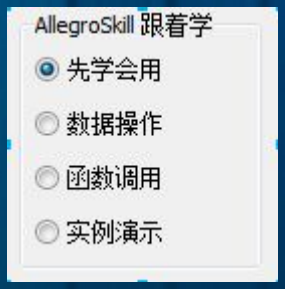
Allegro Skill 跟着学



作者 林晓伟
Q Q 16831621
官网 www.allegroskill.com

目录

§1-1 不得不说的废话

从事 allegro layout 的人不多，从事 Allegro layout 并会使用 skill 语言编程的就更少了。如果你只是编程爱好者，那还是止步吧，非工作需求请绕行。因为本 Skill 语言过于冷门只适用于 Allegro 软件。本书只适用于从事 allegro layout 工作的同行朋友翻阅。还希望你有一定其它计算机语言基础，最好是 C 语言。

.. 毛遂自荐

计算机及其应用专业的我，从事 Allegro layout 十余年，偏爱玩转计算机软件。有任何疑问都可加我的 QQ:16831621，skype:tatarlxw.随时联系。

. 出书原因

因为工作需要接触了 skill 语言，苦于网络上没有一本完整的 skill 学习资料，就结合了 lisp、turb C、skill-axl 三种语言写了这本《 Cadence Skill 跟着学》，算是笔记。iv

. 建站根本

因为学了书，挣了几百元，也结交了不少同道想学 Skill 的朋友。经常被询问书是否有更新？就把这书建了个网站，想借此结交更多的朋友。为能更好的服务于大家，不想再借助他人网站发布，更有条理的分享自己的笔记。 www.allegroskill.com 网站的内容与域名一样定位仅 AllegroSkilll 语言，涉及内容面很窄，还很冷门，建站的总体费用还很高，所以还希望大家能花点小钱捐建一下，以保证本站的可持继发展。

.. Skill 能做什么？

SKILL 语言是 Cadence 提供给用户的一个开发接口，利用其本身提供的接口函数和 SKILL 语言完成自动化操作的功能。可为 PCB 工程师们提高工作效率，缩短工程图面开发周期，减少出错率等，品质保证等各方面都有一定的帮助。

. 如何学习 SKILL 语言呢？

首先您需要一本入门的书，对就是这套《 Cadence Skill 跟着学》 丛书，这是目前网络上内容

最丰富的一本 Cadence Skill 教学书籍，最关键的是：“笔者还活着”，正在建站中！希望能对您有所帮助。就让这本书与您一起深入探讨 SKILL 语言 。花点小钱购买一本吧，哈！

. 如何学好计算机语言？

个人以为学好一门计算机语言，当然就是边看书边敲敲打打，通过使用它来完成实际的任务。您并非需要学习并牢记每一条语句，可以通过本站快速搜索您想要的命令就好；不知您是否也有同样的认为！细心的您会发现本书在描述语法时，例子特别多。笔者也是初学，为了弄清某命令的用法，就敲较多的例子。关键还是要有工作需求才能坚持。

§1-2 Skill 简介

·.. Skill 语言可以做什么？

Skill 是"嵌入式"的高级开发语言，支持一套类似 C 语言的语法，大大降低了初学者学习的难度，同时高水平的编程者可以选择使用类似 Lisp 语言的全部功能。所以 SKILL 语言既可以用作最简单的工具语言，也可以作为开发任何应用的、强大的编程语言。

SKILL 可以与底层系统交互，也提供了访问 Cadence 各个工具的丰富接口。在 Cadence 环境下，允许 Skill 自定义用户界面，并包含强大的跟踪、调试和分析工具。Skill 还可以让你访问和控制所有的工具、配置用户环境、设计数据库。用户可以通过 Skill 语言来访问，并且可以开发自己的基于 Cadence 平台的快捷、实用工具。

· Skill 语言和 Lisp 语言的关系

Cadence 提供二次开发的 SKILL 语言，它是一种基于通用人工智能语言——Lisp 的交互式高级编程语言(LISP 即 List Processing－表处理，是最早和最重要的符号处理编程语言之一，它于 1958 年由美国的 J. McCarthy 提出，LISP 在人工智能 AI 方面获得广泛应用)。

。Skill 函数提供两种表示法，

一种是代数表示法，现在大多数语言采取这种方式，即 func(*arg1 arg2 ...*)，

```
procedure( fibonacci(n)
if( (n == 1 || n == 2) then
  1
```

```
    else fibonacci(n-1) + fibonacci(n-2)
    )
)
```

另一种是前缀表示法，类似于 Lisp 语言，即(func *arg1 arg2* ...)。

```
(defun fibonacci (n)
  (cond
    ((or (equal n 1) (equal n 2)) 1)
     (t (plus (fibonacci (difference n 1))
       (fibonacci (difference n 2))))
  )
)
```

这里可以看到类似 Lisp 语言的表示法后面有很多右括号，而且函数和参数容易混淆，所以一般推荐还是用常用的类 C 语言代数表示法。

Skill 程序就像一个 list 表，类似 Lisp 语言，程序的操作就像操作数据(list)一样，您可以动态地创建、修改、收集、评估的函数定义和表达。

§1-3 调试窗体

. 语法约定

在 Cadence skill 帮助中出现在语法描述中的字样：

text	说明文字必须准确输入
z_argument	说明文字用于替换相应的参数，前缀（z_)表示数据类弄的参数可以接受。
[ ]	表示可选参数。
|	分离选项的选择
{ }	使用竖线各封闭的选项，你必须选择一项。
...	表示重复前面的描述，

=>	函数返回值，及结果
text	指示手册，菜单命令，表单按钮各表单域的名称

. 函数调用

你可以通过 键盘、窗体、菜单、命令解释器、skill 程序等方式去执行 skill 代码来实现功能。

Bindkeys
一个 Bindkeys 关联键盘事件，发送 Skill 功能
Forms
有些功能需要您填写字段，在弹出的表单中提供数据。
Menus
当您在菜单中选择一个项目时，系统会发送相关的 Skill 功能。
CIW
你可以直接输入 Skill 功能到 CIW,进行立即评估。
Skill process
您可以启动一个单独的 UNIX 的过程 ，可以直接提交 skill 功能的解释。
Skill Interpreter
您可能通过回载 skill 源码文件，对提交的 skill 功能进行集中评估。

