UNIX Administration Course

Version 1.0

mapleson@gamers.org
Tel: (+44) (0)1772 893297
Fax: (+44) (0)1772 892913
WWW: http://www.futuretech.vuurwerk.nl/

Detailed Notes for Day 2 (Part 3)

UNIX Fundamentals: System Monitoring Tools.

Running a UNIX system always involves monitoring how a system is behaving on a daily basis. Admins must keep an eye on such things as:

- disk space usage
- system performance and statistics, eg. CPU usage, disk I/O, memory, etc.
- network performance and statistics
- system status, user status
- service availability, eg. Internet access
- system hardware failures and related maintenance
- suspicious/illegal activity

Figure 37. The daily tasks of an admin.

This section explains the various system monitoring tools, commands and techniques which an admin can use to monitor the areas listed above. Typical example administration tasks are discussed in a later lecture. The focus here is on available tools and what they offer, not on how to use them as part of an admin strategy.

Disk Space Usage.

The df command reports current disk space usage. Run on its own, the output is expressed in terms of numbers of blocks used/free, eg.:

```
yoda # df
Filesystem             Type  blocks     use     avail  %use Mounted on
/dev/root               xfs  8615368  6116384  2498984  71  /
/dev/dsk/dks4d5s7       xfs  8874746  4435093  4439653  50  /home
milamber:/mapleson      nfs  4225568  3906624   318944  93  /mapleson
```

Figure 38. Using df without options.

A block is 512 bytes. But most people tend to think in terms of kilobytes, megabytes and gigabytes, not multiples of 512 bytes. Thus, the -k option can be used to show the output in K:

```
yoda # df -k
Filesystem             Type  kbytes     use     avail  %use Mounted on
/dev/root               xfs  4307684  3058192  1249492  71  /
/dev/dsk/dks4d5s7       xfs  4437373  2217547  2219826  50  /home
milamber:/mapleson      nfs  2112784  1953312   159472  93  /mapleson
```

Figure 39. The -k option with df to show data in K.

The df command can be forced to report data only for the file system housing the current directory by adding a period:

```
yoda # cd /home && df -k .
Filesystem            Type  kbytes     use     avail  %use Mounted on
/dev/dsk/dks4d5s7      xfs  4437373  2217547  2219826  50  /home
```

Figure 40. Using df to report usage for the file system holding the current directory.

The du command can be used to show the amount of space used by a particular directory or file, or series of directories and files. The -k option can be used to show usage in K instead of 512byte blocks just as with df. du's default behaviour is to report a usage amount recursively for every sub-directory, giving a total at the end, eg.:

```
yoda # du -k /usr/share/data/models
436     /usr/share/data/models/sgi
160     /usr/share/data/models/food
340     /usr/share/data/models/toys
336     /usr/share/data/models/buildings
412     /usr/share/data/models/household
864     /usr/share/data/models/scenes
132     /usr/share/data/models/chess
1044    /usr/share/data/models/geography
352     /usr/share/data/models/CyberHeads
256     /usr/share/data/models/machines
1532    /usr/share/data/models/vehicles
88      /usr/share/data/models/simple
428     /usr/share/data/models/furniture
688     /usr/share/data/models/robots
7760    /usr/share/data/models
```

Figure 41. Using du to report usage for several directories/files.

The -s option can be used to restrict the output to just an overall total for the specified directory:

```
yoda # du -k -s /usr/share/data/models
7760    /usr/share/data/models
```

Figure 42. Restricting du to a single directory.

By default, du does not follow symbolic links, though the -L option can be used to force links to be followed if desired.

However, du does examine NFS-mounted file systems by default. The -l and -m options can be used to restrict this behaviour, eg.:

```
ASH # cd /
ASH # du -k -s -l
0       CDROM
0       bin
0       debug
68      dev
0       disk2
2       diskcopy
0       dumpster
299     etc
0       home
2421    lib
```

```
2579    lib32
0       opt
0       proc
1       root.home
4391    sbin
565     stand
65      tmp
3927    unix
397570  usr
6346    var
```

Figure 43. Forcing du to ignore symbolic links.

The output in Fig 43 shows that the /home directory has been ignored.

Another example: a user can find out how much disk space their account currently uses by entering: du -k -s ~/

Swap space (ie. virtual memory on disk) can be monitored using the swap command with the -l option.

For full details on these commands, see the relevant man pages.

Commands relating to file system quotas are dealt with in a later lecture.

System Performance.

This includes processor loading, disk loading, etc.

