代码如诗

小楼一夜听春雨

C Runtime Library、C Runtime

1)运行时库就是 C run-time library，是 C 而非 C++ 语言世界的概念:取这个名字就是因为你的 C 程序运行时需要这些库中的函数.

2)C 语言是所谓的“小内核”语言，就其语言本身来说很小（不多的关键字，程序流程控制，数据类型等）；所以，C 语言内核开发出来之后，Dennis Ritchie 和 Brian Kernighan 就用 C 本身重写了 90% 以上的 UNIX 系统函数，并且把其中最常用的部分独立出来，形成头文件和对应的 LIBRARY，C run-time library 就是这样形成的。

3)随后，随着 C 语言的流行，各个 C 编译器的生产商/个体/团体都遵循老的传统，在不同平台上都有相对应的 Standard Library，但大部分实现都是与各个平台有关的。由于各个 C 编译器对 C 的支持和理解有很多分歧和微妙的差别，所以就有了 ANSI C；ANSI C （主观意图上）详细的规定了 C 语言各个要素的具体含义和编译器实现要求，引进了新的函数声明方式，同时订立了 Standard Library 的标准形式。所以C运行时库由编译器生产商提供。至于由其他厂商/个人/团体提供的头文件和库函数，应当称为第三方 C 运行库（Third party C run-time libraries）。

4)C run-time library里面含有初始化代码，还有错误处理代码(例如divide by zero处理)。你写的程序可以没有math库，程序照样运行，只是不能处理复杂的数学运算，不过如果没有了C run-time库，main()就不会被调用，exit()也不能被响应。因为C run-time library包含了C程序运行的最基本和最常用的函数。

5)到了 C++ 世界里，有另外一个概念:Standard C++ Library,它包括了上面所说的 C run-time library 和 STL。包含 C run-time library 的原因很明显，C++ 是 C 的超集，没有理由再重新来一个 C++ run-time library. VC针对C++ 加入的Standard C++ Library主要包括：LIBCP.LIB, LIBCPMT.LIB和 MSVCPRT.LIB

6)Windows环境下，VC提供的 C run-time library又分为动态运行时库和静态运行时库。
动态运行时库主要是DLL库文件msvcrt.dll(or MSVCRTD.DLL for debug build),对应的Import library文件是MSVCRT.LIB(MSVCRTD.LIB for debug build)
静态运行时库(release版)对应的主要文件是：
LIBC.LIB (Single thread static library, retail version)
LIBCMT.LIB (Multithread static library, retail version)

msvcrt.dll提供几千个C函数，即使是像printf这么低级的函数都在 msvcrt.dll里。其实你的程序运行时，很大一部分时间时在这些运行库里运行。在你的程序(release版)被编译时，VC会根据你的编译选项（单线程、多线程或DLL)自动将相应的运行时库文件（libc.lib,libcmt.lib或Import library msvcrt.lib)链接进来。

编译时到底哪个C run-time library联入你的程序取决于编译选项：
/MD, /ML, /MT, /LD (Use Run-Time Library)
你可以VC中通过以下方法设置选择哪个C run-time library联入你的程序：
To find these options in the development environment, click Settings on the Project menu. Then click the C/C++ tab, and click Code Generation in the Category box. See the Use Run-Time Library drop-down box.

公告

昵称: 小 楼 一 夜 听 春 雨
园龄: 11年8个月
粉丝: 695
关注: 3
+加关注

‹ 2009年5月 ›

日	一	二	三	四	五	六
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类

C/C++(135)
D3D(116)
deep learning(4)
GO(2)
linux(119)
OGRE(10)
Python(34)
Windows(9)
架构设计(6)
开源(20)
其他(44)
数据库(3)
算法&数据结构(15)
网络(52)
英语(6)
更多

随笔档案

2020年11月(1)
2020年6月(2)
2020年5月(1)
2020年1月(1)

从程序可移植性考虑,如果两函数都可完成一种功能，选运行时库函数好,因为各个 C 编译器的生产商对标准C Run-time library提供了统一的支持.

