脚本使用指南

这些脚本已经经过调试通过， 在使用过程中可能会出现一些 BUG， 但只要坚持使用， 每次出现 BUG 时候，都进行调试修改， 相信这些脚本都能成为 KDS BJ TEAM 的得力小工具， 会给我们的工作带来很大的方便，里面一些设计思想和实现技巧，相信对大家还是有帮助的，至少我通过阅读服务器上的脚本学到了很多设计思想和实现方法。

这个文档包含以下五个脚本的使用说明：

1. windows 回收站的 shell 脚本实现（trash 命令）
2. daily 日志检测脚本（DailyLogview.sh）
3. 运行 Daily 前的检测脚本 (DailyCheck.sh)
4. kdsd10@obi22, kdsdevl@x22,kdsdevl@obi22,kdsdevl@y71 四系统，拷贝表脚本（XBY.sh）
5. 报告检测小工具 (CheckReport.sh， RulesMake.sh 等)

trash 命令替代 linux rm 命令实现 windows 回收站的功能

一. 功能简介

1. 将删除的文件放在回收站中
2. 恢复删除的文件
3. 实现 linux rm 命令的功能, 使用起来几乎和 linux 系统自带的 rm ,命令完全一样
4. 新增功能: rm -l, rm -e, rm -c
5. 该脚本每次在运行时候会检查$HOME/.trash 目录下文件大小之和, 若
   超过最大容量, 脚本会自动将日志文件中所记录文件中的前一半文件从回
   收站中清除,所以建议删除大文件(相对于回收站最大容量而言)直接用
   命令/bin/rm 而不要用 rm.

二. 使用方法:

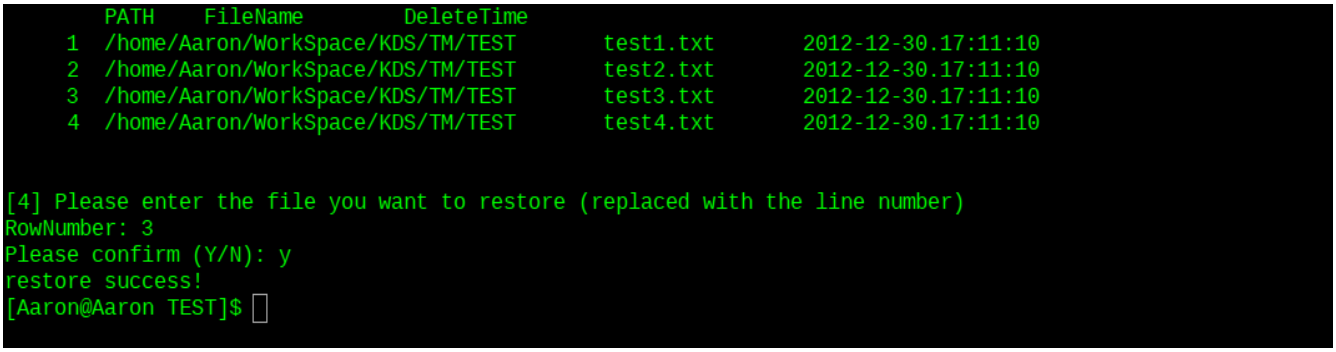
1. 将 trash 文件放到 $HOME/bin/
2. 在$HOME/.bashrc 文件中加入 alias rm=”$HOME/bin/trash”, 重新登陆终端或执行 bash 命令。
3. 执行命令 rm -e 配置回收站的最大容量,单位 K
4. 回收站的默认目录为:$HOME/.trash, 默认配置文件为:$HOME/.trash/trash.conf
   默认 log 文件为:$HOME/.trash/trash.log
5. 怎样恢复文件:
   在 linux 终端中输入 rm -l， 然后 在 RowNumber: 后面键入要删除文件所在的行标识：988 键入 y/Y 然后按回车键 恢复成功.
   如果想只查看删除列表, 则键入 rm -l 后直接按回车键或者键入 Q/q
6. 更详细的参数介绍请键入：rm --help

三. 注意事项

1. 想要手动清空$HOME/.trash 目录需要用/bin/rm 命令, 请不要尝试用 rm -r $HOME/.trash 的方法.
2. 该脚本不支持 rm -r -f , rm -rfi (选项组合超过 2 个)格式.