The most common command used by admins/users to observe CPU usage is ps, which displays a list of currently running processes along with associated information, including the percentage of CPU time currently being consumed by each process, eg.:

```
ASH 6# ps -ef
    UID   PID  PPID  C    STIME TTY      TIME CMD
   root     0     0  0 08:00:41 ?       0:01 sched
   root     1     0  0 08:00:41 ?       0:01 /etc/init
   root     2     0  0 08:00:41 ?       0:00 vhand
   root     3     0  0 08:00:41 ?       0:03 bdflush
   root     4     0  0 08:00:41 ?       0:00 munldd
   root     5     0  0 08:00:41 ?       0:02 vfs_sync
   root   900   895  0 08:03:27 ?       1:25 /usr/bin/X11/Xsgi -bs [etc]
   root     7     0  0 08:00:41 ?       0:00 shaked
   root     8     0  0 08:00:41 ?       0:00 xfsd
   root     9     0  0 08:00:41 ?       0:00 xfsd
   root    10     0  0 08:00:41 ?       0:00 xfsd
   root    11     0  0 08:00:41 ?       0:00 pdflush
   root   909   892  0 08:03:31 ?       0:02 /usr/etc/videod
   root  1512  1509  0 15:37:17 ?       0:00 sh -c /var/X11/xdm/Xlogin
   root   158     1  0 08:01:01 ?       0:01 /usr/etc/ypbind -ypsetme
   root    70     1  0 08:00:50 ?       0:00 /usr/etc/syslogd
   root  1536   211  0 16:06:04 pts/0   0:00 rlogind
   root   148     1  0 08:01:00 ?       0:01 /usr/etc/routed -h -[etc]
   root   146     1  0 08:01:00 ?       0:00 /usr/etc/portmap
   root   173   172  0 08:01:03 ?       0:01 /usr/etc/nfsd 4
   root   172     1  0 08:01:03 ?       0:01 /usr/etc/nfsd 4
   root   174   172  0 08:01:03 ?       0:01 /usr/etc/nfsd 4
   root   175   172  0 08:01:03 ?       0:01 /usr/etc/nfsd 4
   root   178     1  0 08:01:03 ?       0:00 /usr/etc/biod 4
   root   179     1  0 08:01:03 ?       0:00 /usr/etc/biod 4
   root   180     1  0 08:01:03 ?       0:00 /usr/etc/biod 4
   root   181     1  0 08:01:03 ?       0:00 /usr/etc/biod 4
   root   189     1  0 08:01:04 ?       0:00 bio3d
   root   190     1  0 08:01:04 ?       0:00 bio3d
```

```
root   191     1  0 08:01:04 ?       0:00 bio3d
root   202     1  0 08:01:05 ?       0:00 /usr/etc/rpc.statd
root   192     1  0 08:01:04 ?       0:00 bio3d
root   188     1  0 08:01:03 ?       0:00 bio3d
root   311     1  0 08:01:08 ?       0:00 /usr/etc/timed -M -F yoda
root   211     1  0 08:01:05 ?       0:02 /usr/etc/inetd
root   823     1  0 08:01:33 ?       0:13 /usr/lib/sendmail -bd -q15m
root  1557  1537  9 16:10:58 pts/0   0:00 ps -ef
root   892     1  0 08:03:25 ?       0:00 /usr/etc/videod
root  1513  1512  0 15:37:17 ?       0:07 /usr/Cadmin/bin/clogin -f
root  1546   872  0 16:07:55 ?       0:00 /usr/Cadmin/bin/directoryserver
root  1537  1536  1 16:06:04 pts/0   0:01 -tcsh
root   903     1  0 08:03:27 tablet  0:00 /sbin/getty ttyd1 co_9600
  lp   460     1  0 08:01:17 ?       0:00 /usr/lib/lpsched
root  1509   895  0 15:37:13 ?       0:00 /usr/bin/X11/xdm
root   488     1  0 08:01:19 ?       0:01 /sbin/cron
root  1556  1537 28 16:10:56 pts/0   0:01 find /usr -name *.txt -print
root   895     1  0 08:03:27 ?       0:00 /usr/bin/X11/xdm
root   872     1  0 08:02:32 ?       0:06 /usr/Cadmin/bin/directoryserver
```

Figure 44. Typical output from the ps command.

Before obtaining the output shown in Fig 44, I ran a find command in the background. The output shows that the find command was utilising 28% of available CPU resources; tasks such as find are often limited by the speed and bandwidth capacity of the disk, not the speed of the main CPU.

The ps command has a variety of options to show or not show various information. Most of the time though, ’ps -ef’ is adequate to display the kind of information required. Note that other UNIX variants use slightly different options, eg. the equivalent command on SunOS would be ’ps -aux’.

One can use grep to only report data for a particular process, eg.:

```
ASH 5# ps -ef | grep lp

     lp   460     1  0 08:01:17 ?       0:00 /usr/lib/lpsched
```

Figure 45. Filtering ps output with grep.

This only reports data for the lp printer scheduler.

However, ps only gives a snapshot of the current system state. Often of more interest is a system’s dynamic behaviour. A more suitable command for monitoring system performance over time is ’top’, a typical output of which looks like this:

```
IRIX ASH 6.2 03131015 IP22 Load[0.22,0.12,0.01] 16:17:47  166 procs
    user   pid  pgrp   %cpu proc  pri  size   rss    time  command
    root  1576  1576  24.44    *   20   386    84    0:02  find
    root  1577  1577   0.98    0   65   432   100    0:00  top
    root  1513  1509   0.18    *   60  4322  1756    0:07  clogin
    root   900   900   0.12    *   60  2858   884    1:25  Xsgi
    root   146   146   0.05    *   60   351    77    0:00  portmap
    root   158     0   0.05    *   60   350    81    0:00  ypbind
    root  1567  1567   0.02    *   60   349    49    0:00  rlogind
    root     3     0   0.01    *  +39     0     0    0:03  bdflush
    root   172     0   0.00    *   61     0     0    0:00  nfsd
    root   173     0   0.00    *   61     0     0    0:00  nfsd
    root   174     0   0.00    *   61     0     0    0:00  nfsd
    root   175     0   0.00    *   61     0     0    0:00  nfsd
```

Figure 46. top shows a continuously updated output.

From the man page for top:

"Two header lines are displayed. The first gives the machine name, the release and build date

information, the processor type, the 1, 5, and 15 minute load average, the current time and the number of active processes. The next line is a header containing the name of each field highlighted."

The display is constantly updated at regular intervals, the duration of which can be altered with the -i option (default duration is 5 seconds). top shows the following data for each process:

"user name, process ID, process group ID, CPU usage, processor currently executing the process (if process not currently running), process priority, process size (in pages), resident set size (in pages), amount of CPU time used by the process, and the process name."

Just as with the ps command, top shows the ID number for each process. These IDs can be used with the kill command (and others) to control running processes, eg. shut them down, suspend them, etc.

There is a GUI version of top called gr_top.

Note that IRIX 6.5 contains a newer version of top which gives even more information, eg.:

```
IRIX WOLFEN 6.5 IP22           load averages: 0.06 0.01 0.00       17:29:44
58 processes:  56 sleeping, 1 zombie, 1 running
CPU: 93.5% idle,  0.5% usr,  5.6% ker,  0.0% wait,  0.0% xbrk,  0.5% intr
Memory: 128M max, 116M avail, 88M free, 128M swap, 128M free swap

  PID       PGRP USERNAME PRI  SIZE   RES STATE    TIME WCPU% CPU% COMMAND
 1372       1372 root      20 2204K 1008K run/0    0:00  0.2  3.22 top
  153        153 root      20 2516K 1516K sleep    0:05  0.1  1.42 nsd
 1364       1364 root      20 1740K  580K sleep    0:00  0.0  0.24 rlogind
```

Figure 47. The IRIX 6.5 version of top, giving extra information.

