



Linux Administration

Controlling Processes

Lecturer:

Di Dio Lavore, Galtarossa

Politecnico di Milano

www.polimi.it

© 2003 - Di Dio Lavore, Galtarossa

Summary



-
- Introduction
 - The INIT process
 - Signals
 - **kill** command
 - **nice** and **renice** commands
 - **ps** command
 - ▶ Examples
 - **pstree** command
 - **top** command
 - ▶ Example
 - Explanation of **ps** and **top** output
 - X-based Interfaces

Introduction



- A process is the abstraction used by Linux to represent a running program
- A process consists of an address space and a set of data structure within the kernel

- The kernel's internal data structures record various piece of information about each process:
 - ▶ Process's address space map
 - ▶ Current status (sleeping,runnable...)
 - ▶ Execution priority
 - ▶ Resources used
 - ▶ The owner
 - ▶ ...

Introduction (2)



- Many of the parameters associated with a process directly affect its execution
- The parameters that are most interesting from a system administartor's point of view are:
 - ▶ PID - Process ID number
 - ▶ PPID - Parent PID
 - ▶ UID and EUID - real and Effective User ID
 - ▶ GID and EGID - real and Effective Group ID
 - ▶ Nice value
 - ▶ Control terminal

NOTE:

An existing process must clone itself to create a new process
The clone can then exchange the program it is running for a different one

The INIT process



- When the system boots the kernel autonomously creates a **INIT** process
- **INIT** is always process number 1
- All processes are descendants of **INIT**
- **INIT** plays important role in process management:
 - ▶ *When a process completes, it calls a routine named **_exit...***
 - ▶ *If the parent of a process dies first, it accepts the orphaned process...*

Signals



- Signals are process-level interrupt requests
- When a signal is received, one of two things can happen:
 - ▶ *If the receiving process has designated a handler routine for that signal, the handler is called (“catch” the signal)*
 - ▶ *Otherwise, the kernel takes some default action*
- Many signals terminate the process
- To prevent signals from arriving, programs can request that they be either ignored or blocked:
 - ▶ *Ignored* = discarded
 - ▶ *Blocked* = queued for delivery

Signals (2)



Signals that every administrator should know General Process Signals

HUP (1)	Hangup <i>can be caught, ignored or blocked</i>
INT (2)	Interrupt <i>can be caught, ignored or blocked</i>
QUIT (3)	Quit <i>can be caught, ignored or blocked</i>
KILL (9)	Destroy a process <i>cannot be caught, ignored or blocked</i>
TERM (15)	Software termination <i>can be caught, ignored or blocked</i>

The default action for all these signals is abnormal process termination

Signals (3)



Job Control Process Signals

CONT	Continue after STOP <i>process cannot ignore or block this</i>
STOP	Suspend a process <i>cannot be caught, ignored or blocked</i>
TSTP	Keyboard stop <i>“soft” version of STOP (can be caught, ignored or blocked)</i>

The default action for these signals is suspending process execution, except for the CONT signal which defaults to resuming process execution

kill command



- As its name implies, the **kill** command is most often used to terminate a process
- It can send any signal, but by default it sends a **TERM** (this does not guarantee that the target process will die)
- It can be used by normal users on their processes or by *root* on any process
- The syntax is : `kill [-signal] pid`
 - ▶ `signal` is the number or symbolic name of the signal
 - ▶ `pid` is the number of the target process
- A `pid` of `-1` broadcasts the signal to all process except **INIT**

NOTE: Exists the **killall** command that kills processes by name

nice and renice commands



- The “niceness” of a process is a numeric hint to the kernel
- The strange name is derived from the fact that it determines how nice you are going to be to other users of the system
- A high nice value means a low priority for the process
- The range of allowable niceness values is `-20` to `+19`
- A newly created process inherits the nice value of its parent process
- The owner of the process can increase its nice value but cannot lower it
- *root* has complete freedom in setting nice values
- A process’s nice value can be set at the time of creation with the **nice** command and can be adjusted during execution with the **renice** command



