文件a:    
**10/05766798607,11/20050325191329,29/0.1,14/05766798607    
10/05767158557,11/20050325191329,29/0.08,14/05767158557**

文件b:    
**05766798607    
05766798608    
05766798609**通过文件a和文件b对比,输出结果如下：   
**10/05766798607,11/20050325191329,29/0.1,14/05766798607**

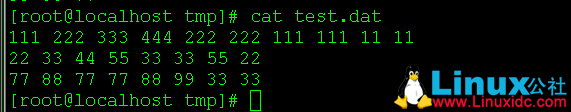
**方法：**a） **awk -F"[/,  ]" '{if(NR==FNR){a[$1]}else if($2 in a)print $0}' b a**

**解析：**NR为主体行号，FNR为单位文件行号。**{if(NR==FNR){a[$1]}** 将b文件中的数据放入数组a，**else if($2 in a)print $0}'** 主要是对b文件内容的操作，判断第二个域中数据是否与a数组中的数据匹配，匹配则打印本行。

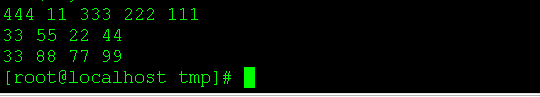
b）精简法：awk -F"[,/ ]" '{a[$1]}{if($2 in a)print $0}' b a

**解析：**-F"[,/ ]" 用，或/ 做分隔符，'{a[$1]}{if($2 in a)print $0}' 用第一个域的值作为数组a的下标，判断第二个域是否与数组a（a数组中的有用数据也就为数组a文件中的，即前三行）中的匹配，因为先读入b文件，且b文件中只有一个域，所以此时判断只与a文件有关，最后打印匹配行信息。

有文件，包含的内容如下：



去除重复列结果如下**：**



方法：awk '{for(i=1;i<=NF;i++)a[$i,NR]++}{for(j in a){split(j,b,SUBSEP);if(b[2]==NR)printf b[1]" "} printf "\n"}' file

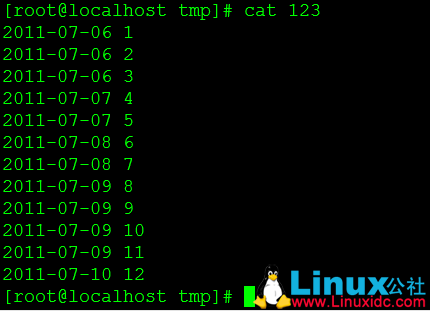
说明：本方法巧妙的利用了awk的数组，a[$i,NR]将读入的数据以域和行号为下标，只要本行的列中有重复数据则以该数据和该行行号为下标的数组元素加1。后面通过for循环取出下标，用split(j,b,SUBSEP)将两个下标分离赋值给新的数组b，此时b[1]中的数据就是我们想要的（不重复的列），最后判断b[2]与行号相等的则打印b[1]。执行命令后结果如下：

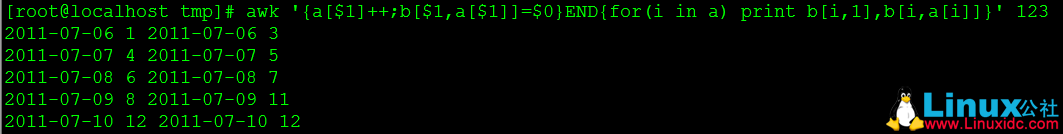
如果只想打印重复列，只需a数组的之是否大于1：awk '{for(i=1;i<=NF;i++)a[$i,NR]++}{for(j in a){split(j,b,SUBSEP);if(b[2]==NR&&a[b[1],NR]>1)printf b[1]" "} printf "\n"}' file

补充：

awk的二维数组使用

awk的多维数组在本质上是一维数组，更确切一点，awk在存储上并不支持多维数组。awk提供了逻辑上模拟二维数组的访问方式。例如，array[2,4] = 1这样的访问是允许的。awk使用一个特殊的字符串SUBSEP (\034)作为分割字段，在上面的例子中，关联数组array存储的键值实际上是2\0344。  
  
类似一维数组的成员测试，多维数组可以使用 if ( (i,j) in array)这样的语法，但是下标必须放置在圆括号中。  
类似一维数组的循环访问，多维数组使用 for ( item in array )这样的语法遍历数组。与一维数组不同的是，多维数组必须使用split()函数来访问单独的下标分量。split ( item, subscr, SUBSEP)。





要求：每个数据保留最初的和最后的，就是日期相同的只打印第一个和最后一个（类似打卡记录筛选）

方法：awk '{a[$1]++;b[$1,a[$1]]=$0}END{for(i in a) print b[i,1],b[i,a[i]]}' 123

解析：第一个数组a以$1下标，并累加。第二个数组b为二位数组，以$1，a[$1]为下标，并把$0赋值给b。因为a数组累加，所以相同下标的出现多少 次，其值就为多少。b数组中相同第一个下标的最后一个的第二个下标就是a[第一个]的值，所以最打印的b[i,1],b[i,a[i]] 就是想要的结果。