C Runtime Library来历, API, MFC, ATL关系

CRT原先是指Microsoft开发的C Runtime Library，用于操作系统的开发及运行。后来在此基础上开发了C++ Runtime Library，所以现在CRT是指Microsoft开发的C/C++ Runtime Library。在VC的CRT/SRC目录下，可以看到CRT的源码，不仅有C的，也有C++的。

CRT原先的目的就是支持操作系统的运行。因为Windows操作系统除汇编部分外，都是用C/C++编写的，所以内核及许多关键服务都在CRT上运行 （它们都采用dll技术动态链接）。此外，用 VC编写的C/C++程序也用到它们（可以动态链接，也可以静态链接，前者运行时需要系统中已安装CRT的dll，后者不需要）。可以说，CRT就是 Microsoft编写Windows时使用的低层类库。然后，它又被当作C++标准库的一个实现包含在了VC系列中；我们在VC中使用的C++标准库， 其实就是CRT的一个真子集（少了C++标准所不包含的代码，特别是大量的低层C代码）

至于CRT与WINDOWS API的关系，与许多人理解的相反，WINDOWS API作为Windows的一部份，是在CRT的基础上开发的。你可以将Windows（及其API）看作一个项目，而这个项目使用的语言是汇编/C/C ++，使用的类库就是CRT。所以，离开CRT，Windows API也无法使用的。

C++标准，是C++的通用语言规范，指导所有C ++使用者。而CRT的其中一部分可以看作是Microsoft开发的一个C++标准库实现（其实也确实如此，Microsoft在开发CRT时，参考了 正在标准化过程中的C++语言规范）。它与C++标准有一定的差距，部分原因是，在C++没有完成标准化之前，CRT已经开发并投入使用了。为了向下兼容 以前的Windows代码，早期的CRT与C++标准总有一定的差距。但是CRT确实在不断的改进中。VC6带的CRT与C++标准还有比较大的差距，而 VC8的几乎完全符合C++标准了。

综上，CRT（Microsoft's C/C++ Runtime Library）的一个真子集（主要是C++ Runtime Library）是一个符合（或至少是企图符合）C++标准的C++库。而Windows API（以及Windows的其他许多部分）都是在CRT的基础上开发的。

除了以上介绍的，在使用CRT的过程中，还需要了解的是：

1、CRT的一些组成部分也调用了Windows API。这可能就是有人认为CRT是建立的Windows API基础上的原因。但是实际上，这一部分剥离CRT没有任何的问题。只不过Microsoft将在Windows平台上可以使用的C/C++低层库都加 入到CRT中。因此，CRT中很大一部分是操作系统平台无关的（原始的CRT），是开发Windows本身及其上一切的基础。它们也可以作为一个C/C++库在其他操作系统平台上使用。还有一部分，则是和Windows紧密绑定的，调用Windows API来实现的，可以看作扩展的CRT。之所以将这两部分放在一起，是因为它们都是开发Windows操作系统所需要的，也因为它们也都是Windows 平台上的C/C++程序员所需要的。这种复杂关系是Microsoft的人为因素造成的，不能因此认为CRT是建立在Windows或Windows API基础上的。

2、CRT的大部分内容是跨硬件平台的，但是也有一些部分，是直接用汇编写成、基于硬件平台、并根据特定硬件平台做的优化（而不是将生成机器码的责任完全 交给编译器）。如早期对Indel的x32做了优化，现在由加入对AMD64的优化，这部分则是不跨硬件平台的。

关于ATL

ATL是建立在CRT上的，如果你看了ATL的源码就知道了。至于不用链接，是因为ATL库静态链接了CRT，所以它可以在CRT之外运行。类似这样的误 解在于混淆了作为低层基本库的CRT和作为产品而附带在VC中的CRT。虽然这两者是同样的代码，但是概念是不一样的。

在编写操作系统时，你需要一个合适的低层库，以便完成一些基本的、多次重复的工作。于是，就有了CRT。在最低层的时候，根本连dll这个概念都没有的， 所以CRT的源代码只能做成lib，被静态链接。然后，随着Windows越做越复杂，Microsoft提出了API的概念，它提供Windows开发 者一组接口，可以直接操作Windows，这就是Windows API了。当然，Windows API也是在CRT之上编写的。

接着， Microsoft想给予C/C++程序员以足够的支持，除了原始CRT之外，还要增加在Windows平台上编程所特有的东西，如thread等等。这 些东西都是和平台相关的，只能建立在Windows API上。而这些新增内容，也被放进了CRT中。此时，CRT不仅仅包含最低层平台无关的代码，还包括平台相关的部分。如你调用CRT的 _beginthread，其实内部调用了Windows API的

