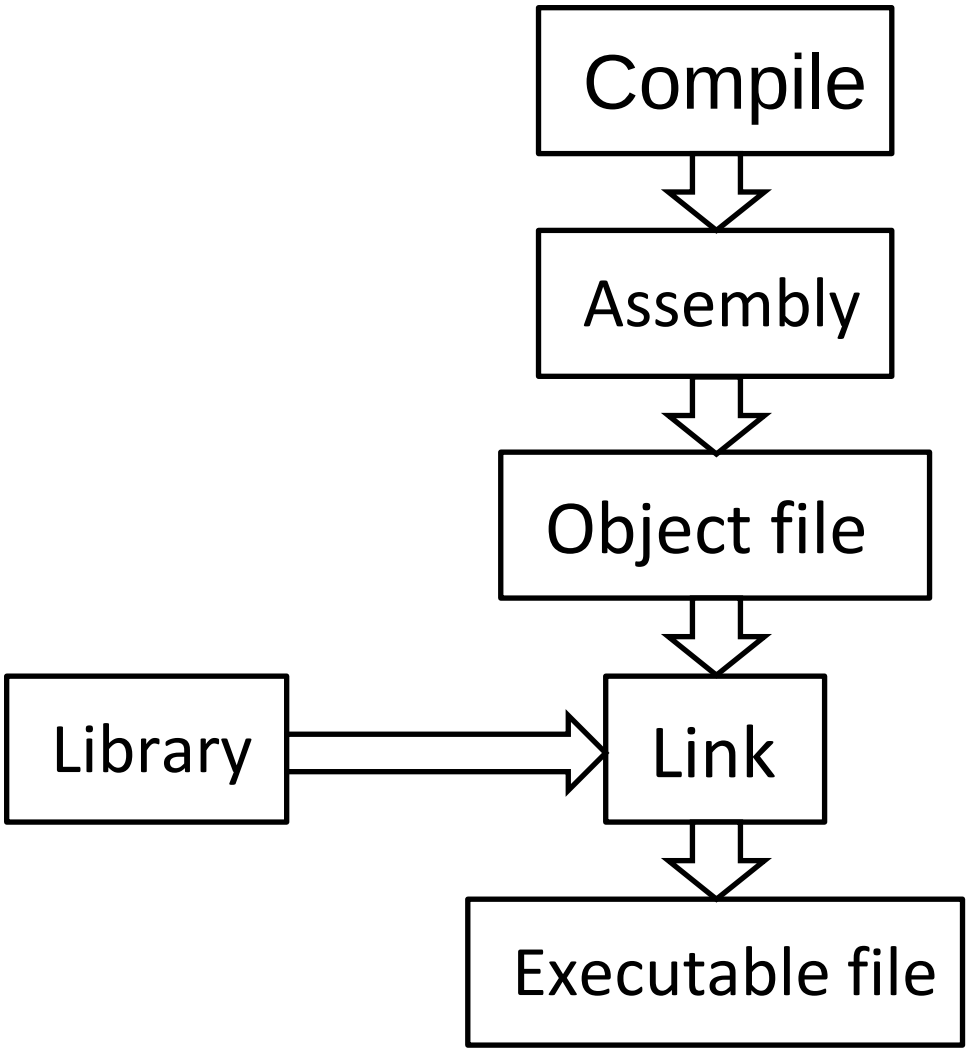
- Used for automating the compilation of C/C++.
- Just execute "makefile" to compile a large number of files
- When making changes to the code, only files that have been modified will be recompiled, saving a lot of time from repetitive compilations.

