9 内存模型和名称空间

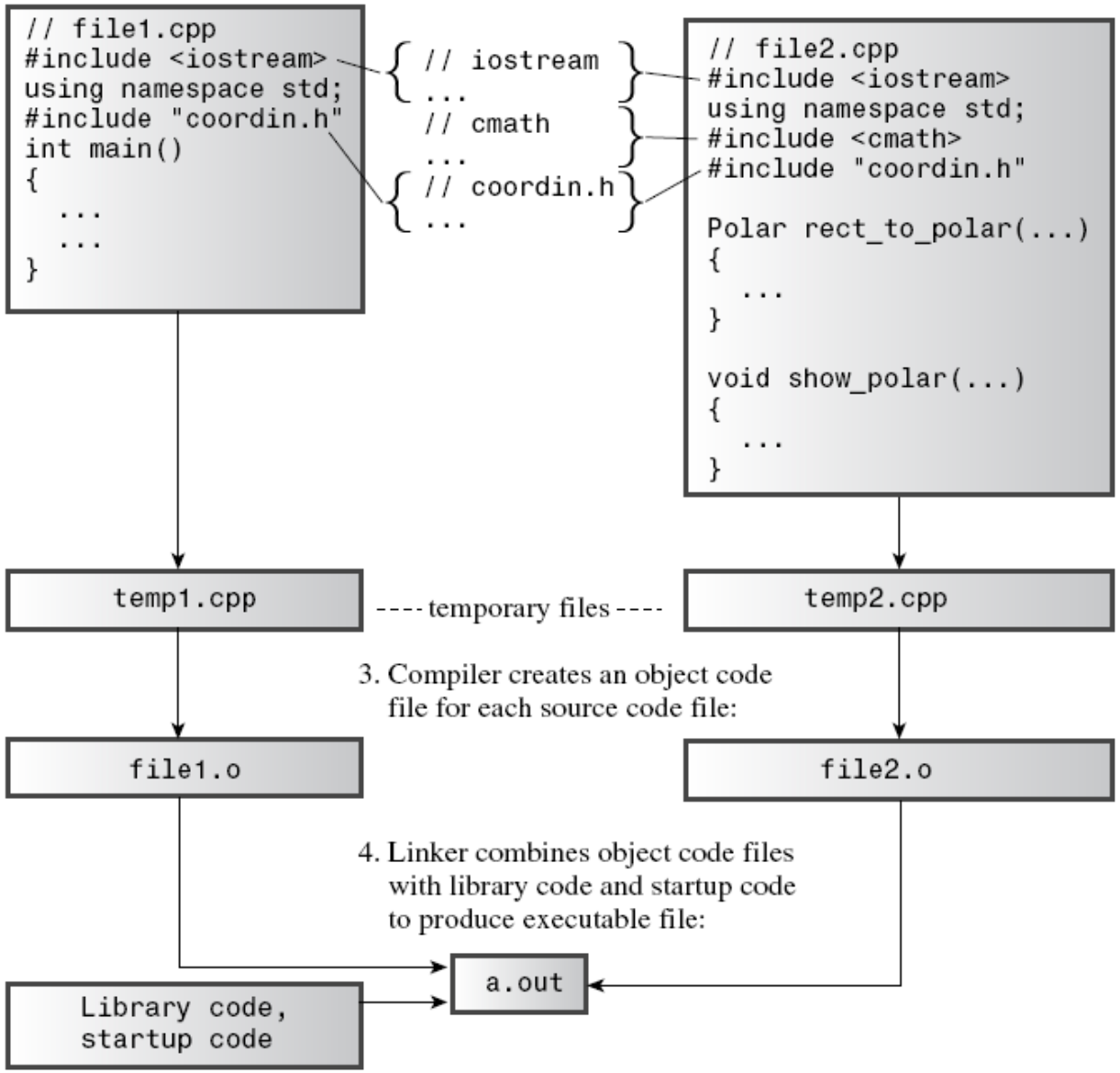
1．单独编译

- 头文件
  - 声明：结构体，类，函数）
  - 不要包含函数定义和变量声明
- 源代码（实现）
  - 变量定义和调用
  - 函数实现
- 源代码（应用）
  - 变量定义和调用
  - 函数调用