3. 如果你可以你甚至可以用该脚本作为备份脚本， 假若想备份 test2.txt 你只需要执行 rm test2.txt， 当然如果真想备份某个文件的话， 最好编写专门的备份脚本。

使用截图：



日志检测脚本 DailyLogview.sh

一. 功能简介

该脚本用来监视日志文件中的出错信息, 使用该脚本需要有基本的 awk 编程能力, 不要担心,很基本的就可以.

1. 该脚本拥有先进的算法, 尤其在高频率监视大 log 文件时有明显的效果。
2. 脚本具有及时向你所指定的 tty 发送出错信息以及 email 出错信息的功能,出错信息记录在 AutoDaily.conf 文件中的 errorfile 变量所定义的文件中，你也可以自己指定出错信息的存放文件。

二. 使用指南

1. 因为该脚本是专门为 daily 写的因此， 该脚本需要放在 kdsdevl@x22 上， 放在其它地方需在命令行要指定日志文件。
2. 启动 daily 脚本后直接执行/home/kdsdevl/alex/AutoDaily/DailyLogview.sh gn errorfile pts/5 即可， 若出现错误， 该脚本会在当前 pts/5 tty 中打印出错信息,将错误信息记录在当前目录下的 errorfile 文件中， 发送错误信息到 emaillist 列表中 。

   各位置参数说明：

   gn: 所要检测的 agency（gn/fh/fn）

   errorfile: 所要监测的日志文件

   pts/5: 设置要将错误信息发送的终端名字， 可以指定多个终端，eg: pts/5,pts/6,pts/7
3. 若执行/home/kdsdevl/alex/AutoDaily/DailyLogview.sh gn 命令， 则所检测的日志文件为：/UBSys/staging/siac_daily/logs/daily_dnif.sh_$(date '+%Y%m%d')_gn_x22

   错误信息记录文件为：当前目录下的 log.error 文件

   错误信息发送到当前 tty

   配置说明：

   #设置 DailyLogview.sh 所在的主目录， 其实这 个目录是以后 daily 自动话脚本所存放的主目录。

```
AutoHome=/home/kdsdevl/alex/AutoDaily

#设置需要监测的日志文件， monfile 变量为了尽量简便
monfile="/UBSys/staging/siac_daily/logs/daily_dnif.sh_$(date '+%Y%m%d')_${AGCLASS}_x22"

#监测出来的错误信息的记录文件,log.error 会自动生成在你启动监测脚本的目录下
errorfile=log.error

#errorfile 所记录的最大行数， 若好过该行，则 errirfile 文件会自动减半
maxrecord=5000

#设置监测的频率，单位：s
DELAY=3

#设置是否在终端上打印出错信息， on 打印 off 不打印
tell=on
#设置是否发从邮件， yes 发送， no 不发送
mail=yes

#设置 email list
maillist=akong@kdsglobal.com

#该变量 DailyLogview.sh 脚本中没有用到
timeout=30
```

三. 其他说明：

其实该脚本的功能不止这些， 大家完全可以修改该脚本，因为该脚本在设置时尽量考虑了可重用性，所以只需要进行简单的修改，就可以让该脚本监测其它类型的日志， 大家可能要问，我要设置的错误信息种类在哪里指定呢， 很简单大家可以在脚本中修改 awkstr 变量的值， 错误信息的类型完全由 awkstr 所制定的 awk 脚本所设定， 之前有考虑过用 grep 来提取错误信息，但这样不太灵活，就舍弃了这种想法。

该脚本很适合高频率的监测大型日志文件系统， 个人感觉该脚本再经过加工完全可以作为一个不错的日志监测脚本放到网上供大家使用。

假若刚启动监测脚本时日志文件总共有 10000 行，设置监测频率为 3s， 那么假设在这 3s 内日志文件新增 200 行， 则该脚本查找错误字符串的范围为 10000-10200 而不是 0-10200, 试想假若所要监测的日志文件是个大型的日志文件， 超过 10W 行，而且要不停的监测，那么该脚本会节省很多资源和时间， 而且脚本中的一些实现方法也值得借鉴和学习， 脚本用的是动态变量(通过 shell 宏)来实现此功能的.