Makefile - rule

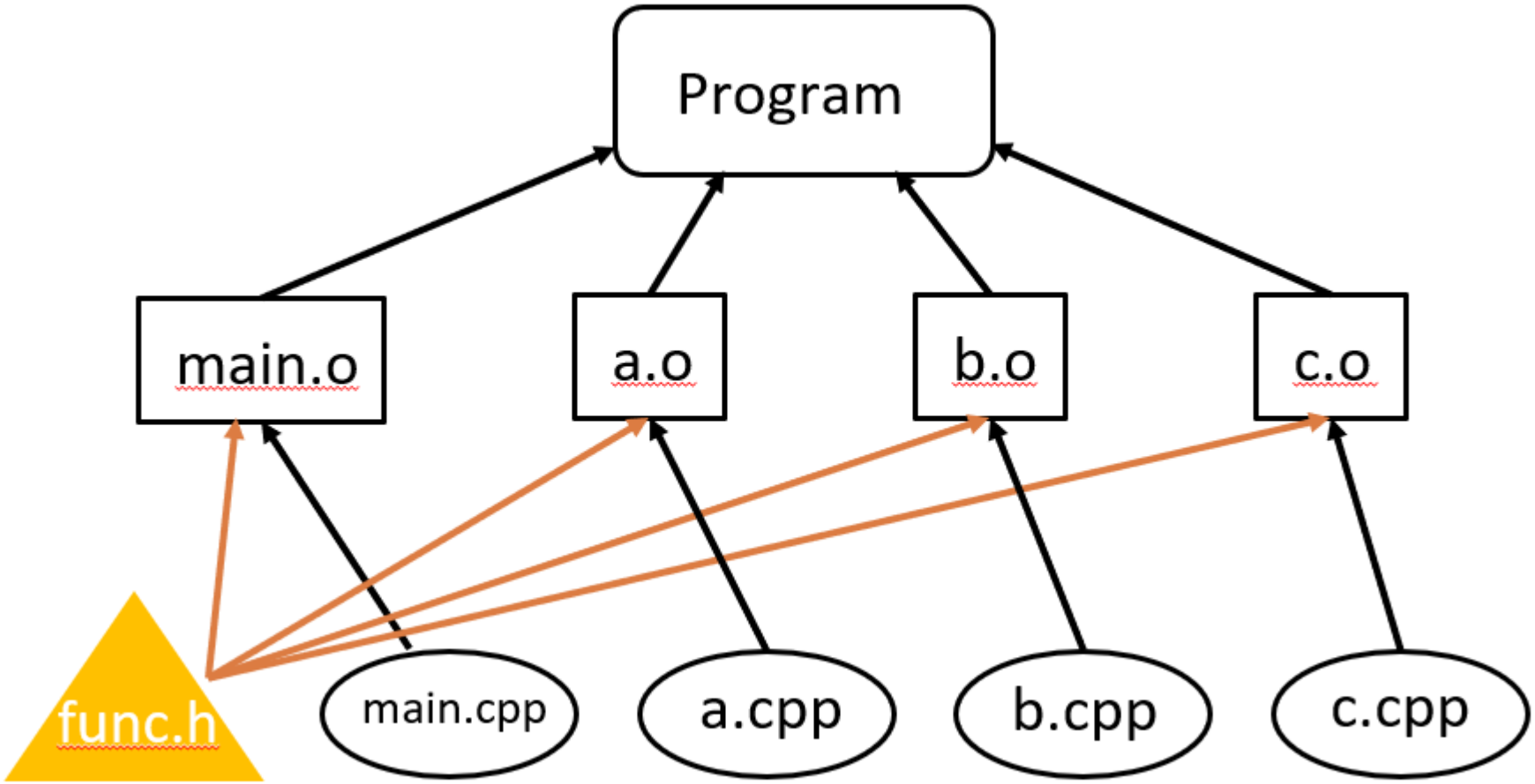
- Filename: makefile / Makefile
- Grammar:
  - ※target: dependencies
  - ※[tab]system command
- Use "#" to make comments.
- "\" can be used to indicate a line continuation. Note that there should not be any spaces after "\".
- Command: make



Compilation process



Example (1/6)



Example (2/6)

- Compile: original

```
g++ a.cpp b.cpp c.cpp func.h main.cpp -o main
```

- Compile: makefile

1-OOP_exercise > 7-lab7 > 1-example > M makefile

```
1    all:main
2    main:main.cpp a.cpp b.cpp c.cpp func.h
3    [tab]g++ main.cpp a.cpp b.cpp c.cpp func.h -o main
```