2019年11月(1)
2019年9月(5)
2018年6月(1)
2018年5月(7)
2018年3月(3)
2018年2月(1)
2018年1月(13)
2017年12月(3)
2017年11月(6)
2017年10月(5)
2017年9月(1)
更多

useful

陈硕的Blog

最新评论

1. Re:如何读书？ 我一年读500本书，你呢?
《如何读一本书》
--一片-枫叶
2. Re:如何读书？ 我一年读500本书，你呢?
读书之事，随性而为之 何必刻意为之 开心就好
--全力以赴001
3. Re:python模块及包的导入
学到知识，感谢解惑
--始不垂翅
4. Re:linux文件锁flock
flock是放在inode里的，不是放在文件描述符里的，file_struct 只是一个文件的打开上下文，锁一个文件理应锁在他的本体结构上，也就是 inode 上，而多个进程可以打开多个 上下文，也就...
--执生
5. Re:C/C++服务器开发的必备利器–libconfig
你好。这个库如何生成.a静态链接库?
--naiquan

阅读排行榜

1. linux 创建连接命令 ln -s 软链接(512823)
2. Linux 中如何卸载已安装的软件(148637)
3. sscanf，sscanf_s及其相关用法(132546)
4. Jsoncpp的使用(105231)
5. 虚拟机与Docker有何不同(78544)

评论排行榜

1. C++游戏开发需要阅读的书籍(10)
2. 静态链接与动态链接的区别(8)
3. CxImage(6)
4. Python中的__new__()方法的使用(5)
5. D3D中的世界矩阵，视图矩阵，投影矩阵(5)

推荐排行榜

1. 静态链接与动态链接的区别(27)
2. TCP粘包问题分析和解决（全）(20)
3. sscanf，sscanf_s及其相关用法(16)
4. linux 创建连接命令 ln -s 软链接(13)
5. 虚拟机与Docker有何不同(11)

CreateThread。加入这些东西后，CRT仍然被用作编写操作系统；但是显然，那些调用了Windows API的部分已经失去移值性了。

然后，CRT被封装成产品，随编译器一起发布。此时CRT产品的LIB和DLL都是Windows格式的，你不能在Windows以外的平台上使用EXE 或DLL吧，这就是CRT和CRT产品的区别。Windows API的产品，或是Windows的其他许多组成部分也是一些LIB/DLL文件，这些都是表面的东西，是与Windows绑定在一起的。但是，如果你认 为是先有Windows或Windows API，才有CRT的，那你就本末倒置了。除非你对CRT的定义就是那些LIB/DLL产品，而不包括用来产生它们的代码。

就象C++编译器用来编译用C++写的编译器自身一样，Windows（及其上的编译器）用来作为平台开发和编译CRT，并也用CRT来写Windows 自身（当然第一个CRT和第一个用来编译Windows的编译器不是在Windows上开发的）。就象"我"也可以先写一个类库，然后在它基础上写一个操 作系统，在这个操作系统上进一步扩充这个类库，然后将它配合编译器发布出去，发展一些我的操作系统的支持者，顺便再赚点收入。或者以另一种模式发布另一个 库（只是我在原来那个库上开发的一个产品，由于我独立地发布这个新库，许多人会不知道这个新库与旧库的关系。这很好，因为编程本身就是尽量隐藏细节，尽量 做到对使用者透明的），吸引不同风格的开发者。这样我的付出得到了最大的回报——由于我没有发布操作系统的源代码，所以许多用户认为我不仅做了系统，还做 了编译器，还开发了一个类库。做了那么多事，回报是应该的。其实他们不知道，类库是编写操作系统所必须的，编译器也是必须的，这些必须的东西却可以在操作 系统之外获得更多的回报，真是太完美了！这是什么？这就是商业精神！当然这些误解对我是有好处的，我就不必到处宣扬真相了。反正我把类库的源码都发布了， 也没有骗过人吧。我不过是在那个原始类库中加进了一些与我的操作系统相关的东西，以方便在我的系统上编写程序的人们，这是我的好心吧；至于有人可能产生进 一步的误解，就不是我需要考虑的了......

