2007.9.4  
  
**引  言**  
嵌入文档（Here Documents）技术是Unix/Linux平台中脚本语言BASH提供的一个特征，是用于在当前脚本内部处理重定向的一种手段。由于Unix从设计之初就是遵循“使用小而简单的工具进行无缝的集成”的理念，完成一项工作是需要大量的文本处理工具、流处理工具进行合作，通过标准输入、输出进行连接，将一个工具的处理结果转向另一个工具进行加工，直到得到最终的处理结果。因此，I/O重定向（redirection）是SHELL脚本的一个重要语言特征。但是，重定向技术也使整个处理过程显得凌乱、琐碎，对于初学者往往把握不好重定向的使用时机，甚至滥用重定向，而使得整个脚本里存在大量的中间文件，极大的损害了脚本语言的“优雅”程度。这是SHELL脚本学习过程中的一个常见问题。  
为解决这个问题，SHELL脚本在设计上尽可能的减少中间文件，增加了“Here Documents”这一特征。Here Documents提供了在脚本内部保存原始文件，以及在脚本内部提供重定向的机制，是精简脚本的有效方式，也是反映脚本编程水平的重要特征。在各种资料中“Here Documents”有不同的翻译名称，例如“嵌入文档”、“内部文档”、“现场文档”等等。为了兼顾字面上的相近和使用上的形象，本文使用“嵌入文档”这一名称，并通过若干实例展示这一功能的使用方法。  
  
**一、问题的提出**  
事情是这样的，在8月初，笔者负责某信息学竞赛考试服务器的准备和维护工作（[http://gait.buaa.edu.cn:8765](http://gait.buaa.edu.cn:8765/)），在考试结束需要将服务器上的考试成绩数据导出来，发送给竞赛主办方。服务器上安装的竞赛考试系统是在Linux下开发的专用系统，其中保存的成绩数据都是采用自定义格式的文本文件，并且由于系统开发并不是非常完善，没有提供很好的导出成绩的功能。而主办方当然希望看到一个直观的Excel文件的总成绩表。因此，需要紧急的手工处理一下，将服务器上的文本文件转换成Excel表出来。在这种情况下，当然首选的是SHELL脚本文件。  
在考试系统里面，成绩的统计结果已经可以方便的在一个统计工具的WEB页面里查看。这个统计工具当初设计的还是非常实用的，可以根据考试人员的需求，在页面上查询考试人员的成绩、分数，等等。但是这个工具当时实现时存在一些问题，它只是列出了成绩信息，而没有把考试人员的一些基本信息列出来，包括考试人员的联系方式等注册属性。但是这个信息还是要给主办方的，而且要求都显示在同一个Excel文件中。改程序已经来不及了，因此这次只能手工处理配置文件了。  
根据历次的经验，对于这种批量文本的转换处理，使用SHELL脚本的效率还是非常高的。因此立即着手进行。  
  
**二、需求和目标**  
首先要解决的是弄清考试系统里保存的数据文件都是什么格式。通过与系统的开发人员相切磋，从系统中找出了与本次处理相关的文件格式。  
  
2.1 考生基本信息文件  
  
在当前目录下有所有考生的基本注册信息文件，每个考生保存为一个文件。在系统中，考生有唯一标识是一个ID，是一个32位整数，使用这个整数保存成该ID的"%8X.REG"形式的文件名（当时使用这种格式主要是为了显示的直观和规整）。  
例如，我本人的ID是4，那么我的存储文件名为“00000004.REG”。  
信息文件的内容如下所示：  
  
--------------------------------------------------------  
考生基本信息文件（00000004.REG）  
--------------------------------------------------------  
1        loginid:jgj   
2        pwd:kkaACXiDLyk=   
3        repwd:kkaACXiDLyk=   
4        name:靳国杰   
5        sex:男   
6        id:130229197911092212   
7        birthday:19791109   
8        school:北航   
9        class:BY0306   
10        phone:01082317614   
11        mobile:13691250730   
12        address:北航逸夫馆   
13        code:100083   
14        mail:jin@cse.buaa.edu.cn  
--------------------------------------------------------  
  
关于文件格式的说明：   
1.        上面这个文件中共包括14行，每一行形如“注册属性名称：属性值”，即冒号左边是字段名称，冒号右边是值。该文件一共14行，存储了每个人员的14项注册信息。这些信息都是要给主办方的。  
2.        通过名称可以直观的看出，注册属性包括登录名、密码、确认密码（与密码相同）、姓名、性别、身份证号、出生日期、学校、班级、电话、手机、地址、邮政编码、邮箱。  
在本次考试中一共注册了650人，因此，注册文件共有650个，文件名是从00000001.REG到0000028A.REG。  
  
2.2 汇总文件  
  
为了给竞赛主办方的人员一个清晰的材料，需要将这650个文件汇总，形成一份总表TOTAL.CSV，每人一行，15个字段各占一列。为方便起见，同一行中各字段以“～”分隔，从而能够方便的导入到Excel中，形成给主办方的最终结果。汇总文件TOTAL.CSV如下表所示：  
  
