管道的力量——记一次脚本的实现

由来

记得在《UNIX编程艺术》一书中，有关于unix系统中的管道的讲解，unix习惯于将任务划分成一个个的不同的任务，然后使用管道将这些任务的输入输出连接起来，简洁而有力，今天遇到一个任务，任务是这样子的。在数据库中有一个worker表，这个表里面由于程序的版本原因，现在的表结构比上一版本的表结构多了几个字段，这几个字段默认为null的，所以之前的用户这几个字段的数据就是null了，现在就是需要将这些数据修复成正确的数据。

动手

这些数据的来源是根据一个key，通过https请求，从另外一个系统中获得，这个key在worker表中存在，并且是唯一的。https请求回来的结果的原始数据是json格式的，然后通过des加密，然后再经过base64编码，最后将每个byte转换成两个‘0-F’的字符。最终返回的数据就是一串‘0-F’组合的字符串。

因此，这个任务就分成了如下几步：1、从数据库中获取需要修复的用户的key，2、通过https请求获取加密的字符串，3、将这个字符串解密，将其中一部分的信息转换成json格式字符串，4、将这些字符串更新对应key的记录。

本来计划使用一个python脚本实现，但是由于使用的linux系统没有root权限，也没有安装MySQLdb，但是有urllib2库，所以https请求交给python来实现。由于解密这段在另外一个java文件中已经实现了，而重新在学习python的des解密和base64编解码也需要一定的时间，因此，综合以上的限制，就将这个任务分解成上面的4个任务，所有的mysql操作（1和4）使用shell脚本实现，https请求这部分由python实现，解密部分由java实现。

它们之间的连接就通过管道。第一个任务的输入是从mysql来的，所以第一个任务一个简单的mysql命令就可以实现了，如下： mysql -hxxx -P3306 -uxxx -pxxx database -e "select fid from worker where fid is not null and custominfo is null or custominfo = '' " | sed "1d"

这里就用到了管道，将mysql的结果输入到sed中，将第一行（第一行是字段名）删除，这里明白了原来管道只处理标准输出，因此直接运行mysql命令和通过管道交给sed之后输出的结果不一样（前者会有“|”列成表格，但是为什么输出到sed或者输出到文件就没有了？！），这个命令的结果就是所有的key的集合。

第二个任务是通过python实现的，将每一个key从标准输入中读取，然后通过https请求访问服务端获取加密之后的数据data，输出的格式是key&data，之所以使用'&'作为分隔符，是因为所有的中间结果中不会包含这个字符，所以分割字符的时候不会导致意料之外的错误。

第三个任务是通过java程序实现的，因为解密的接口已经实现了，只需要输入data字段调用这个接口就可以得到解密之后的json字符串，然后还需要通过jackson库解密这个json字符串，放到一个map中，再取出需要存储到数据库的内容序列化成新的字符串输出。所以这个java程序同样从标准输入中读取每一行，然后按照‘&’字符进行分割，对后一个data字段处理，得到结果使用key&out写入到标准输出中，其中key是不变的，out是处理之后得到的json字符串。

第四个任务再次通过shell实现，从标准输出中读取每一行，通过mysql命令更新数据库。

总结

通过这次让我见识到了管道的强大，将任务分割成小人物之后也更加方便测试，也认识到使用纯文本的方式作为输入输出是多么的友好，另外，有时候遇到问题的时候要尽量换个角度思考解决问题的方案，就像本例，要是以前可能就想方设法使用一个python脚本实现全部的功能了，但是换个角度想一下可能就会有更加方便的实现了。最后，要尽可能的实现代码的复用（不要重新发明轮子），就是因为在解密上遇到了障碍才想起来不如直接使用已经写好的java程序，以后问题的时候要首先思考一下有没有类似问题的现有的解决方案，省时省力。