TeXmacs的入门讲座

主讲人：沈达

- TeXmacs 的安装(+)
- TeXmacs 的简单使用 (++)
- TeXmacs 文档的源码表示 (+++)
- TeXmacs 的内置作图工具 (++)
- 扩展 TeXmacs (+++)
- 获得帮助以及帮助 TeXmacs (+)

TeXmacs 的安装

1. 下载最新源代码, 尽可能下载SVN repo上的最新代码, 下载可能会比较慢
2. 编译

   如果你已经从软件源装过一次 TeXmacs 那么需要安装的依赖可能是下面这些：

```
$ sudo apt-get install guile-1.8-dev libfreetype6-dev # 安装依赖
$ ./configure
$ make -j4
$ sudo make install
```

3. (可选) 下载Fandol字体, 解压到这个目录 `$HOME/.TeXmacs/fonts/truetype`
4. 配置 TeXmacs 的 desktop 文件

   这样就可以双击 `.tm` 文件打开文档, 否则需要在终端下敲命令

   Troubleshootings : 请先参考豆瓣 TeXmacs 小组的 FAQ

TeXmacs 的简单使用1——界面

TeXmacs的界面从上到下分别为：

标题栏. 如果标题后面有一个 * 表示该文档已经被修改过

工具栏.

样式相关的工具栏. 从左至右分别是 :文件、编辑、转到.....

模式相关的工具栏. 主要方便你插入各种对象

对象相关的工具栏. 主要方便你编辑各种对象

Buffer.

状态栏.

交互式命令.

TeXmacs 的简单使用2——样式

样式 :同样的文档内容, 使用不同的样式, 就会由于样式对各种对象显示方式的定义不同而显示不同

宏包 :宏包是在样式基础上的微调

TeXmacs 的标准样式 :

article.

generic.

beamer. 本幻灯片所使用的样式

tmdoc. TeXmacs 的帮助文档所使用的样式

source. TeXmacs 文档的源码表示

TeXmacs 的简单使用3——模式

TeXmacs 主要有三种模式：

default.