. CIW 命令解释器窗口

你可以在 allegro PCB 编辑器的命令行里直接运行 AXL-SKILL 函数，语法如下：
skill (<function><arguments>)

也可在 allegro 中输入 skill 就可进入了 AXL-SKILL 的运行环境，在这样的环境中可以直接调用 AXL-SKILL 命令/函数，可用 exit 命令返回到 allegro 命令环境。

```
Command > skill Skill > exit
t
Command >
```

另外还可以输入 **set telskill** 可以得到一个尺寸大小可调的 skill 开发窗口。编者强烈见意大家在学习本册【数据操作】章节时使用这种状态上机调试本书内容。

Command > **set** telskill ;进入 skill 开发窗口

注：若没有弹出窗口，尝试在 allegro 菜单里面，选择 setup-> user prference->skill-> telskill----OK

§1-4 目录简介

.. 目录使用

. 绝对路径：

是指目录下的绝对位置，直接到的目标位置。Unix 中以"/"开始的路径，Windows 中以磁盘"C:,D:,..."开始的路径，网络邻居的电脑以 "\\" 开始。

. 相对路径

不是绝对路径的情况就是相对路径 ， 相对路径就是指由这个文件所在的路径引起的跟其它文件（或文件夹）的路径关系。使用相对路径可以为我们带来非常多的便利。

以"~/"开始的路径表示被查询的路径是用户的 home 文件夹;

以"~username"开始的路径，如果 username 正好是一个用户的名字，那么被查询的路径将是该用户的 home 文件夹;

以"./"开始的路径表示被查询的位置是当前的工作目录

以"../"开始的路径表示被查询的位置为当前工作目录的父文件夹

. 常用路径

Allegro 安装时会带有两个目录：安装目录(cdsroot)与工作目录(home)，

先了解一下你机器上的 allegro 的几个比较常用到的目录。

Allegro 命令：Echo 显示各种预设目录，下表以笔者本机 allegro 16.5 为例，帮您展现各文件与目录所在：

目录名称	查看命令	例
安装目录	echo $cdsroot	C:\Cadence\SPB_16.5
工作目录	echo $home	D:/SPB_Data16.5
pcbenv 目录	echo $localenv	D:/SPB_Data16.5/pcbenv
菜单目录	echo $menupath	D:\SPB_Data16.5\pcbenv\cuimenus\

. 设置 skill 路径

当传递一个相对路径给 load 函数时，系统解析 SKILL 目录列表。一般我们会在 SKILL 初始化的时候通过 setSkillPath 和 getSkillPath 建立目录。因为 load 要找到 SKILL 来装载，通常我们都不喜欢通过绝对路径来装载一个文件，因为那样不便与日后目录的重整，且包含的字符太多了。

skill 支持用户定义若干不同的路径来存放 skill 源文件，并通过 skill 的路径管理函数来设置各个路径的访问顺序。如果同一个文件在不同路径下都存在，按照先到先得原则，即使用第一个被找到的文件。

设置 skill 查询路径的函数 setSkillPath，如

```
setSkillPath("." "~" "C:/Skill")
```

获取 skill 查询路径列表的函数 getSkillPath，如

```
setSkillPath("." "~" "C:/Skill");
getSkillPath()=>("." "~" "C:/Skill")
```

就设置了 3 个 skill 查询路径，先后顺序分别为"."－当前工作路径，"~"－home 文件夹，"C:/Skill"－用户自定义文件夹。

实例：设置一个新路径到目录列表中

```
setSkillPath( append( trSamplesPath getSkillPath() ) )
```

可以将 trSamplesPath 改为自己想添加的目录。然后将代码直接放在 allegro.ilinit 中。之后在自己添加的目录就可以使用相对路径加载。

.. 目录操作函数

目录修正	axlOSSlash	更改 DOS 目录反斜杠为 UNIX 斜线
		p = axlOSSlash("C:\\Cadence\\SPB_16.5") -> "C:/Cadence/SPB_16.5"
创建子目录	createDir	在当前打开的目录下创建 test 子目录，也可以使用绝对路径。
		createDir("./test") createDir("c:/test")
删除目录	deleteDir	删除目录，也可以使用绝对路径。如果这是个空文件夹，且用户具有删除它的权限
		deleteDir("./test") deleteDir("c:/test")
目录存在性	isDir	检查目录是否存在，存在返回 t
		isDir("c:/test")
路径存在性	isLink	检查路径是否存在，存在返回 t
		isLink("/usr/bin")=> nil isLink("/usr/spool")=> t ;假设"/usr/spool"路径存在。
目录读写性	isReadable	检查目录是否可读写
		isReadable("c:/test") isExecutable("c:/test")
获取工作目录	getWorkingDir	getWorkingDir=>"D:/SPB_Data16.3/a301.001a"
改变工作目录	changeWorkingDir	changeWorkingDir("D:/SPB_Data16.3/") getWorkingDir=>"D:/SPB_Data16.3/"
打开目录	http	使用资源管理器打开当前工作目录
		http . ;打开当前工作目录

文件浏览	getDirFiles	浏览目录获取指定文件夹下所有文件名函数
		getDirFiles("D:/SPB_Data16.3/a301.001a") ("." ".." "001.clp" "002.clp" "1.txt" "a.brd" "a301.001a_0811.brd" "a301.001a_0812.brd" "a301.001a_0813.brd""a301.001a_\275\273\275\323.doc" "allegro.jrl" "allegro_S05016.3_AllegroMiniDump.dmp""autosave.brd" "backup")

上述列表中含：

1.目录列表有： "."（当前的工作目录）".." （父目录）与子目录列表如："backup"

2.中文文件名自动转 2 进制转义字符表示。

§1-5 文件操作函数

. 常见文件

这里只列举小些与 Allegro Skilll 语言相关的几个常见文件：

类型	功能	源码格式
.il	skill 源码	skill
.form	表单文件	BNF
.color	颜色配置文件	skill
.clp	剪切板文件	skill
allegro.ilinit	随 allegro 启动自动加载	skill
env	随 allegro 启动自动加载	command