所以还是看看CRT的源码吧——看看那些针对硬件平台的汇编；看看VC的标准C++库和CRT关系；再看看其他操作系统的源代码，想想CRT中的哪些部分 可以支持用来写操作系统，而如果我自己写系统，又需要哪些东西；甚至你可以看看DOS的源代码，想想和CRT的相似性，以及历史渊源。可惜不能看到 Windows的源代码，否则一切就清楚了。

最后再说一句，C++当然不是Microsoft的专利。但是Microsoft选择了C++，并取得了成功，这是肯定的了：象CRT，象VC，象 Windows，象Office，象 SQLServer......这一方面说明了C++的优势，一方面也是Microsoft自身的因素在起作用。然后，它当然要紧抓C++的大旗，大力宣 扬它自己的C++，并排斥其他的C++。这就是帝国的"风范"了。所以对Microsoft，总是即恨且爱，总希望哪天它会良心发现——当然这只是幻想罢 了。不过，肯定该肯定的，否定该否定的，总是应该的。但就产品而言，Microsoft不是最好的，但大多都是最成功的，在看到它的不足的同时，也要看到 它的优点。存在的即使不是合理的，也一定有它的合理性。所以，不能简单用一两句话评价Microsoft及它的成功。惟有一点是可以肯定的，它决定选择C ++，真是太英明了！

\- -

C Runtime Library

对于上面的这个太头困惑了很久，现在转贴一张

抄来的 共同学习哈

****

1)运行时库就是 C run-time library，是 C 而非 C++ 语言世界的概念:取这个名字就是因为你的 C 程序运行时需要这些库中的函数.

2)C 语言是所谓的"小内核"语言，就其语言本身来说很小（不多的关键字，程序流程控制，数据类型等）；所以，C 语言内核开发出来之后， Dennis Ritchie 和 Brian Kernighan 就用 C 本身重写了 90% 以上的 UNIX 系统函数，并且把其中最常用的部分独立出来，形成头文件和对应的 LIBRARY，C run-time library 就是这样形成的。

3)随后，随着 C 语言的流行，各个 C 编译器的生产商/个体/团体都遵循老的传统，在不同平台上都有相对应的 Standard Library，但大部分实现都是与各个平台有关的。由于各个 C 编译器对 C 的支持和理解有很多分歧和微妙的差别，所以就有了 ANSI C；ANSI C （主观意图上）详细的规定了 C 语言各个要素的具体含义和编译器实现要求，引进了新的函数声明方式，同时订立了 Standard Library 的标准形式。所以C运行时库由编译器生产商提供。至于由其他厂商/个人/团体提供的头文件和库函数，应当称为第三方 C 运行库（Third party C run-time libraries）。

4)C run-time library里面含有初始化代码，还有错误处理代码(例如divide by zero处理)。你写的程

序可以没有 math库，程序照样运行，只是不能处理复杂的数学运算，不过如果没有了C run-time 库，main()就不会被调用，exit()也不能被响应。因为C run-time library包含了C程序运行的最基本和最常用的函数。

5)到了 C++ 世界里，有另外一个概念:Standard C++ Library,它包括了上面所说的 C run- time library 和 STL。包含 C run-time library 的原因很明显，C++ 是 C 的超集，没有理由再重新来一个 C ++ run-time library. VC针对C++ 加入的Standard C++ Library主要包括：LIBCP.LIB, LIBCPMT.LIB和 MSVCPRT.LIB

6)Windows环境下，VC提供的 C run-time library又分为动态运行时库和静态运行时库。

动态运行时库主要是DLL库文件msvcrt.dll(or MSVCRTD.DLL for debug build),对应的Import library文件是MSVCRT.LIB(MSVCRTD.LIB for debug build)

静态运行时库(release版)对应的主要文件是：

LIBC.LIB (Single thread static library, retail version)

LIBCMT.LIB (Multithread static library, retail version)

msvcrt.dll提供几千个C函数，即使是像printf这么低级的函数都在msvcrt.dll里。其实你的程序运行时，很大一部分时间时在这些运 行库里运行。在你的程序(release版)被编译时，VC会根据你的编译选项(单线程、多线程或DLL)自动将相应的运行时库文件 (libc.lib,libcmt.lib或Import library msvcrt.lib)链接进来。

编译时到底哪个C run-time library联入你的程序取决于编译选项：

/MD, /ML, /MT, /LD (Use Run-Time Library)

你可以VC中通过以下方法设置选择哪个C run-time library联入你的程序：

To find these options in the development environment, click Settings on the Project menu. Then click the C/C++ tab, and click Code Generation in the Category box. See the Use Run-Time Library drop-down box.

从程序可移植性考虑,如果两函数都可完成一种功能，选运行时库函数好,因为各个 C 编译器的生产商对标准C Run-time library提供了统一的支持.

小 楼 一 夜 听 春 雨
关注 - 3
粉丝 - 695
+加关注

0 0

« 上一篇： 多线程学习笔记1
» 下一篇： 多线程笔记2

posted @ 2009-05-01 20:18 小 楼 一 夜 听 春 雨 阅读(790) 评论(0) 编辑 收藏 举报