第2章

使用交互环境

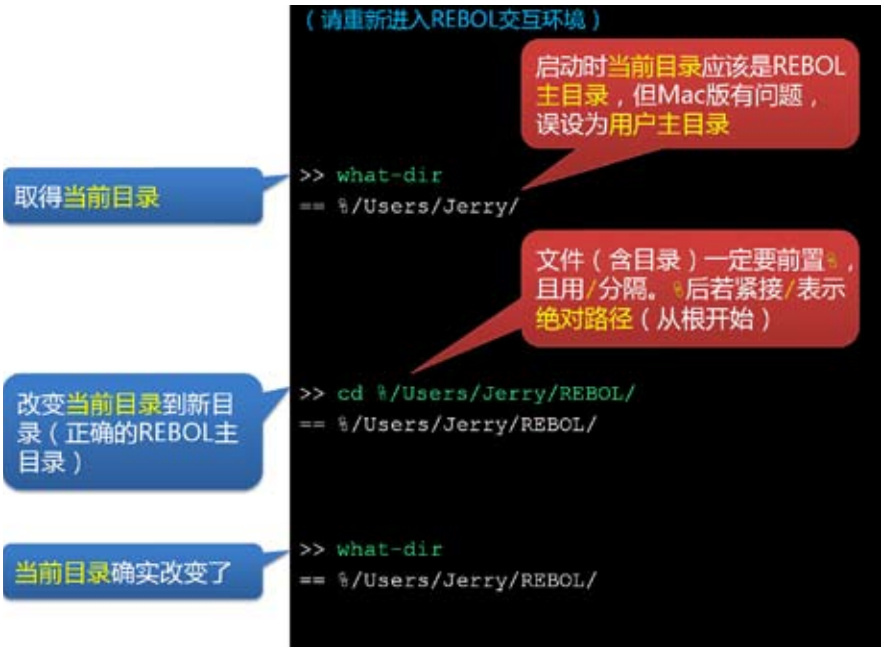
许多脚本语言的解释器都提供了交互环境，我们可通过此环境与计算机沟通交流。为什么叫做交互环境？因为我们在这里通过编程语言与计算机沟通，让它做任何事情，它都会立刻照办并给予我们反馈。

运行 REBOL 解释器的方式很简单，用鼠标双击 REBOL 解释器的文件即可，你会看到一个文字窗口，这个窗口就是 REBOL 的交互环境。一般来说，微软 Windows 的文字窗口底色是黑色，苹果 Mac OS X 文字窗口的底色是白色。本书一律用黑色底色表示 REBOL 交互环境的窗口。

窗口上出现大量的文字信息，包括 REBOL 的版本，版权声明，用法等。最后出现 `>>`，以及一个闪烁的光标，表示可输入代码。你可以在光标后面输入 REBOL 程序。

动手做下面的两个实验：

- 输入 `1 + 2`，再按下回车键。这是一个很简短的程序。
- 输入 `q` 或者 `quit`（`q` 是 `quit` 的简写），再按下回车键，就可以退出 REBOL。

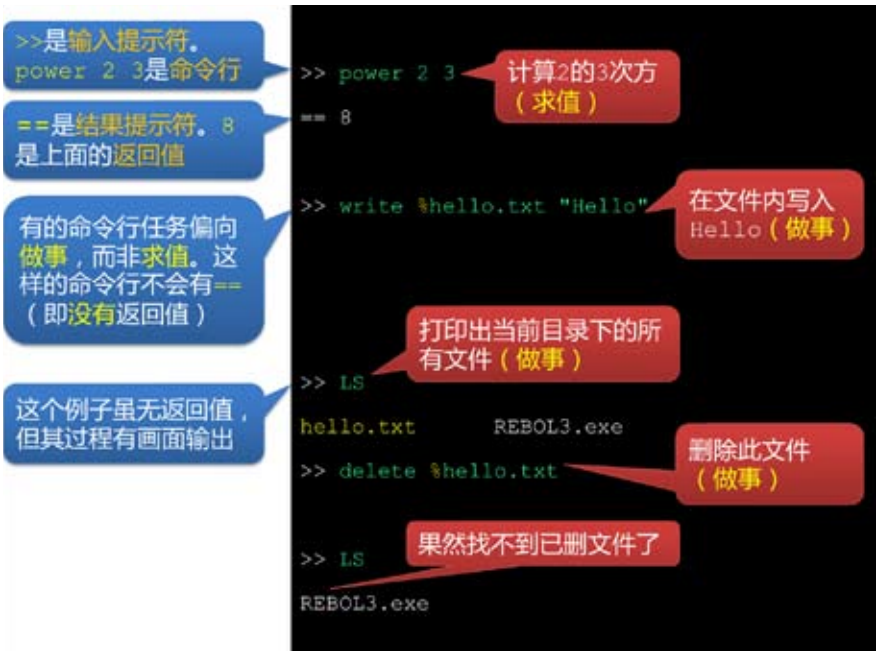


如果你遵照前面的操作方式，已经退出 REBOL 解释器，那么现在重新打开它吧！

我们可以通过 `what-dir` 来得知当前目录（dir）路径是什么（what）。所谓当前目录就是默认目录。若文件没有指定路径，默认该文件是在当前目录下。

Windows 版的 REBOL 初始时当前目录是没问题的，就是 REBOL 主目录。但 Mac 版就有问题了，居然是用户主目录。你可以通过 `cd` 函数来调整当前目录。`cd` 是改变目录（change directory）的意思。

