Erlang编程

Erlang的并发特性源自于语言本身而非操作系统。它把现实世界模拟成一系列的进程，其间仅靠交换消息进行互动。由于erlang简化了并行编程，在erlang世界中，存在并行进程但是没有锁，没有同步方法，也不存在共享内存污染的可能，因为erlang根本没有共享内存。

Erlang程序可以由几百万个超轻量级进程组成，这些进程可以运行在单处理器、多核处理器或者处理器网络上。

**变量**

Erlang中所有的变量都必须用大写字母表示，且不能修改。一个变量如果含有一个被赋予的值，就称为绑定变量，否则，则被称为自由变量，一开始，所有的变量都是自由变量

**原子**

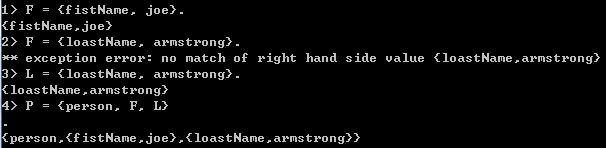
Erlang中的原子是全局有效的，而且无需使用宏定义或者包含文件。原子是一串以小写字母开头，后跟数组字母或下划线(\_)或者邮件符号(@)的字符。例如，red、december、cat、joe@somehost及a\_long\_name等

使用单引号引起来的字符也是原子，使用这种形式，我们就能使得原子可以用大写字母作为开头或者包含非数字字符。例如：’Monday’、’+’、’an atom with spaces’，你还可以将原本不需要使用引号的原子引起来，’a’实际上就等同于a

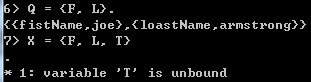
**元组（tuple）**

将若干个以逗号分割的值用一对花括号括起来就形成了一个元组。例如，想要描述某个人的名字和他的身高，可以用{joe、1.82}。这个元组包括了一个原子核一个浮点值

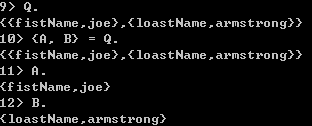
如果创建的一个新元组引用了一个已绑定的变量，那么新元组就会享有这个变量所引用的数据结构。



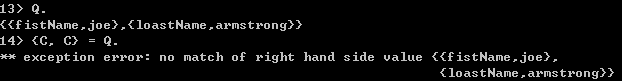
如果在创建数据结构时试图引用一个未定义的变量，系统就会给出一个错误

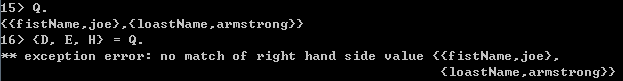


从元组中提取字段值

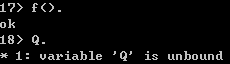


位于等号两边的元组必须含有相同数量的元素，两边相对应的元素必须绑定相同的值

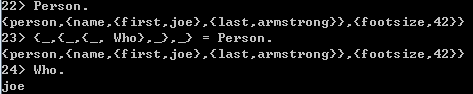




采用f()可以取消所有的变量



\_作为占位符，表示那些我们不关心的变量，符号\_称为匿名变量。与常规变量不同，在同一个模式中的不同地方，各个\_所绑定的值不必相同

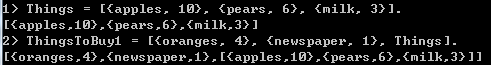


**列表**

将若干个以逗号分割的值用一对方括号括起来，就形成了一个列表，列表的第一个元素称为列表的头，那么你可以想象一下如果从列表中移除头，所剩下的东西就是列表的尾，列表的头。列表的头可以是任何东西，但是列表的尾通常还是一个列表。访问列表的头是一个非常高效的操作，因此，实际上所有的列表处理函数都是从提取列表头开始的，先会头进行处理，然后继续处理列表的尾。

如果T是一个列表，那么[H]T]也是一个列表，这个列表以H为头，以T为尾。竖线符号可以将列表的头和尾分割开来，而[]则是空列表。

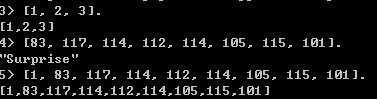
可以用[E1,E2…En|T]这种形式向T的起始位置加入多个新元素。



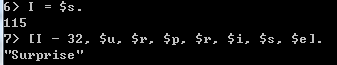
**字符串**

严格的讲，erlang中没有字符串，字符串实际上就是一个整数列表。用双引号将一串字符括起来就是一个字符串。在某些编程语言中，字符串既可以使用单引号也可以使用双引号，而在erlang中，必须使用双引号。

Shell打印一串列表值时，只有列表中的所有整数都是可打印字符时，它才把这个列表当做字符串来打印。



我们还可以通过$来表示字符的整数值：



字符串的字符时Latin-1(ISO-8859-1)编码的字符，erlang所关心的，只是以某种编码方式编码的一串整数值列表，如果碰巧是在Latin-1编码下，那么他们应该可以正确显示。

**模块**

模块是erlang中代码的基本单元，我们编写的所有函数都存在于模块之中。模块文件通常存放在以.erl为扩展名的文件中。

要运行一个模块，首先需要编译它，编译成功之后的模块文件其扩展名是.beam