- **ps** is the system administrator's main tool for monitoring processes
- It shows the PID, UID, priority and control terminal of process
- It also gives other information (memory usage, CPU time consumed, current status...)
- **ps** is used with two useful set of arguments: **aux** or **lax**
- **ps** offers only a one-time snapshot of the system

ps aux (example)



```
# ps aux

USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.3  0.0  1288   484 ?        S    19:18   0:05 init
root         2  0.0  0.0      0     0 ?        SW   19:18   0:00 [keventd]
root         3  0.0  0.0      0     0 ?        SW   19:18   0:00 [kapmd]
root         4  0.0  0.0      0     0 ?        SWN  19:18   0:00 [ksoftirqd_CPU0]
root         5  0.0  0.0      0     0 ?        SW   19:18   0:00 [kswapd]
root         6  0.0  0.0      0     0 ?        SW   19:18   0:00 [bdflush]
root         7  0.0  0.0      0     0 ?        SW   19:18   0:00 [kupdated]
root         8  0.0  0.0      0     0 ?        SW<  19:18   0:00 [mdrecoveryd]
root        12  0.0  0.0      0     0 ?        SW   19:18   0:00 [kjournald]
root       157  0.0  0.1   1700   960 ?        S    19:18   0:00 devfsd /dev
root       253  0.0  0.0      0     0 ?        SW   19:18   0:00 [khubd]
root       459  0.0  0.0      0     0 ?        SW   19:18   0:00 [kjournald]
rpc        731  0.0  0.1   1412   524 ?        S    19:18   0:00 portmap
root       747  0.0  0.1   1356   584 ?        S    19:18   0:00 syslogd -m 0
root       755  0.0  0.2   2020  1192 ?        S    19:18   0:00 klogd -2
root       790  0.0  0.0   1336   496 ?        S    19:18   0:00 gpm -t imps2 -m /
xfs        897  0.0  0.8   5792  4484 ?        S    19:18   0:00 xfs -port -1 -dae
daemon     942  0.0  0.0   1312   504 ?        S    19:18   0:00 /usr/sbin/atd
root       945  0.0  0.1   2292   664 ?        S    19:18   0:00 /usr/bin/kdm -nod
root       968  0.0  0.0   1500   492 ?        S    19:18   0:00 saslauthd -a pam
```

```
a - select all processes on a terminal, including those of other users
u - display user-oriented format
x - select processes without controlling ttys
```

ps lax (example)



```
# ps lax

  F  UID  PID  PPID  PRI  NI   VSZ  RSS  WCHAN  STAT  TTY      TIME COMMAND
100  0     1     0    8    0  1288  484  do_sel  S     ?        0:05 init
040  0     2     1   10    0    0     0  contex  SW    ?        0:00 [keventd]
040  0     3     1    9    0    0     0  apm_ma  SW    ?        0:00 [kapmd]
040  0     4     1   19   19    0     0  ksofti SWN   ?        0:00 [ksoftirqd_CPU0]
040  0     5     1    9    0    0     0  kswapd  SW    ?        0:00 [kswapd]
040  0     6     1    9    0    0     0  bdfly   SW    ?        0:00 [bdfly]
040  0     7     1    9    0    0     0  kupdat  SW    ?        0:00 [kupdated]
040  0     8     1   -1  -20    0     0  md_thr  SW<   ?        0:00 [mdrecoveryd]
040  0    12     1    9    0    0     0  end     SW    ?        0:00 [kjournald]
140  0   157     1    8    0  1700  960  devfsd  S     ?        0:00 devfsd /dev
040  0   253     1    9    0    0     0  end     SW    ?        0:00 [khubd]
040  0   459     1    9    0    0     0  end     SW    ?        0:00 [kjournald]
140  71   731     1    9    0  1412  524  do_pol  S     ?        0:00 portmap
040  0   747     1    9    0  1356  584  do_sel  S     ?        0:00 syslogd -m 0
140  0   755     1    9    0  2020  1192  do_sys  S     ?        0:00 klogd -2
140  0   790     1    9    0  1336  496  nanosl  S     ?        0:00 gpm -t imps2 -m /dev/psaux

l - long format
a - select all processes on a terminal, including those of other users
x - select processes without controlling ttys
```

ps ax (BSD syntax example)



```
# ps ax

  PID  TTY      STAT  TIME COMMAND
   1  ?        S      0:05 init
   2  ?        SW     0:00 [keventd]
   3  ?        SW     0:00 [kapmd]
   4  ?        SWN    0:00 [ksoftirqd_CPU0]
   5  ?        SW     0:00 [kswapd]
   6  ?        SW     0:00 [bdfly]
   7  ?        SW     0:00 [kupdated]
   8  ?        SW<    0:00 [mdrecoveryd]
  12  ?        SW     0:00 [kjournald]
 157  ?        S      0:00 devfsd /dev
 253  ?        SW     0:00 [khubd]
 459  ?        SW     0:00 [kjournald]
 734  ?        S      0:00 portmap
 750  ?        S      0:00 syslogd -m 0
 758  ?        S      0:00 klogd -2
 793  ?        S      0:00 gpm -t imps2 -m /dev/psaux
 900  ?        S      0:00 xfs -port -l -daemon -droppriv -user xfs
 945  ?        S      0:00 /usr/sbin/atd
 955  ?        S      0:00 /usr/bin/kdm -nodaemon
 970  ?        S      0:00 saslauthd -a pam -T
 989  ?        S<     0:04 /etc/X11/X -deferglyphs 16 -auth /var/run/xauth/A:0-k
 993  ?        S      0:00 xinetd -stayalive -reuse -pidfile /var/run/xinetd.pid
 994  ?        S      0:00 -:0
1045  ?        S      0:00 cupsd
1255  ?        S      0:00 crond
1281  ?        S      0:00 /usr/bin/perl /usr/share/webmin/miniserv.pl /etc/webm
1387  tty1    S      0:00 /sbin/mingetty tty1
```