math. 大部分输入$符号就会进入数学模式, 代码中先输入$再按Tab

graphics. 通过菜单 绘制图片 进入图形模式

command. 按下 \ 进入

source. 在source样式中就是source模式

Q : 如何输入特殊符号,如 $

A : 先输入$, 再按Tab

以插入一个超链接为例：

菜单. 插入→链接→超链接

⟨hlink||⟩

命令. \ h i n k RET

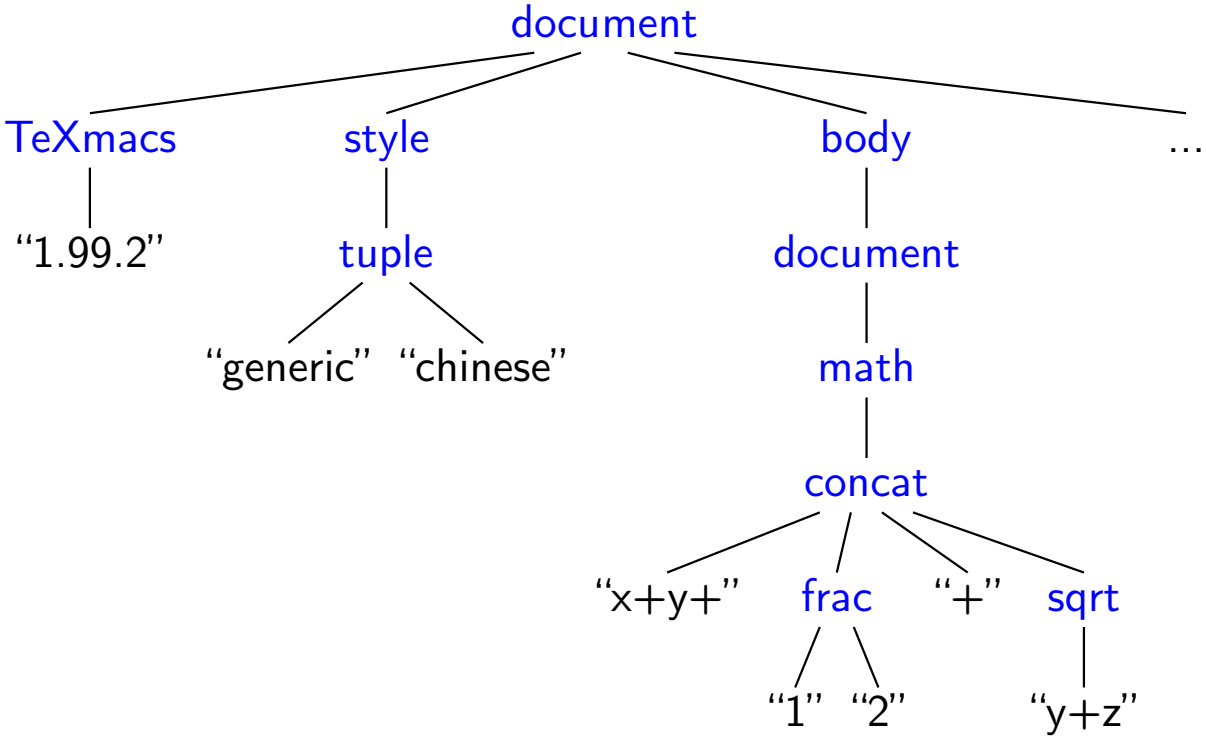
⟨hlink||⟩

快捷键. Meta-I >

⟨hlink|中国科大|http://ustc.edu.cn⟩

Tips：在菜单中可以看到快捷键，还可以在配置文件中自定义快捷键。

TeXmacs 文档的源码表示1 —— 内部结构



TeXmacs 文档的源码表示2———序列化

下面我们以一个简单的文档`hello.tm`演示文档的源码表示：

1. 切换文档的样式为`source`
2. 使用`source`的`scheme` style

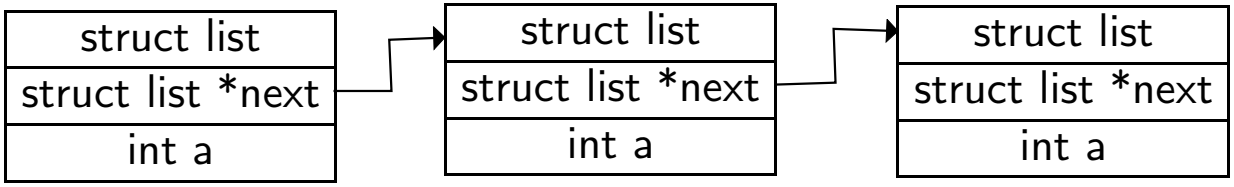
序列化：就是将文档树保存成文件的方法

默认格式. 缺点是不以Unicode方式保存, 中文没法看。优点是最接近可编辑的文档。

TeXmacs Scheme 格式. 其后缀名为stm,这种格式最为接近TeXmacs文档的内部表示———树, 在扩展TeXmacs很有帮助

XML 格式. 优点其一是XML本身的优点，其二是这是三种格式中唯一以Unicode方式存储中文的。用版本控制的时候，这可使得 Git Diff 信息更有意义

TeXmacs 的内置作图工具



上面是一个链表的例子, 可以用TEXMACS的内置作图工具完成。

演示：

- 基本对象
- 在其它内容上方绘制
- 如果一个对象被另外一个对象覆盖了, 该怎么办？
- 绘制链表
- 看一下链表这个例子的源码

TeXmacs和Scheme:

Extended Guile Scheme.

Guile Scheme用于构建TeXmacs的插件。

相当于后台。

和Emacs比较, Guile Scheme相当于是Elisp。

TeXmacs Scheme.