值得强调一点：REBOL 规定文件（目录也算文件的一种）一定要前置 `%` 符号，且不管操作系统采用 `/` 还是 `\` 当做目录分隔符号，REBOL 一律采用 `/`。`%` 后面紧跟着 `/` 则表示绝对路径。



REBOL 解释器窗口也称为 REBOL 交互环境。在交互环境下，你会看到 >> 与 ==，分别是输入提示符与结果提示符。顾名思义，输入提示符提示我们后面可以输入程序，结果提示符提示我们后面出现的是执行的结果（即返回值）。在输入提示符后面，我们可以输入一行 REBOL 代码，再按下回车键，这行代码被称为命令行。

以此图为例，第一个命令行是 power 2 3（2 的前后有空格），计算 2 的 3 次方；第二个命令行建立一个文件 hello.txt，内容是 Hello；第三个命令行是 LS，列出当前目录下的所有文件（会看到刚才建立的 hello.txt）；第四个命令行是删除 hello.txt 文件。第五个命令行再度列出当前目录下的所有文件（会看到 hello.txt 消失了）。

有的命令行在运算的过程中有返回值，有的则没有返回值。简单地理解这件事：以求值或查询为目的的代码（power、what-dir、+），会有返回值；但以做事为主要目的的代码（write、LS、delete），就可能不会有返回值。

在交互环境下，不管要求 REBOL 解释器做什么事，都可以马上看到结果；出现错误（error）也可以马上看到报错。这里的四个例子，都是基于各自不同的原因导致程序出错而收到的错误信息。

- 第一个错误信息是 Access error，详细说明中提到“Not Found”（找不到）。我们链接到一个不存在的网址，当然无法取回网页，所以收到这样的错误警告。
- 第二个错误信息依然是 Access error，详细说明中提到“Timeout”（网络连接超时）。
- 第三个错误信息是 Syntax error，表示语法错误。我们在交互环境中乱打一通，不符合语法，所以收到这样的错误警告。
- 第四个错误信息是 Math error，表示数学错误。读小学时我们就知道不能拿 0 做除数，所以收到这样的错误警告。

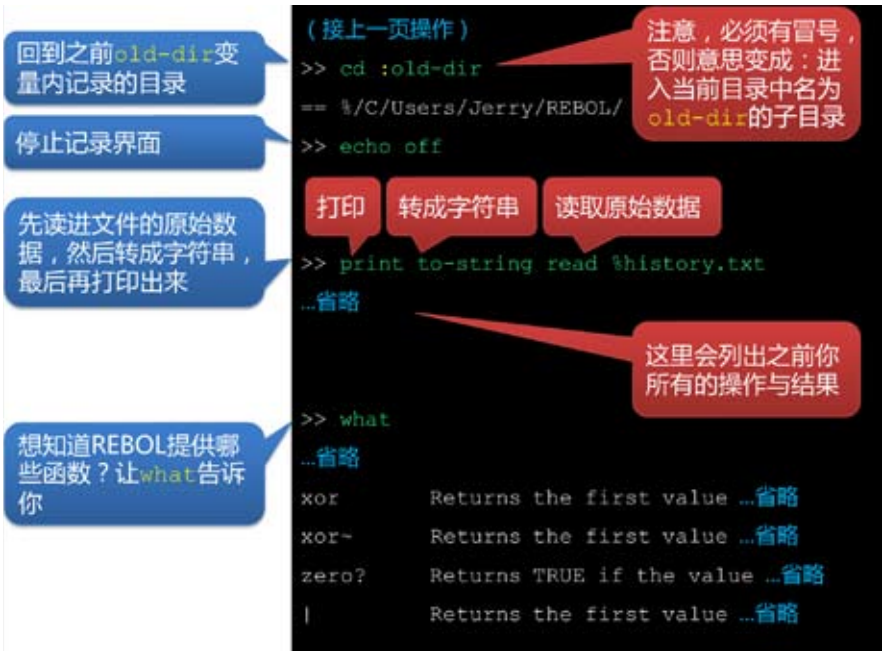
我们是 REBOL 语言的初学者，一开始会常看到错误信息，这是很自然的事，不要因此觉得沮丧。随着我们对 REBOL 语言越来越熟悉，看到报错的概率也就越来越小了。