--------------------------------------------------------  
汇总文件（TOTAL.CSV）  
--------------------------------------------------------  
ID~登录名~密码~确认密码~姓名~性别~……   
00000001~bootluck~QkxldMs0JllDRj5pIczVIw==~QkxldMs0JllDRj5pIczVIw==~Sun~男~……  
00000002~xinghe~RA9S0UZEIHVDRj5pIczVIw==~RA9S0UZEIHVDRj5pIczVIw==~xinghe~男~……  
00000003~alice607~9sCyToqCsQd5ouKr/39oaw==~9sCyToqCsQd5ouKr/39oaw==~alice607~女~……  
00000004~jgj~kkaACXiDLyk=~kkaACXiDLyk=~靳国杰~男~……  
……  
……  
……  
……  
……  
……  
0000028A~holybear~a1aACXiDLyk=~a1aACXiDLyk=~李圣雄~男~……  
--------------------------------------------------------  
  
  
关于文件格式的说明：   
1.        上面这个文件中共包括651行，第一行是所有注册字段（以“～”分隔）。之后的每一行对应一个考生的所有注册信息。  
2.        通过名称可以直观的看出，注册属性包括登录名、密码、确认密码（与密码相同）、姓名、性别、身份证号、出生日期、学校、班级、电话、手机、地址、邮政编码、邮箱。  
  
3.3 汇总要求  
  
因此，在这个工作中，需要对现有的数据文件进行如下的汇总过程，以形如最终提交的汇总文件。算法如下：  
① 对每个考生信息文件，抽取冒号右边的信息，得到所有注册字段名；抽取冒号右边的信息，得到所有属性值；  
② 对抽出的14个字段，以适当的分隔符分隔（比如“～”），拼成一行，总为汇总文件的首行；  
③ 对每一个考生的属性值，以“～”分隔，形成一条考生信息记录，作为汇总文件的一行；  
④ 通过编写的Shell脚本，对所有650个考生信息文件执行上述过程，最终形成一个有651行的CSV（逗号分隔字段）文件。  
形成的CSV文件导入到Excel中就是要发给主办方的文件。由于CSV导入Excel的操作比较简单，这里就不再赘述，下面重点讲解Shell脚本的编写过程。  
  
**三、技术基础**  
我们采用Red Hat Linux和Unix下面广泛使用的BASH来编写脚本。首先讲解有关Here Documents的语法基础和基本使用方法。  
  
3.1 语法结构  
在命令行中执行man bash，可以找到BASH手册中有关Here Documents的语法定义，其一般形式如下：

|  |
| --- |
| Here Documents        This  type  of  redirection  instructs the shell to read input from the        current source until a line containing  only  word  (with  no  trailing        blanks)  is seen.  All of the lines read up to that point are then used        as the standard input for a command.         The format of here-documents is:                <<[-]word                       here-document               delimiter         No parameter expansion, command substitution, arithmetic expansion,  or        pathname expansion is performed on word.  If any characters in word are        quoted, the delimiter is the result of quote removal on word,  and  the        lines  in the here-document are not expanded.  If word is unquoted, all        lines of the here-document are subjected to parameter  expansion,  com-        mand  substitution,  and arithmetic expansion.  In the latter case, the        character sequence \<newline> is ignored, and \ must be used  to  quote        the characters \, $, and `.         If the redirection operator is <<-, then all leading tab characters are        stripped from input lines and  the  line  containing  delimiter.   This        allows  here-documents within shell scripts to be indented in a natural        fashion. |

为了便于说明，这段说明翻译成中文如下所示：

|  |
| --- |
| #man bash     Here Documents（嵌入文档）        Here Documents作为重定向的一种方式，指示shell从源文件的当前位置开始读取输出，直到遇到只包含一个单词的文本行时结束。在该过程中读到的所有文本行都将作为某一个命令的标准输入而使用。        here-documents的使用形式:                <<[-]word                       here-document               delimiter         Shell将不对word 进行任何参数和命令替换、表达式计算，以及文件名扩展。如果在word中出现被引号包含的字符，则delimiter 将被视为在word中去除引号所包含部分的结果，并且here-document中的文本将不被扩展。如果word不包含引号，则here-document中的所有文本都将进行常规的参数扩展、命令替换、表达式计算。在后一种情况下，字符序列\<newline>将被忽略, 并且必须对元字符\, $, 和`使用\进行转义。        如果重定向操作符是<<-, 则输入行和delimiter行中的所有前缀TAB字符都将被忽略。这样源代码中的here-documents就可以按照优雅的嵌入方式进行对齐。 |

3.2 技术的应用  
下面我们考察一下在Shell脚本中使用Here Documents的必要性所在。Shell脚本对于文本文件的处理是非常方便的，因此实现本文所要完成的工作还是比较简单的，只需对所有文件进行一次循环过程。但是问题在于，从①-②，②-③的每一部，实际上都是使用cut、echo、cat等进行流处理，每个处理过程的输出作为下一步骤的输出。这里面自然的要使用重定向的方法。在以前对BASH用的不熟的时候，往往笨拙的使用临时文件存储这些中间结果。比如： 00000004.REG -> 00000004.cut -> 00000004.line -> all.txt。这样处理的过程中，代码显的非常杂乱，而且有时候不得不写出多个脚本来处理重定向，这种情况使得问题更为加重。  
因此，这次下决心要将代码写的精简一些。在BASH内部可以使用Here Documents进行重定向，使流的输入输出都是在同一脚本内进行的，不用拆开成多个脚本，也省略了大量的中间文件，使代码非常清晰、紧凑。  
  