而TeXmacs Scheme则是由TeXmacs解释, 并渲染显示。

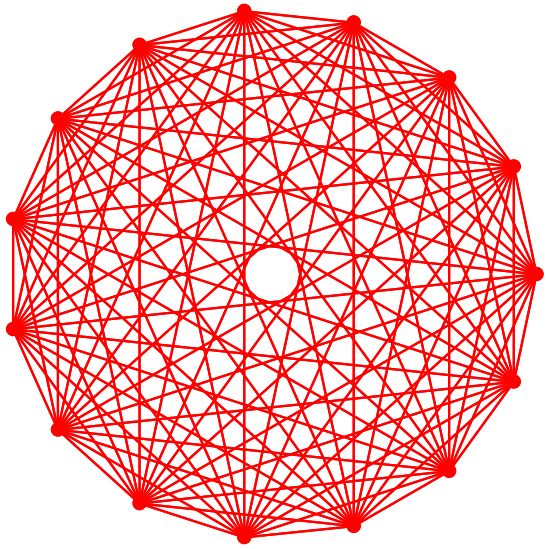
相当于前端。

和LaTeX比较, TeXmacs Scheme就是LaTeX文档的源码。不同的是TeXmacs可以所见即所得。

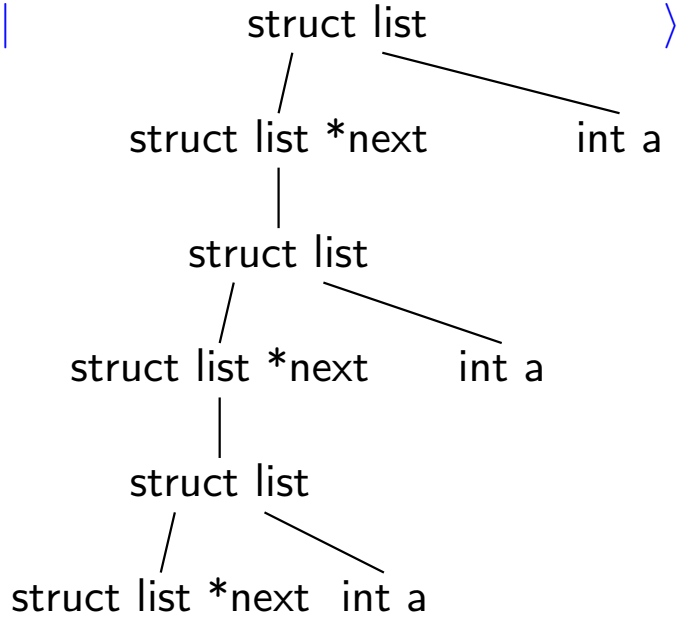
扩展 TeXmacs 2——rose

输入：⟨rose|3.5|15⟩

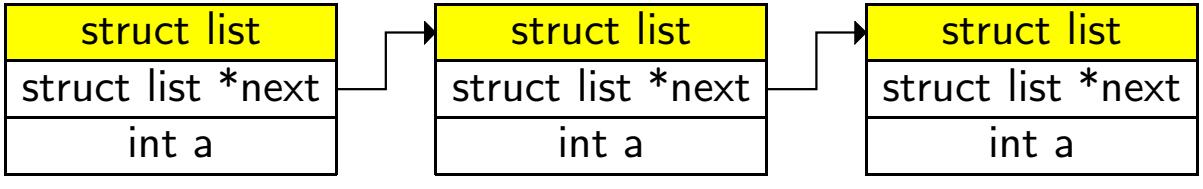
输出：



输入：⟨struct-graph|



输出：



⟨while|*condition*|*body*⟩ (重复求值)

在这个结构下，只要给定的条件*condition*始终满足，*body*就会重复接着排版.例如，申明了

```
⟨assign|count|
    ⟨macro|from|to|
        ⟨with|i|from|
            ⟨concat|
                ⟨while|⟨less|i|to⟩|i, ⟨assign|i|⟨plus|i|1⟩⟩⟩|
                to⟩⟩⟩⟩
```

则代码⟨count|1|50⟩会产生

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, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50

通过Guile Scheme可以做什么：

自定义快捷键.

```
Scheme] (kbd-map ("h i ." (insert "Hello World")))
```

自定义菜单.

撰写函数. 比如前面展示的rose和struct-graph实际上都是由Guile Scheme写成的

这些文件放在哪里：

```
~/.TeXmacs/progs/my-init-texmacs.scm
~/.TeXmacs/progs/my-init-buffer.scm
```

