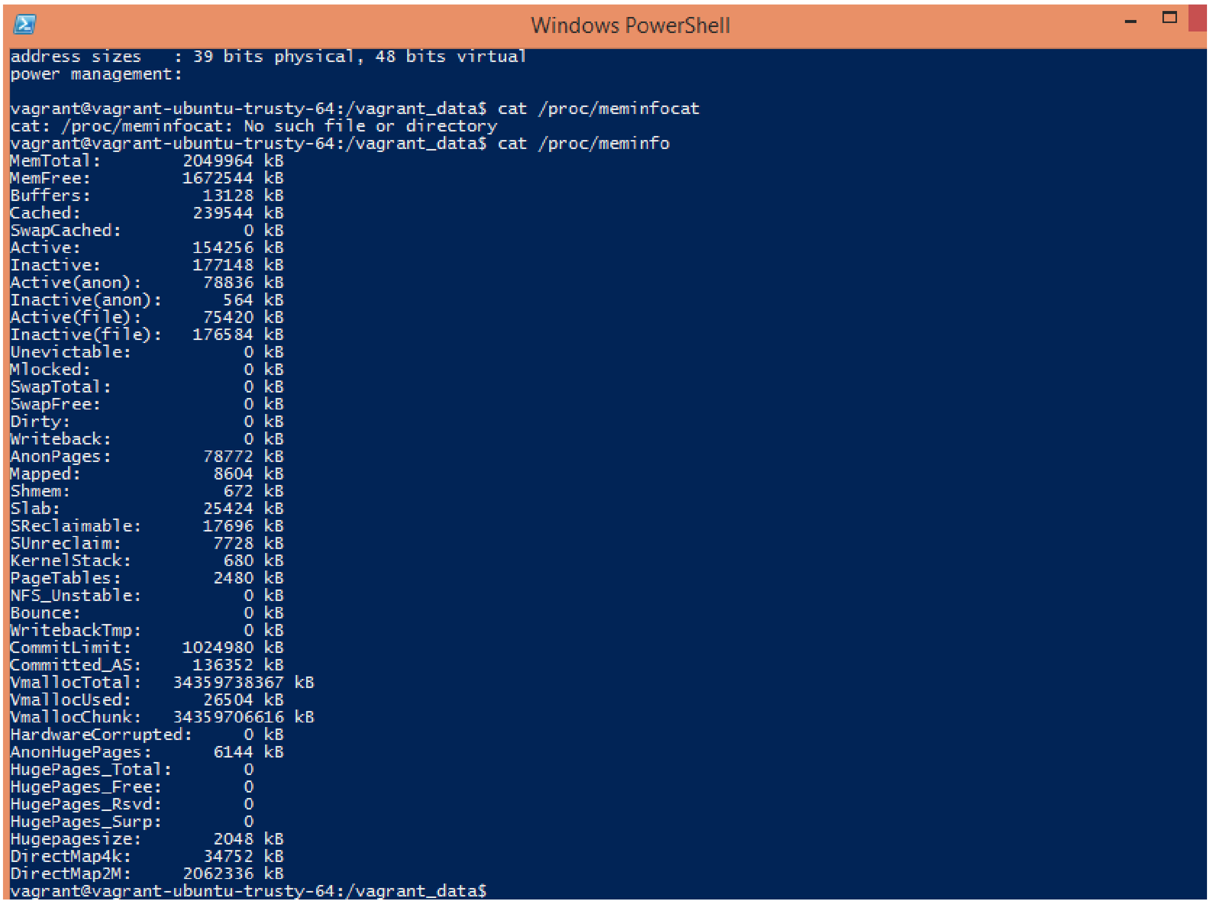
ASSIGNMENT II

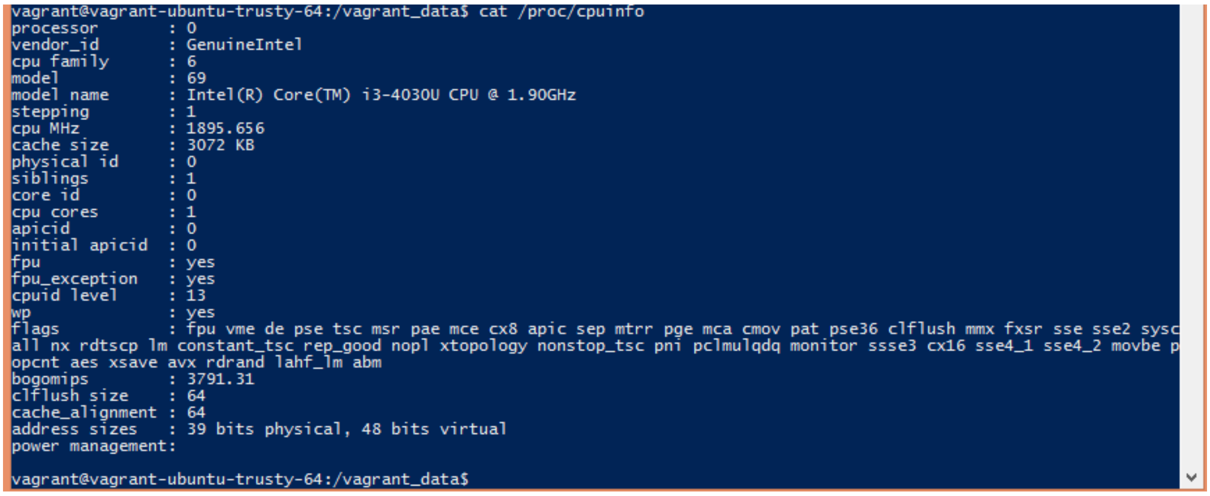
PART 1

The amount of RAM used in vagrant is 2048.



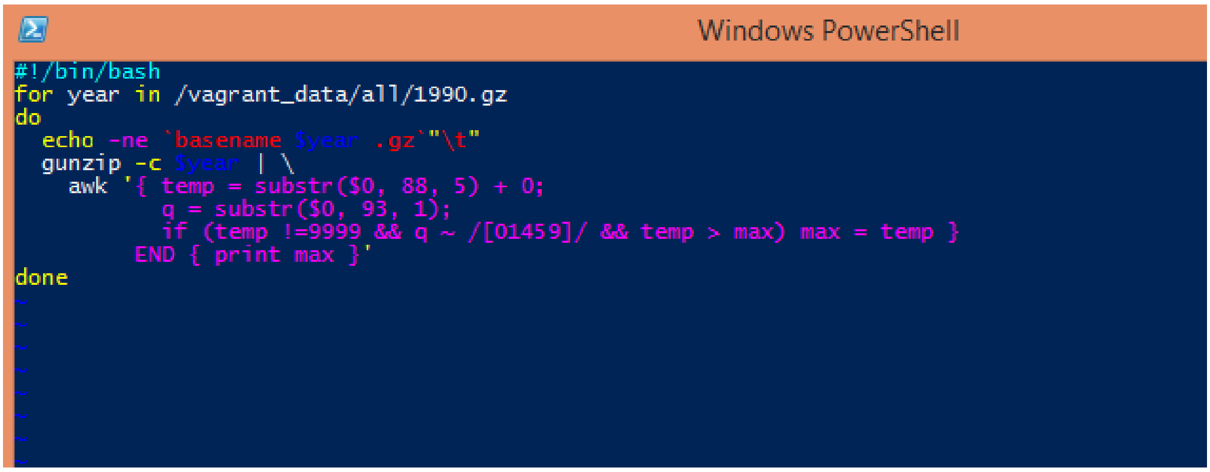
```
address sizes   : 39 bits physical, 48 bits virtual
power management:

vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$ cat /proc/meminfocat
cat: /proc/meminfocat: No such file or directory
vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$ cat /proc/meminfo
MemTotal:        2049964 kB
MemFree:         1672544 kB
Buffers:           13128 kB
Cached:           239544 kB
SwapCached:            0 kB
Active:           154256 kB
Inactive:         177148 kB
Active(anon):      78836 kB
Inactive(anon):      564 kB
Active(file):      75420 kB
Inactive(file):   176584 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:                 0 kB
Writeback:             0 kB
AnonPages:         78772 kB
Mapped:             8604 kB
Shmem:               672 kB
Slab:              25424 kB
SReclaimable:      17696 kB
SUnreclaim:         7728 kB
KernelStack:         680 kB
PageTables:         2480 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     1024980 kB
Committed_AS:     136352 kB
VmallocTotal:   34359738367 kB
VmallocUsed:       26504 kB
VmallocChunk:   34359706616 kB
HardwareCorrupted:     0 kB
AnonHugePages:      6144 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
DirectMap4k:       34752 kB
DirectMap2M:     2062336 kB
vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$
```

CPU speed is 1895.656.



