南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

C/C++ Program Design

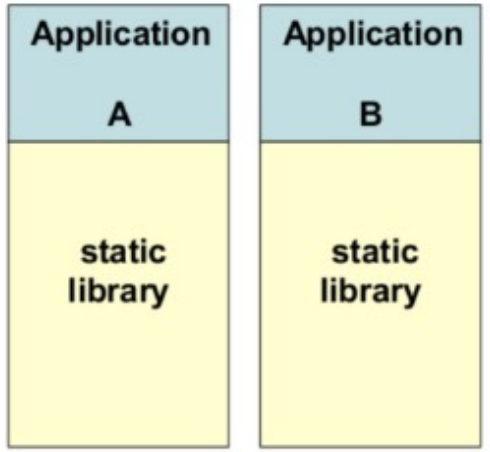
Lab 6, static library

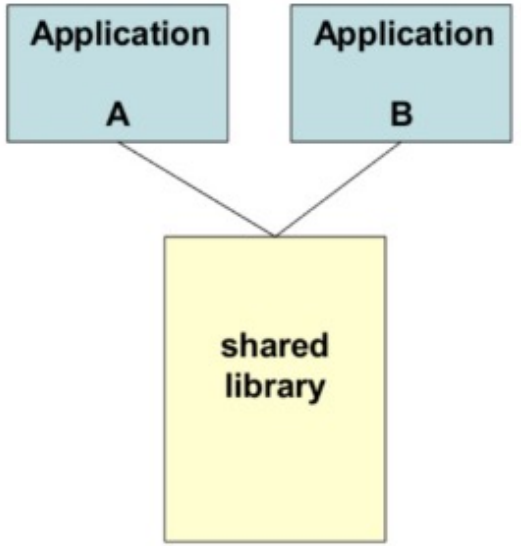
廖琪梅，王大兴

Static library and Dynamic library

Static Linking and Static Libraries (also known as an **archive**) is the result of the linker making copy of all used library functions to the executable file. Static Linking creates larger binary files, and need more space on disk and main memory. Examples of static libraries are, ***.a*** files in Linux and ***.lib*** files in Windows.

Dynamic linking and Dynamic Libraries Dynamic Linking doesn't require the code to be copied, it is done by just placing name of the library in the binary file. The actual linking happens when the program is run, when both the binary file and the library are in memory. If multiple programs in the system link to the same dynamic link library, they all reference the library. Therefore, this library is shared by multiple programs and is called a "**shared library**" . Examples of Dynamic libraries are, ***.so*** in Linux and ***.dll*** in Windows.





	advantages	disadvantages
Static Library	1. Make the executable has fewer dependencies, has been packaged into the executable file. 2. The link is completed in the compilation stage, and the code is loaded quickly during execution.	1. Make the executable file larger. 2. Being a library dependent on another library will result in redundant copies because it must be packaged with the target file. 3. Upgrade is not convenient and easy. The entire executable needs to be replaced and recompiled.
Dynamic Library	1.Dynamic library can achieve resource sharing between processes, there can be only one library file. 2. The upgrade procedure is simple, do not need to recompile.	1. Loading during runtime will slow down the execution speed of code. 2. Add program dependencies that must be accompanied by an executable file.

Building a static library

- Suppose we have written the following code:

```
// mymath.h
#ifndef __MY_MATH_H__
#define __MY_MATH_H__
float arraySum(const float *array,
size_t size);
#endif
```

```
// mymath.cpp
#include <iostream>
#include "mymath.h"

float arraySum(const float *array, size_t
size)
{
    if(array == NULL)
    {
        std::cerr << "NULL pointer!" <<
std::endl;
        return 0.0f;
    }
    float sum = 0.0f;
    for(size_t i = 0; i < size; i++)
        sum += array[i];
    return sum;
}
```


```cpp
// main.cpp
#include <iostream>
#include "mymath.h"
int main()
{
    float arr1[8]{1.f, 2.f, 3.f, 4.f, 5.f, 6.f, 7.f, 8.f};
    float * arr2 = NULL;

    float sum1 = arraySum(arr1, 8);
    float sum2 = arraySum(arr2, 8);

    std::cout << "The result1 is " << sum1 << std::endl;
    std::cout << "The result2 is " << sum2 << std::endl;

    return 0;
}
```

Building a static library

- In previous class we do the following:
- This will compile the “main.cpp” and “mymath.cpp” into “main”
- And then run “main”

