一、常见问题

`sizeof` 运算符：

- `sizeof` 是一个编译时求值的运算符，用于获取数据类型或对象的字节大小。
- 在编译阶段，编译器会根据数据类型或对象的定义确定其大小，并在编译时期计算 `sizeof` 表达式的值。
- 由于 `sizeof` 是在编译时期确定的，因此其结果不依赖于程序运行时的具体情况。

函数重载的机制：

- 函数重载是一种**编译期**的特性，它允许在同一作用域中声明多个同名函数，但它们的参数类型或个数不同。
- 在编译阶段，编译器根据函数调用的上下文和参数的类型、数量等信息来确定调用哪个重载函数。
- 因此，函数重载的决定是在编译期进行的，而不是在运行时。

Hash 表的 Rehash 代价高的解决方案：

- 一种解决方案是采用动态扩容策略，即当 Hash 表的负载因子达到某个阈值时，触发扩容操作。
- 另一种解决方案是选择更好的 Hash 函数和初始容量，以尽量减少 Rehash 的发生。
- 还可以考虑采用更高效的 Hash 表实现，如 Cuckoo Hashing、Hopscotch Hashing 等，以降低 Rehash 的代价。

解析 HTTP 请求时的数据未完全接收的情况：

- 当用户一次性没有传输完整的 HTTP 请求数据时，服务器可以采取以下策略：
  - 通过设置合理的超时时间来等待数据完整传输，超时后进行错误处理或关闭连接。
  - 在接收到部分数据后，持续等待直到数据完整传输，或者根据 HTTP 请求头中的 Content-Length 字段来判断数据是否完整。
  - 在收到数据后立即进行处理，但需要实现缓冲机制来存储未完整的数据，等待后续数据的到达，然后再进行完整的解析和处理。
  - 如果无法确定数据是否完整，可以返回错误响应或等待更多数据的到达。
- 对于长连接的情况，也可以考虑使用分块传输编码（chunked transfer encoding）等方式来处理不完整的数据。

指针为什么危险

指针可以运算（++、--）从而指向未知的内存；

右值引用

右值引用的特点之一是可以延长右值的生命周期、`减少对象的复制，提升程序性能`

```
X make_x()
{
  X x1;
  return x1;
}
int main()
{
  X &&x2 = make_x();
  x2.show();
}
```

若无右值：构造、return-拷贝构造； X x2 = make_x();---又拷贝构造；

若有右值：构造、return-拷贝构造； X x2 = make_x();---无拷贝构造；

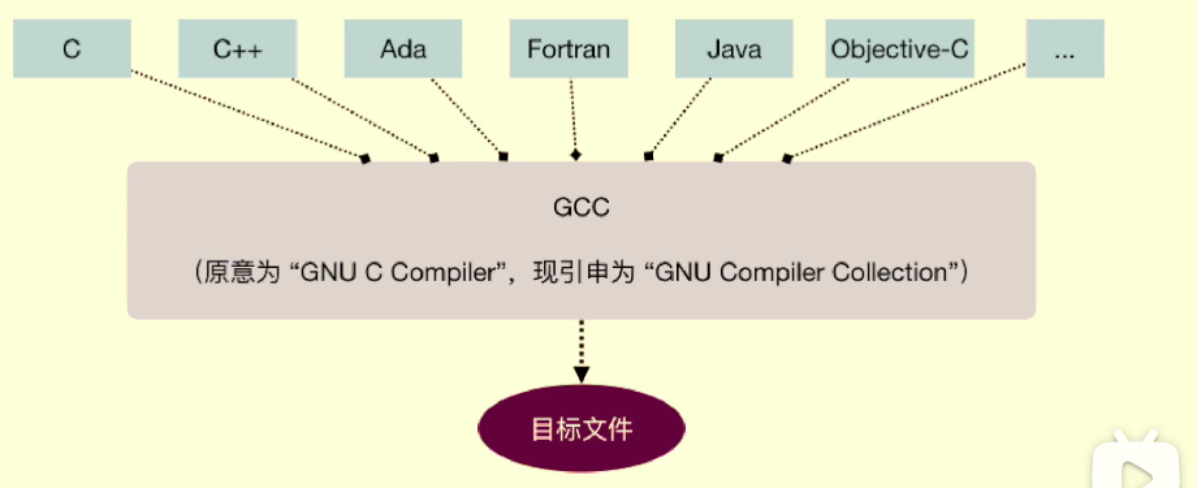
在理解这段代码之前，让我们想一下如果将x &&x2 = make_x0这句代码替换为x x2 = make x0会发生几次构造。

在没有进行任何优化的情况下应该是3次构造

- 首先make x函数中x1会默认构造一次，
- 然后return x1会使用复制构造产生临时对象，
- 接着x x2 = make x0会使用复制构造将临时对象复制到x2，最后临时对象被销
  毁。

```
int&& a = 1; // 右值引用
int b = 2;
int&& c = move(b); // 左值转右值
```

GCC



如果操作系统是类 Unix 系统，那么很有可能已经安装了 GCC。你可以在命令行终端键入 cc --version 命令来查看版本信息

1 预处理阶段

- 在`预处理阶段`，GCC 会对 C 语言源代码中以 # 开始的代码行进行处理，`对宏进行替换。`
- 通常，C 语言程序的预处理输出文件是以 .i结尾的文件。我们可以用 GCC 的 -E 选项生成预外理输出文件 (如果不加 -E 选项，GCC 默认将输出写到标准输出)，例如:

```
$ gcc -E -o simple example.i simple example.c
```

2 编译阶段

- 在`编译阶段`，GCC 会将 C 语言程序根据目标 CPU 架构 (默认是宿主机，即编译源代码的主机)生成对应的汇编语言文件
- 生成的汇编语言文件通常`以 s 结尾`，如果想要查看这个文件，我们可以使用`GCC的 -S 选项`，例如:

```
$gcc -S simple_example.c
```

3 汇编阶段

- 在汇编阶段，GCC 会调用宿主机的汇编器把汇编语言文件翻译成二进制文件，该文件是一个对象文件，其中包含机器码以及描述外部链接对象的符号表。
- 如果直接生成目标程序，每个 C 语言源代码文件对应的对象文件都是临时文件。我们可以用GCC 的 -c选项单独生成对象文件，例如:

```
$ gcc -c simple_example.c
```

4链接阶段

- 如果直接用汇编阶段生成的对象文件组成可执行程序，
- 其中一些片段的顺序是错误的，并且还有可能缺少一些对象 (比如，simple example 中的 printf 需要链接标准链接库，其中大部分内容在静态库 libc.a或者动态链接库 libc.so中)，
- 因此需要将它们重新排序并补充完整。这个过程就称为链接。

默认情况下，GCC 生成的可执行程序名称是 a.out，我们可以用 -o 选项指定自己想要的名字，例如:

```
$ gcc -o app simple_example.c
```

此后，我们就可以用 app 这个名字来调用这个 C 语言程序了

```
$./app
```