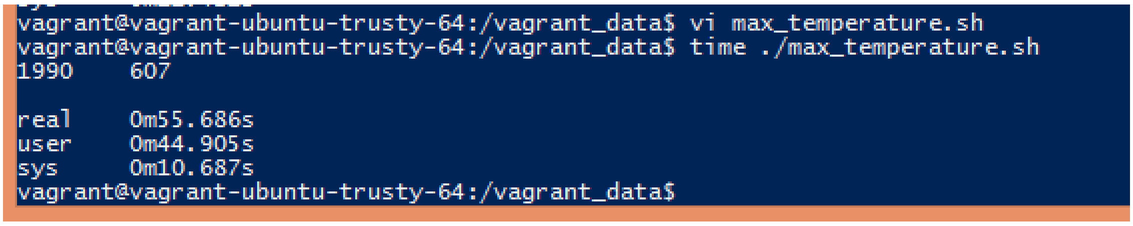
```
vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$ cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 69
model name      : Intel(R) Core(TM) i3-4030U CPU @ 1.90GHz
stepping        : 1
cpu MHz         : 1895.656
cache size      : 3072 KB
physical id     : 0
siblings        : 1
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 sysc
all nx rdtscp lm constant_tsc rep_good nopl xtopology nonstop_tsc pni pclmulqdq monitor ssse3 cx16 sse4_1 sse4_2 movbe p
opcnt aes xsave avx rdrand lahf_lm abm
bogomips        : 3791.31
clflush size    : 64
cache_alignment : 64
address sizes   : 39 bits physical, 48 bits virtual
power management:

vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$
```

Run against the 1990 data set.



```
#!/bin/bash
for year in /vagrant_data/all/1990.gz
do
  echo -ne `basename $year .gz`"\t"
  gunzip -c $year | \
    awk '{ temp = substr($0, 88, 5) + 0;
           q = substr($0, 93, 1);
           if (temp !=9999 && q ~ /[01459]/ && temp > max) max = temp }
         END { print max }'
done
~
~
~
~
~
~
~
~
```

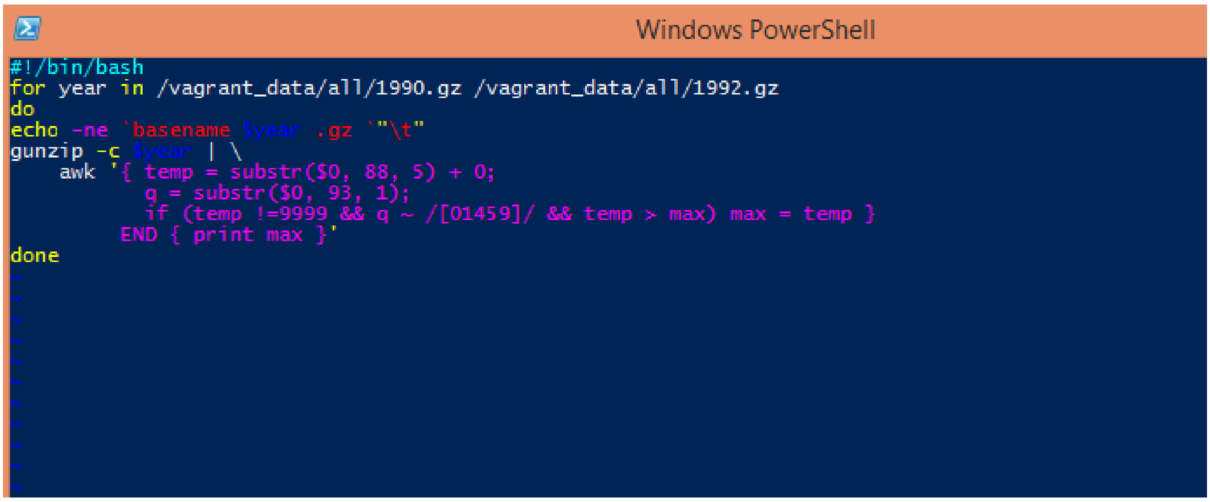
The above is the script used to run for the year 1990.



```
vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$ vi max_temperature.sh
vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$ time ./max_temperature.sh
1990    607

real    0m55.686s
user    0m44.905s
sys     0m10.687s
vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$
```

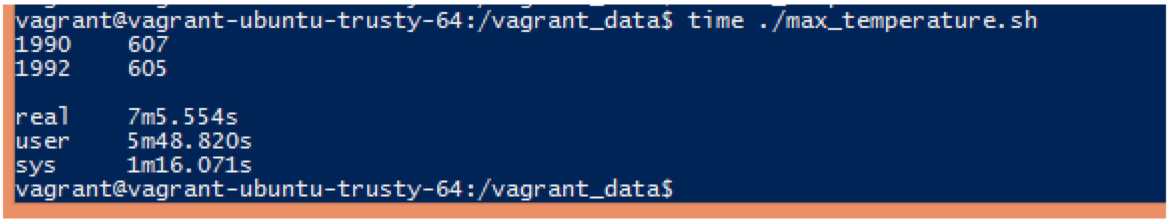
The above screen shot gives the time required to run for the year 1990.

