猿课

例题一：

【题目要求】

请按照这样的日期格式（xxxx-xx-xx）每日生成一个文件，例如今天生成的文件为 2017-12-20.log， 并且把磁盘的使用情况写到到这个文件中，（不用考虑 cron，仅仅写脚本即可）

【习题分析】

本题有两个核心知识点：

1. 如何自动表示当天的日期

2. 磁盘使用情况

打印日期的命令为 date，示例命令如下：

```
# date
2017 年  12 月  20 日  星期三  16:26:55 CST
```

而题目中要求的格式为应该是：2017-12-20，date 命令是有这样的功能的，示例命令如下：

```
# date +%Y-%m-%d
2017-12-20
```

或者：

```
# date +%F
2017-12-20
```

磁盘使用情况，我们用命令 df -h 实现，示例命令如下：

```
# df -h
文件系统          容量   已用   可用  已用%  挂载点
/dev/vda1          99G   1.8G    92G      2% /
devtmpfs          911M       0   911M      0% /dev
```

```
tmpfs           920M     0   920M     0% /dev/shm
tmpfs           920M   336K   920M     1% /run
tmpfs           920M     0   920M     0% /sys/fs/cgroup
tmpfs           184M     0   184M     0% /run/user/0
```

【习题答案】

有了上面的分析之后，我们最终得到本题答案：

```
#! /bin/bash
d=`date +%F`
logfile=$d.log
df -h > $logfile
```

【答案解析】

把当天日期赋值给变量 d，从而定义每日的日志文件名，最终把磁盘使用情况的结果直接输入到该日志里。这里的>，比较特殊它可以把该符号左边的结果写入到该符号右边的文件里。

扩展知识点：

1. shell 中反引号可以表示一个命令的结果，通常给变量赋值，示例命令如下：

```
# n=`wc -l /etc/passwd|awk '{print $1}'`
# echo $n
23
```

2. date 命令还有诸多用法，示例如下：

```
# date +%H ##小时
16
```

date +%M ##分钟

38

date +%S ##秒

55

date +%T ##时间

16:39:31

date +%w ##星期

3

date -d "-1 day" +%F ##一天以前

2017-12-19

3. >为正确重定向，我们运行一条命令时，有正确的输出信息也有错误的输出信息，>会把正确的输出信息写入到指定文件里，与其对应的还有一个错误重定向符号 2>，顾名思义它会把错误信息写入到指定文件里。示例如下：

ls /etc/passwd /etc/nofile ##其中/etc/nofile 是不存在的，所以会报错

ls: 无法访问/etc/nofile: 没有那个文件或目录

/etc/passwd

ls /etc/passwd /etc/nofile > /tmp/log 2>/tmp/error

cat /tmp/log

/etc/passwd

cat /tmp/error

ls: 无法访问/etc/nofile: 没有那个文件或目录

例题二：

【题目要求】

有日志 1.log，部分内容如下：

112.111.12.248 – [25/Sep/2013:16:08:31 +0800]formula-x.haotui.com

"/seccode.php?update=0.5593110133088248" 200"http://formula-

x.haotui.com/registerbbs.php" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;

SV1;)"

61.147.76.51 – [25/Sep/2013:16:08:31 +0800]xyzdiy.5d6d.com

"/attachment.php?aid=4554&k=9ce51e2c376bc861603c7689d97c04a1&t=1334564048&f

id=9&sid=zgohwYoLZq2qPW233ZIRsJiUeu22XqE8f49jY9mouRSoE71"

301"http://xyzdiy.5d6d.com/thread-1435-1-23.html" "Mozilla/4.0 (compatible; MSIE 6.0;

Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)"

统计出每个 IP 的访问量有多少

【习题分析】

这种分析日志的需求，在平时工作中很常见，而且找运维工作时的笔试题里面出现频率也非常多。

根据日志内容，可以看到 IP 地址就是第一段内容，所以只需要把 1.log 的第一段给过滤出来，然后近一步统计每一个 IP 的量即可。

过滤第一段，使用 awk 就可以很容易得到，而统计每个 IP 的访问量则需要排序然后再计算数量，排序使用 sort 命令，统计每个 IP 访问量用 uniq。

【习题答案】

awk '{print $1}' 1.log |sort -n |uniq -c |sort -n

【答案解析】

1. awk 命令在分段方面还是比较有优势的，这里的{print $1}讲第一段打印出来，awk 可以用-F 指定分隔符，如果不指定分隔符，默认就以空白字符（比如空格、Tab 等），本题中，IP 地址就是在第一段。