1. Give UNIX compile command for two source files:

```
CC file1.cpp file2.ccp
```

2. Preprocessor combines included files with source code:



多个文件编译

- (P9.1 coordin.h P9.2 file1.cpp P9.3 file2.cpp)
- 头文件管理

2．存储持续性、作用域和链接性

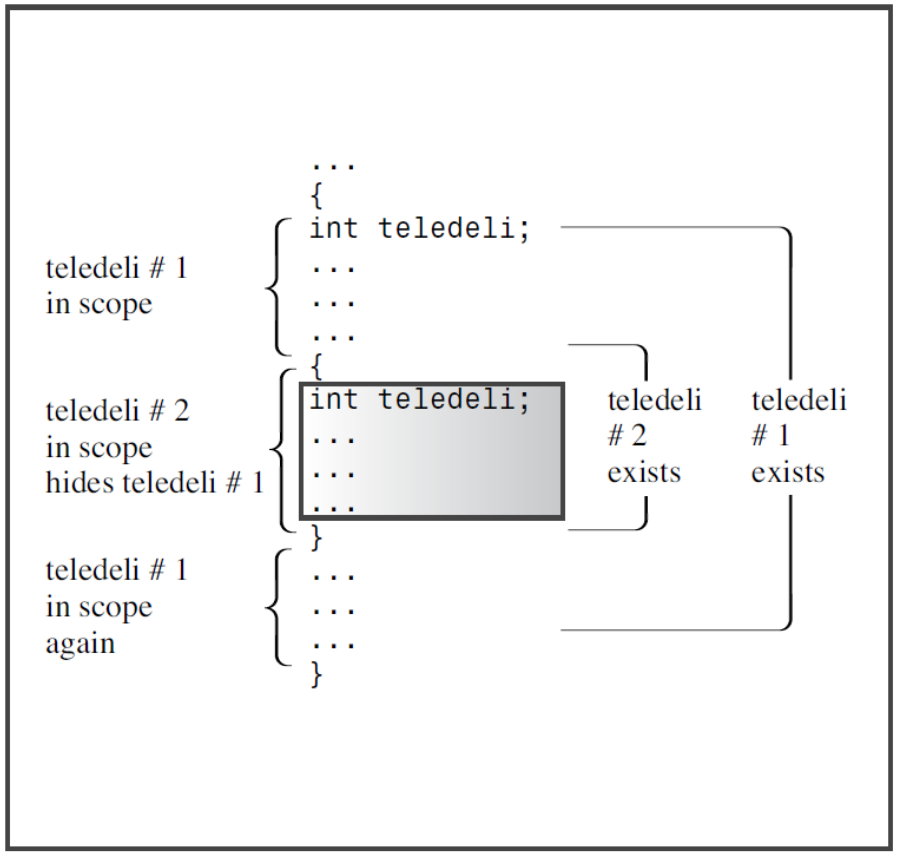
- 内存方案：存储类别如何影响信息在文件间共享
- 存储持续性：指一个变量在内存中存在的时间
- 自动存储持续性：自动的
  - 函数定义中：局部变量，函数参数
  - 执行到时创建，执行完时释放
- 静态存储持续性
  - 函数定义外部定义的变量；static定义的变量
  - static，一旦运行就存在
- 线程存储持续性(PASS)
- 动态存储持续性
  - new申请，直到delete删除（自由存储/堆）

2.1 作用域和链接

- 作用域
  - 描述名称在编译单元在多大范围内可见。
  - 即一个标识符在什么范围起作用（可以被识别和使用）
  - 全局变量，名字空间：全局
  - 代码块{}：局部。函数原型，自动变量
  - C++函数：整个名称空间（全局，类）
- 链接性
  - 名称如何在不同单元间共享