Second time against 1990 and 1992



```
#!/bin/bash
for year in /vagrant_data/all/1990.gz /vagrant_data/all/1992.gz
do
echo -ne `basename $year .gz `"\t"
gunzip -c $year | \
    awk '{ temp = substr($0, 88, 5) + 0;
           q = substr($0, 93, 1);
           if (temp !=9999 && q ~ /[01459]/ && temp > max) max = temp }
         END { print max }'
done
~
~
~
~
~
~
~
~
~
~
~
```

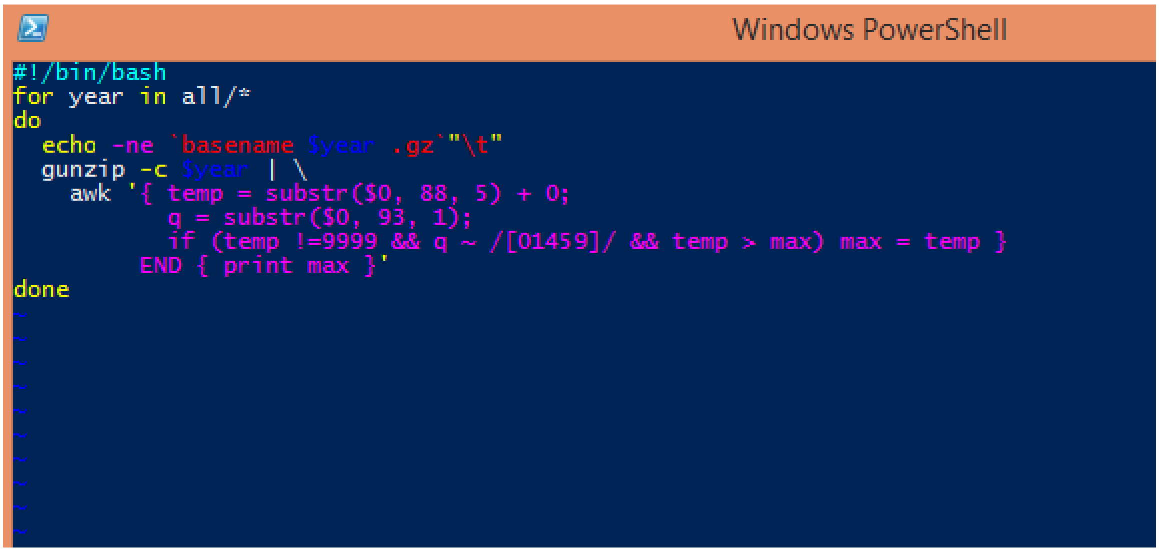
The above is the script to determine the time for comparsion for the year 1990 and 1992.