该脚本可以很方便的实现对其他日志文件的监控， 要想实现这一点，你只需要：进脚本中设置 awkstr 为你所用的 awk 脚本, 该脚本我更希望能成为一个小工具而不仅仅是脚本。

运行 Daily 前的检测脚本(DailyCheck.sh)

1. 功能简介:

a. 检测 ninja 服务器是否能 ping 通, (ninja5, ninja13, ninja15, ninja16, ninja18, ninja19, ninja23)

b. 检测每个剩余硬盘空间, 当硬盘空间小于 2G 时会提示出错信息

c. 检测 x22/obi22 是否能和 xnodes/obinodes ping 通

d. 检测 daily 是否已经启动, 若启动则会提示哪个 agency daily running

e. 检测数据合法性

检测数据是否到来, 若到来则检测数据大小和日期,若数据时间戳不符合正常的情况则会报错,若数据大小小于 500K 则会报错, (经观察发现数据都是大于 500K 的, 若是出错,自己可以认为进去 check, 再决定数据是否合法)

f. run 简单 cohort 测试是否能正常 run 通, 若 run 不通则会报错, 可能是因为 node down. Or others

若有其他需要检测的项目可以在脚本中进行追加.

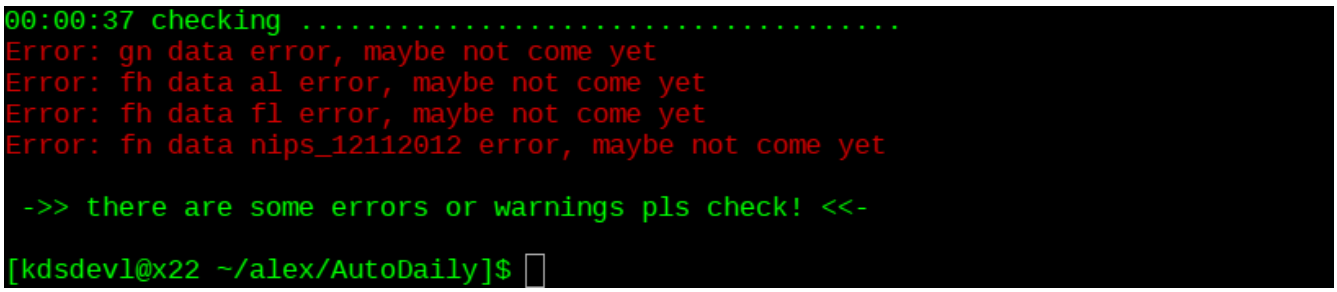
二. 使用方法

该脚本可以在 POX 和 POD 上运行, 2 个系统目录结构是相同的,否则需要修改脚本

三. 其他说明

该脚本设计了一个打印检测进度条及秒表的函数（里面的实现方法可供大家借鉴）, 大家可以用该函数实现其他脚本的"进度条及秒表"

四. 运行截图



XBY.sh 拷贝表脚本(如果愿意脚本可以实现很多不错的功能,至少所有的备份表的工作都可以以该脚本为框架来实现)

一. 功能简介

1. 该脚本可以在 kdsd10@obi22, kdsdevl@y71, kdsdevl@obi22, kdsdevl@x22, 4 个系统之间进行表的拷贝(相互的)
2. 该脚本完全根据.nodes 和 kernelite.ini 来编写， 在第一次运行时会自动建立一个小型的备份系统， 系统所在的文件夹是固定的:$HOME/.NodeBack, 备份系统所生成的脚本及日志文件全放在此目录下
3. 该脚本实现了表拷贝和拷贝进度查询的 2 大基本功能
4. 因为该脚本编写时尽量考虑了可移植行及通用性， 所以只要按正常流程建立 新的 UB 系统

like kdsdevl@y71 该脚本就可以直接使用， 不需要进行任何更改.