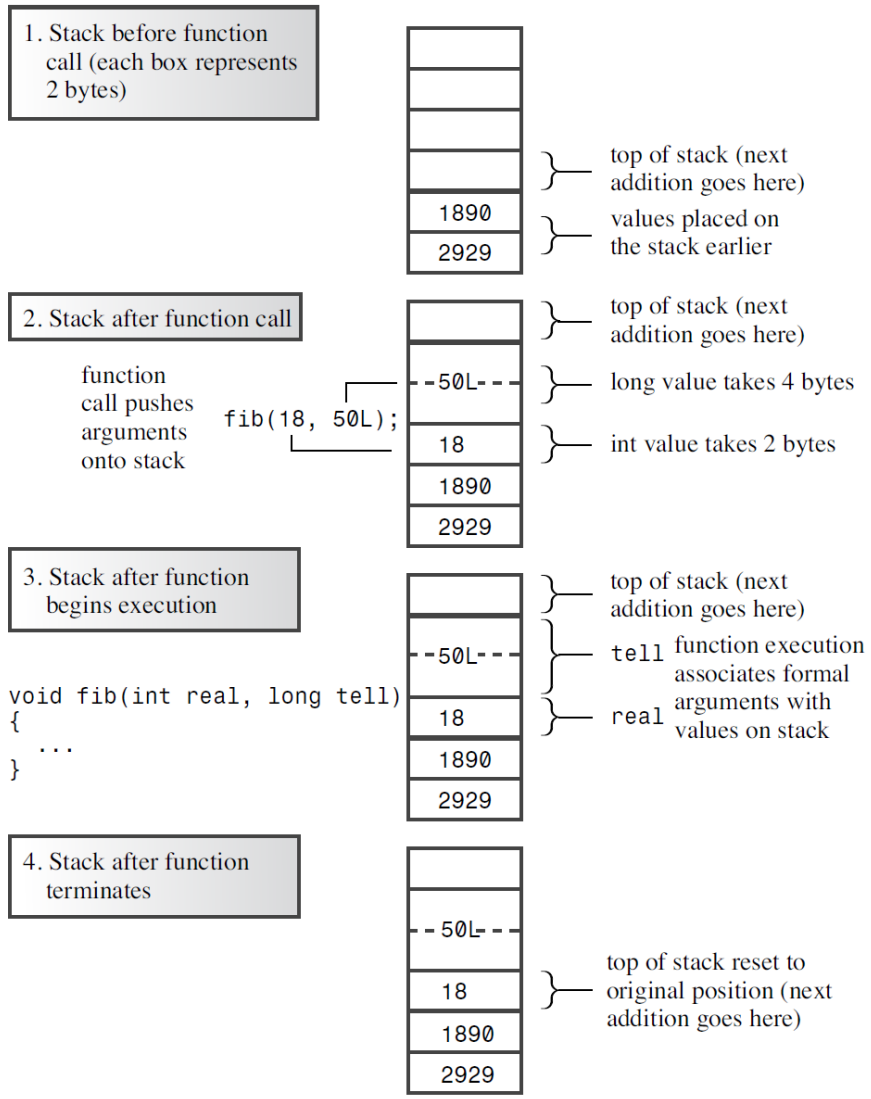
2.2 自动存储持续性【生命周期】

- 函数中声明的参数和变量
  - 存储持续性为自动
  - 作用域为局部，没有链接性
- 代码块中的变量
  - 存在时间和作用域被限制在该代码块内。
  - 同名变量：新定义隐藏以前的定义
- 自动变量初始化
  - 声明时可以初始化
- 自动变量和栈关联
- P9.4 autoscp.cpp



2.3 静态持续变量

- 三种链接性
  - 外部（其他文件可访问）
  - 内部（当前文件访问）
  - 无



2.4 静态持续性、外部链接性

- 外部变量：链接性为外部的变量
  - 静态，作用域整个文件
- 变量声明
  - 定义声明，定义：分配存储空间
  - 引用声明，声明：不分配存储空间。
    - extern
- P9.5 external.cpp P9.6 support.cpp

```cpp
// file1.cpp
#include <iostream>
using namespace std;

// function prototypes
#include "mystuff.h"

// defining an external variable
int process_status = 0;

void promise ();
int main()
{
   ...
}

void promise ()
{
   ...
}
```

This file defines the variable `process_status`, causing the compiler to allocate space for it.

```cpp
// file2.cpp
#include <iostream>
using namespace std;

// function prototypes
#include "mystuff.h"

// referencing an external variable
extern int process_status;

int manipulate(int n)
{
   ...
}

char * remark(char * str)
{
   ...
}
```

This file uses `extern` to instruct the program to use the variable `process_status` that was defined in another file.

2.5 静态持续性、内部链接性

- P9.7 twofile1.cpp P9.8 twofile2.cpp
- static作用域为整个文件
  - 链接性为内部

2.6 静态存储持续性、无链接性

- P9.9 static.cpp
- static作用于代码块

2.7 说明符和限定符

- 存储说明符
  - auto（在C++11 中不再是说明符）
  - register
  - static
  - extern
  - thread_local（C++11 新增的）
  - mutable

2.8 函数和链接性

- static函数

2.9 语言链接性

- C++语言链接性
- extern “C” void spiff(int)

