Root and Administrator Tasks: Process Management

- UNIX “root” privileged accounts (Chapter 3 in USAH)

- If a process has a userid of 0, this removes most restrictions such as permission checks from processes. These are generally called “root” processes; root processes can, `tattoueba`:

Root and Administrator Tasks: Process Management

1. Mount and unmount file systems – however, that's not quite true on Linux machines; the mount(8) program is now being suid to root on some (many) distributions, and if the keyword `user` is specified in a mount point defined in `/etc/fstab`, then the mount program will allow a user to mount or unmount that specific filesystem.

2. Root processes can set a process’s filesystem root to a subdirectory of a filesystem via chroot()

Root and Administrator Tasks: Process Management

3. Create device files (/dev, **mknod**)
4. Set the system clock
5. Can access any local file

Root and Administrator Tasks: Process Management

6. Change file ownership
7. Raise resource limits (datasize, stacksize, coresize) – no other userid than 0 can do so; other userids can only lower resources limits
8. Lower nice values (raising priority)

Root and Administrator Tasks: Process Management

9. Change system's hostname
10. Run `halt`, `shutdown`, `telinit`
11. Manage print subsystems

Root and Administrator Limitations: Process Management

12. Many other programs check to see if the current process is running under uid 0 (the code to check for this usually looks something like “if geteuid() == 0) ...”)

Root and Administrator Limitations: Process Management

☞ What limitations and restrictions are there to such root processes?

1. UNIX suffers from “userid 0 has all powers”, so root account (and its password(s)) is focus of security breakins.

Root and Administrator Limitations: Process Management

2. Usually root on another machine won't (and shouldn't!) trust you
3. Should be careful that when acting as "root" that you know your $PATH. Beware of file paths in $PATH, especially the current working directory (".", aka as "pwd" or "cwd").

How to become "root"?

Generally, people use something along the way of **sudo**, **su**, or **login**.

1. Login as "root", if allowed in `/etc/ttytab` (BSD), `/etc/default/login` (Solaris) or `/etc/securetty` (RedHat Linux). Chapter 7 in USAH contains more (and old) information about hard-wired terminals and ttytab/gettytab/securetty.

How to become root? login, su, sudo

2. Execute the su command

   ☞ "su" = Substitute User

How to become root? login, su, sudo

- "su" with minus flag ("su - fc5") invokes a "login" session
- Good idea to "su - root". The advantages of a "login" shell:
  - Paths are those of root, not your current processes

How to become root? login, su, sudo

- Set up items such as “safe” aliases for dangerous programs such as
  - rm → rm -i
  - cp → cp -i
  - mv → mv -i

sudo: pseudo su, or how to set up safer su

☞ Previously, was often not a part of a vendor-supplied UNIX (RedHat and CentOS do include it – and Ubuntu tries to make it de rigueur)

☞ Allows a class of users to execute a set of commands with root privileges (flexible enough though to do more)

☞ Logs the use of the "sudo" command (but does not log the commands executed by the shells that are started by **sudo** !)

- ☞ Does raise some vulnerabilities (yet-another setuid program)

sudo: pseudo su, or how to set up safer su

```
# sudoers file.
#
# This file MUST be edited with the ’visudo’ command as root.
#
# See the sudoers man page for the details on how to write a sudoers file.
#

# Host alias specification

# User alias specification
```

sudo: pseudo su, or how to set up safer su

```
# Cmnd alias specification

# Defaults specification

# User privilege specification
root        ALL=(ALL) ALL

# Uncomment to allow people in group wheel to run all commands
# %wheel        ALL=(ALL)        ALL
```

sudo: pseudo su, or how to set up safer su

```
# Same thing without a password
# %wheel        ALL=(ALL)        NOPASSWD: ALL

# Samples
# %users  ALL=/sbin/mount /cdrom,/sbin/umount /cdrom
# %users  localhost=/sbin/shutdown -h now

user1        monet=/usr/local/bin/suroot, /bin/su
```

System Load Average

"load average" == average size of ready queue over sample period

☞ Shows the 1, 5, and 15 minute load averages

☞ Can see with **w**, **uptime**, or **top**

☞ What's a reasonable load average? → depends on the machine and the type of jobs running

Idle Time

- Percentage of time the system is idle
- Can see with “iostat -c 1”, “top”, or “vmstat 1”
- What do you want this number to be? (again, it depends on machine’s raison d’etre)

Idle Time

```
[root@smtpin MailScanner]# iostat -c 1
Linux 2.6.9-55.0.2.ELsmp (smtpin.cs.fsu.edu)    06/02/2008

avg-cpu:  %user   %nice    %sys %iowait   %idle
          12.72    0.00    1.66    1.33   84.30

avg-cpu:  %user   %nice    %sys %iowait   %idle
          30.00    0.00    4.50    4.00   61.50

avg-cpu:  %user   %nice    %sys %iowait   %idle
          14.46    0.00    1.75    2.00   81.80
```

Process Monitoring: ps

☞ **ps** comes from process status; page 53 in USAH has comprehensive information

☞ Shows a window into process table via the filesystem – remember, **ps** these days generally is just walk through the `/proc` pseudo-filesystem

Process Monitoring: ps

☞ Rich command options set; unfortunately, there are different options depending on whether the OS is BSD or System V based.

