所属系统：

文档类型：

**Lua系列**

**培训教程**

**Lua++学习指南**

|  |  |  |
| --- | --- | --- |
| 文件状态：  [√] 草稿  [ ] 正式发布  [ ] 正在修改 | 文件编号： |  |
| 当前版本： | 1.0 |
| 作 者： | 张子良 |
| 审 批： |  |
| 审批日期： | 2011-04-22 |

目 录

[第一章 tolua++基础 1](#_Toc291247314)

[1.1 tolua++简介 1](#_Toc291247315)

[1.2 tolua++资源 1](#_Toc291247316)

[1.3 tolua++编译 1](#_Toc291247317)

[1.3.1 tolua++.exe 1](#_Toc291247318)

[1.3.2 tolua++.lib 3](#_Toc291247319)

[1.4 tolua++实例 4](#_Toc291247320)

[1.4.1 Tarray 4](#_Toc291247321)

[第二章 Tolua++专题 7](#_Toc291247322)

[第三章 图表目录 8](#_Toc291247323)

# tolua++基础

## tolua++简介

tolua++ is an extended version of tolua,a tool to integrate c/c++ code with Lua. Tolua++ includes new features oriented to c++ such as:

* Support for std::string as a basic type
* Support for class templates

Tolua++用来解决Lua脚本文件调用C/C++变量、函数、类和方法的一个工具。

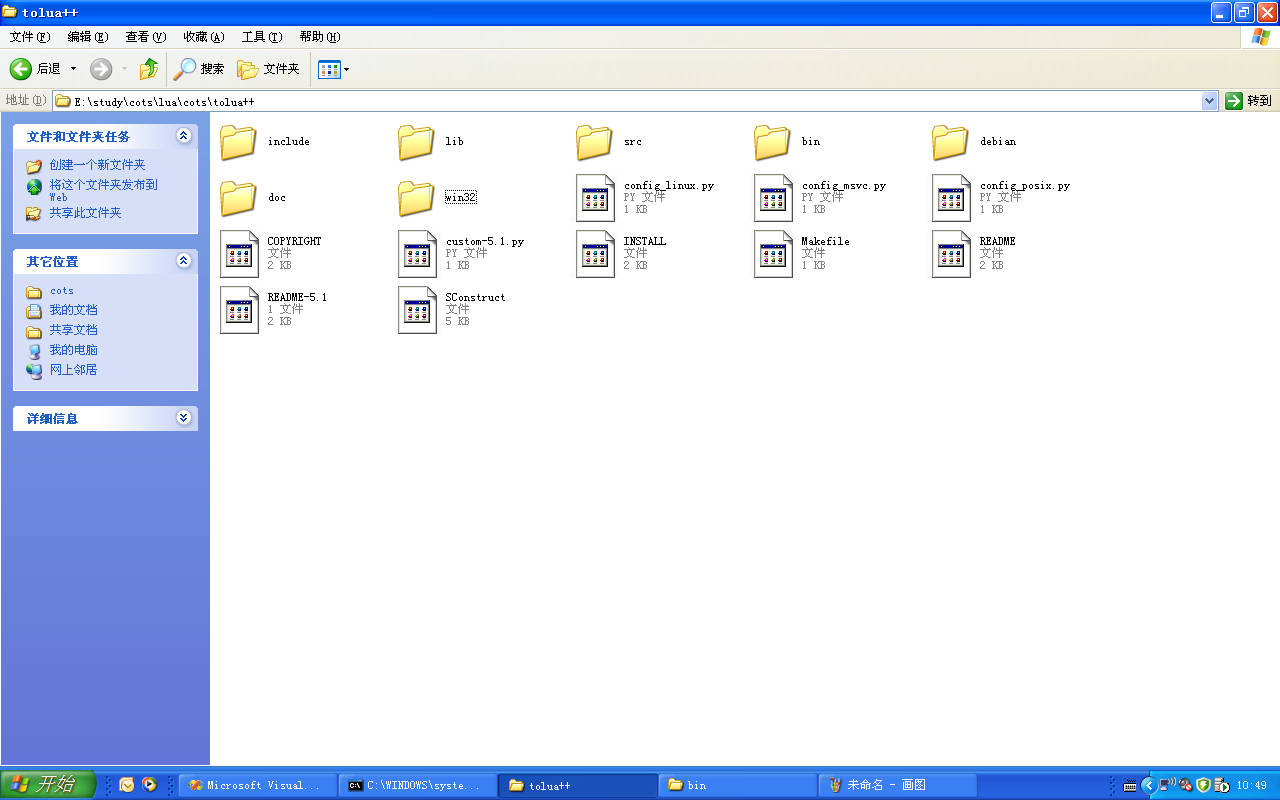
## tolua++资源

URL：http://www.codenix.com/~tolua/

## tolua++编译

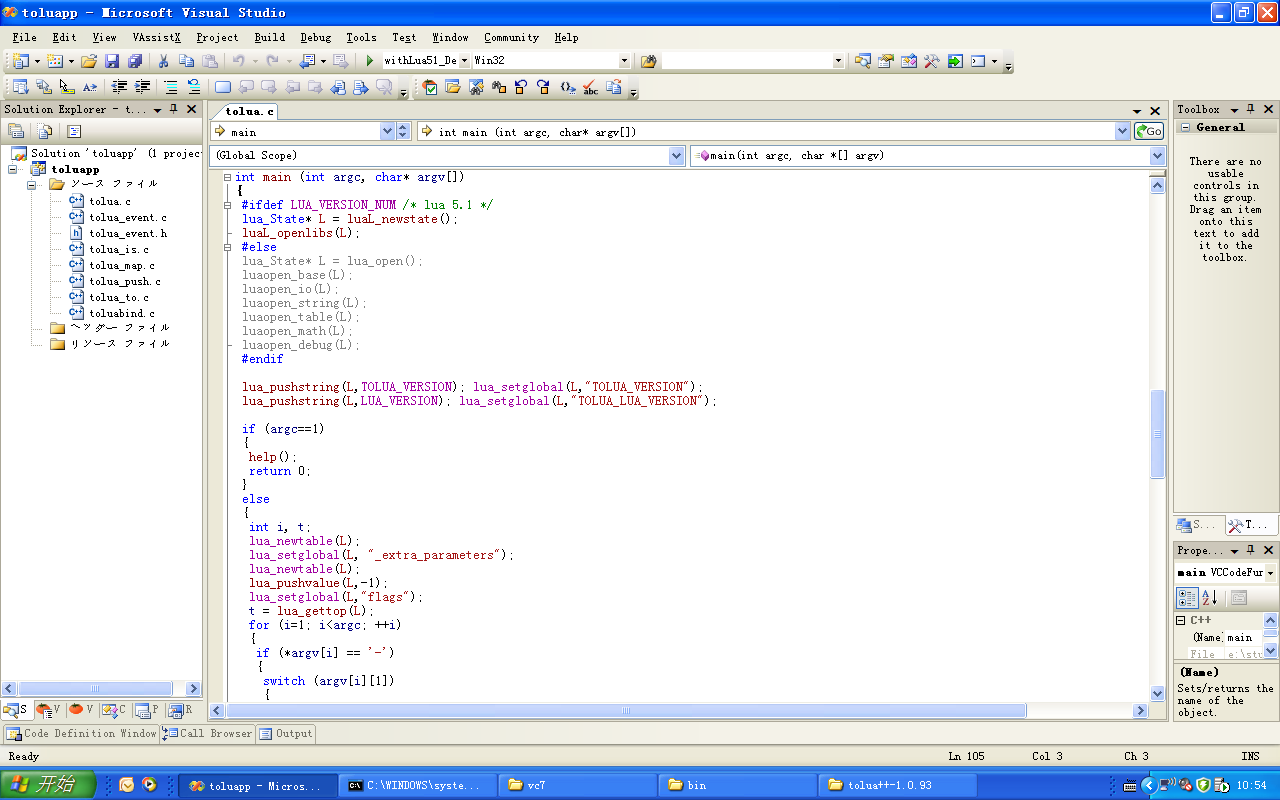
### tolua++.exe

本文使用tolua++ 1.0.92的版本，首先下载源码，解压后发现，tolua++目录如下图所示：



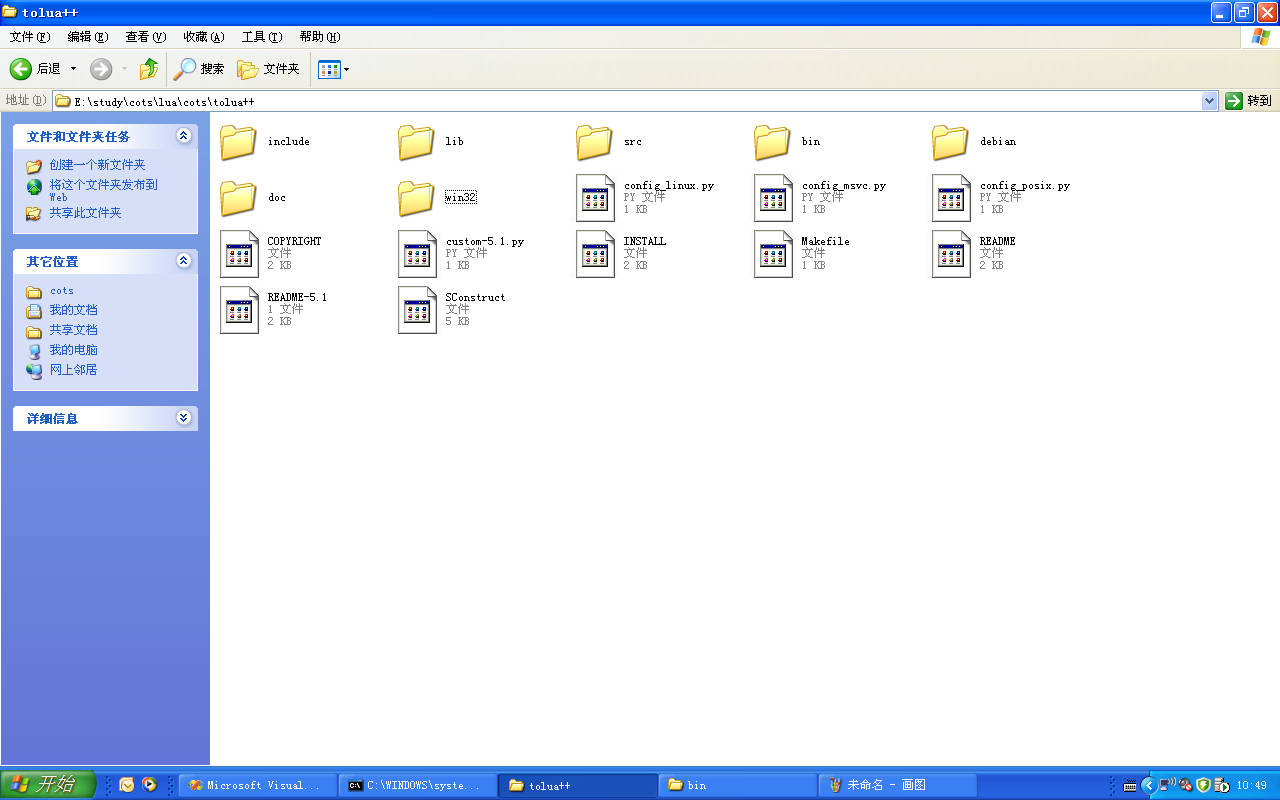
1. tolua++文件目录

在Win32目录下有VC7文件夹，VC7文件夹下有基于VS2003的工程文件，打开工程文件，可以直接编译生成tolua++.exe。如下图所示：



1. tolua++工程界面

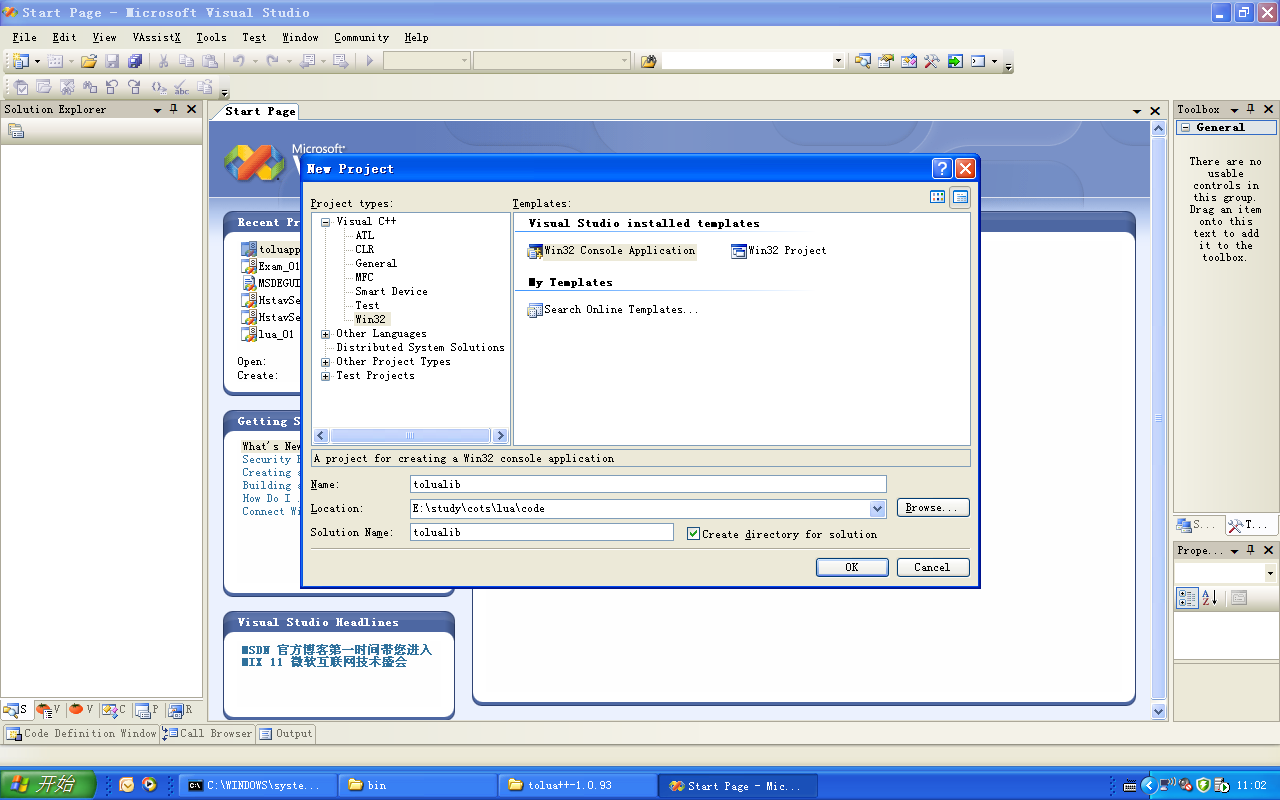
注意：编译时需要设定Lua的头文件和库文件路径，如下图所示：



1. tolua++库文件设置

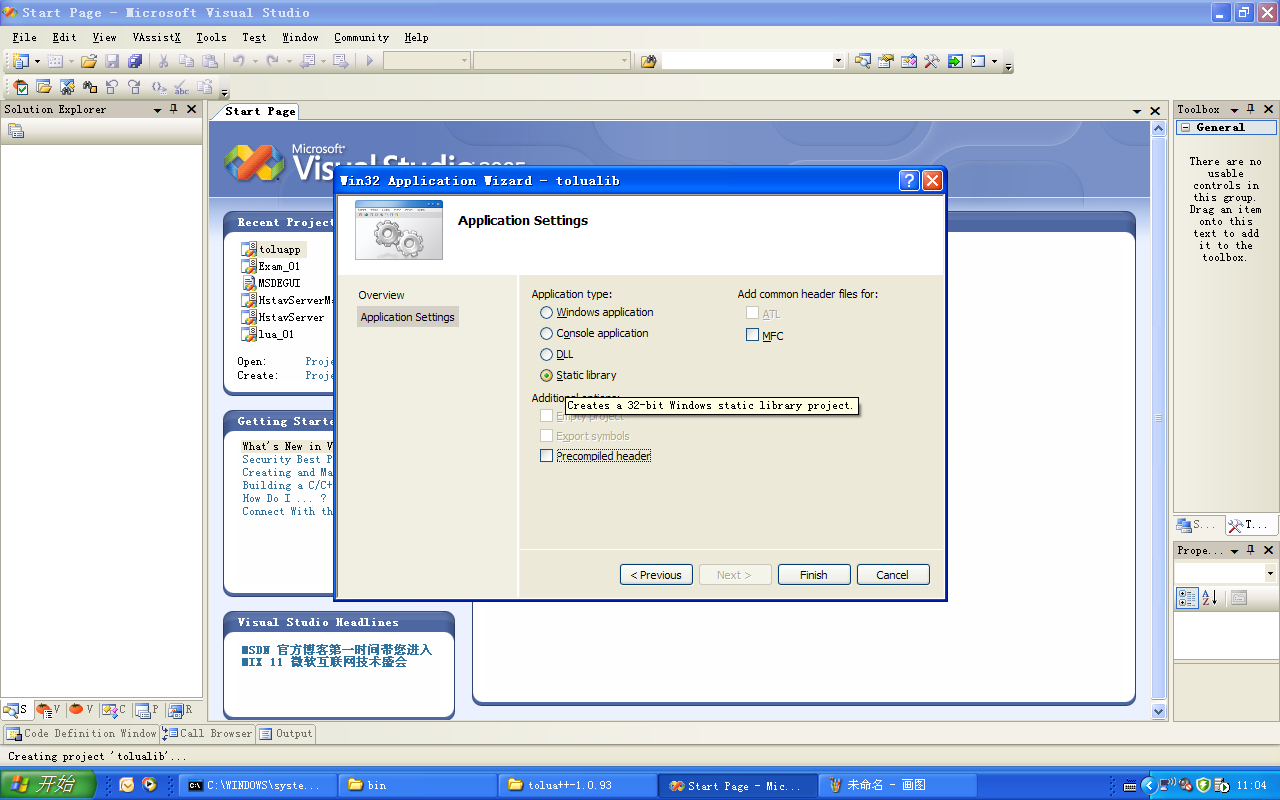
### tolua++.lib

第一步：新建基于win32控制台工程。如下图所示：



1. 新建工程