. 检测文件函数

文件的读/写均为 UNIX 文本文件。

文件存在	isFile，isFileName	检测文件是否存在函数
		isFileName("c:/test/1.txt") =>nil ;c:/test/1.txt 实际不存在
可被读	isReadable	检测文件或文件夹的是否可以被读取的函数
		isReadable("c:/test/1.txt") =>nil
可读写	isExecutable	判断文件是否可读写。
		isExecutable("c:/test/1.txt") =>nil
可被写	isWritable	检测文件或文件夹的是否可以被写入的函数
		isWritable("c:/test/1.txt")
大文件检查	isLargeFile	检测文件是否大于 2GB 或更大
		isLargeFile("largeFile") => t
文件长度	fileLength	返回文件长度
		fileLength("largeFile") => 3072000000
删除文件	deleteFile	删除指定文件
		deleteFile("c:/test/1.txt")
修改文件名	renameFile	重命名文件 renameFile(S_old S_new) => t | nil

§1-6 运行环境

. Cadence 安装设置

因为各公司的设置存在很大的差异。让我们一起先找到您正在使用的 Allegro 所常用的三个文件：

文件名	功能	全文件名
allegro.men	Allegro 菜单配置	%localenv%\menus\allegro.mem 或:%cdsroot%\share\local\pcb\menus\allegro.men
env	Allegro 用户设置	%localenv%\env
allegro.ilinit	加载 skill 文件	%localenv%\allegro.ilinit

正版 Allegro 安装时，要求 License 安装在工作组或局域网的服务器上。客户端软件安装在各自机器上，在启用软件时通过指定的网卡 MAC 地址来鉴别是否已注册。因此用户配置文件所放置的 pcbenv 目录可以有三种选择：

1. 在 server 机器上，修改时可使整个局域网内的用户统一修改，适用于公司整理修改管理；

2.在 Allegro 安装目录内，修改时可能无法还原系统自带的设置，适用于安装快速设置；

3. 用户自定义工作目录，设置 $home 位置，适用于个性化设置。

还是先获取个人设置目录 pcbenv 的路径在哪，以笔者机器为例：

```
command>echo $localenv=>D:/SPB_Data16.5/pcbenv
```

依上表所示，笔者 16.5 版的三个文件为：

文件名	功能	全文件名
allegro.men	Allegro 菜单配置	D:\SPB_Data16.5\pcbenv\cuimenus\allegro.mem
env	Allegro 用户设置	D:\SPB_Data16.5\pcbenv\env
allegro.ilinit	加载 skill 文件	D:\SPB_Data16.5\pcbenv\allegro.ilinit

注:如果您没有在用户设置目录中找到 allegro.ilinit,可使用记事本创建一个空文件，*.ilinit 文件与 *.il 格式是一样的，同是 skill 源码文件

注：Allegro 14 版的菜单文件为 adlayout.men，Allegro 15 版以上为 allegro.mem。

. Allegro 启动过程

当你启动 Allegro PCB 编辑器时，会读取 allegro PCB 编辑器的 env 文件，然后运行 allegro.ilinit 文件。最后自动加载菜单配置文件 allegro.men，

注：不能在 env 文件中插入 skill 命令。通常会在$home/pcbenv/allegro.ilinit 文件中插入加载 skill

文件的语句。

§1-7 Skill 编辑器

skill 语言 源码为超级文本格式，只要使用 记事本 ，就可直接编辑，然后在 Allegro 中 加载编译，再执行。

. notepad++ 调试

只是记事本用于编程实属不易，那就随便找个源码编辑器吧！这类软件还真多，有 20 几款都还不错，笔者选最新版的 "notepad++ v6.7.4" ，选择理由，免费版，省去注册或坐等新版破解版。

下载附件即可使用。 (附件下载请访问 www.allegroskill.com 论坛。

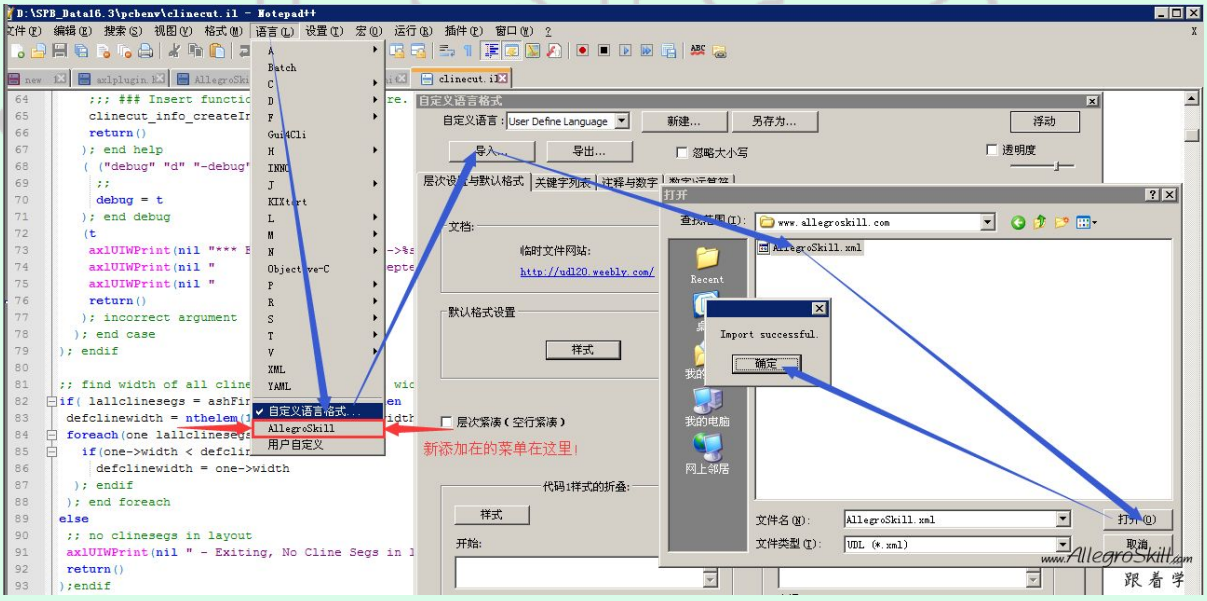
调试方法：

1. 自行去网上下载 notepad++ 软件，

2. 点菜单 语言> 自定义语言格式 > 导入... > 选择本站下载的附件 AllegroSkill.xml > 确定 >重启 notepad++ 软件

3. 下次再开启 notepad++ 软件，就可自动 将 il clp ilinit 等文件源码彩化。

若没有被彩化可选 点菜单 语言>AllegroSkill 即可。



§1-8 用户习惯性设定

用户习惯性设定一般在设置在 env 文件里。在进行 User Perferences Editor 设定后，设定动作均会在 env 文件里出现相关的 set/unset 语句。关于 User Perferences Editor 部分会在 Allegro-用户配置 版块详解。

