Nested for loops iterate through the IP address and users extracting the lines with each IP address and user combination. If the number of attempts for this IP/User combination is > 0, the time of the first occurrence is extracted with `grep`, head, and cut. If the number of attempts is > 1, then the last time is extracted using tail instead of head.

This login attempt is then reported with the formatted `printf` command.

Finally, the temporary file is removed.

Monitoring remote disk usage health

Disks fill up and sometimes wear out. Even RAIDed storage systems can fail if you don't replace a faulty drive before the others fail. Monitoring the health of the storage systems is part of an administrator's job.

The job gets easier when an automated script checks the devices on the network and generates a one-line report, the date, IP address of the machine, device, capacity of device, used space, free space, percentage usage, and alert status. If the disk usage is under 80 percent, the drive status is reported as `SAFE`. If the drive is getting full and needs attention, the status is reported as `ALERT`.

Getting ready

The script uses SSH to log in to remote systems, collect disk usage statistics, and write them to a log file in the central machine. This script can be scheduled to run at a particular time.

The script requires a common user account on the remote machines so the `disklog` script can log in to collect data. We should configure auto-login with SSH for the common user (the *Password-less auto-login with SSH* recipe of `Chapter 8`, *The Old-Boy Network*, explains auto-login).

How to do it...

Here's the code:

```
#!/bin/bash
#Filename: disklog.sh
#Description: Monitor disk usage health for remote systems
```

```
logfile="diskusage.log"

if [[ -n $1 ]]
then
  logfile=$1
fi

   # Use the environment variable or modify this to a hardcoded value
user=$USER

#provide the list of remote machine IP addresses
IP_LIST="127.0.0.1 0.0.0.0"
# Or collect them at runtime with nmap
# IP_LIST=`nmap -sn 192.168.1.2-255 | grep scan | grep cut -c22-`

if [ ! -e $logfile ]
then
  printf "%-8s %-14s %-9s %-8s %-6s %-6s %-6s %s\n" \
    "Date" "IP address" "Device" "Capacity" "Used" "Free" \
    "Percent" "Status" > $logfile
fi
 (
for ip in $IP_LIST;
do
 ssh $user@$ip 'df -H' | grep ^/dev/ > /tmp/$$.df

 while read line;
 do
 cur_date=$(date +%D)
 printf "%-8s %-14s " $cur_date $ip
 echo $line | \
     awk '{ printf("%-9s %-8s %-6s %-6s %-8s",$1,$2,$3,$4,$5); }'

 pusg=$(echo $line | egrep -o "[0-9]+%")
 pusg=${pusg/\%/};
 if [ $pusg -lt 80 ];
 then
 echo SAFE
 else
 echo ALERT
 fi

 done< /tmp/$$.df
done

) >> $logfile
```

The `cron` utility will schedule the script to run at regular intervals. For example, to run the script every day at 10 a.m., write the following entry in `crontab`:

```
00 10 * * * /home/path/disklog.sh /home/user/diskusg.log
```

Run the `crontab -e` command and add the preceding line.

You can run the script manually as follows:

```
$ ./disklog.sh
```

The output for the previous script resembles this:

```
01/18/17 192.168.1.6    /dev/sda1   106G    53G    49G     52%     SAFE
01/18/17 192.168.1.6    /dev/md1    958G  776G   159G     84%     ALERT
```

How it works...

The `disklog.sh` script accepts the log file path as a command-line argument or uses the default log file. The `-e $logfile` checks whether the file exists or not. If the log file does not exist, it is initialized with a column header. The list of remote machine IP addresses can be hardcoded in `IP_LIST`, delimited with spaces, or the `nmap` command can be used to scan the network for available nodes. If you use the `nmap` call, adjust the IP address range for your network.

A for loop iterates through each of the IP addresses. The `ssh` application sends the `df -H` command to each node to retrieve the disk usage information. The `df` output is stored in a temporary file. A `while` loop reads that file line by line and invokes `awk` to extract the relevant data and output it. An `egrep` command extracts the percent full value and strips `%`. If this value is less than 80, the line is marked `SAFE`, else it's marked `ALERT`. The entire output string must be redirected to the `log` file. Hence, the `for` loop is enclosed in a subshell `()` and the standard output is redirected to the log file.

See also

- The *Scheduling with a cron* recipe in `Chapter 10`, *Administration Calls*, explains the `crontab` command