pstree command



- **ps**tree shows running processes as a tree
- The tree is rooted at either PID or INIT if PID is omitted
- If a user name is specified, all process trees rooted at processes owned by that user are shown

- Example:

```
# pstree
init--atd
  |--bdfush
  |--cron
  |--cupsd
  |--devfsd
  |--gpm
  |--kalarmd
  |--kapmd
  |--9*[kdeinit]
  |--kdeinit--artsd
  |   |--5*[kdeinit]
  |   |--kdeinit---bash---pstree
  |   |--ksnapshot
  |--kdm--X
```

top command



- **top** command provides a regularly updated summary of active processes and their use of resources
- By default, the display is updated every 5 seconds
- The most active processes appear at the top
- It accepts input from the keyboard, so *root* can observe how some actions (signals or **renice**) affect the overall condition of the system
- *root* can run **top** with the **q** option to queue it up to the highest possible priority

top (example)



```
# top

7:40pm up 22 min, 3 users, load average: 0.02, 0.12, 0.09
64 processes: 62 sleeping, 2 running, 0 zombie, 0 stopped
CPU states: 5.1% user, 2.9% system, 0.0% nice, 91.8% idle
Mem: 516276K av, 179612K used, 336664K free, 0K shrd, 10272K buff
Swap: 506008K av, 0K used, 506008K free, 71268K cached

  PID USER      PRI  NI  SIZE  RSS SHARE STAT  %CPU %MEM    TIME COMMAND
  986 root        19 -10 88988 12M 2748 S <   4.7  2.4   0:34 X
 1797 matteo    18  0 87864 14M 12004 R    2.5  2.8   0:03 kdeinit
 1995 root        12  0  1032 1032   816 R    0.5  0.1   0:02 top
 1748 matteo    9  0 90188 16M 13816 S    0.1  3.3   0:01 kdeinit
    1 root         8  0   484  484   420 S    0.0  0.0   0:05 init
    2 root         9  0     0     0     0 SW    0.0  0.0   0:00 keventd
    3 root         9  0     0     0     0 SW    0.0  0.0   0:00 kapmd
    4 root        19 19     0     0     0 SWN   0.0  0.0   0:00 ksoftirqd_CPU0
    5 root         9  0     0     0     0 SW    0.0  0.0   0:00 kswapd
    6 root         9  0     0     0     0 SW    0.0  0.0   0:00 bdflush
    7 root         9  0     0     0     0 SW    0.0  0.0   0:00 kupdated
    8 root        -1 -20     0     0     0 SW<   0.0  0.0   0:00 mdrecoveryd
   12 root         9  0     0     0     0 SW    0.0  0.0   0:00 kjournald
  157 root         8  0   960  960   768 S    0.0  0.1   0:00 devfsd
  253 root         9  0     0     0     0 SW    0.0  0.0   0:00 khubd
  459 root         9  0     0     0     0 SW    0.0  0.0   0:00 kjournald
  731 rpc         9  0   524  524   444 S    0.0  0.1   0:00 portmap
  747 root         9  0   584  584   476 S    0.0  0.1   0:00 syslogd
  755 root         9  0  1192 1192   416 S    0.0  0.2   0:00 klogd
  790 root         9  0   496  496   432 S    0.0  0.0   0:00 gpm
  897 xfs         9  0 4484 4484  1028 S    0.0  0.8   0:00 xfs
  942 daemon       9  0   504  504   436 S    0.0  0.0   0:00 atd
```

Explanation of ps and top output



- **PID:** *The process ID of each task*
- **USER:** *The user name of the task's owner*
- **PRI:** *The priority of the task*
- **NI:** *The nice value of the task. Negative nice values are higher priority*
- **SIZE:** *The size of the task's code plus data plus stack space, in kilobytes*
- **RSS:** *The total amount of physical memory used by the task, in kilobytes*

Explanation of ps and top output (2)



- **SHARE:** *The amount of shared memory used by the task*
- **STAT:** *The state of the task is shown here (S for sleeping, R for running, Z for zombies...)*
- **%CPU:** *The task's share of the CPU time since the last screen update, expressed as a percentage of total CPU time per processor*
- **TIME:** *Total CPU time the task has used since it started*
- **%MEM:** *The task's share of the physical memory*
- **COMMAND:** *The task's command name*