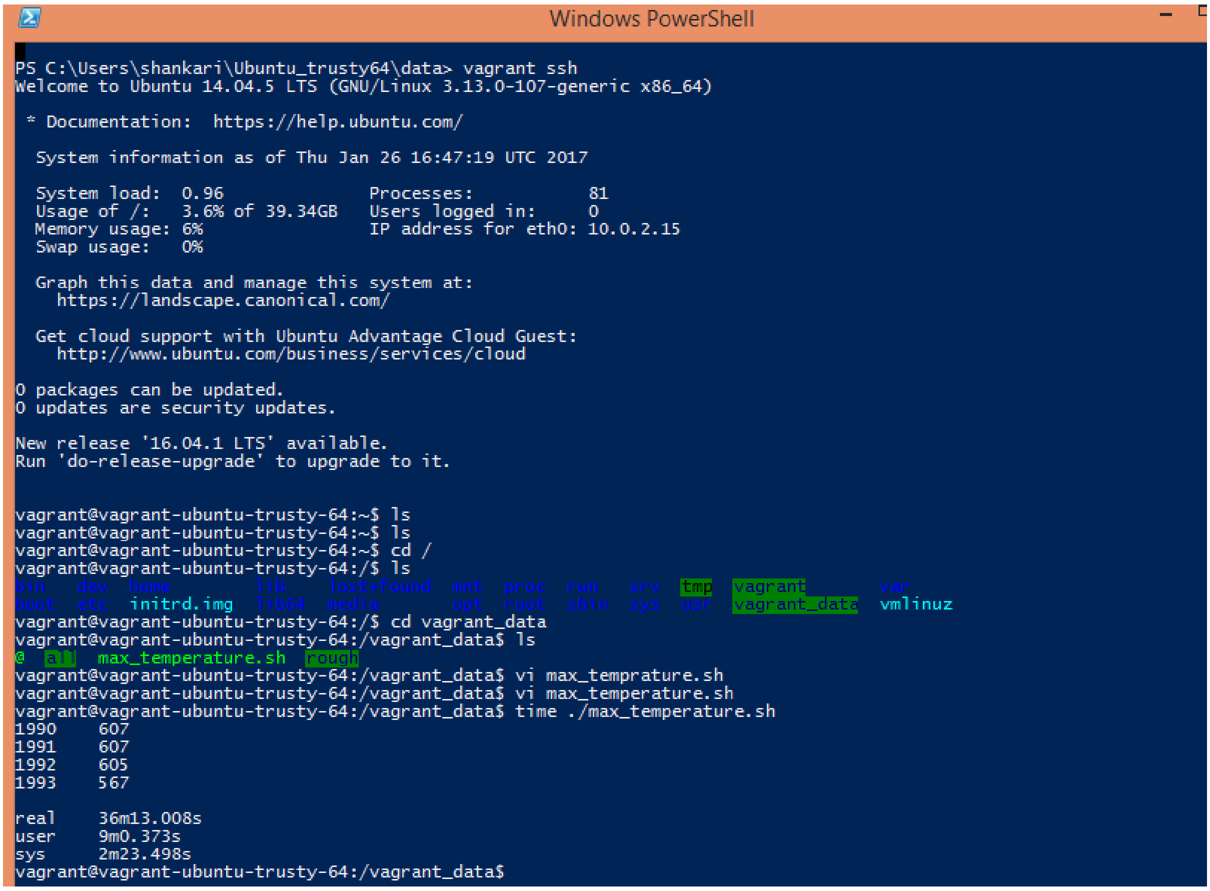
```
vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$ time ./max_temperature.sh
1990    607
1992    605

real    7m5.554s
user    5m48.820s
sys     1m16.071s
vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$
```

The time required to run the data above for the years 190 and 1992 are given above.

Third time against 1990, 1991, 1992, 1993.



```
#!/bin/bash
for year in all/*
do
  echo -ne `basename $year .gz`"\t"
  gunzip -c $year | \
    awk '{ temp = substr($0, 88, 5) + 0;
           q = substr($0, 93, 1);
           if (temp !=9999 && q ~ /[01459]/ && temp > max) max = temp }
         END { print max }'
done
~
~
~
~
~
~
~
~
~
~
```