A program which offers much greater detail than top is osview. Like top, osview constantly updates a whole range of system performance statistics. Unlike top though, so much information is available from osview that it offers several different ’pages’ of data. The number keys are used to switch between pages. Here is a typical output for each of the five pages:

Page 1 (system information):

```
Osview 2.1 : One Second Average      WOLFEN 17:32:13 04/21/99 #5  int=5s
Load Average             fs ctl    2.0M
  1 Min      0.000      fs data    7.7M
  5 Min      0.000       delwri      0
  15 Min     0.000      free      87.5M
CPU Usage                data      26.0M
  %user       0.20       empty    61.4M
  %sys        0.00      userdata  20.7M
  %intr       0.00      reserved     0
  %gfxc       0.00      pgallocs     2
  %gfxf       0.00   Scheduler
  %sxbrk      0.00     runq          0
  %idle      99.80     swapq         0
System Activity        switch        4
  syscall       19     kswitch      95
  read           1     preempt       1
  write          0   Wait Ratio
  fork           0     %IO          1.2
  exec           0     %Swap        0.0
  readch        19     %Physio      0.0
  writech       38
  iget           0
System Memory
  Phys       128.0M
   kernel     10.1M
    heap       3.9M
     mbufs    96.0K
```

```
    stream  40.0K
   ptbl      1.2M
```

Figure 48. System information from osview.

Page 2 (CPU information):

```
Osview 2.1 : One Second Average      WOLFEN 17:36:27 04/21/99 #1   int=5s
CPU Usage
  %user       0.00
  %sys      100.00
  %intr       0.00
  %gfxc       0.00
  %gfxf       0.00
  %sxbrk      0.00
  %idle       0.00
```

Figure 49. CPU information from osview.

Page 3 (memory information):

```
Osview 2.1 : One Second Average      WOLFEN 17:36:56 04/21/99 #1   int=5s
System Memory         iclean         0
  Phys      128.0M *Swap
   kernel    10.5M *System VM
    heap      4.2M *Heap
     mbufs  100.0K *TLB Actions
     stream  48.0K *Large page stats
    ptbl      1.3M
    fs ctl    1.5M
   fs data    8.2M
    delwri       0
   free      77.1M
    data     28.8M
    empty    48.3M
   userdata  30.7M
   reserved     0
   pgallocs    450
Memory Faults
  vfault      1.7K
  protection   225
  demand       375
  cw            25
  steal        375
  onswap         0
  oncache     1.4K
  onfile         0
  freed          0
  unmodswap      0
  unmodfile      0
```

Figure 50. Memory information from osview.

Page 4 (network information):

```
Osview 2.1 : One Second Average      WOLFEN 17:38:15 04/21/99 #1   int=5s
TCP
  acc. conns     0
  sndtotal      33
  rcvtotal       0
  sndbyte      366
  rexmtbyte      0
  rcvbyte        0
UDP
  ipackets       0
  opackets       0
  dropped        0
  errors         0
IP
  ipackets       0
```

```
  opackets      33
  forward        0
  dropped        0
  errors         0
NetIF[ec0]
  Ipackets       0
  Opackets      33
  Ierrors        0
  Oerrors        0
  collisions     0
NetIF[lo0]
```

Figure 51. Network information from osview.

Page 5 (miscellaneous):

```
 Osview 2.1 : One Second Average      WOLFEN 17:38:43 04/21/99 #1   int=5s
 Block Devices
   lread      37.5K
   bread          0
   %rcache    100.0
   lwrite         0
   bwrite         0
   wcancel        0
   %wcache      0.0
   phread         0
   phwrite        0
 Graphics
   griioctl       0
   gintr         75
   swapbuf        0
   switch         0
   fifowait       0
   fifonwait      0
 Video
   vidioctl       0
   vidintr        0
   drop_add       0
*Interrupts
*PathName Cache
*EfsAct
*XfsAct
*Getblk
*Vnodes
```

Figure 51. Miscellaneous information from osview.

osview clearly offers a vast amount of information for monitoring system and network activity.

There is a GUI version of osview called gr_osview. Various options exist to determine which parameters are displayed with gr_osview, the most commonly used being -a to display as much data as possible.

Programs such as top and osview may be SGI-specific (I’m not sure). If they are, other versions of UNIX are bound to have equivalent programs to these.

Example use: although I do virtually all the administration of the server remotely using the office Indy (either by command line or GUI tools), there is also a VT-style terminal in my office connected to the server’s serial port via a lengthy cable (the Challenge S server itself is in a small ante room). The VT display offers a simple text-only interface to the server; thus, most of the time, I leave osview running on the VT display so that I can observe system activity whenever I need to. The VT also offers an extra communications link for remote administration should the network go down, ie. if the network links fail (eg. broken hub) the admin Indy cannot be used to communicate with the server, but the VT still can.

Another tool for monitoring memory usage is gmemusage, a GUI program which displays a

graphical split-bar chart view of current memory consumption. gmemusage can also display a breakdown of the regions within a program's memory space, eg. text, data, shared memory, etc.

Much lower-level tools exist too, such as sar (system activity reporter). In fact, osview works by using sar. Experienced admins may use tools like sar, but most admins will prefer to use higher-level tools such as top, osview and gmemusage. However, since sar gives a text output, one can use it in script files for automated system information gathering, eg. a system activity report produced by a script, executed every hour by the cron job-scheduling system (sar-based information gathering scripts are included in the cron job schedule as standard). sar can be given options to report only on selected items, eg. the number of processes in memory waiting for CPU resource time. sar can be told to monitor some system feature for a certain period, saving the data gathered during that period to a file. sar is a very flexible program.

Network Performance and Statistics.

osview can be used to monitor certain network statistics, but another useful program is ttcp. The online book, "IRIX Admin: Networking and Mail", says:

> "The ttcp tool measures network throughput. It provides a realistic measurement of network performance between two stations because it allows measurements to be taken at both the local and remote ends of the transmission."

To run a test with ttcp, enter the following on one system, eg. sevrin:

```
ttcp -r -s
```

Then enter the following on another system, eg. akira:

```
ttcp -t -s sevrin
```

After a delay of roughly 20 seconds for a 10Mbit network, results are reported by both systems, which will look something like this:

```
SEVRIN # ttcp -r -s
ttcp-r: buflen=8192, nbuf=2048, align=16384/0, port=5001  tcp
ttcp-r: socket
ttcp-r: accept from 193.61.252.2
ttcp-r: 16777216 bytes in 18.84 real seconds = 869.70 KB/sec +++
ttcp-r: 3191 I/O calls, msec/call = 6.05, calls/sec = 169.39
ttcp-r: 0.1user 3.0sys 0:18real 16% 118maxrss 0+0pf 1170+1csw

AKIRA # ttcp -t -s sevrin
ttcp-t: buflen=8192, nbuf=2048, align=16384/0, port=5001  tcp  -> sevrin
ttcp-t: socket
ttcp-t: connect
ttcp-t: 16777216 bytes in 18.74 real seconds = 874.19 KB/sec +++
ttcp-t: 2048 I/O calls, msec/call = 9.37, calls/sec = 109.27
ttcp-t: 0.0user 2.3sys 0:18real 12% 408maxrss 0+0pf 426+4csw
```

Figure 52. Results from ttcp between two hosts on a 10Mbit network.