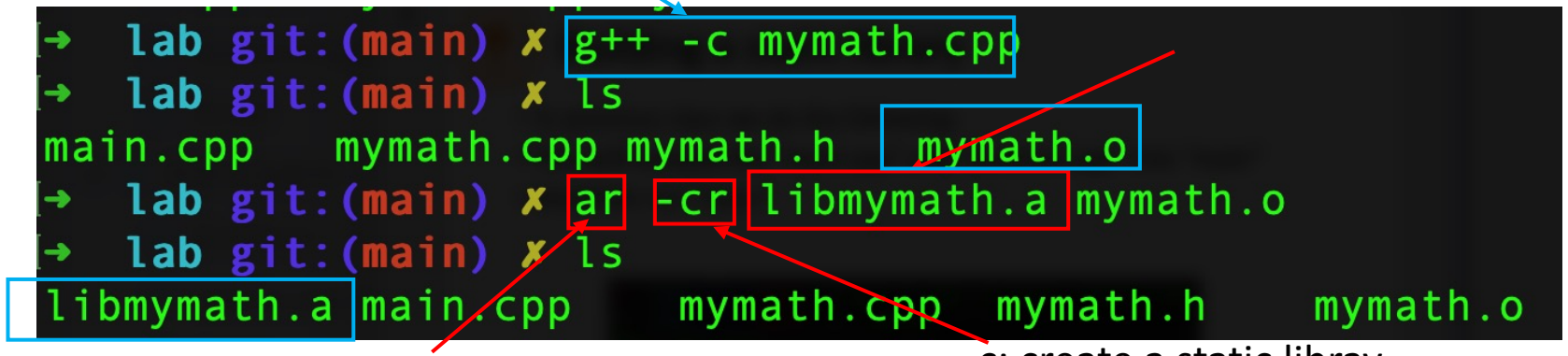
```
➜  lab git:(main) ✗ g++ *.cpp -o main -std=c++11
➜  lab git:(main) ✗ ./main
NULL pointer!
The result1 is 36
The result2 is 0
```

Building a static library

- A static library is created by **.o** file.
- Remember to use **“ar”** command with arguments “**-cr**” when building it.
- Now we should see “libmymath.a” in the current directory

Compile the source file to the object file.

The name of **.a** must be started with “**lib**” followed by the .cpp name in which a function is defined.



ar is a linux command.

c: create a static libray.
r: add the object file to the static library.

Using a static library

- Now we can use ".a" static library.
- Let's compile "main" again:

"-lmymath" indicates to use "libmymath.a" or "libmymath.so"

"-L." indicates to find a library file in the current directory.