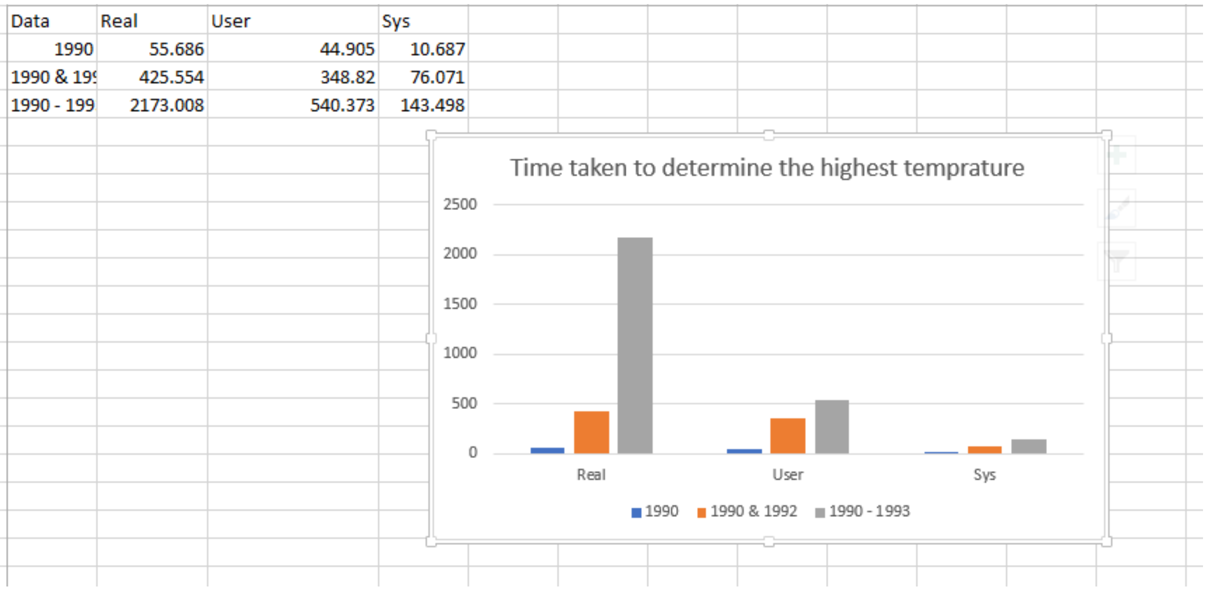
```
PS C:\Users\shankari\Ubuntu_trusty64\data> vagrant ssh
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 3.13.0-107-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Thu Jan 26 16:47:19 UTC 2017

  System load:  0.96              Processes:           81
  Usage of /:   3.6% of 39.34GB   Users logged in:     0
  Memory usage: 6%                IP address for eth0: 10.0.2.15
  Swap usage:   0%

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


vagrant@vagrant-ubuntu-trusty-64:~$ ls
vagrant@vagrant-ubuntu-trusty-64:~$ ls
vagrant@vagrant-ubuntu-trusty-64:~$ cd /
vagrant@vagrant-ubuntu-trusty-64:/$ ls
bin   dev  home        lib    lost+found  mnt  proc  run   srv  tmp  vagrant       var
boot  etc  initrd.img  lib64  media       opt  root  sbin  sys  usr  vagrant_data  vmlinuz
vagrant@vagrant-ubuntu-trusty-64:/$ cd vagrant_data
vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$ ls
@  all  max_temperature.sh  rough
vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$ vi max_temprature.sh
vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$ vi max_temperature.sh
vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$ time ./max_temperature.sh
1990    607
1991    607
1992    605
1993    567

real    36m13.008s
user    9m0.373s
sys     2m23.498s
vagrant@vagrant-ubuntu-trusty-64:/vagrant_data$
```

The time taken for all the years is 36m and 3.008s.

ANALYSIS

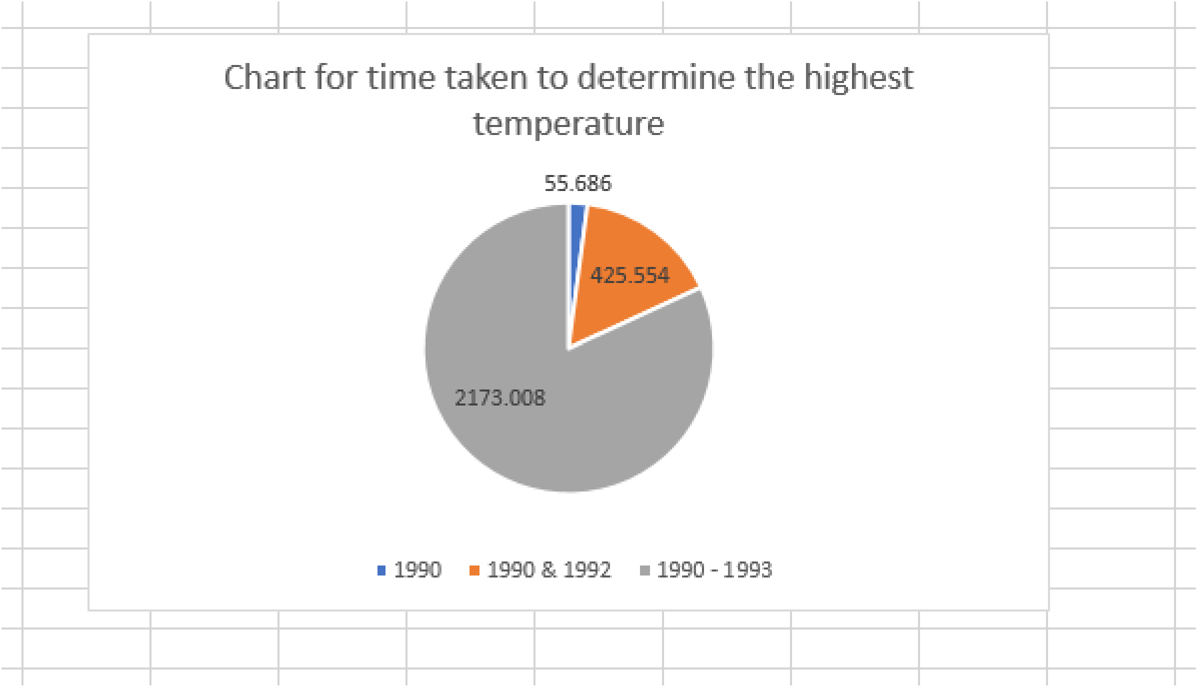
The time values obtained are:

Data	Real(Sec)	User(Sec)	System(Sec)
1990	55.686	44.905	10.687
1990 & 1992	425.554	348.82	76.071
1990 - 1994	2173.008	540.373	143.498



Data	Real	User	Sys
1990	55.686	44.905	10.687
1990 & 19	425.554	348.82	76.071
1990 - 199	2173.008	540.373	143.498

The bar graph gives the time to determine the highest temperature for the data set given. The X axis is the three set of data given and the Y axis is the time taken seconds to determine the highest temperature for each year.

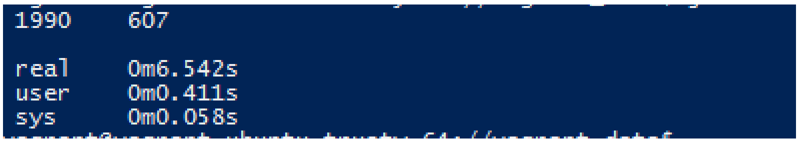


The pie chart for the highest temperature for the data gives us the idea that more the data more the time it requires to execute.

The awk script works well for data of smaller size that is clearly displayed as for a 100 MB data that is it takes only 55.686 seconds to find the highest temperature. On the other hand, for a 500 MB data the time taken is 425.554 and for a 900MB data the time taken is 2173.008. When the size of data increases the time taken increases enormously. So to handle large set of data we need to use some technology that are more efficient to handle large amount of data rather than using awk.

PART II

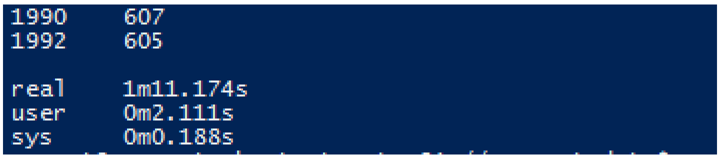
For the year 1990.



```
1990    607

real    0m6.542s
user    0m0.411s
sys     0m0.058s
```

The above is the screenshot of time values obtained to execute the code to find the maximum temperature for the year 1990.

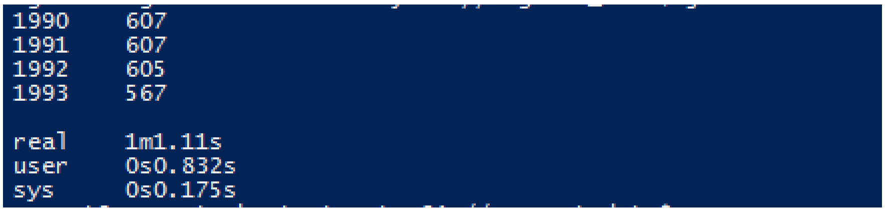
For the year 1990 & 1992.



```
1990    607
1992    605

real    1m11.174s
user    0m2.111s
sys     0m0.188s
```

The above is the screenshot of time values obtained to execute the code to find the maximum temperature for the year 1990 and 1992.

For the year 1990- 1992.