. Script 键鼠操作录制文件

可先录制一段键鼠操作，与类似 office 里宏的类似。比如只单显某布的布线，供日后调用快捷切换层面。

操作：菜单 File -> Script 进入 Scripting 录制面板）-> 输入待录制文件名 -> 点 [Record] 开始录制

-> 开始键鼠操作直至得到想要的效果

-> 再进入 Scripting 录制面板 -> 点[Stop]按钮结束全程录制。

录制结果被保存在 "xxx.scr" 文件中，可使用 replay 命令调用。

```
command>replay("xxx.scr")
```

.. 快捷键设定

. alias 字符串别名

在 Allegro13-14 版及其前生，只有 alias 字符串别名设定命令，可将长字符串以较短的字符表示，

比如：将 F2 键设定为布线命令"add connect"，

```
command>alias F2 "add connect"
```

由于 F1 被系统默认为“help”功能无法进行再设定，一般键盘只能设定 F2-F12，home，page up,page down, end, insert, delete, 四个方向键，还有与 Ctrl,Alt,Shift 相结合。虽可设定的键还蛮多，但都不易于左手单手操作。但传说中有一种专业的键盘，键盘左手边还多出两竖 12 个功能键，可设定到 F13-F24。小生用过，感觉相当爽。

当然在 Allegro 15 版之后有了 funckey 快捷键设定命令就用普通键盘更好用。

. funckey 快捷键设定

先区分一下 alias 与 funckey 的区别

```
command>alias a "zoom in"  ;在命令行 CIW 中输入"a" 再按"enter", 即可放大
command>funckey b "zoom out" ;在设计工作区按"a"键，即可缩小
```

由此可见有了 funckey 与 alias 配合使用，可使整个键盘上的除 Ctrl,Alt,Shift,win,popup,fn,caps...等特殊键外均可做快捷键设定。

```
command>alias ;可查看 alias 与 funckey 的所有已设定
```

例：比如单显 top 布线层快捷设定在"1"键：

```
先录制单显 top 布线层的键鼠操作：先全关闭显于所有层，再勾先显示与 top 布线相关层面
（pin via drc 零件框 CnsRgn 区 等）
并将文件保存在工作目录下，如："OnlyDisplayTop.scr"，最后有 funckey 进行调用。
command>funckey 1 "replay(OnlyDisplayTop.scr)"
```

笔者见意公司内统一设定几个快捷键，统一设定有助于各工程师位客串他人机器时容易上手。

§1-9 Skill 语法操作简介

入门就算了，抬抬腿吧只想会用的大侠。这里仅简介一些非了解不可以知识点。

变量(variable)

变量是保存了其它值的表达式，如 x=3, x 即变量。变量的值是在编译时期被储存的，而在执行时期时回传返回值。

Skill 的变量不需要事先声明，Skill 第一次用到是会自动生成变量。变量可以由字符、数字、"_" 和 "?" 组成，注意第一个字符不能是数字和"?"。由于 Cadence 所开发的 Skill 中的变量、函数

都是第一个字母小写，以_为开头的是 Cadence 的专用函数，为了避免冲突，建议函数和变量命名都以大写字母开头。有时 code package 前头使用三个小写字母更清晰些。

. 定义局部变量

局部变量的定义,使用 let 或 prog 函数，

let 和 prog 的区别在于函数的返回值，如果不需要在函数内使用跳转函数或者返回值的话使用 let 即可。Let 的运行速度要快于 prog，当函数中不使用 go 或者 return 时，建议最好使用 let。

let 的返回值是 let()中的最后一句表达式，prog 的返回值比较多样，还包括 return 命令的返回值。相对于 let，prog 还多了项功能，在 prog()中多了一个循环功能，支持 go 命令。

. 转义字符

SKILL 中的转义字符与 C 语言基本一样。它是一种特殊的字符常量，一般通过一个反斜杠和一个或多个字符组成。转义字符具有特定的含义，不同于字符原有的意义，故称"转义"字符。比如我们在编程中常常用到的换行符"\n"就是一个转义字符。转义字符主要用来表示那些用一般字符不便于表示的控制代码。

常用的转义字符表

\a	响铃(BEL)	007
\b	退格(BS)	008
\f	换页(FF)	012
\n	换行(LF)	010
\r	回车(CR)	013
\t	水平制表(HT)	009
\v	垂直制表(VT)	011
\\	反斜杠	092
\?	问号字符	063
\'	单引号字符	039
\"	双引号字符	034

\0	空字符(NULL)	000
\ddd	任意字符	三位八进制
\xhh	任意字符	二位十六进制

需要借助转义字符(\)才可以打印的字符，以及双引号被定义成字符串时都需要和转义字符一起，如"\"", "\\" 分别定义了字符串" 和\。

字符串限制

例子

比如常见的目录结构，一般为 c:\\windows\\，其中的两个反斜杠代表的就是输出一个反斜杠，其输出的目录就为 c:\windows。这就是为什么使用反斜杠指定目录时，需要同时使用两个。

```
Skill > str="C:\\Cadence\\SPB_16.3\\doc\\skhelp"=>"C:\\Cadence\\SPB_16.3\\doc\\skhelp"
Skill > printf str
=>C:\Cadence\SPB_16.3\doc\skhelp
=>t
Skill > axlGetVariable("cdsroot")
"d:\\Cadence\\SPB_16.3"
Skill >
```

. 字符串合成：

strcat(s_string1 [s_string2] ...)=> t_result,

```
strcat( 'ab "xyz" )        => "abxyz"
```

