在Window平台中搭建Lisp学习环境（Notepad++篇）

原作：庆森，修改：陆志广

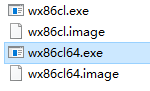
以上内容在原作者基础上略微修改，如有不妥之处请联系我

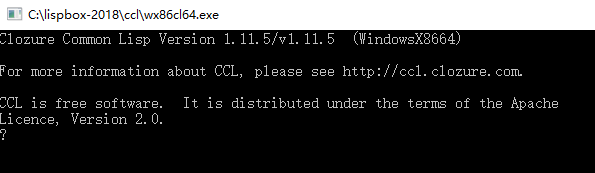
（QQ1635533973）共同学习进步。

在开始阅读本文之前，笔者先谈谈本文的目的吧。如果说你是刚刚了解到Lisp语言，没有折腾过linux的朋友，然后又已经看过Lisp相关的书籍，好想实践下书籍中的代码，那么本文适合你。相信有很朋友都跟我一样无论是工作、学习还是娱乐都习惯在Window平台上。不然折腾emacs的使用，很有可能到最后会把你学习lisp的兴趣也折腾没有了。

　　在这里我用的是ccl（ANSI Common Lisp 语言的实现）。目前官方有提供[下载](https://ccl.clozure.com/download.html)。安装也非常简单，这里就不列出步骤了。以下是目录支持系统。

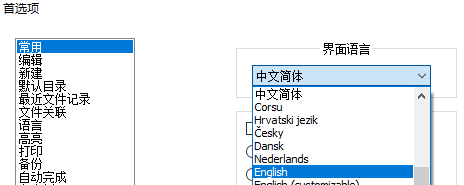
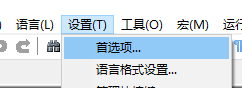


安装完之后如下：

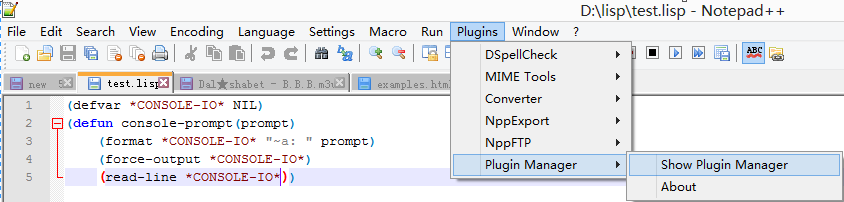


现在也可以直接在里面运行LISP语句了，来试试吧！

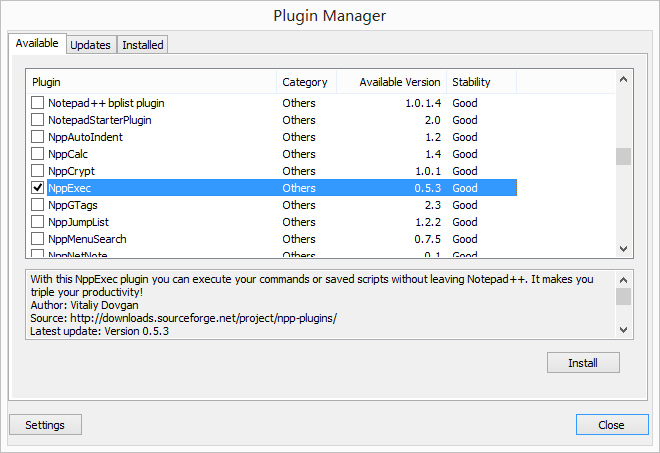
接下来安装Notepad++(文本编辑器),没有安装的朋友[点击这里](https://notepad-plus-plus.org/" \t "_blank)。官网可能不好进，大家可以直接百度。安装完之后我们还要给Notepad++安装NppExec插件（如果已经安装了的，请忽略这步）



大家先把中文的改成英文的，不然后面的NppExec插件可能会装不上，建议以管理员权限运行软件安装，中间如果失败可尝试重启系统，我的是Windows10 64位都可以安装好的。