2. sort 命令是排序的命令，-n 选项表示以数字的形式排序，如果不加-n，则以 ASCII 排序，本题中的 IP 地址以数字的形式排序更容易区分。

3. uniq 命令是用来去重复的，一个文本中如果有多行内容是一模一样的，使用 uniq 命令就可以把相同内容的行给删除掉，只留一行。而-c 选项的作用是计算重复的行数，所以在此题中使用 uniq -c 正好可以计算 IP 地址的访问数量。不过，大家一定要注意，uniq 去重的前提是首先要排序。

4. 本题答案里最后没得 sort -n 意思是按访问量大小来排序，请求量越大的 IP 排在越后面，如果要想排在前面，可以加一个-r 选项，即 sort –nr

猿课

例题三：

【题目要求】

写一个脚本计算一下 linux 系统所有进程占用内存大小的和。

【习题分析】

本题有一个核心点，就是想办法把所有进程使用的内存统计出来，top 或者 ps 命令都可以获取每一个进程的内存使用大小。统计完内存后，然后用 for 循环把所有内存相加，最终得到一个内存之和就是本题的答案。

【习题答案】

```
#! /bin/bash
sum=0
for mem in `ps aux |awk '{print $6}' |grep -v 'RSS'`
do
    sum=$[$sum+$mem]
done
echo "The total memory is $sum."
```

【答案解析】

1. ps aux 命令可以打印出所有进程的信息，其中第六列，也就是 RSS 那列就是内存使用大小，其中第一行 RSS 我们不需要，所以用 grep -v 去掉。

2. 我们拿到内存大小后，接下来就是做一个加法运算。这里用到了 for 循环，有多少个进程就循环多少次，每次循环，变量 mem 被赋予新的值（对应那个进程内存大小）。sum 初始值设置为 0，

每循环一次，sum 的值都要加上本次循环对应的 mem 的值，这样循环结束后 sum 的值就是所有内存的总大小了。

【小技巧】

我们在写脚本时，是需要反复在命令行下面运行一些命令的，比如在本题中，我们首先想到要使用 ps aux 命令把所有进程信息打印出来，但是内存在哪一列我们无概念，只有见到实际的输出内容后，才判定第六列为内存列，所以就可以使用 awk '{print $6}'把第六列打印出来，但是此时第一行的 RSS 是多余的，固又使用了 grep -v 把 RSS 去掉。

总之，写任何一个脚本，都不是一气呵成，是需要不断地运行和调试，方得始终。

例题四：

【题目要求】

设计一个脚本，监控远程的一台机器(假设 ip 为 180.163.26.39)的存活状态，当发现宕机时发一封邮件给你自己。

提示：

1. 你可以使用 ping 命令　ping -c10 180.163.26.39

2. 发邮件脚本可以参考 https://coding.net/u/aminglinux/p/aminglinux-book/git/blob/master/D22Z/mail.py

3. 脚本可以搞成死循环，每隔 30s 检测一次

【习题分析】

题目中其实已经给出简单的思路，对于这种监控类的脚本，我认为套路就一个：首先设定一个标准阈值，然后通过一些手段获取到要监控目标的属性值。再拿这个属性值和标准阈值进行比较，如果不正常就要做出相应的动作，或发邮件或执行某个命令。

本题是要监控一个 IP 的存活状态，而题目中给出的建议是用 ping 命令，所以不妨先运行一下这个命令，看看输出结果是什么？如下：

```
# ping -c10 180.163.26.39
PING 180.163.26.39 (180.163.26.39) 56(84) bytes of data.
64 bytes from 180.163.26.39: icmp_seq=1 ttl=51 time=10.2 ms
64 bytes from 180.163.26.39: icmp_seq=2 ttl=51 time=12.2 ms
64 bytes from 180.163.26.39: icmp_seq=3 ttl=51 time=10.4 ms
64 bytes from 180.163.26.39: icmp_seq=4 ttl=51 time=12.4 ms
64 bytes from 180.163.26.39: icmp_seq=5 ttl=51 time=10.1 ms
```

64 bytes from 180.163.26.39: icmp_seq=6 ttl=51 time=13.4 ms

64 bytes from 180.163.26.39: icmp_seq=7 ttl=51 time=10.2 ms

64 bytes from 180.163.26.39: icmp_seq=8 ttl=51 time=12.0 ms

64 bytes from 180.163.26.39: icmp_seq=9 ttl=51 time=10.5 ms

64 bytes from 180.163.26.39: icmp_seq=10 ttl=51 time=12.3 ms

--- 180.163.26.39 ping statistics ---

10 packets transmitted, 10 received, 0% packet loss, time 9017ms

rtt min/avg/max/mdev = 10.138/11.411/13.469/1.154 ms

看到上面的结果后，你应该把重点放到最后面两行，其中"0% packet loss"是一个总结性的输出，字面意思是说丢包率为 0%。所以，我们就可以拿这个丢包率来说事。如果不存活了，那这个丢包率就是 100%，但实际网络环境中只要丢包率超过 20%，就会有很大的问题。由此，可以把标准阈值设置为 20。