合成所有提供的字符串生成一个新的字符串

. 合成 skill 语句

有时可以很方便的将几个 skill 语名合成一个单一的 skill 来声明，使用大括号｛｝来集

合 skill 语句至一个单一的 skill 声明中。单一语句的返回值是该组中的最后一个 Skill 语句的返回值。您可以将这个返回值保存至一个变量中。

这个例子计算 BBOX 的像素高度，

```
bBoxHeight = {
        bBox = list( 100:150 250:400)
        ll = car( bBox )
        ur = cadr( bBox )
        lly = yCoord( ll )
        ury = yCoord( ur )
        ury - lly }
```

. 声明 skill 语句

要通过引用函数名，使用声明一个过程与函数体关联，该函数体和函数名构成了一个 SKILL 功能。

```
procedure( ComputeBBoxHeight( )
        bBox = list( 100:150 250:400)
        ll = car( bBox )
        ur = cadr( bBox )
        lly = yCoord( ll )
        ury = yCoord( ur )
        ury - lly }
        ) ; procedure
bBoxHeight = ComputeBBoxHeight()
;注，要执行的函数体，其函数名后紧跟().
```

. 定义带参数函数

为了使您的函数更回灵活，可以在函数体加形式参数识别某些变量。

```
procedure( ComputeBBoxHeight( bBox )
      ll        = car( bBox )
      ur        = cadr( bBox )
      lly       = yCoord( ll )
      ury       = yCoord( ur )
      ury - lly
      ) ; procedure
bBox = list( 100:150 250:400)
bBoxHeight = ComputeBBoxHeight( bBox )
;要执行你的函数，你必须提供的参数值。
```

. 加载你的 skill 源码

这个加载功能评估在源码中的每条语句，通常用于定义函数集。

返回 t,表明所有表达式计算没有错误，如果有任何错误就中止之后的表达式计算，即加载不成功，需要修改错误代码后再次调试。

本书在源码中使用 => 符号表示返回值，与 skill 帮助档用法一致。

§1-10 加载 skill 程序_allegro.ilinit

想加载 Skill 程序，总得有份 xxx.il 文档，网上神人也很多，听说 Skill 语言已被开发很深了，再开发那就得走创新用法与内部需求路线。去各大论坛找找还是有收获得，本论坛可能会比较无耻，将会收集各大论坛的帖子与附件，帖子整理一下放在"采集整理"版块，Skill 程序去重一下放在"软件库"版块，方便大家恶心自己。

.. Skill 函数

. 自定义命令

> axlCmdRegister(); register Skill 函数为 allegro 的命令

如果之前命令已被注册，那么将会用新的命令替换之前的。

```
axlCmdRegister(t_allegroCmd  ts_callback  ?cmdType t_cmdType  ?doneCmd
```

```
ts_doneCmd  ?cancelCmd s_cancelCmd )
⇒ t/nil
```

t_allegroCmd：该处为在 allegro 使用的命令

ts_callback：该处为调用 SKILL 中的哪个函数

?cmdType：命令类型，共分为两类。"interactive"，为默认类型；"general" 当命令调用时会立即执行命令，会立即停止 allegro 正在执行的命令。一般用于含有有弹出窗口的情况下。可选项

?doneCmd：当执行 DONE 命令后调用的 Skill 函数，可选项

?cancelCmd：当执行 CANCEL 命令后调用的 Skill 函数。可选项。

当返回值为 t 时说明命令注册成功，否则失败，请检查各个参数是否正确。

下面是一个简单的例子：

```
axlCmdRegister( "pal_via_net" 'ch_via_net ?cmdType "interactive" ?doneCmd
'viadone ?cancelCmd 'viacancel)
```

可使用该函数为 Skill 注册一个 allegro 命令，如果之前命令已被注册，那么将会用新的命令替换之前的。因此笔者见意使用一个统一的个性前缀，如本节中使用的"pal"。自成一个命令系列必避 allegro 自带或他人命令冲突。

. 执行外部程序命令 system sh

system 跳出 Allegro 直接执行 cmd 命令

sh 等待执行 cmd 的命令结束后，返回至程序继继下一条语句。

.. 跟学写两个 Skill 小程序

先跟着笔者自己动手写一个所有计算机语言开始的学习的第一个程序“Hello Word”小 Skill 程序吧！

新建文本文档 d:\MyTools\HelloWorld.il skill 源码如下

```
;神奇的 Hello World
axlCmdRegister(  "palHelloWorld"  'palHelloWorld)
```

```
defun(palHelloWorld  ()
axlUIConfirm("Hello World!")
axlUIConfirm(strcat("This is my first Skill Program
_"  axlGetVariable("username")))
)
```

再来写个新建文本文档 d:\MyTools\OpenHome.il skill 源码如下

```
;选定当前工作图档
axlCmdRegister( "palOpenHome" 'palOpenHome)
defun(palOpenHome()
FileName=buildString(parseString( axlGetDrawingName() "/") "\\")
expBoard=strcat("explorer.exe/select,/n,/e," FileName)
axlShell(strcat("system " expBoard))
)
```

热心的朋友看这到可能会提意:"可以在语句后添加适当的注释",哈,这里仅是加载 skill 程序。不过还是要简介几条常见的,

.. 加载 skill

. skill 加密

加密 SKILL 程序，即是给 SKILL 程序加上密码，使用命令格式：

格式： encrypt(inputFileoutputFile passwd) ;其中 passwd 可以省略：

```
skill>encrypt("test.il" "test.ile" "pass")skill>load("test.ile" "pass")
```

好像 cadence 没有发布加密后的 skill 如何反编译。但如果加密时没有使用密码，还是可以使用 compress 命令进行反编译的，只是换行格式没了。不过好像很少见有没有密码的加密档哦。

. 源码压缩

压缩文件即可以减少文件大小

格式：compress(inputFileoutputFile)