Here Documents的一般形式如下： 

|  |
| --- |
| 0001  commands <<ID  0002     here documents  .......     ......  000n  ID<\n> |

在上面这个语法描述中，0001行的commands是任何SHELL命令。ID是一个字符串标识，标志着嵌入文档的开始（习惯上，常用的标识是EOF，等等）。从0002行可以写任何文本，这里写入的内容都将作为标准输入传送给commands。在最后一行写标识符ID，代码嵌入文本结束。   
  
在使用Here Documents编写脚本的时候注意以下两点：   
  
1.        在000n行上，必须只有一个ID，也就是说，ID前面不能有空格，ID后面必须是回车符。否则SHELL处理时会出错。这一点在实际时要尤为注意。  
2.        嵌入文档内可以写普通文本（比如说，写程序的使用Usage），也可以是标准的SHELL语句。如果是SHELL语句，要使用反引号（`）括起来，这时候语句的标准输出也将作为嵌入文档送给commands。这个特性非常有用。   
  
下面举一个简单的例子： 

|  |
| --- |
| [exam@Exam-Server user]$ nl -b a  aa.sh      1  #!/bin/bash      2       3  cat <<EOF      4    Now:      5    `date`      6  EOF [exam@Exam-Server user]$ sh aa.sh   Now:   三  7月 26 19:23:24 CST 2006 |

上面是一个最简单的例子，但是已经基本概括了Here Documents的使用方法。在aa.sh中，EOF作为嵌入文档的标识ID。4、5行中的Here Documents是一行标准文本“Now”和一个命令date的输出。两部分输出送给了cat命令去执行。  
  
  
四、问题的解决   
通过反复的编写、调试，基本上最终的脚本go.sh如下表所示：

|  |
| --- |
| [exam@Exam-Server user]$ nl -b a go.sh      1  #!/bin/sh       2       3  echo "ID~"`grep title  reg\_prop.xml  | cut -d\" -f2 | ./toline.sh`       4  for v in 0\*[0-9A-F].REG; do      5    echo -n "$v~"      6       7    while true; do      8      read line      9      10      if [ "$?" == "1" ]; then     11        break     12      fi     13      14      echo -n $line"~"     15    done  <<EOF     16    `cut -d: -f2 $v`     17  EOF     18      19    echo     20  done |

由于SHELL脚本起源于上个世纪70、80年代，因此可读性比当前的编程语言一般要差一些，因此下面对这个脚本中的内容作一下简单的说明：  
  
1. 第4行-20行的for循环是对650个文件依次进行处理。（对于分隔符，本来是想用分号，或者逗号，或者空格。结果这些字符在注册信息里全出现了，后来选了“~”号）。   
  
2. 第7-15行是while循环，执行的基本功能是把每个信息文件拼成一行。   
  
3. 16行是执行cut命令，提取文件中冒号右边的内容，把输出作为Here Documents送给while循环。   
  
4. 在while循环中，第8行是使用read命令，每次读取一行，使用“~”拼成一行。注意这里有个特殊的处理：因为在信息文件中某一项注册项可能是空的，这样16行的输出可能中间有空行。因此，判断read的结束不能使用if [ "$line" == ""] 的方法。经过调查，发现可以从read的返回值获取是否到达行末的状态。如果是遇到Ctrl-D结束，那么返回值是1。如果是正常的回车键结束，即使没有任何内容，返回值也是0。因此，正确的判断是 if [ "$?" == "1" ];   
  
5. 17行的EOF是代表Here Documents的结束。注意此处不能像写C程序一样，为了美观而把EOF向右缩进，因为SHELL规定EOF必须是该行唯一出现的内容，连空格都不能有。因此，这个EOF就暂时突出在这一行了。（通过将输入操作符改为“<<-”可以解决这个问题，但是这里也无伤大雅）。  
  
6. 文件中的toline.sh是自已编写的另外一个常用脚本，用于把输入的若干文本行拼成一行，并且以“～”分隔。这个脚本的实现比较简单，建议留给读者自己练习编写。  
五、总结  
Shell脚本是进行服务器维护和日常管理的专用语言，是每一个Unix/Linux服务器管理员必须掌握的基本功。鉴于这种工具对文本文件处理的方便性，在应急处理一些工作时也是首选工具，本文中工作要求就是一个典型的应用。在Shell脚本的众多语法结构中，Here Documents可以说是比较深入的内容了，因此在网上的资料中并不算十分常见，即使是在Red Hat内置的大量脚本中，这种技术也不是用的特别普遍。但是深入掌握一种语法结构必将大大提高工作效率。在本文的形成过程中参考了网上其它一些文档和资料，读者可以参考更多的关于Shell脚本深入编程的文章，掌握这一利器，为我所用。