Full details of the output are in the ttcp man page, but one can immediately see that the observed network throughput (around 870KB/sec) is at a healthy level.

Another program for gathering network performance information is netstat. The online book, "IRIX Admin: Networking and Mail", says:

"The netstat tool displays various network-related data structures that are useful for monitoring and troubleshooting a network. Detailed statistics about network collisions can be captured with the netstat tool."

netstat is commonly used with the -i option to list basic local network information, eg.:

```
yoda # netstat -i
Name Mtu   Network     Address            Ipkts Ierrs    Opkts Oerrs  Coll
ec0  1500  193.61.252  yoda.comp.uclan  3906956     3  2945002     0 553847
ec3  1500  193.61.250  gate-yoda.comp.   560206     2   329366     0 16460
lo0  8304  loopback    localhost         476884     0   476884     0     0
```

Figure 53. The output from netstat.

Here, the packet collision rate has averaged at 18.8%. This is within acceptable limits [1].

Another useful command is 'ping'. This program sends packets of data to a remote system requesting an acknowledgement response for each packet sent. Options can be used to send a specific number of packets, or send as many packets as fast as they are returned, send a packet every so often (user-definable duration), etc.

For example:

```
MILAMBER # ping yoda
PING yoda.comp.uclan.ac.uk (193.61.252.1): 56 data bytes
64 bytes from 193.61.252.1: icmp_seq=0 ttl=255 time=1 ms
64 bytes from 193.61.252.1: icmp_seq=1 ttl=255 time=1 ms
64 bytes from 193.61.252.1: icmp_seq=2 ttl=255 time=1 ms
64 bytes from 193.61.252.1: icmp_seq=3 ttl=255 time=1 ms
64 bytes from 193.61.252.1: icmp_seq=4 ttl=255 time=1 ms
64 bytes from 193.61.252.1: icmp_seq=5 ttl=255 time=1 ms
64 bytes from 193.61.252.1: icmp_seq=6 ttl=255 time=1 ms

----yoda.comp.uclan.ac.uk PING Statistics----
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 1/1/1 ms
```

Figure 54. Example use of the ping command.

I pressed CTRL-C after the 7th packet was sent. ping is a quick and easy way to see if a host is active and if so how responsive the connection is.

If a ping test produces significant packet loss on a local network, then it is highly likely there exists a problem of some kind. Normally, one would rarely see a non-zero packet loss on a local network from a direct machine-to-machine ping test.

A fascinating use of ping I once observed was at The Moving Picture Company (MPC) [2]. The admin had written a script which made every host on the network send a ping test to every other host. The results were displayed as a table with host names shown down the left hand side as well as along the top. By looking for horizontal or diagonal lines of unusually large ping times, the admin could immediately see if there was a problem with a single host, or with a larger part of the network. Because of the need for a high system availability rate, the script allows the admin to spot problems almost as soon as they occur, eg. by running the script once every ten seconds.

When the admin showed me the script in use, one column had rather high ping times (around 20ms). Logging into the host with rlogin, ps showed everything was ok: a complex process was

merely consuming alot of CPU time, giving a slower network response.

System Status and User Status.

The rup command offers an immediate overview of current system states, eg.:

```
yoda # rup
yoda.comp.uclan.ac.u    up  6 days,  8:25,    load average: 0.33, 0.36, 0.35
gate-yoda.comp.uclan    up  6 days,  8:25,    load average: 0.33, 0.36, 0.35
wolfen.comp.uclan.ac    up          11:28,    load average: 0.00, 0.00, 0.00
conan.comp.uclan.ac.    up          11:28,    load average: 0.06, 0.01, 0.00
akira.comp.uclan.ac.    up          11:28,    load average: 0.01, 0.00, 0.00
nikita.comp.uclan.ac    up          11:28,    load average: 0.03, 0.00, 0.00
gibson.comp.uclan.ac    up          11:28,    load average: 0.00, 0.00, 0.00
woo.comp.uclan.ac.uk    up          11:28,    load average: 0.01, 0.00, 0.00
solo.comp.uclan.ac.u    up          11:28,    load average: 0.00, 0.00, 0.00
cameron.comp.uclan.a    up          11:28,    load average: 0.02, 0.00, 0.00
sevrin.comp.uclan.ac    up          11:28,    load average: 0.69, 0.46, 0.50
ash.comp.uclan.ac.uk    up          11:28,    load average: 0.00, 0.00, 0.00
ridley.comp.uclan.ac    up          11:28,    load average: 0.00, 0.00, 0.00
leon.comp.uclan.ac.u    up          11:28,    load average: 0.00, 0.00, 0.00
warlock.comp.uclan.a    up           1:57,    load average: 0.08, 0.13, 0.11
milamber.comp.uclan.    up           9:52,    load average: 0.11, 0.07, 0.00
merlin.comp.uclan.ac    up          11:28,    load average: 0.01, 0.00, 0.00
indiana.comp.uclan.a    up          11:28,    load average: 0.00, 0.00, 0.02
stanley.comp.uclan.a    up           1:56,    load average: 0.00, 0.00, 0.00
```

Figure 55. The output from rup.

The load averages for a single machine can be ascertained by running ’uptime’ on that machine, eg.:

```
MILAMBER 84# uptime
  8:05pm  up 10:28,  6 users,  load average: 0.07, 0.06, 0.25
MILAMBER 85# rsh yoda uptime
  8:05pm  up 6 days,  9:02,  2 users,  load average: 0.47, 0.49, 0.42
```

Figure 56. The output from uptime.

The w command displays current system activity, including what each user is doing. The man page says, "The heading line shows the current time of day, how long the system has been up, the number of users logged into the system, and the load averages." For example:

```
yoda # w
  8:10pm  up 6 days,  9:07,  2 users,  load average: 0.51, 0.50, 0.41
User     tty from            login@   idle   JCPU   PCPU  what
root     q0  milamber.comp.  7:02pm              8         w
cmprj    ftp UNKNOWN@ns5ip.  7:29pm                        -
```

Figure 57. The output from w showing current user activity.