该命令主要会去除标识符如：空格，缩进等，去除注解等。命令产生文件为 ASCII 格式，所以

也可以对加密文件进行压缩。

如果你的源码打算加密，那见意就先压一下，再加密吧

```
skill> load("D:/welcome.il") ;加载
skill> compress("D:/welcome.il"  "D:/welcome.ilc")  ;压缩文件
skill> encrypt("D:/welcome.ilc" "D:/welcome.ile" "www.allegroskill.com");加密文件
skill> load("D:/welcome.ile" "www.allegroskill.com") ;加载有密文的 skill 文件
```

. 跨平台函数 axlShell 与命令 skill

```
command>skill skill 函数
skill>axlShell ("command 命令")
```

. 加载 skill 程序

大家都使用 load("") 这程 C 语言格式调用 skill，其它还可以使用 Lisp 语言格式调用 (load "")

例：在 allegro.ilinit 加入刚刚创建的两个 skill 小源码

```
setSkillPath( append( "d:/MyTools" getSkillPath() ) )
(load "OpenHome.il")
(load "HelloWorld.il ")
```

注意例子中的斜杠方向。Allegro 的前身是从工作站 LUNIX 系统移至 windows 系统的。

§1-11 手工修改菜单_allegro.mem

日后会介绍如何设置动态菜单，在管理员安装机器时，先手工修改好菜单文件，然后复制公司内部统一菜单文件是必要的。或是如果只是一两处需要修改，使用手工菜单修改还是蛮方便的。

先了解一下 allegro 菜单格式如下

//注释信息	; //为注释语句
POPUP "主菜单名"	;显示在 Allegro 菜单栏上面
BEGIN	;主菜单开始，在 BEGIN 与 END 间定义菜单

```
  MENUITEM "菜单名", "执行的命令"          ;菜单项
  ……
  MENUITEM SEPARATOR                      ;横线
  ……
  POPUP "二级菜单名"                      ;二级菜单，可嵌入二、三级菜单.....
  BEGIN                                   ;二级菜单开始
   MENUITEM "菜单名", "执行的命令"        ;三级菜单项
   ……                                     ;
  END                                     ;二级菜单结束
END                                       ;主菜单结束
```

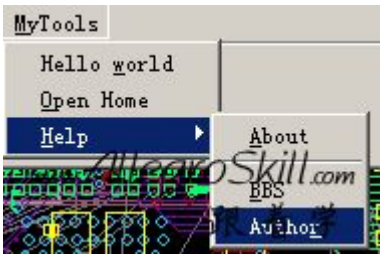
Allegro 带有一个菜单配置文件 allegro.mem 在每次启动时会调用该文件加载菜单，可输入 echo $menupath 命令查找菜单文件的位置

```
Command > echo $menupath
D:\SPB_Data16.5\pcbenv\cuimenus\.C:/Cadence/SPB_16.5/share/local/pcb/menus ;查找结果可能会有多个目录，第一个就是启动时调用位置，
```

用文本编辑器（如记事本、notepad++）打开编缉 D:\SPB_Data16.5\pcbenv\cuimenus\allegro.men，

通常大家会在 help 菜单后添加菜单，免乱了 allegro 已有菜单，那就在最后一行"END"前加代码，

来个实例吧，跟着糟菜一起学修改菜单:在 help 后添加"Mytools"，修改前请先备份菜单配置文件后再修改。




```
// -----MyTools menu for Allegro Begin-----
POPUP "&MyTools"
BEGIN
    MENUITEM "Hello &world", "palHelloWord"
    MENUITEM "&Open Home","palOpenHome"
```

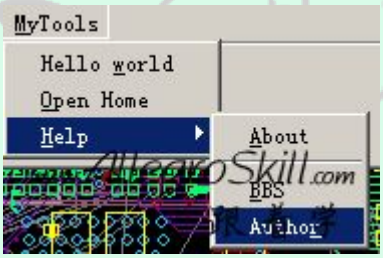
```
    POPUP "&Help"
        BEGIN
        MENUITEM "&About","echo 'MyTools 1.0'"
        MENUITEM SEPARATOR
        MENUITEM "&BBS", "http http://www.allegroskill.com/forum.php"
        MENUITEM "Autho&r","http http://wpa.qq.com/msgrd?V=1&uin=16831621"
        END
END
// -----MyTools menu for Allegro end-----
```

笔者一般为能类似"// -----MyTools menu for Allegro Begin/end-----" 这样一对注释做自定义菜单的开始与结尾，方便日后再次修改时容易找到上次的修改处。

由于您的不熟练操作，本操作就存在一定风险，如果你在修改菜单后，再次打开 allegro 编辑器时发现没有看到菜单，说明菜单修改有误。须认真查找格式不对之处。

§1-12 动态修改菜单

与手工修改菜单一样添加下图示例菜单。



方法一：适用于逐个添加菜单的程序下面是菜单源文件

```
;在 allegro.inilit 中添加(load "D:/mySkill/UserMenu.il")
defun(loadMyMenu  (t_menuName)
palMenu = axlUIMenuFind(  nil  -1)
```

```
palMenu1 = axlUIMenuInsert(palMenu 'popup  "MyTools")
palMenu2 = axlUIMenuInsert(palMenu1  "Hello &world","palHelloWorld")
palMenu2 = axlUIMenuInsert(palMenu1  "&Open Home","palOpenHome")
axlUIMenuInsert(palMenu2'end  )
palMenu2 = axlUIMenuInsert(palMenu1 'popup  "Help")
axlUIMenuInsert(palMenu2"About","echo 'MyTools1.0'")
axlUIMenuInsert(palMenu2"BBS","httphttp://www.allegroskill.com/forum.php?")
axlUIMenuInsert(palMenu2"Autho&r","httphttp://wpa.qq.com/msgrd?V=1&uin=1683
1621")
axlUIMenuInsert(plMenu2'separator  )
axlUIMenuInsert(palMenu2'end  )
axlUIMenuInsert(palMenu1'end  )
);defun
axlTriggerSet('menu 'loadMyMenu)
```