2.10 存储方案和动态分配

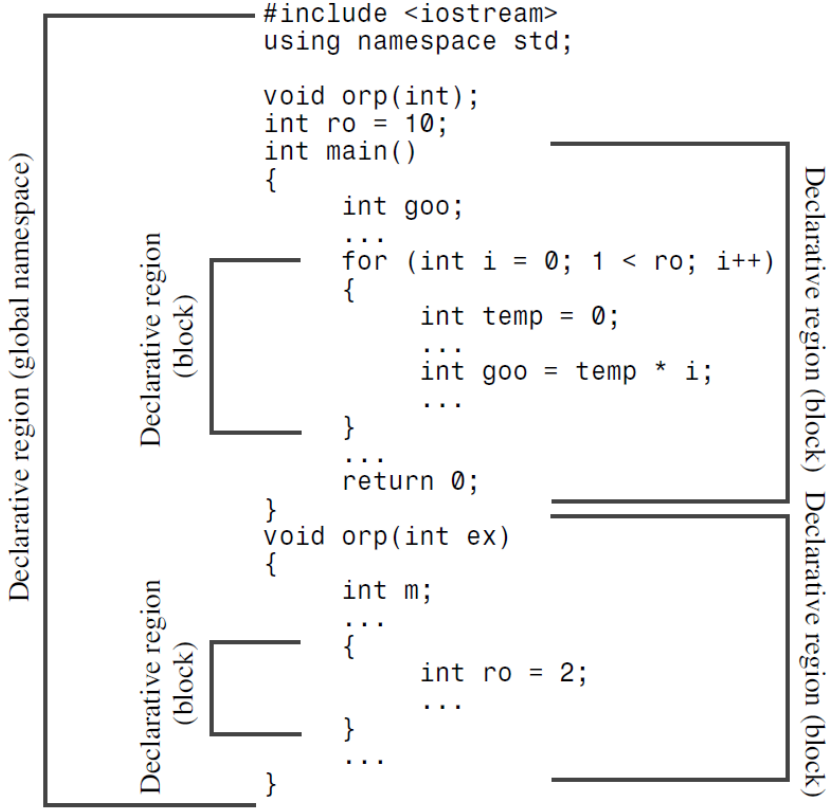
P9.10 newplace.cpp

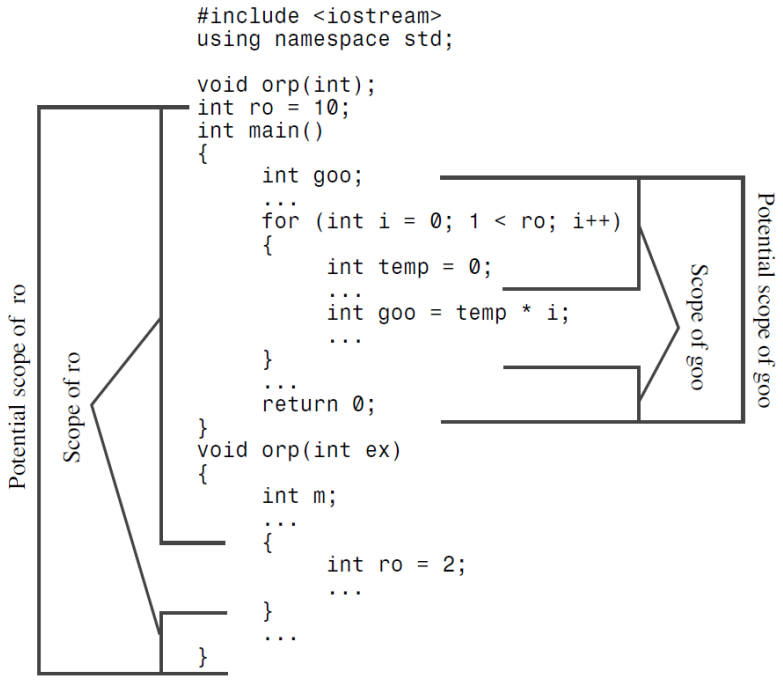
3 名称空间

- 不同模块可能使用相同标识符
- namespace

3.1 传统的c++名称空间

- 声明区域
  - 可以进行声明的区域
- 潜在作用域
  - 声明点开始，到声明区域的结尾
- 作用域





3.2 新的名称空间特性

- namespace
- 定义
- 使用
  - 作用域解析运算符::
  - using
    - 可能导致重名而覆盖

3.3 名称空间示例

- P9.11 namesp.h P9.12 namesp.cpp P9.13 namessp.cpp
- 多文件，多名称空间
- using debts::Debt;
- using namespace debts;

3.4 名称空间及其前途

- 指导原则
  - 简化大型编程项目的管理工作

4. 总结