5. 该脚本可以在任何地方执行命令， 而且在一个新的 UB 环境中， 只需要往新的 UBHOST 中 copy XBY.sh 脚本便可以直接备份，很是方便， tblo 所支持的正则表达式， 都可以写入 FILE 文件中
6. 该脚本可以记录一切 rsync 信息，方便今后检查表的同步情况。
7. 脚本可以自动判断是本地同步还是远程同步。

二. 使用方法（假若从 kdsdevl@obi22 往 kdsd10@obi22 上拷贝数据）

1. ssh kdsd10@obi22; cp XBY.sh $HOME/bin/
2. XBY.sh backup kdsdevl@obi22 FILE 1 1 36 layout

各参数说明

backup: 调用 XBY.sh 中的 backup 函数 进行数据拷贝

kdsdevl@obi22: 从 kdsdevl@obi22 上往当前的 SysGovernor 上拷贝数据

（也就是说， 假若你在 kdsd10@obi22 上执行该命令， 那么你就是往 kdsd10@obi22 上拷贝数据）

FILE： 所需要拷贝表的列表

1: 从第几个 node 上

1: 跨度为几个 node

36: 到第几个 node 结束

layout： 要拷贝的类型为 layout, 若该处值为 layout/db.ini 则 FILE 应该为空文件 .若是拷贝表 该参数可以不写

若要检测表的拷贝进度请执行：

XBY.sh check 1 1 36

check： 调用 XBY.sh 中的 check 函数打印备份进度

1 : 开始 node id

1: 跨度

36: 终止 node id

检查同步进度：

XBY.sh check 1 1 36

输出信息如下：

1 20121230 01:35:30 1/1 /UBSys/NodesPOD/1/Disks/0/hets/une_idx

各列代表的意思为：

1: node id

20121230 01:35:30： 同步发生的时间

1/1: 备份进度， 第几个表/表总数

/UBSys/NodesPOD/1/Disks/0/hets/une_idx ： 所备份表所在的文件夹

三. 注意事项

1. 设计上该脚本可以在四个 系统之间任意拷贝数据， 但为了安全期间，该脚本杜绝往 kdsdevl@x22 上拷贝数据， 所以若在 kdsdevl@x22 上执行该脚本， 脚本会报错并退出.
2. 该脚本的使用在通过正常方式创建的 UB 系统， 比如 kernelite.ini 中若指定 DBDir = /UBSys/NodesPPOD/1 而非 DBDir = /UBSys/NodesPOD/1 那么这种方式就不属于通过正常方式创建的 UB 系统.
3. 因为目前从 kdsdevl@y71 上 ping 不通有些 lnodes, 所有其他系统往 kdsdevl@y71 上拷贝数据还没有测试。 若想往 kdsdevl@y71 上拷贝数据首先应该确保 node ID 相同的 lnode 和其他 node 之间能够互相通过 ssh 无密码登陆对方 host
4. FILELIST 可以是表名的列表例如： fnll_fix_gen_pdb_201301 也可以是 tblo fnll_fix_gen_pdb_201301 > FILEST, 这种格式的列表