Target

Dependencies

System command

```
14:18 melody2354@vda04 [~/1-OOP_exercise/7-lab7/1-example] >$ make
g++ a.cpp b.cpp c.cpp func.h main.cpp -o main
```

Example (3/6)

- Transform into multiple targets

1-OOP_exercise > 7-lab7 > 1-example > M makefile

```
1   all:main
2   main:main.o a.o b.o c.o
3       g++ main.o a.o b.o c.o -o main
```

```
14:24 melody2354@vda04 [~/1-OOP_exercise/7-lab7/1-example] >$ make
g++ -c main.cpp
g++ -c a.cpp
g++ -c b.cpp
g++ -c c.cpp
g++ main.o a.o b.o c.o -o main
```

```
9       g++ -c b.cpp
10  c.o:c.cpp func.h
11      g++ -c c.cpp
12  clean:
13      rm -rf *.o main
```

Example (4/6)

- a.cpp change
  → Only the files that are referenced need to be recompiled

```
14:24 melody2354@vda04 [~/1-OOP_exercise/7-lab7/1-example] >$ make
g++ -c a.cpp
g++ main.o a.o b.o c.o -o main
```

1-OOP_exercise > 7-lab7 > 1-example > M makefile

```
1   all:main
2   main:main.o a.o b.o c.o
3       g++ main.o a.o b.o c.o -o main
4   main.o:main.cpp func.h
5       g++ -c main.cpp
6   a.o:a.cpp func.h
7       g++ -c a.cpp
8   b.o:b.cpp func.h
9       g++ -c b.cpp
10  c.o:c.cpp func.h
11      g++ -c c.cpp
12  clean:
13      rm -rf *.o main
```

Example (5/6)

- Clean

```
clean:
    rm -rf a.o b.o c.o main.o main
```

→ Simplified

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

```
14:46 melody2354@vda04 [~/1-OOP_exercise/7-lab7/1-example] >$ ls
a.cpp  a.o  b.cpp  b.o  c.cpp  c.o  func.h  main*  main.cpp  main.o  makefile
14:46 melody2354@vda04 [~/1-OOP_exercise/7-lab7/1-example] >$ make clean
rm -rf *.o main
14:46 melody2354@vda04 [~/1-OOP_exercise/7-lab7/1-example] >$ ls
a.cpp  b.cpp  c.cpp  func.h  main.cpp  makefile
```

Example (6/6)