方法二：将菜单建成 list 放置于参数文件中，给程序调用

下面是菜单源文件

```
;在 allegro.inilit 中添加(load"D:/mySkill/UserMenu.il")
defun(loadMyMenu  (t_menuName)
          SKILLToolsMenu = '(
          (popup  "MyTools")
            ("Hello&world","palHelloWorld")
            ("&OpenHome","palOpenHome")
            (popup  "&Help")
            ("About","echo'MyTools 1.0'")
            (separator)
            ("BBS","httphttp://www.allegroskill.com/forum.php?")
            ("Autho&r","httphttp://wpa.qq.com/msgrd?V=1&uin=16831621")
              (end)
          (end)
          )
         let(  (palMenu)
          palMenu  = axlUIMenuFind(nil  -1)
```

```
            axlUIMenuInsert(palMenu SKILLToolsMenu)
          )
)
axlTriggerSet('menu     'loadMyMenu)
```

注释一下两种方法吧

菜单源码一注释：

```
;在 allegro.inilit 中添加(load "D:/mySkill/UserMenu1.il")
defun(loadMyMenu  (t_menuName)     ;定义菜单
palMenu = axlUIMenuFind(  nil  -1)  ;定位菜单位置于 Allegro 主菜单尾部，即"Help"后。
palMenu1 = axlUIMenuInsert(palMenu 'popup  "MyTools")  ;添加自定义主菜单" MyTools"
palMenu2 = axlUIMenuInsert(palMenu1  "Hello &world","palHelloWorld")  ;添加二级子菜单
 “Hello &world”
palMenu2 = axlUIMenuInsert(palMenu1  "&Open Home","palOpenHome")  ;添加二级子菜单“Open
Home"
axlUIMenuInsert(palMenu2 'end  )  ;添加二级子菜单结果，本语句可忽略
palMenu2 = axlUIMenuInsert(palMenu1 'popup  "&Help")  ;i 添加带有子菜单的二级子菜
单  "&Help"
axlUIMenuInsert(palMenu2  "About","echo 'MyTools 1.0'")  ;添加三级子菜单  "About"，示例内
部命令的直接使用。
axlUIMenuInsert(plMenu2 'separator  )  ;添加菜单分组横线
axlUIMenuInsert(palMenu2  "BBS","http  http://www.allegroskill.com/forum.php?")  ;添加三
级子菜单  "BBS"，示例如何访问网页。
axlUIMenuInsert(palMenu2  "Autho&r","http  http://wpa.qq.com/msgrd?V=1&uin=16831621")  ;
添加三级子菜单  "BBS"，示例如何与作者取得联系。
axlUIMenuInsert(palMenu2 'end  )  ;添加二级子菜单结果，本语句可忽略
axlUIMenuInsert(palMenu1 'end  )  ;添加主菜单结果，本语句可忽略
);defun
axlTriggerSet('menu 'loadMyMenu)  ;某个事件触发相应函数，本命令是自动加载菜单的关键！
```

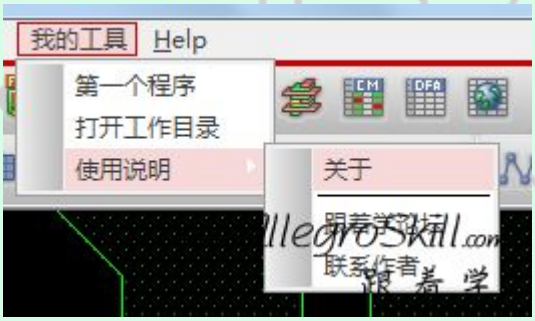
菜单源码二注释

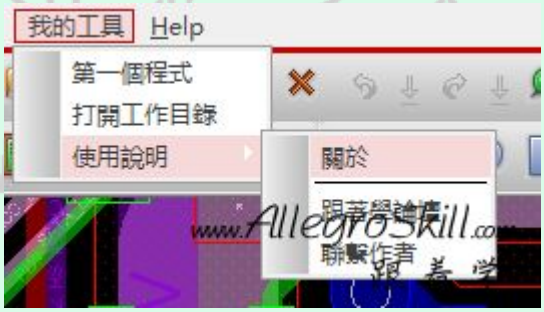
```
;在 allegro.inilit 中添加(load "D:/mySkill/UserMenu2.il")
```

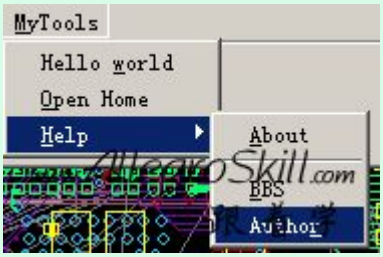
```
defun(loadMyMenu (t_menuName)  ;定义菜单
      SKILLToolsMenu = '(  ;定义一个 list,
      (popup "MyTools")  ;添加自定义主菜单"MyTools"
       ("Hello &world","palHelloWorld")  ;添加二级子菜单"Hello &world"
       ("&Open Home","palOpenHome") ;添加二级子菜单"Open Home"
       (popup "&Help") ;i 添加带有子菜单的二级子菜单 "&Help"
       ("About","echo 'MyTools 1.0'") ;添加三级子菜单 "About"，示例内部命令的直接使用。
       (separator)    ;添加菜单分组横线
       ("BBS","http http://www.allegroskill.com/forum.php?") ;添加三级子菜单 "BBS"，示例如何访问网页。
       ("Autho&r","http http://wpa.qq.com/msgrd?V=1&uin=16831621") ;添加三级子菜单 "BBS"，示例如何与作者取得联系。
        (end) ;添加二级子菜单结果
      (end) ;添加主菜单结果
      )
     let( (palMenu) ;定义局部变量
      palMenu = axlUIMenuFind(nil -1) ;定位菜单位置于 Allegro 主菜单尾部，即"Help"后。
      axlUIMenuInsert(palMenu SKILLToolsMenu)
     )
)
axlTriggerSet('menu  'loadMyMenu) ;某个事件触发相应函数，本命令是自动加载菜单的关键！
```

§1-13 多国语言的实现

以上篇中提到的动态修改菜单方式一为例，实现多国语言的菜单。







先请看下面的 l 中文转义字符例子：

```
printf("Cadence\n\tskill\n\t\t\270\372\316\322\321\247")
Cadence
    skill
        跟我学
t
```

其中 Cadence\n\tskill\n 每输出一行换行后再添加相应的水平制表符，再输出另外的内容。最后一行"\270\372\316\322\321\247"代表的是八进制的 ASCII 码，该部分 ASCII 码为汉子"跟着学"的八进制代码。