找到菜单栏中的Plugins(插件)->Plugin Manager(插件管理)->Show Plugin Manager

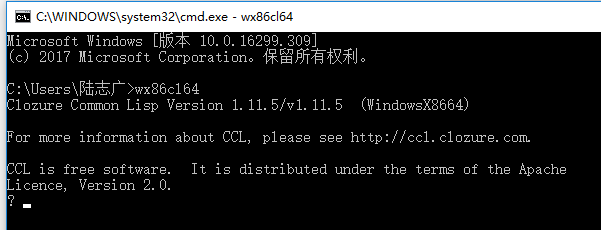


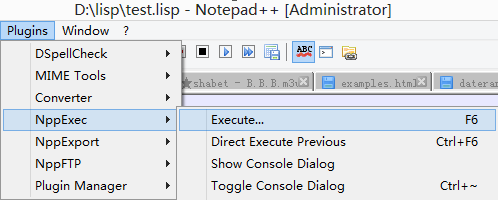
找到NppExec这个插件，然后点击Install。安装完之后记得要重启Notepad++。

在下一步设置前，我们要先把编译器设置成全局变量才行。

右键电脑，属性，高级系统设置，环境变量，系统变量，Path，编辑，新建，输入程序运行目录，如：C:\lispbox-2018\ccl即可。

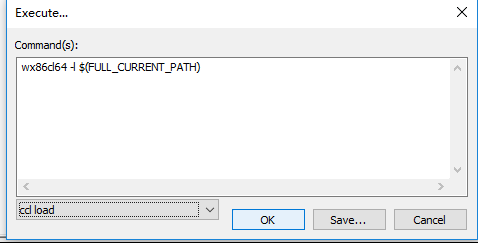
设置好后运行CMD，输入程序名，测试是否可以用。右键开始，运行，CMD





现在来配置Notepad++编译lisp,如上图找到Plugins->NppExec->Execute。

通过-l选项我们可以把源文件(\*.lisp)或者编译后的文件(\*.fas)加载到clisp环境中。

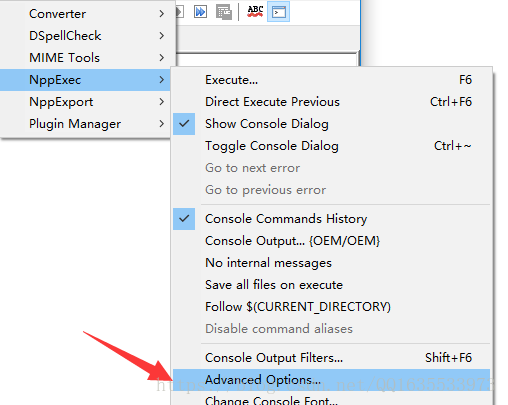


对于练习Lisp的语法我感觉-l选项就够了，所以我们配置的命令就是

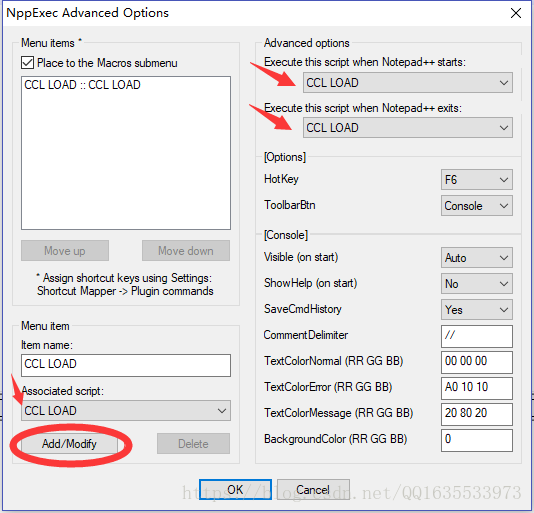
wx86cl64 -l $(FULL\_CURRENT\_PATH)