With the -W option, w shows the ’from’ information on a separate line, allowing one to see the full domain address of ftp connections, etc.:

```
yoda # w -W
  8:11pm  up 6 days,  9:08,  2 users,  load average: 0.43, 0.48, 0.40
User     tty       login@   idle   JCPU   PCPU  what
root     ttyq0     7:02pm              8         w -W
         milamber.comp.uclan.ac.uk
cmprj    ftp22918  7:29pm                        -
         UNKNOWN@ns5ip.uclan.ac.uk
```

Figure 58. Obtaining full domain addresses from w with the -W option.

The rusers command broadcasts to all machines on the local network, gathering data about who is

logged on and where, eg.:

```
yoda # rusers
yoda.comp.uclan.ac.uk           root
wolfen.comp.uclan.ac.uk         guest guest
gate-yoda.comp.uclan.ac.uk      root
milamber.comp.uclan.ac.uk       root root root root mapleson mapleson
warlock.comp.uclan.ac.uk        sensjv sensjv
```

Figure 59. The output from rusers, showing who is logged on where.

The multiple entries for certain users indicate that more than one shell is active for that user. As usual, my login data shows I'm doing several things at once.

rusers can be modified with options to:

- report for all machines, whether users are logged in or not (-a),
- probe a specific machine (supply host name(s) as arguments),
- display the information sorted alphabetically by:
  - host name (-h),
  - idle time (-i),
  - number of users (-u),
- give a more detailed output in the same style as the who command (-l).

Service Availability.

The most obvious way to check if a service is available for use by users is to try and use the service, eg. ftp or telnet to a test location, run up a Netscape sessions and enter a familiar URL, send an email to a local or remote account, etc. The ps command can be used to make sure the relevant background process is running for a service too, eg. 'nfsd' for the NFS system. However, if a service is experiencing problems, simply attempting to use the service will not reveal what may be wrong.

For example, if one cannot ftp, it could be because of anything from a loose cable connection to some remote server that's gone down.

The ping command is useful for an immediate check of network-related services such as ftp, telnet, WWW, etc. One pings each host in the communication chain to see if the hosts respond. If a host somewhere in the chain does not respond, then that host may be preventing any data from getting through (eg. a remote proxy server is down).

A useful command one can use to aid in such detective work is traceroute. This command sends test packets in a similar way to ping, but it also reports how the test packets reached the target site at each stage of the communication chain, showing response times in milliseconds for each step, eg.:

```
yoda # traceroute www.cee.hw.ac.uk
traceroute to osiris.cee.hw.ac.uk (137.195.52.12), 30 hops max, 40 byte packets
1 193.61.250.33 (193.61.250.33) 6 ms (ttl=30!) 3 ms (ttl=30!) 4 ms (ttl=30!)
```

```
2 193.61.250.65 (193.61.250.65) 5 ms (ttl=29!) 5 ms (ttl=29!) 5 ms (ttl=29!)
3 gw-mcc.netnw.net.uk (194.66.24.1) 9 ms (ttl=28!) 8 ms (ttl=28!) 10 ms (ttl=28!)
4 manchester-core.ja.net (146.97.253.133) 12 ms 11 ms 9 ms
5 scot-pop.ja.net (146.97.253.42) 15 ms 13 ms 14 ms
6 146.97.253.34 (146.97.253.34) 20 ms 15 ms  17 ms
7 gw1.hw.eastman.net.uk (194.81.56.110) 20 ms (ttl=248!) 18 ms 14 ms
8 cee-gw.hw.ac.uk (137.195.166.101) 17 ms (ttl=23!) 31 ms (ttl=23!) 18 ms (ttl=23!)
9 osiris.cee.hw.ac.uk (137.195.52.12) 14 ms (ttl=56!) 26 ms (ttl=56!) 30 ms (ttl=56!)
```

If a particular step shows a sudden jump in response time, then there may be a communications problem at that step, eg. the host in question may be overloaded with requests, suffering from lack of communications bandwidth, CPU processing power, etc.

At a lower level, system services often depend on background system processes, or daemons. If these daemons are not running, or have shut down for some reason, then the service will not be available.

On the SGI Indys, one example is the GUI service which handles the use of on-screen icons. The daemon responsible is called objectserver. Older versions of this particular daemon can occasionally shut down if an illegal iconic operation is performed, or if the file manager daemon experiences an error. With no objectserver running, the on-screen icons disappear.

Thus, a typical task might be to periodically check to make sure the objectserver daemon is running on all relevant machines. If it isn't, then the command sequence:

```
/etc/init.d/cadmin stop
/etc/init.d/cadmin start
```

restarts the objectserver. Once running, the on-screen icons return.

A common cause of objectserver shutting down is when a user's desktop layout configuration files (contained in .desktop- directories) become corrupted in some way, eg. edited by hand in an incorrect manner, or mangled by some other operation (eg. a faulty Java script from a home made web page). One solution is to erase the user's desktop layout configuration directory, then login as the user and create a fresh .desktop- directory.

objectserver is another example of UNIX GUI evolution. In 1996 SGI decided to replace the objectserver system entirely in IRIX 6.3 (and later) with a new service that was much more reliable, less likely to be affected by errors made in other applications, and fully capable of supporting new 'webified' iconic services such as on-screen icons that are direct links to ftp, telnet or WWW sites.

In general, checking the availability of a service requires one to check that the relevant daemons are running, that the appropriate configuration files are in place, accessible and have the correct settings, that the relevant daemon is aware of any changes which may have been made (perhaps the service needs to be stopped and restarted?) and to investigate via online information what may have caused services to fail as and when incidents occur. For every service one can use, the online information explains how to setup, admin and troubleshoot the service. The key is to know where to find that information when it is needed.

A useful source of constantly updated status information is the /var/adm/SYSLOG file. This file is where any important system events are logged. One can configure all the various services and daemons to log different degrees of detailed information in the SYSLOG file. Note: logging too much detail can cause the log file to grow very quickly, in which case one would also have to ensure that it did not consume valuable disk space. The SYSLOG file records user logins, connections via ftp, telnet, etc., messages logged at system bootup/shutdown time, and many other

things.

Vendor Information Updates.

Most UNIX vendors send out periodic information booklets containing indepth articles on various system administration issues. SGI's bulletin is called Pipeline. Such information guides are usually supplied as part of a software support contract, though the vendor will often choose to include copies on the company web site. An admin should read any relevant articles from these guides - they can often be unexpectedly enlightening.