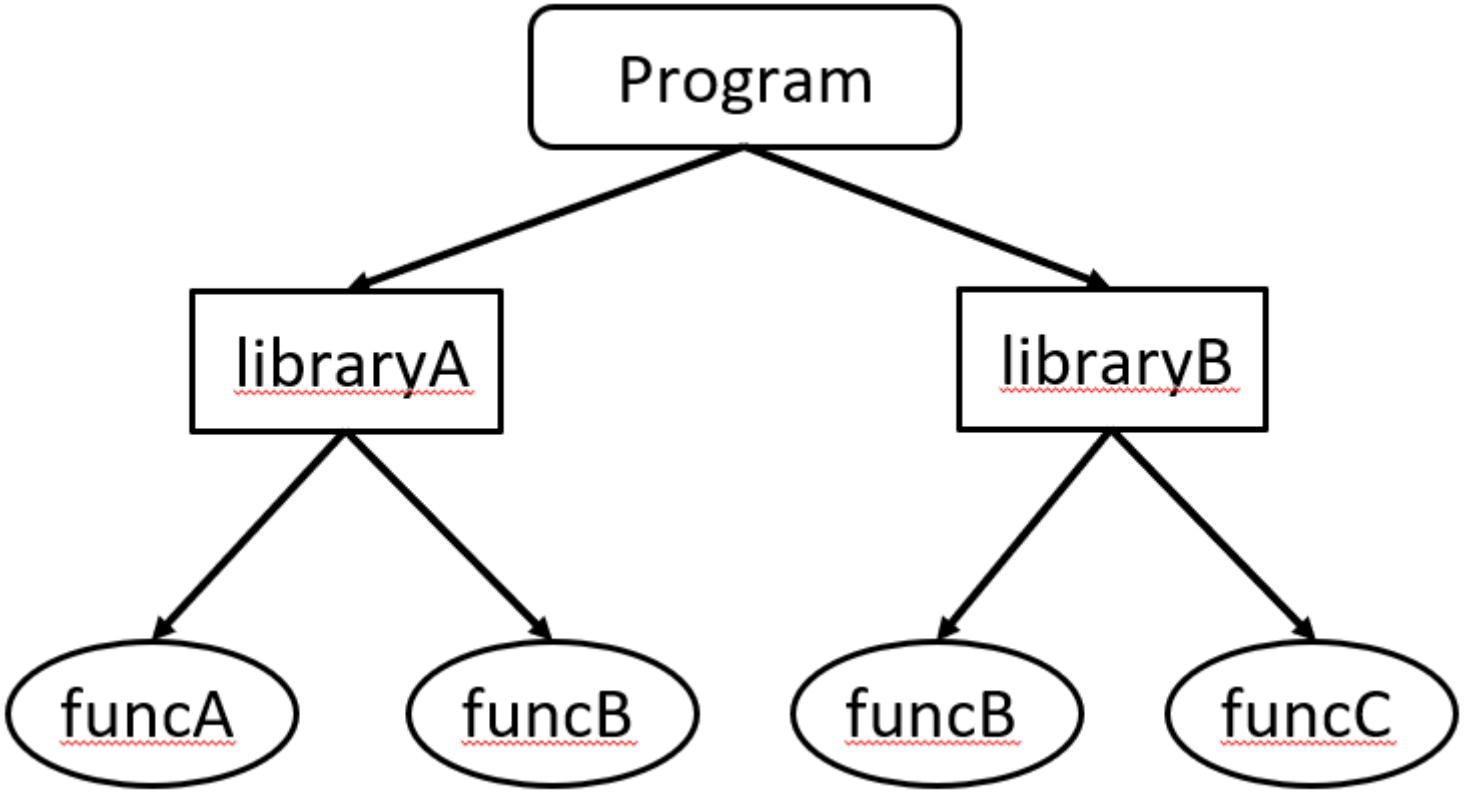
- Variable

1-OOP_exercise > 7-lab7 > 1-example > makefile

```
1   all:main
2   CC = g++
3   target = main
4
5   $(target):main.o a.o b.o c.o
6       $(CC) main.o a.o b.o c.o -o $(target)
7   main.o:main.cpp func.h
8       $(CC) -c main.cpp
9   a.o:a.cpp func.h
10      $(CC) -c a.cpp
11  b.o:b.cpp func.h
12      $(CC) -c b.cpp
13  c.o:c.cpp func.h
14      $(CC) -c c.cpp
15  clean:
16      rm -rf *.o $(target)
```

Namespace

- To obtain functions or variables with the same name under different libraries.



Namespace

- It will result in errors related to predefined functions, if not using namespace.

libraryA.h

```
void Func(){
    cout << "hi" << endl;
}
```

libraryB.h

```
void Func(){
    cout << "hello" << endl;
}
```

main.cpp

```
#include "libraryA.h"
#include "libraryB.h"
```

Compile: error

```
libraryB.h: In function ‘void Func()’:
libraryB.h:12:6: error: redefinition of ‘void Func()’
```

Namespace

- Using namespace in library →Success

libraryA.h

```
namespace libraryA{
    void Func(){
        cout << "hi" << endl;
    }
}
```

libraryB.h

```
namespace libraryB{
    void Func(){
        cout << "hello" << endl;
    }
}
```

main.cpp

```
#include "libraryA.h"
#include "libraryB.h"

int main(){
    libraryA::Func();
    libraryB::Func();
    return 0;
}
```