这就相当于Emacs的`init.el`文件了。

另外, TeXmacs的插件放在`~/.TeXmacs/plugins`, 样式文件在`~/.TeXmacs/packages`下

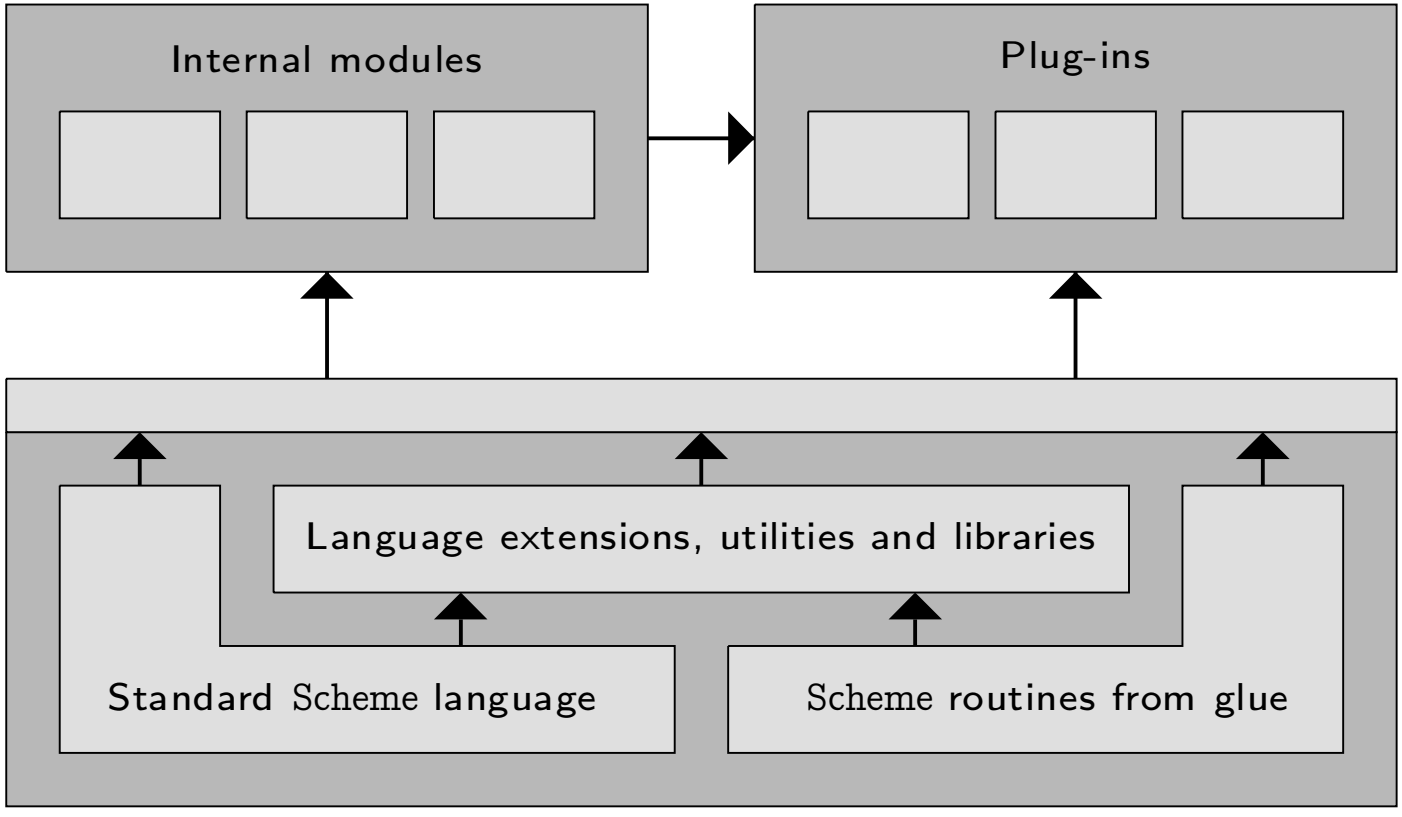


图 **1.** Schematic organization of the Scheme API.

⟨rose|3.5|15⟩

TeXmacs Scheme: ⟨assign|*rose*|⟨macro|*x*|*y*|⟨extern|rose|*x*|*y*⟩⟩⟩

展开就是：⟨extern|rose|3.5|15⟩

而rose是一个在外部的文件中定义的函数,我们看看这个函数生成了什么：

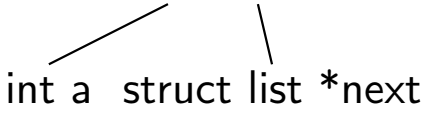
```
Scheme] (rose (tree "3.5") (tree "15"))
```

前面我们已经看过了图形的源码表示，所以说，所谓通过编程作图实际上就是在这些数据上进行操作。我们称之为数据驱动的编程模型。

在设计一个实用的函数的时候，我们首先要考虑这个函数的接口。因为TeXmacs接收的是用户的编辑，输出的是格式化的内容。而TeXmacs本身提供非常优秀的编辑接口。所以在设计接口的时候要因地制宜。

比如，struct-graph就利用了内置的tree作为输入。

⟨struct-graph| struct list ⟩



编写函数的一般格式:

```
(tm-define (fun arg1 arg2 ...)
  (:secure #t)
  ...)
```

如果不用(:secure #t)会倒是代码在运行的时候返回insecure script。

另外,帮助→扩展语言这一节文档非常重要,有必要读一下。

最后, 就是/usr/share/TeXmacs/progs的各种scheme源码。

获得帮助.

- 首选：TeXmacs的邮件列表
- TeXmacs的豆瓣小组

帮助**TeXmacs.**

- 使用TeXmacs
- 翻译TeXmacs的文档
- 撰写TeXmacs插件, 为TeXmacs报Bug
- 成为TeXmacs的开发者

此文档由制作 GNU TEXMACS (参见 `http://www.texmacs.org`).