运行截图

```
12031 checking Env, pls wait ....
12031 mkdir /home/kdsd10/.NodeBack & /home/kdsd10/.NodeBack/log ....
12031 copy /home/kdsd10/bin/XBY.sh to /home/kdsd10/.NodeBack/ ...
12088 generate /home/kdsd10/.NodeBack/kdsd10@obi22.list .....done
14130 generate /home/kdsd10/.NodeBack/kdsdevl@x22.list .....done
12031 generate SingleNode.sh ....
12031 mkdir all nodes ~/.NodeBack & ~/.NodeBack/log, pls wait ....
12031 generate FILELIST.txt,pls wait ....
12031 check FILELIST.txt format ...
12031 upload SingleNode.sh to kdsd10@obi23:$HOME/.NodeBack/ ...
12031 upload FILELIST.txt to kdsd10@obi23:$HOME/.NodeBack/ ...
12031 file list and script upload done
12031 send rsync request according to your requirement
[kdsd10@obi22 ~]$ XBY.sh check 1 1 1
16393 copy /home/kdsd10/bin/XBY.sh to /home/kdsd10/.NodeBack/ ...
1 20121230 01:35:30 1/1 /UBSys/NodesPOD/1/Disks/0/hets/une_idx
[kdsd10@obi22 ~]$
```

报告检测小工具（该脚本的灵活性非常大）

一. 功能简介

1. 该脚本可以检测报告， 检测报告的格式在可遇见的未来都可以检测。
2. 报告检测基本分为 2 大类， a. 自身检测 b. 和其他脚本的对比检测。
3. 该脚本在检测前你需要进行配置， 任何报告的自身检测，以及任何报告和任何报告的对比检测，以及检测的内容和筛选条件，都可以在此进行配置，在配置文本中所指定的筛选条件是按 awk 语法来填写的，比如你想要 A 报告中的第 3 个 field 和 B 报告中的第四个 field 想等， 那么 你需要这么写：

   $3==$4 而不是$3=$4
4. 该脚本所在的.ReportQA 目录，必须放在$HOME 下， 因为脚本中的变量已经写死了， 而且脚本中用到的路径也是从$HOME/.ReportQA 目录开始的相对路径。
5. 在检测脚本时， 你需要进入$HOME/.ReportQA 目录下进行操作。
6. 里面的设计思想也是借鉴了 UBEOD 系统中的~/bin/scripts/gen-kini.sh 脚本，实现上包含了 shell 脚本中常用的知识和技巧。

二. 使用方法(假设要对比 WFPrepaymentArm/fharm.txt 和 WFServicerArm/fh.txt 2 个报告。自身和 2 者

之间的对比)

使用时基本分为 2 步操作：

A. 生成筛选规则

B. 进行对比

1. cd $HOME; tar -xvzf ~/ReportQA.tar.gz; cd .ReportQA;cp -r ~/WFPrepaymentArm .; cp -r ~/WFServicerArm .

   下面先进行 WFPrepaymentArm/fharm.txt 的自身检测

2. 创建筛选条件文本：test

   内容如下：

```
$3>20000
$4>-100 && $4 < 100
$5>-100 && $5 < 100
$6 != $9
$10 > 0
$2 == "All" && $3 > 20000
~
```

BTW： 里面的筛选条件可以随便指定 (没有指定的 field 该脚本不做检测), 所指定的条件是需要满足的条件，若满足该条件， 则脚本会输出出现错误的行 以及对应的筛选条件。

3. 执行命令：

   ./RulesMake.sh WFPrepaymentArm/fharm.txt test > TEST

   TEST 中内容为：

```
<WFPrepaymentArm-fharm.txt>
$3>20000
$4>-100 && $4 < 100
$5>-100 && $5 < 100
$6 != $9
$10 > 0
$2 == "All" && $3 > 20000
</WFPrepaymentArm-fharm.txt>
```

若想顺便生成 WFServicerArm/fh.txt 报告的检测脚本， 只需执行命令：
./RulesMake.sh WFServicerArm/fh.txt test2 >> TEST

此时 TEST 内容为：

```
<WFPrepaymentArm-fharm.txt>
$3>20000
$4>-100 && $4 < 100
$5>-100 && $5 < 100
$6 != $9
$10 > 0
$2 == "All" && $3 > 20000
</WFPrepaymentArm-fharm.txt>

<WFServicerArm-fh.txt>
$3>50000
$4>-100 && $4 < 100
$5>-100 && $5 < 100
$6 != $9
$3 > $4 + 1000
</WFServicerArm-fh.txt>
```