☞ The BSD “ps” has these columns (which is generally true for the other “ps” variations):

1. Process state. First letter indicates the runnability of the process:

Process Monitoring: ps

- R - Runnable processes.
- T - Stopped processes.
- P - Processes in page wait.
- D - Processes in non-interruptable waits;
- S - Processes sleeping less than about 20 seconds.

Process Monitoring: ps

- I - Processes sleeping more than 20 seconds
- Z - zombie (process with NO resources other than a proc slot)

2. Swapped? Second letter indicates whether a process is swapped out;
    - blank - loaded in memory

Process Monitoring: ps

- W - Process is swapped out.
- '>' - Process has specified a soft limit on memory (imposed by the “limit” command)

Process Monitoring: ps

3. Niced? Third letter indicates whether a process is running with altered CPU scheduling priority (nice, renice)
   - blank - normal
   - N - The process priority is reduced

Process Monitoring: ps

- '<' - The process priority has been raised artificially.

4. You can use the "renice" command to change a process' nice value:

Process Monitoring: ps

```
renice +19 PID    ##  lowest priority (nice)
renice -19 PID    ## highest priority (not nice)
```

Example “ps” output from long, long ago

```
USER       PID %CPU %MEM   SZ  RSS TT STAT START   TIME COMMAND
-----------------------------------------------------------------------
kuncick   7467 40.1 1.3   124  364 pc D <  09:32   0:15 find / -name foo
kuncick   7419 16.6 1.3   124  364 pc D N  09:28   0:11 find ...
kuncick   7529 39.4 1.2   100  320 pc R    09:35   0:01 find / -name foo
kuncick   7528 35.8 1.2   112  324 pc R    09:35   0:15 find / -name foo
root         1  0.0 0.0    52    0 ?  IW   Dec 18  0:24 /sbin/init -
root         2  0.0 0.0     0    0 ?  D    Dec 18  0:08 pagedaemon
root        75  0.0 0.4    48  108 ?  S    Dec 18 12:09 in.routed
bynum     7328  0.0 0.1    48   24 pb S    09:24   0:00 rlogin
```

Example “ps” output from the more recent past

```
F   UID   PID  PPID PRI  NI   VSZ  RSS WCHAN  STAT TTY        TIME COMMAND
--------------------------------------------------------------------------
5     0  1778     1  24   0  1528  512 -      S    ?          0:00 /usr/sbin/apmd -p
 10 -w 5 -W -P /etc/sysconfig/apm-scripts/apmscript
5     0  1866     1  15   0  2144  880 -      S    ?          0:00 xinetd -stayalive
 -pidfile /var/run/xinetd.pid
1    51  1902     1  15   0  5992 2284 -      S    ?          0:00 sendmail: Queue
 runner@00:01:00 for /var/spool/clientmqueue
1    49  1949     1  25   0  5296 4012 -      S    ?          0:00 /usr/bin/jserver
```

Example “ps” output from the more recent past

```
4   500  2089  2068  15   0 18364 8948 schedu S    ?          0:00 /usr/bin/gnome-s
SSH_AGENT_PID=2140 HOSTNAME=sophie.cs.fsu.edu TERM=dumb SHELL=/bin/bash HISTSIZE=10
QTDIR=/usr/lib/qt-3.1 USER=langley LS_COLORS= SSH_AUTH_SOCK=/tmp/ssh-sQIL2089/agent
PATH=/usr/kerberos/bin:/usr/local/bin:/usr/bin:/bin:/usr/X11R6/bin:/home/langley/bi
MAIL=/var/spool/mail/langley PWD=/home/langley INPUTRC=/etc/inputrc XMODIFIERS=@im=
LANG=en_US.UTF-8 LAMHELPFILE=/etc/lam/lam-helpfile GDMSESSION=Default
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass SHLVL=1 HOME=/home/langley
LOGNAME=langley LESSOPEN=|/usr/bin/lesspipe.sh \%s DISPLAY=:0 G_BROKEN_FILENAMES=1
XAUTHORITY=/home/langley/.Xauthority
0     0  7601  7365  15   0 11320 5528 -      S    pts/1      0:00 emacs -nw
```

Many more features to “ps”