可通过转码软件获取汉字的转义字符

```
Skill > axlUIPrompt("Code:",sprintf(nil,"%L",axlUIPrompt("Please Input Chinese:")))
```

因为这里可能会用到较多的变量，所以再次重申一下变量的定义。

Skill 的变量不需要事先声明，Skill 第一次用到是会自动生成变量。变量可以由字符、数字、"_" 和 "?" 组成，注意第一个字符不能是数字和"?"。由于 Cadence 所开发的 Skill 中的变量、函数都是第一个字母小写，以_为开头的是 Cadence 的专用函数，为了避免冲突，建议函数和变量命名都以大写字母开头。有时 code package 前头使用三个小写字母更清晰些。

```
;在 allegro.inilit 中添加(load "D:/mySkill/UserMenu3.il")

;把所有菜单项名称均用一个变量来表示。
strMytools = "MyTools" ;我的工具
strHelloWorld = "Hello &world" ;第一个程序
strOpenHome = "&Open Home" ;打开工作目录
strHelp = "&Help" ;使用说明
strAbout = "About" ;关于
strBBS = "BBS" ;跟着学论坛
strAuthor = "Autho&r" ;联系作者

defun(loadMyMenu (t_menuName)
palMenu = axlUIMenuFind( nil -1)
palMenu1 = axlUIMenuInsert(palMenu 'popup "MyTools")
palMenu2 = axlUIMenuInsert(palMenu1 "Hello &world","palHelloWorld")
palMenu2 = axlUIMenuInsert(palMenu1 "&Open Home","palOpenHome")
axlUIMenuInsert(palMenu2'end )
palMenu2 = axlUIMenuInsert(palMenu1 'popup "Help")
```

```
axlUIMenuInsert(palMenu2"About","echo 'MyTools1.0'")
axlUIMenuInsert(palMenu2"BBS","httphttp://www.allegroskill.com/forum.php?")
axlUIMenuInsert(palMenu2"Autho&r","httphttp://wpa.qq.com/msgrd?V=1&uin=16831621")
axlUIMenuInsert(plMenu2'separator )
axlUIMenuInsert(palMenu2'end )
axlUIMenuInsert(palMenu1'end )
);defun
axlTriggerSet('menu 'loadMyMenu)
```

将语言配置放置另一个文件内， 如 "D:/mySkill/Language.ini"

```
LanguageType="Traditional Chinese"

;简体中文
if( LanguageType == "Simplified Chinese" then

 strMytools = "\316\322\265\304\271\244\276\337" ;我的工具
 strHelloWorld = "\265\332\322\273\270\366\263\314\320\362" ;第一个程序
 strOpenHome = "\264\362\277\252\271\244\327\367\304\277\302\274" ;打开工作目录
 strHelp = "\312\271\323\303\313\265\303\367" ;使用说明
 strAbout = "\271\330\323\332" ;关于
 strBBS = "\270\372\327\305\321\247\302\333\314\263" ;跟着学论坛
 strAuthor = "\301\252\317\265\327\367\325\337" ;联系作者

 )

;繁体中文
if( LanguageType == "Traditional Chinese" then
 strMytools = "\316\322\265\304\271\244\276\337" ;我的工具
 strHelloWorld = "\265\332\322\273\202\200\263\314\312\275" ;第一個程式
 strOpenHome = "\264\362\351_\271\244\327\367\304\277\344\233" ;打開工作目錄
 strHelp = "\312\271\323\303\325f\303\367" ;使用說明
 strAbout = "\352P\354\266" ;關於
 strBBS = "\270\372\326\370\214W\325\223\211\257" ;跟著學論壇
 strAuthor = "\302\223\300M\327\367\325\337" ;聯繫作者

)
```


修改 LanguageType 的值"Traditional Chinese"就可在下次重启 allegro 时动态切换菜单语言，这里示例 英文，简体中文，繁体中文，你还可以按自己的需求自定义其它国家的语种。

§1-14 实例演示

哈，总是要保留点干货放在我的官网上，不是吗？要不要大家都会了还去访问我的官网做什？

《Allegro Skill 跟着学》 §1-14 只想会用实战演示

http://www.allegroskill.com/forum.php?mod=viewthread&tid=935&extra=page%3D1&_dsign=c58d9e71

请记住：www.allegroskill.com 学习 AllegroSkill 语言，这一站就够了！期待您的加入。

附 访问官网.邀请加入

1. 本书所提到的实例，在官网均可下载，

《Allegro Skill 跟着学》先学会用课题附件

http://www.allegroskill.com/forum.php?mod=viewthread&tid=918&extra=page%3D1&_dsign=896f97f7

你任何有不明白的地方或是想分享 skill 相关知识也欢迎访问 www.allegroskill.com

2. 本书摘自 www.allegroskill.com 之【先学会用】版块的以下帖子

《Allegro Skill 跟着学》§1-1 不得不说的废话
《Allegro Skill 跟着学》§1-2 Skill 简介
《Allegro Skill 跟着学》§1-3 语言环境认知
《Allegro Skill 跟着学》§1-4 目录操作
《Allegro Skill 跟着学》§1-5 文件操作
《Allegro Skill 跟着学》§1-6 运行环境
《Allegro Skill 跟着学》§1-7 Skill 编辑器_Notepad++
《Allegro Skill 跟着学》§1-8 用户习惯性设定_env
《Allegro Skill 跟着学》§1-9 Skill 语法操作简介
《Allegro Skill 跟着学》§1-10 加载 skill 程序_allegro.ilinit
《Allegro Skill 跟着学》§1-11 手工修改菜单_allegro.mem
《Allegro Skill 跟着学》§1-12 动态修改菜单
《Allegro Skill 跟着学》§1-13 多国语言的实现
《Allegro Skill 跟着学》只想会用实战演示
《Allegro Skill 跟着学》先学会用课题附件

本书售价三元，如果您能支持一下那就最好了，如果不想那就算了。

购书入口:

http://www.allegroskill.com/forum.php?mod=viewthread&do=tradeinfo&tid=509&pid=516&_dsign=9715350f

3. 本站 www.allegroskill.com 力求打破网络论坛不系统的格局。你还在等什么，快快加入吧！