4. 检测规则已经生成， 接下来便可以进行检测了， 检测时候的命令为
5. ./CheckReport.sh WFPrepaymentArm TEST
   WFPrepaymentArm 文件夹下有很多文件 假若有 fharm.txt g2arm.txt fnarm.txt， 该脚本会对所有文件一一进行扫描， 若在 TEST 中搜到和其对应的检测规则，则进行检测， 否则只报警。所以此时只会对 WFPrepaymentArm/fharm.txt 进行检测。

   对 WFPrepaymentArm/fharm.txt 的检测结果为：

```
FH 10/1 IO 10   2007    1311652147      5       1.9     43.7    41.8    42.2    45.0    35.5    46.0    48.4    41.8    41.2    37.7 2
7.9     107     100     98      5.8     6.34    66      245815  93      730     98      6 ##### WFPrepaymentArm/fharm.txt ##### $2 ==
All && $3 > 20000
FH 10/1 IO 10   2006    1107342569      -19     -9.5    41.1    50.6    47.2    47.6    44.9    49.6    56.4    45.7    46.1    42.5 2
7.1     100     109     110     5.9     6.36    77      241716  89      735     99      2 ##### WFPrepaymentArm/fharm.txt ##### $3>200
00
FH 10/1 IO 10   2006    1107342569      -19     -9.5    41.1    50.6    47.2    47.6    44.9    49.6    56.4    45.7    46.1    42.5 2
7.1     100     109     110     5.9     6.36    77      241716  89      735     99      2 ##### WFPrepaymentArm/fharm.txt ##### $2 ==
All && $3 > 20000
FH 10/1 IO 10   2005    584532499       -7      -2.7    37.1    39.8    38.7    43.8    41.0    40.0    38.5    37.9    39.4    35.6 1
9.5     91      90      94      5.3     5.74    86      238306  79      736     98      3 ##### WFPrepaymentArm/fharm.txt ##### $2 ==
All && $3 > 20000
FH 10/1 IO 10   2004    5415128 -13     -9.0    59.4    68.4    0.1     0.0     38.9    38.5    0.2     49.6    39.7    38.2    7.7  1
45      118     94      5.2     5.57    96      233791  71      763             0 ##### WFPrepaymentArm/fharm.txt ##### $2 == All && $
3 > 20000
Warning: WFPrepaymentArm/g2arm.txt have not configure rules conf,pls check.
Warning: WFPrepaymentArm/WFPrepaymentArm.pdf have not configure rules conf,pls check.
[Aaron@Aaron .ReportQA]$
```

第一个#####前面内容为符合筛选条件的行 ， 2 个#####中间的内容为出错的文件， 最后一个#####之后的内容为符合的筛选条件

两个报告之间的对比

WFServicerArm/fh.txt 和 WFPrepaymentArm/fharm.txt 之间进行对比

4. 生成检测规则， 建立规则文本 ct ， 执行命令：
   ./RulesMake.sh WFPrepaymentArm/fharm.txt WFServicerArm/fh.txt ct > CT
   文件 ct 内容为：

```
key:f$1=s$1,f$2=s$2,s$3="ALL"
f$2=s$2+10
```

5.