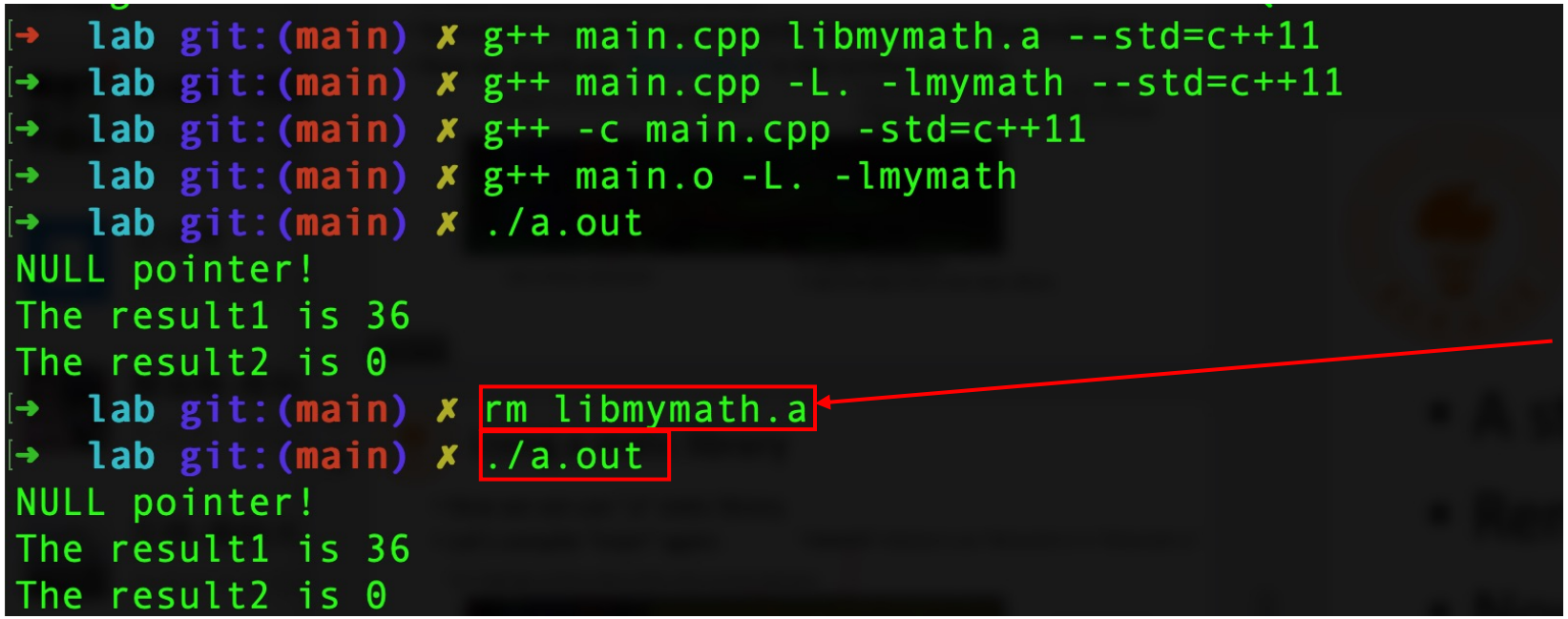
```
➜  lab git:(main) ✗ g++ main.cpp libmymath.a --std=c++11
➜  lab git:(main) ✗ g++ main.cpp -L. -lmymath --std=c++11
➜  lab git:(main) ✗ g++ -c main.cpp -std=c++11
➜  lab git:(main) ✗ g++ main.o -L. -lmymath
➜  lab git:(main) ✗ ./a.out
NULL pointer!
The result1 is 36
The result2 is 0
```

The 3 methods are equivalent.

- **-L**: indicates the directory of libraries
- **-l**: indicates the library name, the compiler can give the "**lib**" prefix to the library name and follows with **.a** as extension name.

Using a static library

If the static library is removed, the program can run normally.



remove the static library file.

To create a static library from multiple object files:

```
ar -cr libtest.a test1.o test2.o
```

Exercise 1

```
#include <iostream>
using namespace std;
#define SIZE 5
int main()
{
    int const *pa = new int[SIZE]{3,5,8,2,6};
    int total = sum(pa,SIZE);
    cout << "sum = " << total << endl;
    return 0;
}
int sum(const int *pArray, int n)
{
    int s = 0;
    for(int i = 0; i < n; i++)
        s += pArray[i];
     return s;
}
```

Fix bugs of the program and run it correctly
without memory leak.

Note: In a function with pointer parameters, the pointers should be checked first. If there is no such statement, please add it.

Exercise 2

```
#include <iostream>
using namespace std;
int * create_array(int size)
{
    int arr[size];

    for(int i = 0; i < size; i++)
        arr[i] = i * 10;
    return arr;
}
int main()
{
    int len = 16;
    int *ptr = create_array(len);
    for(int i = 0; i < len; i++)
        cout << ptr[i] << " ";
    return 0;
}
```

What compilation warnings occur when you compile the program? Why?
What will happen if you ignore the warning and run the program?
Fix bugs of the program and run it correctly without memory leak.

Exercise 3

```
#include <iostream>
#define SIZE 5
void sum(const int *, const int *, int);
int main()
{
    int a[SIZE] = {10,20,30,40,50};
    int b[SIZE] = {1,2,3,4,5};
    std::cout << "Before calling the function, the
contents of a are:" << std::endl;
    for(int i = 0; i < SIZE; i++)
        std::cout << a[i] << " ";
    // passing arrays to function
    sum(a,b,SIZE);

    std::cout << "\nAfter calling the function, the
contents of a are:" << std::endl;
    for(int i = 0; i < SIZE; i++)
    std::cout << a[i] << " ";
    std::cout << std::endl;
    return 0;
}
void sum(const int *pa, const int *pb, int n)
{
    for(int i = 0; i < n; i++)
    {
        *pa += *pb;
        pa++;
        pb++;
    }
}
```

Can the program be run correctly?

How to fix the bugs?

Exercise 4

Define a function that swaps two values of integers. Write a test program to call the function and display the result.

You are required to compile the function into a static library “libswap.a”, and then compile and run your program with this static library.