System Hardware Failures.

When problems occur on a system, what might at first appear to be a software problem may in fact be a hardware fault. Has a disk failed? The fx program can be used to check disk status (block read tests, disk label checks, etc.)

Has a network cable failed? Are all the cable connections firmly in place in the hub? Has a plug come loose?

In late 1998, the Ve24 network stopped operating quite unexpectedly one morning. The errors made it appear that there was a problem with the NFS service or perhaps the main user files disk connected to the server; in fact, the fault lay with the Ve24 hub.

The online guides have a great deal of advice on how to spot possible hardware failures. My advice is to check basic things first and move onto the more complex possibilities later. In the above example, I wasted a great deal of time investigating whether the NFS service was responsible, or the external user files disk, when in fact I should have checked the hub connections first. As it happens, the loose connection was such that the hub indicator light was on even though the connection was not fully working (thus, a visual check would not have revealed the problem) - perhaps the fault was caused by a single loose wire out of the 8 running through the cable, or even an internal fault in the hub (more likely). Either way, the hub was eventually replaced.

Other things that can go wrong include memory faults. Most memory errors are not hardware errors though, eg. applications with bugs can cause errors by trying to access some protected area of memory.

Hardware memory errors will show up in the system log file /var/adm/SYSLOG as messages saying something like 'Hardware ECC Memory Error in SIMM slot 4'. By swapping the memory SIMMs around between the slots, one can identify which SIMM is definitely at fault (assuming there is only one causing the problem).

The most common hardware component to go wrong on a system, even a non-PC system, is the disk drive. When configuring systems, or carrying out upgrades/expansions, it is wise to stick with models recommended by the source vendor concerned, eg. SGI always uses high-quality Seagate, IBM or Quantum disk drives for their systems; thus, using (for example) a Seagate drive is a good way to ensure a high degree of reliability and compatibility with the system concerned.

Sometimes an admin can be the cause of the problem. For example, when swapping disks around or performing disk tasks such as disk cloning, it is possible to incorrectly set the SCSI ID of the disk.

SGI systems expect the system disk to be on SCSI ID 1 (though this is a configurable setting); if the internal disk is on the wrong SCSI ID, then under certain circumstances it can appear to the system as if there are multiple disks present, one on each possible ID. If hardware errors are observed on bootup (the system diagnostics checks), then the first thing to do is to reboot and enter the low-level 'Command Monitor' (an equivalent access method will exist for all UNIX systems): the Command Monitor has a small set of commands available, some of which can be used to perform system status checks, eg. the hinv command. For the problem described above, hinv would show multiple instances of the same disk on all SCSI IDs from 1 to 7 - the solution is to power down and check the SCSI jumpers carefully.

Other problems can occasionally be internal, eg. a buildup of dust blocking air vents (leading to overheating), or a fan failure, followed by overheating and eventually an automatic system shutdown (most UNIX systems' power supplies include circuitry to monitor system temperature, automatically shutting down if the system gets too hot). This leads on to questions of system maintenance which will be dealt with on Day 3.

After disk failures, the other most common failure is the power supply. It can sometimes be difficult to spot because a failure overnight or when one isn't around can mean the system shuts down, cools off and is thus rebootable again the next morning. All the admin sees is a system that's off for no readily apparent reason the next morning. The solution is to, for example, move the system somewhere close at hand so that it can be monitored, or write a script which tests whether the system is active every few seconds, logging the time of each successful test - if the system goes down, the admin is notified in some way (eg. audio sound file played) and the admin can then quickly check the machine - if the power supply area feels overly hot, then that is the likely suspect, especially if an off/on mains switch toggle doesn't turn the system back on (power supplies often have circuitry which will not allow power-on if the unit is still too hot). If the admin wasn't available at the time, then the logged results can show when the system failed.

All SGIs (and UNIX systems in general) include a suite of hardware and software diagnostics tests as part of the OS. IRIX contains a set of tests for checking the mouse, keyboard, monitor, audio ports, digital camera and other basic hardware features.

Thankfully, for just about any hardware failure, hardware support contracts cover repairs and/or replacements very effectively for UNIX systems. It's worth noting that although the Computing Department has a 5-day support contract with SGI, all problems I've encountered so far have been dealt either on the same day or early next morning by a visiting support engineer (ie. they arrived much earlier than they legally had to). Since November 1995 when I took charge of the Ve24 network, the hardware problems I've encountered have been:

- 2 failed disks
- 1 replaced power supply
- 2 failed memory SIMMs (1 failed SIMM from two different machines)
- 1 replaced keyboard (user damage)
- 1 failed monitor
- 1 suspect motherboard (replaced just in case)

- 1 suspect video card (replaced just in case)
- 1 problematic 3rd-party disk (incorrect firmware, returned to supplier and corrected with up-to-date firmware; now operating ok)
- 1 suspect hub (unknown problem; replaced just in case)

Given that the atmosphere in Ve24 is unfiltered, often humid air, and the fact that many of the components in the Indys in Ve24 have been repeatedly swapped around to create different configurations at different times, such a small number of failures is an excellent record after nearly 4 years of use.

It is likely that dirty air (dust, humidity, corrosive gases) was largely responsible for the disk, power supply and memory failures - perhaps some of the others too. A build up of dust can combine with airborne moisture to produce corrosive chemicals which can short-circuit delicate components.

To put the above list another way: 14 out of the 18 Indys have been running non-stop for 3.5 years without a single hardware failure of any kind, despite being housed in an area without filtered air or temperature control. This is very impressive and is quite typical of UNIX hardware platforms.

Installing systems with proper air filtering and temperature control can be costly, but the benefit may be a much reduced chance of hardware failure - this could be important for sites with many more systems and a greater level of overall system usage (eg. 9 to 5 for most machines).

Some companies go to great lengths to minimise the possibility of hardware failure. For example, MPC [2] has an Origin200 render farm for rendering movie animation frames. The render farm consists of 50 Origin200 servers, each with 2 R10000 CPUs, ie. 100 CPUs in total. The system is housed in a dedicated room with properly filtered air and temperature control. Almost certainly as a result of this high-quality setup, MPC has never had a single hardware failure of any kind in nearly 3 years of operation. Further, MPC has not experienced a single OS failure over the same period either, even though the system operates 24hours/day.

This kind of setup is common amongst companies which have time-critical tasks to perform, eg. oil companies with computational models that can take *six months* to complete - such organisations cannot afford to have failures (the problem would likely have to be restarted from scratch, or at least delayed), so it’s worth spending money on air filters, etc.