假若 ./RulesMake.sh WFPrepaymentArm/fharm.txt WFServicerArm/fh.txt ct > CT 命令中第 1 个参数所指定的文件的第 1 个 field 和第 2 个参数所指定的文件的第 1 个 field 相等， 第 1 个参数所指定的文件的第 2 个 field 和第 1 个参数所指定的文件的第 2 个 field 相等, 且第 2 个参数所指定文件中的第 3 个 field 值应为 All, 则这 2 行数据可以进行对比， 那么你应该这样来书写：
key:f$1=s$1,f$2=s$2,s$3="All"
（Note： f$1 :代表第 1 个参数所指定文件中的第 1 个 field. s$2: 代表第 2 个参数所指定文件中的第 2 个 field）

f$2==s$2+10： 这行指定符合条件的 2 个数据所应满足的条件， 若满足则按顺序合并这 2 行， 并打印。
如果$1 是里面含有%， 那么你在指定规则时不应该是这样：$1>22% 而应该是$1>22

CT 内容如下：

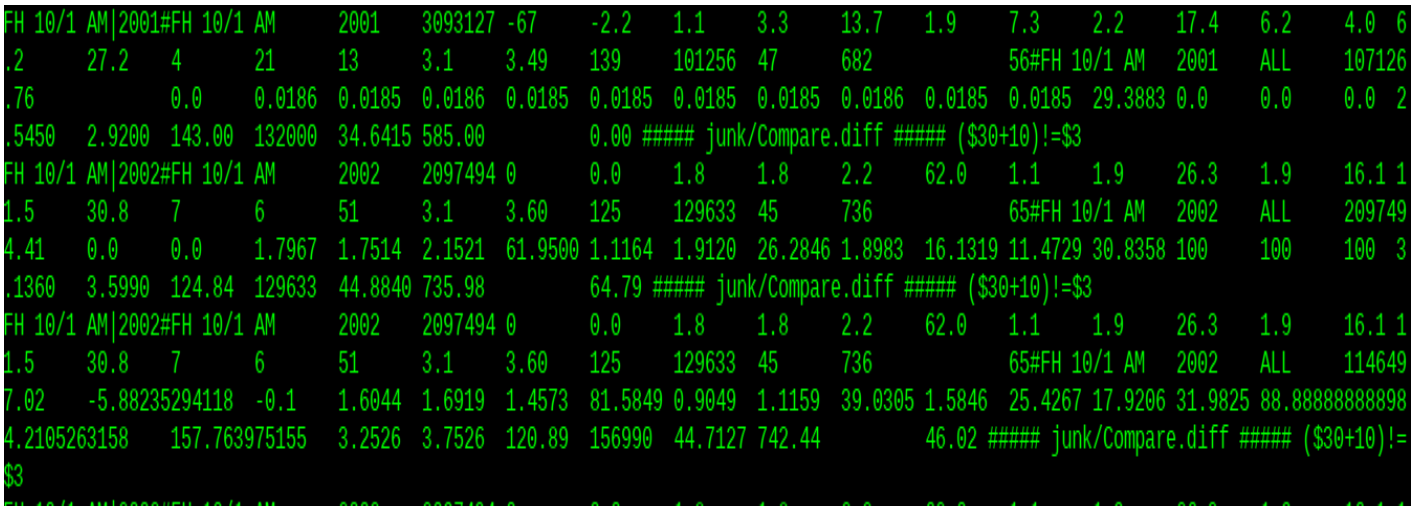
```
<WFPrepaymentArm-fharm.txt-V-WFServicerArm-fh.txt>
key:WFPrepaymentArm-fharm.txt|$1=WFServicerArm-fh.txt|$1,WFPrepaymentArm-fharm.txt|$2=WFServicerArm-fh.txt|$2,WFServicerArm-fh.txt|$3=
"ALL"
WFPrepaymentArm-fharm.txt|$2=WFServicerArm-fh.txt|$2+10

</WFPrepaymentArm-fharm.txt-V-WFServicerArm-fh.txt>
```

同样你也可以继续生成其他对比文件的对比规则， 只需执行命令：
./RulesMake.sh WFPrepaymentArm/fnarm.txt WFServicerArm/fn.txt ct >> CT

5. 执行命令： ./CheckReport.sh WFPrepaymentArm WFServicerArm CT
   进行对比

对比结果为



第一个#####前面内容为符合筛选条件的行 ， 2 个#####中间的内容为出错的文件， 最后一个#####之后的内容为符合的筛选条件