剩下的事情，就是如何把丢包率的那个数字给获取到。至于后面那个发邮件脚本，不是本题所关心的内容，大家可以拿来就用。

【习题答案】

```
#!/bin/bash
ip=180.163.26.39
ma=abc@139.com
while 1
do
    n=`ping -c10 $ip 2>/dev/null |grep 'received'|awk -F 'received, |%' '{print $2}'`
    if [ -z "$n" ]
```

```
    then
        echo "There is sth wrong in the script."
        exit
    fi
    if [ $n -ge 20 ]
    then
        python /usr/local/sbin/mail.py $ma  “$ip down”   “$ip is down”
        #假设 mail.py 已经编写并设置好了
    fi
sleep 30
done
```

【答案解析】

1. 本题中如何截取到那个丢包率的数字是关键所在。

2. 在 ping 命令后面加一个 2>/dev/null 目的是为了把错误信息输出到/dev/null，在 linux 里，这个文件就是一个黑洞设备，无论写多少东西进去都是无底洞。总之这样做在脚本运行时不会输出乱七八糟的错误信息。

3. 如果你理解不了这条命令 ping -c10 $ip 2>/dev/null |grep 'received'|awk -F 'received, |%' '{print $2}'，可以从做到右依次执行每一个管道符左侧的命令，这里有一个小技巧，awk -F 后面指定的分隔符是一个复杂用法，如果你看不懂这个，那咱们写个简单的，如下:

awk -F ':|#|.' 它的意思是，分隔符可以是":"，也可以是"#"，也可以是"."，多个分隔符用竖线"|"来划分。

回到本例，分隔符是"received, "（注意，最后面有个空格），也可以是"%"，所以字符串中无论哪个分隔符出现，都是同等效果的，而我们要的数字就在第二段。

4. 在本例中，把丢包率的数字复制给变量 n，但变量 n 是否被成功赋值，是需要做一个检测的，如果不能成功赋值，那后续的一系列操作都会有问题。[-z "$n"]可以判断变量 n 是否为空。希望大家后续写 shell 脚本时也要考虑到某个关键变量是否被成功赋值的情况。

5. 本例中的 mail.py 内容以及用法，需要大家自行扩展学习。

猿课

例题五：

【题目要求】

找到/123 目录下所有后缀名为.txt 的文件

1. 批量修改.txt 为.txt.bak
2. 把所有.bak 文件打包压缩为 123.tar.gz
3. 批量还原文件的名字，即把增加的.bak 再删除

【习题分析】

1. 查找.txt 的文件，使用 find 命令可以轻松搞定
2. 我们之前学 find 时，也曾多次使用过 xargs 命令，用它很容易批量更改文件名
3. 打包核心命令 tar czvf，关键的在于如何表示所有.bak 文件，如何同时把全部.bak 文件打包压缩
4. 还原文件名似乎有点复杂，需要借助 for 循环，需要把文件名做一个特殊处理，如何获取原文件名是关键点

【习题答案】

```
#!/bin/bash
find /123/ -type f -name "*.txt" > /tmp/txt.list
for f in `cat /tmp/txt.list`
do
    mv $f $f.bak
done
d=`date +%y%m%d%H%M%S`
mkdir /tmp/123_$d
for f in `cat /tmp/txt.list`
```

```
do
    cp $f.bak /tmp/123_$d
done
cd /tmp/
tar czf 123.tar.gz 123_$d/
for f in `cat /tmp/txt.list`
do
    mv $f.bak $f
done
```

【答案解析】

1. 用 find 查找所有的.txt 文件，并把这些文件列表写到/tmp/txt.list 文件中，方便后续调用

2. 把这些.txt 文件做一个遍历，依次修改文件名，如果不这样做，其实一条命令也可以搞定，命令如下：

```
# find /123/ -type f -name "*.txt"|xargs -i mv {} {}.bak
```

3. 为了方便打包压缩所有的.bak 文件，需要把所有的.bak 文件拷贝到另外一个目录下面，为了避免该目录已经存在，所以给目录名加上了一个时间后缀，这里用到了 date 命令，该命令在 shell 脚本中频繁用到，需要大家掌握 date 命令的用法。

4. 本例中把所有.bak 文件拷贝到目录后，再对这个目录打包压缩，这样虽然多了一层目录，但是从实现上就简单了很多。

5. 由于之前已经把所有.txt 文件列表存到了一个文件里，所以再次对这些文件进行第二次改名就容易了许多。如果没有该文件列表，则需要再使用 find 把/123/目录下面的.txt.bak 文件都找出来，然后遍历。