错误信息内会有一些有用的信息，我们常常可以通过这样的信息，排查错误，并修改程序。以后看到错误信息，不要害怕，它们其实是来帮我们的。

只要 REBOL 解释器一关闭，之前的历史记录都会消失，下次又是一个全新的开始。如果你想完整保留你的操作记录（与界面输出记录），可以使用 `echo` 函数，`echo` 后面指定一个文件。上图中的例子把记录保存到 history.txt。如果你没有特别指定文件的路径，那么此文件会被放在当前目录下。当你想停止记录界面输出的时候，应输入的命令行是 `echo off`。

现在要输入些什么呢？随便玩玩：

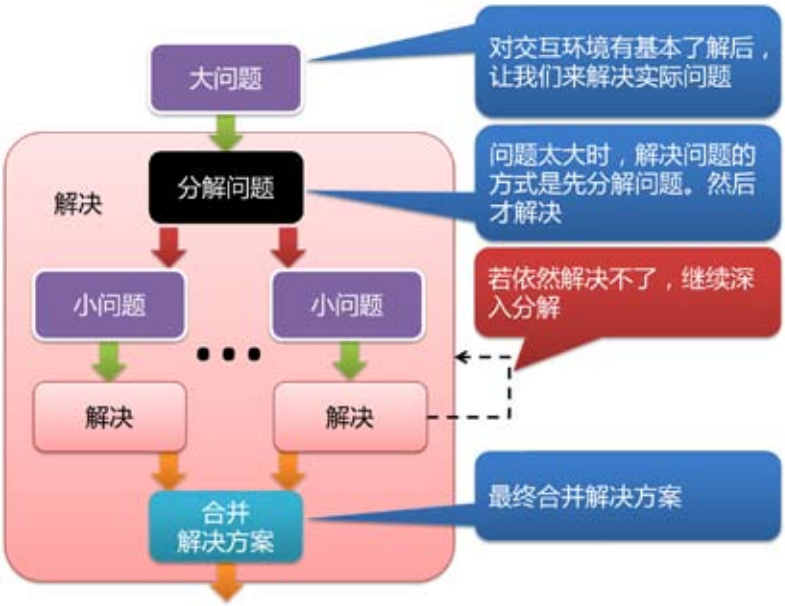
1. 先输入 `old-dir: what-dir`（REBOL 不区分单字的大小写），得到目前的目录路径，并把它记录在 `old-dir` 中。
2. 然后输入 `LS`，得到当前目录下的所有文件。我喜欢把 `LS` 写成大写，而非小写，因为小写的 `l` 一不小心就被错看为数字 1。`LS` 会列出（Listing）当前目录下的所有文件。你会看到其中有一个文件叫做 history.txt，这是我们刚刚通过 `echo` 建立的文件。
3. 接着输入 `cd ..`（注意 `cd` 后面有空格），让 REBOL 解释器把当前目录切换到上一层。`..` 是上一层目录的意思，`.` 是当前目录的意思。
4. 输入 `what-dir`，确定目录已上移一层。
5. 再输入一次 `LS`，会发现列出来的文件清单与刚才不一样了。



6. 通过 `cd :old-dir`，再把目录切换回之前的目录。
7. 通过 `echo off` 将记录功能关闭。
8. 通过 `print to-string read %history.txt`，我们可以把之前记录的内容调出来看。

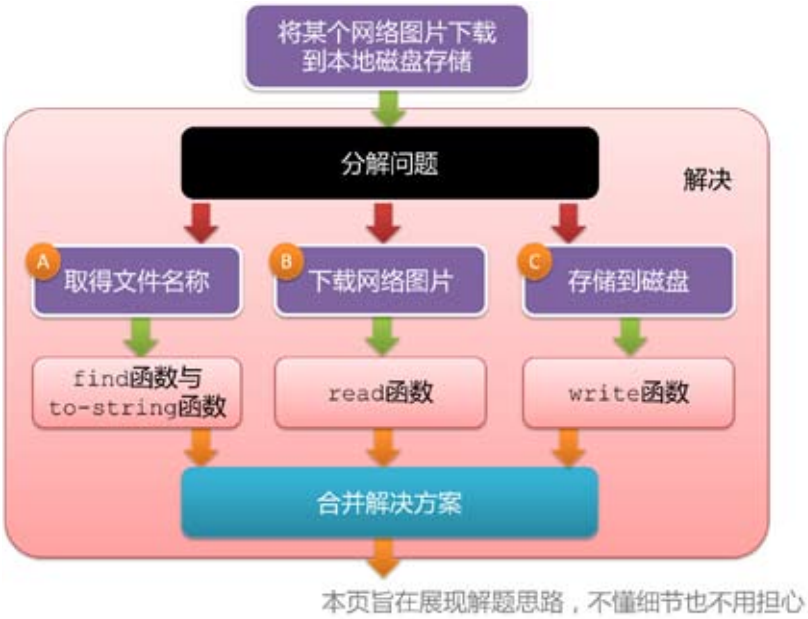
这里我们所练习的一些操作，都是针对文件系统的。REBOL 是一个不错的工具，可以协助我们管理文件。