下面点击Save可以保存下来，名字叫什么好呢？随你们喜欢吧。我就把它命名为：CCL LOAD。

点保存后界面下方就会出现运行框了，会看到界面已经改变了。

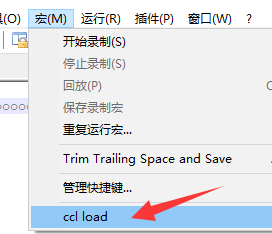


最后我们把CLISP LOAD功能加到Macros（宏）菜单中就可以方便的执行我们写的脚本了。



也差不多了，下面就来展示成果吧。

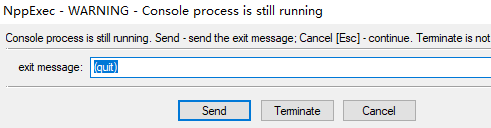
也差不多了，下面就来展示成果吧。



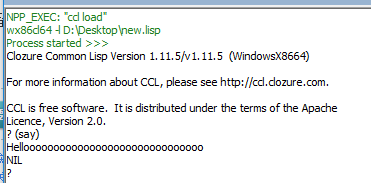
这是改回中文界面后的，大家要确保出现这个之后再改回语言。

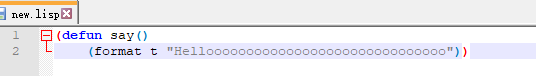
点下CLISP LOAD,激动时刻来了。

点后会出现这个



(quit)，输入这个点Send就可以运行了。

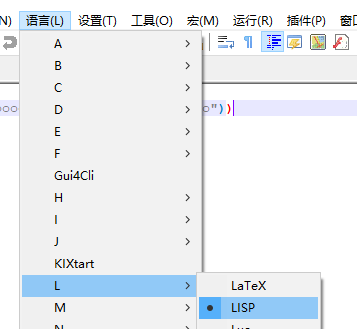




(defun say()  
    (format t "Helloooooooooooooooooooooooooooooo"))

这里定义了一个say函数,打印欢迎信息。最后可以使用**(quit)**或者**(exit)**可以退出CCL模式。

Lisp在Notepad++的语法高亮，这样设置下就可以了。



注：我按原文章测试过CLISP，还有SBCL，都不可用，原因是$(FULL\_CURRENT\_PATH)调用后的变量值是C:\lispbox-2018\ccl这种单反斜线，LOAD加载需要\\双反斜线或者/单斜线才行。好了，结束，有不足的地方欢迎指点，共同学习。

[Notepad++中NppExec的使用之一：基本用法](http://www.cnblogs.com/whenyd/p/3695573.html)

 一直用NPP，很长时间了，最近才学习它的各种插件，这篇文章是根据NppExec的用户指南写的。很多地方是翻译的，但不全是翻译，同时也有些东西没有翻译。

**一、何为NppExec**

简单的说，这个插件可以让用户在NPP中直接运行一些命令和程序，而不用启动这些命令和程序对应的实际工具或编译器。

**1. NppExec是...**

* NppExec是介于Notepad++和外部工具/编译器之间的一个中间件。它允许用户在NPP中直接运行这些工具/编译器。
* NppExec是一个控制台(Console)窗口，它能展示运行过程的输出结果，并将控制台窗口的输入重新定向(redirect)到运行过程中（有一些限制）。
* NppExec是其内部命令的一个解释器。通过"Execute..."(执行)对话框，它能逐条执行多条NppExec命令（也称为NppExec脚本），也能从控制台窗口直接执行单条命令。

**2. NppExec不是...**

* NppExec并不是一个控制台模拟器(console emulator)。NppExec的Console不是一个“真正”的控制台窗口，它没有提供控制台屏幕缓冲区(console screen buffer)。我的理解是NppExec的Console窗口中只是输出代码在真正的控制台中运行的结果。
* NppExec并不是一个命令解释器。虽然它能执行NppExec的命令或脚本，但它不能理解诸如'copy'之类的命令。
* NppExec并不是一个编译器。它能执行外部编译器的代码，但它本身不是编译器，执行这些代码不是它自己的功能。

**二、安装**

在“插件'菜单下打开插件管理器(Plugin Manager)，在可以插件(Available)中找到NppExec，选择后点击下方的"Install"就可以了。安装完成后NPP会提示重启，重启后在插件菜单下就能看到NppExec。