Explanation of ps and top output (3)



- **VSZ:** *Virtual size of the process*
- **TTY:** *Control terminal ID*
- **UID:** *The user ID of the task's owner*
- **WCHAN:** *This shows the address or the name of the kernel function the task currently is sleeping in*
- **F:** *Flags (100 used super-user privileges, 040 forked but...)*
- **PPID:** *The parent process ID each task*
- **START:** *Time the process was started*

X-based Interfaces



General browsable tool: Webmin (<https://localhost:10000/>)

The screenshot shows two browser windows. The left window displays the 'Running Processes' page with a table of running processes. The right window shows the 'Process Information' page for PID 1893.

Process ID	CPU	Command
1777	3.0 %	kdeinit: konqueror --silent
1761	0.5 %	kdeinit: kicker
1893	0.5 %	kdeinit: kedit -caption KEd
1748	0.4 %	/usr/bin/artsd -F 10 -S 40
1756	0.4 %	kdeinit: kdesktop
1783	0.4 %	kdeinit: kio_http https /tmp
1800	0.4 %	kdeinit: kio_http https /tmp

Process Information

Command: kdeinit: kedit -caption KEdit -icon kedit.png -miniicon kedi ...

Process ID: 1893 Parent process: kdeinit: Running...

Owner: matteo CPU: 0.5 %

Size: 97088 kB Run time: 00:00:00

Nice level: 0 (Default) Change

Real group: matteo Process group ID: 1734

Group: 500 TTY: None

Real user: matteo

Buttons: Send Signal (HUP), Terminate Process, Kill Process, Files and Connections

Return to process list

root logged into Webmin 0.990 on localhost.localdomain (Mandrake Linux 9.0)

X-based Interfaces (2)



Specific KDE tool: KDE System Guard

The screenshot shows the 'Process Table' window in KDE System Guard. A warning dialog is displayed over the table, asking 'Do you want to kill the selected process?' with 'Yes' and 'No' buttons.

Name	PID	User%	System%	Nice	VmSize	VmRss	Login	Command
klipper	1756	0.00	0.00	0	96088	12500	matteo	kdeinit: klipper -icon
knotify	1742	0.00	0.00	0	99008	13592	matteo	kdeinit: knotify
ksmserver	1745	0.00	0.00	0	95096	11356	matteo	kdeinit: ksmserver --
kwrited	1760	0.00	0.00	0	95900	12316	matteo	kdeinit: kwrited
kdeinit	1721	0.00	0.00	0	19512	7804	matteo	kdeinit: Running...
kedit	2074	0.00	0.00	0	98160	16456	matteo	kdeinit: kedit -capto
kio_file	2109	0.00	0.00	0	19664	8404	matteo	kdeinit: kio_file file /
kio_thumbnail	1692	0.00	0.00	0	1692	1692	matteo	kdeinit: kio_thumbna
konsole	4964	0.00	0.00	0	4964	4964	matteo	kdeinit: konsole -ico
bash	1616	0.00	0.00	0	1616	1616	matteo	/bin/bash
su	1008	0.00	0.00	0	1008	1008	matteo	su
bash	1644	0.00	0.00	0	1644	1644	root	bash
kwin	3184	0.00	0.00	0	3184	3184	matteo	kdeinit: kwin -sessio
kdesu	1939	0.00	0.00	0	94708	12080	matteo	kdesu
su	1950	0.00	0.00	0	2264	1004	matteo	/bin/su
kdesu_stub	1953	0.00	0.00	0	1316	516	root	/usr/bin/kdesu_stub
konqueror	1956	0.00	0.00	0	111824	30164	root	konqueror
ksnapsot	2112	1.52	0.51	0	36444	14504	matteo	ksnapsot

69 Processes Memory: 206896 KB used, 309380 KB free Swap: 0 KB used, 506008 KB free



- “UNIX System Administration Handbook Third Edition” by Evi Nemeth, Garth Snyder, Scott Seebass, Trent R. Hein
- “Linux Administration Handbook” by Evi Nemeth, Garth Snyder, Trent R. Hein
- “The Linux System Administrator's Guide Version 0.7” by Lars Wirzenius, Joanna Oja, Stephen Stafford (<http://www.tldp.org/guides.html>)

System Commands



- `kill` - terminate a process
- `killall` - kill processes by name
- `nice` - run a program with modified scheduling priority
- `renice` - alter priority of running processes
- `ps` - report process status
- `pstree` - display a tree of processes
- `top` - display top CPU processes