If one does not have filtered air, then the very least one should do is keep the systems clean inside and out, performing system cleaning on a regular basis.

At present, my current policy is to thoroughly clean the Indys twice a year: every machine is stripped right down to the bare chassis; every component is individually cleaned with appropriate cleaning solutions, cloths, air-dusters, etc. (this includes removing every single key from all the keyboards and mass-cleaning them with a bucket of hot water and detergent! And cleaning the keyboard bases inside and out too). Aside from these major bi-annual cleanings, simple regular cleaning is performed on a weekly or monthly basis: removing dirt from the mice (inside especially), screen, chassis/monitor surface, cables and so on; cleaning the desks; opening each system and blowing away internal dust using a can of compressed filtered air, etc.

Without a doubt, this process greatly lengthens the life-span of the systems’ hardware components, and users benefit too from a cleaner working environment - many new students each autumn often

think the machines must be new because they look so clean.

Hardware failures do and will occur on any system whether it's a UNIX platform or not. An admin can use information from online sources, combined with a knowledge of relevant system test tools such as fx and ping, to determine the nature of hardware failures and take corrective action (contacting vendor support if necessary); such a strategy may include setting up automated hardware tests using regularly-executed scripts.

Another obvious source of extensive information about any UNIX platform is the Internet. Hundreds of existing users, including company employees, write web pages [3] or USENET posts describing their admin experiences and how to deal with typical problems.

Suspicious/Illegal Activity.

Users inevitably get up to mischief on occasion, or external agencies may attempt to hack the system. Types of activity could include:

- users downloading illegal or prohibited material, either with respect to national/local laws or internal company policy,
- accessing of prohibited sites, eg. warez software piracy sites,
- mail spamming and other abuses of Internet services,
- attacks by hackers,
- misuse/abuse of system services internally.

There are other possibilities, but these are the main areas. This lecture is an introduction to security and monitoring issues. A more in-depth discussion is given in the last lecture.

As an admin who is given the task of supposedly preventing and/or detecting illegal activities, the first thing which comes to mind is the use of various file-searching methods to locate suspect material, eg. searching every user's netscape bookmarks file for particular keywords. However, this approach can pose legal problems.

Some countries have data protection and/or privacy laws [4] which may prohibit one from arbitrarily searching users' files. Searches of this type are the equivalent of a police force tapping all the phones in an entire street and recording every single conversation just on the off-chance that they might record something interesting; such methods are sometimes referred to as 'fishing' and could be against the law. So, for example, the following command might be illegal:

```
find /home/students -name "*" -print > list
grep sex list > suspected
grep warez list >> suspected
grep xxx list >> suspected
grep pics list >> suspected
grep mpg list >> suspected
grep jpg list >> suspected
grep gif list >> suspected
grep sites list >> suspected
```

As a means of finding possible policy violations, the above script would be very effective, but it's definitely a form of fishing (even the very first line).

Now consider the following:

```
find /home/students -name "bookmarks.html" -print -exec grep playboy {} \;
```

This command will effectively locate any Netscape bookmarks file which contains a possible link to the PlayBoy web site. Such a command is clearly looking for fairly specific content in a very specific file in each user's .netscape directory; further, it is probably accessing a user's account space without her or his permission (this opens the debate on whether 'root' even needs a user's permission since root actually owns all files anyway - more on this below).

The whole topic of computer file access is a grey area. For example, might the following command also be illegal?

```
find . -name "*.jpg" -print > results && grep sex results
```

A user's lawyer could argue that it's clearly looking for any JPEG image file that is likely to be of an explicit nature. On the other hand, an admin's lawyer could claim the search was actually looking for any images relating to tourism in Sussex county, or musical sextets, or adverts for local unisex hair salons, and just accidentally happened to be in a directory above /home/students when the command was executed (the find would eventually reach /home/students). Obviously a setting for a messy court-room battle.

But even ignoring actions taken by an admin using commands like find, what about data backups? An extremely common practice on any kind of computer system is to backup user data to a media such as DAT on a regular basis - but isn't this accessing user files without permission? But hang on, on UNIX systems, the root user is effectively the absolute owner of any file, eg. suppose a file called 'database' in /tmp, owned by an ordinary user, contained some confidential data; if the the admin (logged in as root) then did this:

```
cat /tmp/database
```

the contents of the database file would indeed be displayed.

Thus, since root basically owns all files anyway by default, surely a backup procedure is just the root user archiving files it already owns? If so, does one instead have to create some abstract concept of ownership in order to offer users a concrete concept of what data privacy actually is? Who decides? Nations which run their legal systems using case-law will find these issues very difficult to clarify, eg. the UK's Data Protection Act is known to be 'weak'.

Until such arguments are settled and better laws created, it is best for an admin to err on the side of caution. For example, if an admin wishes to have some kind of regular search conducted, the existence of the search should be included as part of stated company policy, and enshrined into any legal documents which users must sign before they begin using the system, ie. if a user signs the policy document, then the user has agreed to the actions described in that document. Even then, such clauses may not be legally binding. An admin could also setup some form of login script which would require users to agree to a sytsem usage policy before they were fully logged-in.

However, these problems won't go away, partly because of the specifics of how some modern Internet services such as the web are implemented. For example, a user could access a site which automatically forces the pop-up of a Netscape window which is directed to access a prohibited site;

inline images from the new site will then be present in the user's Netscape cache directory in their home account area even though they haven't specifically tried to download anything. Are they legally liable? Do such files even count as personal data? And if the site has its own proxy server, then the images will also be in the server's proxy cache - are those responsible for the server also liable? Nobody knows. Legal arguments on the nature of cache directories and other file system details have not yet been resolved. Clearly, there is a limit to how far one can go in terms of prevention simply because of the way computing technologies work.

Thus, the best thing to do is to focus efforts on information that does not reside inside user accounts. The most obvious place is the system log file, /var/adm/SYSLOG. This file will show all the ftp and telnet sites which users have been accessing; if one of these sites is a prohibited place, then that is sufficient evidence to take action.

The next most useful data resource to keep an eye on is the web server log(s). The web logs record every single access by all users to the WWW. Users have their own record of their accesses in the form of a history file, hidden in their home directory inside the .netscape directory (or other browser); but the web logs are outside their accounts and so can be probably be freely examined, searched, processed, etc. by an admin without having to worry about legal issues. Even here though, there may be legal issues, eg. log data often includes user IDs which can be used to identify specific individuals and their actions - does a user have a legal right to have such data kept private? Only a professional lawyer in the field would know the correct answer.