**三、基本用法**

**1. 获取帮助**

Plugins -> NppExec -> Help/Manual ：打开用户指南。

Plugins -> NppExec -> Help/Docs... ：打开NppExec.txt和NppExec\_TechInfo.txt文件，前者包括NppExec的所有命令、环境变量的语法和说明，后者包括NppExec的一些高级”隐藏“设置。

help：在Console中输入help可以查看以下内容

* 控制台快捷键
* 控制台命令
* 通用命令(General commands)和环境变量

help <command> ：help后加上命令的名字(中间用空格隔开)可以直接查看该命令的帮助文档。

**2. 简介**

NppExec的主要原理与NPP中的“运行”命令(F5)相同：执行给定的命令。通常就是以给定的参数执行一些程序或工具，参数能够以NPP中环境变量的形式来传递。比如，在IE中启动当前文件或者用某个编译器编译当前文件。NppExec提供了额外功能：

* 它有一个控制台窗口显示控制台程序的输出
* 它允许用户逐条执行多条命令
* 它提供了NPP的所有环境变量，也加入了它自己的若干变量
* 它有一些额外命令，只有想不到，没有做不到 :)
* niubility :)

**NppExec中执行命令的四种方式：** （四种方式）

* 使用NppExec的控制台窗口：输入命令，按Enter执行
* 使用NppExec的“Execute...”对话框：可以输入多条命令，保存为NppExec脚本，执行时逐条执行
* 使用NppExec的"npp\_exec <script>"命令：执行脚本
* 使用NppExec的"npp\_exec <file>"命令：执行文本文件保存的脚本

BTW，NppExec的菜单项使用第三种方法执行脚本。

有关内部过程和外部过程在下面“小试牛刀”这一节中。

**3. 小试牛刀**

**3.1 以内部过程的方式执行程序**

在控制台窗口或Execute窗口直接输入程序的名字（必要时添加路径）就可以按内部过程的方式运行程序，输入时也可以带参数。

比如：输入cmd可以运行windows的命令提示窗口（cmd.exe程序），由于cmd本身是控制台程序，因此它会在NppExec的Console窗口中运行，也就可以接着在Console窗口中直接输入cmd的命令来运行，用exit命令退出cmd。 又，输入calc则运行windows自带的计算器。由于计算器不是控制台程序，因此它独立运行。

可以看到，以内部过程运行程序，不管是控制台程序还是非控制台程序，开始都打印Process started >>>，结束后打印<<< Process finished.(Exit code 0)。括号中的Exit code值可能会改变，表示不同的退出状态。这是NppExec在等待内部过程执行完毕。

下面是cmd中查看系统时间的一个例子(控制台中)：

[复制代码](javascript:void(0);)

1 ================ READY ================

2 cmd

3 cmd

4 Process started >>>

5 Microsoft Windows [版本 6.3.9600]

6 (c) 2013 Microsoft Corporation。保留所有权利。

7

8 C:\Users\acton\Desktop>time /t

9 time /t

10 00:13

11

12 C:\Users\acton\Desktop>exit

13 exit

14 <<< Process finished. (Exit code 0)

15 ================ READY ================

[复制代码](javascript:void(0);)

也可以在运行程序时带参数，查看时间可以写成：

[复制代码](javascript:void(0);)

1 ================ READY ================

2 cmd /c time /t

3 cmd /c time /t

4 Process started >>>

5 00:16

6 <<< Process finished. (Exit code 0)

7 ================ READY ================

[复制代码](javascript:void(0);)

.cmd和.bat文件也能这样执行，但不能省略扩展名。

**3.2 以外部过程的方式执行程序**

使用npp\_run <command>，如下，会打开cmd窗口，Console中不会输出Process started和Process finished，NppExec不会等待外部过程的执行。

1 ================ READY ================

2 npp\_run cmd

3 NPP\_RUN: cmd

4 ================ READY ================

**小结：NppExec能理解的只是它的内部命令和环境变量。**

* 当输入其他命令时，比如"copy", "cmd"或"calc"，NppExec尝试将其作为子过程来执行，i.e. 它执行CreateProcess(...)直到子过程结束。
* 当输入类似"npp\_run cmd"或"npp\_run calc"时，NppExec尝试以外部过程的方式执行指定的命令，i.e. 它执行ShellExecute(...)，且不关心这个过程。