除了文件相关的操作，REBOL 还提供许多操作，你可以通过 `what` 函数看到所有的函数功能。



对交互环境有了基本的认识之后，让我们来解决实际的问题。当我们拿到一个程序命题之后，首先考虑怎么解决这个问题，问题如果太大，就需要先分解，各个击破，然后再合并。

如果分解后的问题依然无法简单解决，就继续分解，直到可以解决为止。



如果我们要解决的问题是将某网络图片下载到本地磁盘，可以把这个问题分解为：

A. 取得文件名称。
B. 下载网络图片。
C. 存储到本地磁盘。

对应的解决方式分别为：

A. 通过 `find` 和 `to-string` 函数（你目前不需知道这两个函数的细节）可以取得文件的名称。
B. 通过 `read` 函数可以从一个 URL 下载图片。
C. 通过 `write` 函数可以把图片存到磁盘。

针对上面三个小问题的方案，每个问题分别用一个命令行的代码负责处理。在交互环境下，我们所做的操作会影响后续的操作，而且会持续积累，所以我们可以把任务拆分后再执行。这三个操作虽然分别在三个命令行中处理，但是和放在同一行中处理的效果是一样的。

为了做这个实验，先随便找一个网站图片，我找了 http://www.rebol.com/graphics/reb-logo.gif 这张图片。为了谨慎起见，请先通过网页浏览器输入这个网址，确定这张图片是存在的。特别注意，网址必须完全一样，连大小写也必须一样，因为有些网站是会区分网址大小写的。

先在 REBOL 解释器交互环境中把 `url` 设置好，接着按顺序做这三个操作：

A. 先对 `url` 加工处理，得到文件名称（`file-name`）。

B. 从网络上读取 `url` 的图片。执行完之后，你会看到一大串返回值，这是图片内容的十六进制原始数据，我们不用理会。（注意，有可能出现超时错误，或页面不存在错误。）

C. 执行完 `write` 命令行之后，没有返回值，也没有任何中间信息，就表示成功了，图片已经被存储到本地文件系统中。我们可以在当前目录中找到这张图片的文件。鼠标双击它，打开这张图片，确定内容正确。

有些程序只使用一次，以后不需要反复使用，这类程序称为一次性程序，或抛弃式程序。如果一次性程序需要的代码量很少，我们通常会直接在交互环境中完成它，不需要写成脚本文件。例如从网络下载某些文件，处理本地文件等，交互环境相当适合这些一次性程序。

如果上一页的操作一切顺利，你已经下载了一张图片。要如何下载另一张图片呢？全部重新输入吗？不需要。你只需要输入第一行命令，把 `url` 设置好即可，后续 A、B、C 三个操作可以直接调用历史记录。

如何调用历史记录？通过上下键即可！从历史记录中找到正确的命令行之后，按下回车键。

交互环境中有一些基本的按键操作，我们必须知道。

每次完成一个命令行，最后都要按下回车键（Enter），以告诉 REBOL 解释器输入完毕，开始执行。只要还没按下回车键，都可以使用左右键移动光标，或者使用退格键（Backspace）删除前一个字符。

我们可以使用上下键调出之前输入过的某命令行，以重复使用，或修改后使用。调出历史记录还有另一个方式：F7 键可以调出历史记录窗口，然后用上下键移动命令行，按下回车键选择命令行，或按下退出键（Esc）直接退出历史窗口。Mac OS X 不支持 F7 键调出历史记录功能。

光标的外观为一条线时表示插入模式，外观是方块则表示改写模式。可用 Insert 键切换这两种模式。

在交互环境下，你可以使用鼠标将某段文字框选起来（反白），将光标移动到反白文字上按下鼠标右键，则反白消失，就完成复制（文字已经被记录在内存中）。如果你使用的是 Mac OS X，可以用鼠标将某段文字框起来（反白），将光标移动到反白文字上按下鼠标右键，出现一个菜单，在菜单中选择“复制”即可。

Windows 中粘贴的方式是把光标移动到欲粘贴的位置，然后按下鼠标右键。Mac OS X 的粘贴方式是同时按下 Command 和 V 键。

REBOL 的交互环境使用操作系统的基本按键操作，所以不是很好用，但没关系，因为大多数时候，我们是在文本编辑器中写代码（稍后说明），而不是在交互环境下写代码。