Note: the amount of detail placed into a log file can be changed to suit the type of logging required. If a service offers different levels of logging, then the appropriate online documentation will explain how to alter the settings.

Blocking Sites.

If an admin does not want users to be able to access a particular site, then that site can be added to a list of 'blocked' sites by using the appropriate option in the web server software concerned, eg. Netscape Enterprise Server, CERN web server, Apache web server, etc. Even this may pose legal problems if a country has any form of freedom-of-speech legislation though (non-existent in the UK at present, so blocking sites should be legally OK in the UK).

However, blocking sites can become somewhat cumbersome because there are thousands of web sites which an admin could theoretically have to deal with - once the list becomes quite large, web server performance decreases as every access has to have its target URL checked against the list of banned sites. So, if an admin does choose to use such a policy, it is best to only add sites when necessary, and to construct some kind of checking system so that if no attempt is made to access a blocked site after a duration of, say, two weeks (whatever), then that site is removed from the list of blocked sites. In the long term, such a policy should help to keep the list to a reasonably manageable size. Even so, just the act of checking the web logs and adding sites to the list could become a costly time-consuming process (time = money = wages).

One can also use packet filtering systems such as hardware routers or software daemons like ipfilterd which can accept, reject, or reject-and-log incoming packets based on source/destination IP address, host name, network interface, port number, or any combination of these. Note that daemons such as ipfilterd may require the presence of a fast CPU if the overhead from a busy site is to be properly supported. The ipfilterd system is discussed in detail on Day 3.

System Temporary Directories.

An admin should keep a regular eye on the contents of temporary directories on all systems, ie. /tmp and /var/tmp. Users may download material and leave the material lying around for anyone to see. Thus, a suspicious file can theoretically be traced to its owner via the user ID and group ID of the file. I say theoretically because, as explained elsewhere, it is possible for a user X to download a file (eg. by ftp so as to avoid the web logs, or by telnet using a shell on a remote system) and then 'hand over' ownership of the file to someone else (say user Y) using the chgrp and chown commands, making it look as though a different user is responsible for the file. In that sense, files found outside a user's home directory could not normally be used as evidence, though they would at least alert the admin to the fact that suspect activities may be occurring, permitting a refocusing of monitoring efforts, etc.

However, one way in which it could be possible to reinforce such evidence is by being able to show that user Y was not logged onto the system at the time when the file in question was created (this information can be gleaned from the system's own local /var/adm/SYSLOG file, and the file's creation time and date).

Unfortunately, both users could have been logged onto the same system at the time of the file's creation. Thus, though a possibility, the extra information may not help. Except in one case: video evidence. If one can show by security camera recordings that user X did indeed login 'on console' (ie. at the actual physical keyboard) then that can be tied in with SYSLOG data plus file creation times, irrespective of what user Y was doing at the time.

Certainly, if someone wished to frame a user, it would not be difficult to cause a considerable amount of trouble for that user with just a little thought on how to access files, where to put them, changing ownership, etc.

In reality, many admins probably just do what they like in terms of searching for files, examining users' areas, etc. This is because there is no way to prove someone has attempted to search a particular part of the file system - UNIX doesn't keep any permanent record of executed commands.

Ironically, the IRIX GUI environment *does* keep a record of any file-related actions taken with the GUI system (icons, file manager windows, directory views, etc.) but the log file with this information is kept inside the user's .desktop- directory and thus may be legally out of bounds.

File Access Permissions.

Recall the concept of file access permissions for files. If a user has a directory or file with its permissions set so that another ordinary user can read it (ie. not just root, who can access anything by default anyway), does the fact that the file is globally readable mean the user has by default given permission for anyone else to read the file? If one says no, then that would mean it is illegal to read any user's own public_html web area! If one says yes, and a legal body confirmed this for the admin, then that would at least enable the admin to examine any directory or file that had the groups and others permissions set to a minimum of read-only (read and executable for directories).

The find command has an option called -perm which allows one to search for files with permissions matching a given mode. If nothing else, such an ability would catch out careless users since most users are not aware that their account has hidden directories such as .netscape. An admin ought to

make users aware of security issues beforehand though.

Backup Media.

Can an admin search the data residing on backup media? (DAT, CDR, ZIP, DLT, etc.) After all, the data is no longer inside the normal home account area. In my opinion yes, since root owns all files anyway (though I've never done such a search), but others might disagree. For that matter, consider the tar commands commonly used to perform backups: a full backup accesses every file on the file system by default (ie. including all users' files, whatever the permissions may be), so are backups a problem area?

Yet again, one can easily see how legal grey areas emerge concerning the use of computing technologies.

Conclusion.

Until the law is made clearer and brought up-to-date (unlikely) the best an admin can do is consult any available internal legal team, deciding policy based on any advice given.

References:

1. "Ethernet Collisions on Silicon Graphics Systems", SGI Pipeline magazine (support info bulletin), July/August 1998 (NB: URL not accessible to those without a software support contract):

   http://support.sgi.com/surfzone/content/pipeline/html/19980404EthernetCollisions.html

2. The Moving Picture Company, Soho Square, London. Responsible for some or all of the special effects in Daylight, The English Patient, Goldeneye, The Borrowers, and many other feature films, music videos, adverts, etc. Hardware used: several dozen Octane workstations, many Onyx2 graphics supercomputers, a 6.4TB Ampex disk rack with real-time Philips cinescan film-to-digital-video converter (cinema resolution 70mm uncompressed video converter; 250K's worth), Challenge L / Discrete Logic video server, a number of O2s, various older SGI models such as Onyx RealityEngine2, Indigo2, Personal IRIS, etc., some high-end Apple Macs and a great deal of dedicated video editing systems and VCRs, supported by a multi-gigabit network. I saw one NT system which the admin said nobody used.

3. **The SGI Tech/Advice Centre: Holland #1: http://www.futuretech.vuurwerk.nl/**
   Worldwide Mirror Sites: Holland #2: http://sgi.webguide.nl/
   Holland #3: http://sgi.webcity.nl/
   USA: http://sgi.cartsys.net/
   Canada: http://sgi-tech.unixology.com/

4. The Data Protection Act 1984, 1998. Her Majesty's Stationary Office (HMSO): http://www.hmso.gov.uk/acts/acts1984/1984035.htm