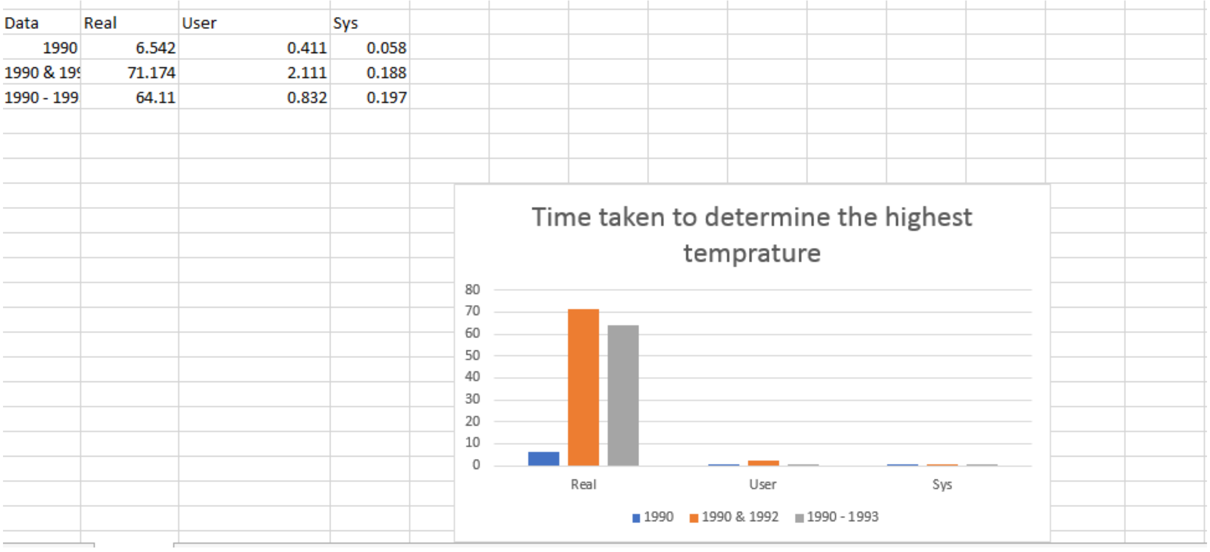
```
1990    607
1991    607
1992    605
1993    567

real    1m1.11s
user    0s0.832s
sys     0s0.175s
```

The above is the screenshot of time values obtained to execute the code to find the maximum temperature for the year 1990 – 1992.

The time values obtained are:

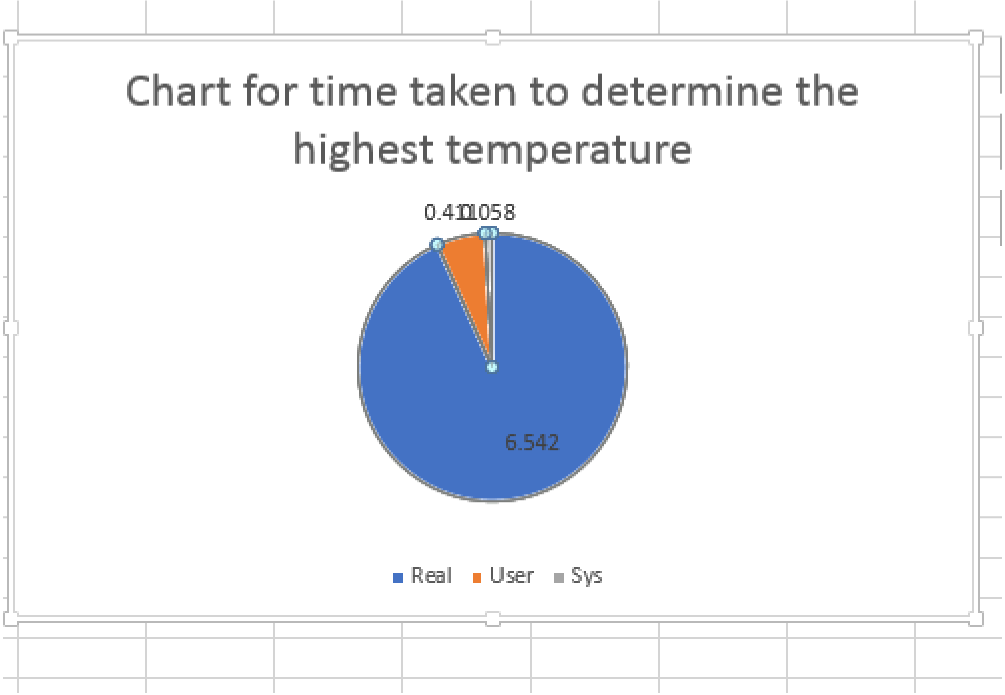
Data	Real (sec)	User(Sec)	System(Sec)
1990	6.542	0.411	0.058
1990 & 1992	71.174	2.111	0.188
1990 - 1994	64.11	0.832	0.175

ANALYSIS



Data	Real	User	Sys
1990	6.542	0.411	0.058
1990 & 19:	71.174	2.111	0.188
1990 - 199	64.11	0.832	0.197

The X axis in the above bar chart is the years and the Y axis is the time to determine the maximum temperature of the specific year.



In case of using java it seems to be much better than awk script. For 100 MB data time taken in real system is 6.542s and for 500 MB the time taken is 71.174 and time taken for 900 MB is 64.11. The time taken to find the maximum temperature for java is lesser compared to the time for moderate size. The main reason according to me for the time reduction is due to using sql and the efficiency in the java code. In case of java the code could be made efficient to improve the performance of the time of execution.