☞ Note that “ps” demonstrates the Heisenberg effect (observing the process table affects the process table, which is also very true of top, especially if you set it to rapidly refresh)

☞ Some favorite “ps” variations:

   ➻ `ps -ef`(System V)

Many more features to "ps"

- `ps -elf` (System V)
- `ps axuw | grep username` (BSD)
- `ps alxwww` (BSD)
- `ps alxwwwe` (BSD, show environmental variables)

“ps” can also act a bit like “pstree”

```
$ ps f
  PID TTY      STAT   TIME COMMAND
21915 pts/1    Ss     0:00 bash
22976 pts/1    S+     0:05  \_ emacs -nw 05-rootadmins.tex
27844 pts/2    Ss     0:00      \_ /bin/bash --noediting -i
17182 pts/2    R+     0:00          \_ ps f
18985 pts/0    Ss     0:00 bash
19153 pts/0    S+     0:00  \_ ssh langley@diablo.cs.fsu.edu
```

Making “ps” ultra-flexible

If you need to tailor “ps” output to arbitrary columns, you can use the o option to specify exactly which columns you would like to display, and `k` option to specify order:

```
$ ps k pid o pid,comm
  PID COMMAND
18985 bash
19153 ssh
21915 bash
22976 emacs
24527 ps
27844 bash
```

```
$ ps k comm o pid,comm
  PID COMMAND
18985 bash
21915 bash
27844 bash
22976 emacs
24729 ps
19153 ssh
```

Other process viewing tools

- **top**
  - Dynamically shows processes, idle time, memory usage, and load averages

Other process viewing tools

- ☞ “pstree” – nice filter that shows family hierarchy of processes

“top” example

```
[root@smtpin MailScanner]# top -b -n 1
top - 07:09:25 up 42 days, 20:18,  2 users,  load average: 1.34, 1.56, 1.79
Tasks: 166 total,   1 running, 165 sleeping,   0 stopped,   0 zombie
Cpu(s): 12.7% us,  1.6% sy,  0.0% ni, 84.3% id,  1.3% wa,  0.0% hi,  0.0% si
Mem:   4149124k total,  3794188k used,   354936k free,   202476k buffers
Swap:  4192956k total,     4960k used,  4187996k free,  2677940k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5675 root      15   0 68888  58m 2788 S    4  1.4   0:03.08 MailScanner
 5686 root      15   0  3520  900  676 R    2  0.0   0:00.01 top
26663 root      16   0 38616  29m 3464 S    2  0.7   0:29.90 MailScanner
    1 root      16   0  1980  548  468 S    0  0.0   0:45.81 init
```

“pstree” example

```
init-+-MailScanner
     |-MailScanner-+-16*[MailScanner]
     |             `-4*[MailScanner---MailScanner]
     +-acpid
     +-atd
     +-clamd
     +-crond
     +-cups-config-dae
     +-cupsd
```

Remedies for sluggish system

☞ Quick Remedies

- ➳ System clogged by many identical jobs → restarting and limiting

Remedies for sluggish system

```
Example:

You log into a mail server and find a high load
average, and many, many sendmail processes running
on the machine. Doing an ‘‘mailq’’ reveals that
there are many, many undelivered messages.
```

Remedies for sluggish system

You can stop sendmail with something like:

```
/etc/init.d/sendmail stop
```

Remedies for sluggish system

or, if you have `killall`:

```
killall sendmail
```

(n.b. – despite the dangers of `killall`, this is actually safer in this instance because you really shouldn't use the `/etc/init.d/sendmail` script if you are running MailScanner, but MailScanner's `/etc/init.d/MailScanner` script by default doesn't have any options to just stop incoming sendmail — but ironically it does have options to start both incoming and outgoing sendmail since slightly special options are needed. MailScanner works by having separate incoming and outgoing queues, and sendmail must be started in a manner that respects that setup.)

Remedies for sluggish system

However, that only stops sendmail.
You now need to clear some of the queue. One way to do this is with the ‘‘-qf’’ option:

```
/usr/sbin/sendmail -qf -v
```

Remedies for sluggish system

This leaves this new sendmail process running in the foreground just as a queue handler, running ‘‘verbosely’’ so that you can see exactly what it is doing, and how long it is taking.

Remedies for sluggish system

Starting a few of these foreground queue handlers will allow the queues to clear more quickly than if you just turn on the regular incoming email and a single background queue handler.

Remedies for sluggish system

- One process has been running for a long time and is accumulating a lot of processor time (for instance, **top** shows it at the "top" of the list consistently) → typically, this can be cleaned up by simply killing that one process. (This is often the result of a controlling terminal having gone awry.)

Remedies for sluggish system

☞ Long-term Remedies

- ➳ Involve more performance analysis
- ➳ May need faster CPU, more memory or faster I/O