Compile: success

```
hi
hello
```

Namespace

- Put the declaration of the function in namespace grouping, and put the definition outside.

```
namespace libraryA{
    void Func();
}
void libraryA::Func(){
    cout << "hi" << endl;
}
```

- Define the function in namespace grouping.

```
namespace libraryA{
    void Func(){
        cout << "hi" << endl;
    }
}
```

Namespace

- Using declaration

```
#include "libraryA.h"
#include "libraryB.h"

int main(){
    libraryA::Func();
    libraryB::Func();
    return 0;
}
```

- Using directive(assign a range)

```
#include "libraryA.h"
#include "libraryB.h"

int main(){
    {
        using namespace libraryA;
        Func();
    }
    {
        using namespace libraryB;
        Func();
    }
    return 0;
}
```

Exercise (1/7) - Description

- In mathematics, a polynomial is an expression of finite length constructed from variables and constants, using only the operations of addition, subtraction, multiplication, and non-negative integer exponents.

- For example, $4x^2 - x + 5$ is a polynomial.

Exercise (2/7) - Specification

- In this problem, your job is :
  1. Put the declaration of PolySeq class and other functions in header file func.h , PolySeq class need to include the following member function.
  2. Implement the following polynomial member functions and other functions in func.cpp.
  3. Implement main function in lab7.cpp.
  4. Write a Makefile that use multiple target and includes a "clean" command to delete all files generated by Makefile.

Exercise (3/7) - Specification

- You must implement the PolySeq class with the following public data members:

PolySeq class	
data	**Description**
int *c	The dynamic array used to store coefficient.
int n	The number of coefficient.

Exercise (4/7) - Specification

- You must implement the PolySeq class with the following public member functions:

PolySeq class	
Functions	**Description**
PolySeq(int)	Constructor. The parameters is the total number of coefficient.
PolySeq()	Constructor with no parameter.
~PolySeq()	Destructor. The dynamic array needed to be deleted.
PolySeq operator+(const PolySeq &)	Return the sum of two polynomials.
PolySeq Derivative()	Return the derivative of the polynomial.
int Integral(int, int)	Return the result of the definite integral of the polynomial. The parameter are lower bound and upper bound of the integral.

Exercise (5/7) - Specification

- You must implement two kinds of getvalue function with two namespaces:

Namespace Poly_Int	
Functions	**Description**
int getvalue(PolySeq &,int)	Return the result of the polynomial with the specified int parameter.
Namespace Poly_Float	
Functions	**Description**
float getvalue(PolySeq &, float)	Return the result of the polynomial with the specified float parameter.

Exercise (6/7) - Specification

- Follow the function calling rules as the following example.

For example: P1 $= 6x + 1$, P2= $3x^2 + 3x + 2$

Functions	Mathematical Expression
P1 + P2	
P1.Derivative()	
P1.Integral(2, 3)	

Exercise (7/7)

- The output should print the following intergers in order.

1. The sum of the first and the second polynomials with parameter x1 and you need to use getvalue with namespace Poly_Int.
2. The derivative of the first polynomial with parameter x3 and you need to use getvalue with namespace Poly_Float.
3. The result of the definite integral of the second polynomial with parameter lower bound x1 and upper bound x2.

Sample Input (cin)	**Sample Output (cout)**
4 n1 3 -2 1 0 c1[ ] $3x^3 - 2x^2 + x^1 + 0x^0$ 3 n2 9 -4 1 c2[ ] $9x^2 - 4x^1 + 1x^0$ 2 3 1.3 x1,x2,x3	47 p1+p2,with x = x1 11.01 The derivative of p1,with x = x3 48 The integral of p2 from x1 to x2

Submission

- Ask TAs for demo
  1. TAs will check if the makefile can be executed properly and it requires the use of multiple targets.
  2. Show TA OJ results
     1. /home/share/demo_OOP112_2 Lab 07
     2. Executable file name: Lab07
  3. TA will check if the getvalue() is used correctly
- Try your best to debug your code by yourself
- Compress all your cpp, makefile and header file to zip and upload to new E3
- Naming rule : studentID_lab7.zip