第二步：工程类型选择static lib，取消Precomoled header选项：



1. 工程类型选择

第三步：添加tolua++\src\lib目录下6个文件到新建工程，添加后效果如下图所示：



1. Tolualib文件目录

第四步：编译，编译后生成tolualib.lib文件

## tolua++实例

### Tarray

第一步：定义接口文件

typedef struct Point Point;

struct Point

{

float x;

float y;

};

extern int a[10];

extern Point p[10];

extern Point\* pp[10];

extern int ma[10];

extern Point mp[10];

extern Point\* mpp[10];

typedef struct Array Array;

struct Array

{

int a[10];

Point p[10];

Point\* pp[10];

};

extern Array array;

extern Array\* parray;

第二步：定义pkg文件

pkg文件是tolua++生成暴露接口代码的主要文件,该文件中定义了你要暴露程序中的那些类和接口，它跟你实际的程序没有任何关联，也就是说它不会去找你真的有没有pkg里所定义的函数，但是在程序编译时会报错。这个文件放在那里都是可以的。为了方便，一般都把你要暴露的类的.h文件拷出来改成pkg就可以了。tarray.pkg文件示例如下所示：

$#include "tarray.h"

struct Point

{

float x;

float y;

};

extern int a[10];

extern const Point p[10];

extern Point\* pp[10];

module M {

extern int ma[10]@a;

extern const Point mp[10]@p;

extern Point\* mpp[10]@pp;

}

struct Array

{

int a[10];

Point p[10];

Point\* pp[10];

};

extern Array array;

extern Array\* parray;

第三步:用tulua++.exe声称.c文件

Tulua++ -o test.c tarray.pkg

第四步：将生成的文件，1中的头文件，tolua++的头文件，lua的头文件及相关的库加入工程。

第五步：编写Lua脚本

for i=1,10 do

assert(a[i]==i)

end

for i=2,10 do

assert(p[i-1].y==p[i].x)

end

for i=1,10 do

assert(M.a[i]==i)

end

for i=2,10 do

assert(M.p[i-1].y==M.p[i].x)

end

for i=1,10 do

assert(pp[i].x==M.p[i].x and p[i].y == M.pp[i].y)

end

for i=1,10 do

assert(array.a[i] == parray.a[i])

assert(array.p[i].x == parray.pp[i].x and array.p[i].y == parray.pp[i].y)

end

for i=1,10 do

array.a[i] = a[10-i+1]

M.a[i] = 10-i+1

assert(array.a[i]==M.a[i])

end

for i=2,10 do

array.p[i] = array.pp[1]

assert(array.p[i].x==0 and array.p[i].y==1)

end

print("Array test OK")

注意：Lua脚本文件中数组是从1开始的，但是在C/C++中数组的下标是从0开始的。所以以上脚本文件需要修改。

第六步：运行工程

输出：Array test OK.

# Tolua++专题

未完待续

# 图表目录

图片目录

[图1： tolua++文件目录 1](#_Toc291234459)

[图2： tolua++工程界面 2](#_Toc291234460)

[图3： tolua++库文件设置 2](#_Toc291234461)

[图4： 新建工程 3](#_Toc291234462)

[图5： 工程类型选择 3](#_Toc291234463)

[图6： Tolualib文件目录 4](#_Toc291234464)

[图7： 结束语 7](#_Toc291234465)



1. 结束语