**4. NppExec的子菜单**

* Execute...：打开 "Execute..."对话框（默认热键F6）
* Direct Execute Previous：执行之前的脚本（默认Ctrl+F6）
* Show Console dialog：打开或隐藏Console窗口（默认Ctrl+~）
* Toggle Console dialog：激活或取消激活Console窗口（依赖Console的"hide toggled Console"选项，但是找不到这个选项在哪里）
* Console Commands History：Console命令历史是否可用（可用时上下键自动输入历史命令）
* Console Output...：打开Console Output/Input encoding对话框，选择输入输出的编码
* No internal messages：不显示命令确认等内部消息。比较如下，前面没有内部消息，后面有内部消息：

[复制代码](javascript:void(0);)

1 ================ READY ================

2 cmd

3 Microsoft Windows [版本 6.3.9600]

4 (c) 2013 Microsoft Corporation。保留所有权利。

5

6 C:\Users\acton\Desktop>exit

7 exit

8 ================ READY ================

9 cmd

10 cmd

11 Process started >>>

12 Microsoft Windows [版本 6.3.9600]

13 (c) 2013 Microsoft Corporation。保留所有权利。

14

15 C:\Users\acton\Desktop>exit

16 exit

17 <<< Process finished. (Exit code 0)

18 ================ READY ================

[复制代码](javascript:void(0);)

* Save all files on execute：执行NppExec脚本前始终保存修改过的文件
* Follow $(CURRENT\_DIRECTORY)：始终以当前目录执行脚本（当前目录用cd命令查看或更改，请help）
* Disable command aliases：命令别名不可用(用NPE\_CMDALIAS查看别名)
* Console Output Filters...：显示NppExec Console Filters对话框(Shift + F6)
* Advanced Options...：高级选项（可以添加菜单项，用于执行脚本）
* Change Console Font...：字体
* Help/Manual Help/Docs...
* Help/About

**5. NppExec Console**

可以用菜单、快捷键和工具栏图标打开，倒数第二个图标就是（反正我的是哈...）。

高级设置中的ToolbarBtn可以选择在工具栏显示Console/Execute/Execute Previous中的一个或不显示任何图标。

主要作用：

* 执行命令（Enter键）
* 当执行内部子过程，且为控制台程序时，显示运行结果（比如Python或cmd的运行结果）
* 显示内部消息
* help
* ...(有一个没看懂)

**6. "Execute..."**

打开方式：F6、菜单或工具栏（需要设置）。

用于执行脚本。

可以在<temporary script>中执行，也可以点击Save保存脚本。**保存后可以在高级选项中设置为NppExec的菜单项，分配热键。** 比如以下脚本：

1 // save current file

2 NPP\_SAVE

3 // copy current file to C:\Backup

4 cmd /c copy "$(FULL\_CURRENT\_PATH)" "C:\Backup\$(FILE\_NAME)" /Y

首先保存当前文件，然后用cmd中的copy命令复制到C:\Backup下，其中//开始的是注释。

**7. NppExec脚本**

NppExec脚本是一个命令集。

* 可以是NppExec的内部命令，比如"cls", "npp\_save", "npe\_debuglog"等
* 也可以是可执行的一些外部程序或命令，比如"cmd", "calc", "C:\tools\tcc\tcc.exe" 。
* 每条命令都可以带参数。比如以下几条命令：

1 INPUTBOX "Input something:" : something

2 NPP\_EXEC "script name" "param 1" "param 2"

3 cmd /c copy /?

4 "C:\tools\tcc\tcc.exe" "$(FULL\_CURRENT\_PATH)" -run

注意：

* 一条命令书写在一行中。
* 注释以//开始。
* 脚本可以保存为txt文件，用npp\_exec执行。
* 脚本存储在插件配置目录下，该目录存储在环境变量$(PLUGINS\_CONFIG\_DIR)中。
* 保存的脚本在npes\_saved.txt中，从其中可以查看脚本保存的格式。临时脚本在npes\_temp.txt中。

**重点是执行脚本的四种方式、两类过程（内部和外部）和脚本基础知识，下次介绍环境变量。**

插件下载地址

https://sourceforge.net/projects/npp-plugins/files/