LINUX FOUNDATION

TRAINING

Managing User Processes

A Linux Foundation Training Publication
www.training.linuxfoundation.org

Overview

Any time a command is run on a Linux system, a process is started. In the case of many administrative commands run from the command line, the process spawned by running the command is short lived; running only as long as it takes the command to execute.

Other commands may take longer to complete, or need to be run in an ongoing, scheduled, or automatic basis. Linux manages the priority given to these processes, the resources they consume, and tracks information about what processes are actually doing on the system.

An administrator needs to be monitor and affect running processes and how they are handled by the system.

Key Ideas

process: A running application or command.

job: A process started by the user from the command line.

priority: The right to claim system resources (CPU, memory) one process has relative to other processes.

niceness: A way of representing priority, on a scale from -20 to 20, with -20 being the highest priority, and most user started processes having a niceness of 0.

foreground: A process started by typing a command into the terminal is generally running in the foreground by default: you wait until it is finished before executing further commands. A process started in the terminal and running in the foreground can be stopped with the “Ctrl+z” key combination.

background: Background processes do not prevent you from starting additional processes. Commands can be executed in the background by appending an “&” to the end of the command.

jobs: The jobs command gives a numbered list of the user initiated commands that are running in the background

fg: The fg command returns a background process to the foreground. If multiple commands are running in the background, use the number given by the jobs command to specify which job you want foreground.

top: A command for checking process resource utilization. By default, displays processes using the most CPU, use “shift+m” to display by memory usage instead.

ps: The process snapshot command gives a list of running processes.

nice: The nice command is used to change the relative priority of a process when starting

the process.

renice: The renice command is used to change the relative priority of a process that is already running.

kill: The kill command stops processes. By default, kill sends a hangup signal to the process. Used with the -9 option, a process is force killed via the kernel directly, rather than by interaction with the process itself.

nohup: The nohup command can be used to start jobs that don't respond to hangup signals, i.e. commands that need to run after the user starting them has logged out.

Example Scenario

Use the commands in the key concepts section to interact with processes on your system.

Now Do It

1. Use the ps command with the aux option to display all running processes.
2. Use the ps command with the -U and -N options to display processes not running as root.
3. Use the nice command in conjunction with your package manager to run a system update with highest priority.
4. Use the watch command in conjunction with the df command to watch disk utilization.
5. Suspend the job started in the previous example.
6. Use the jobs command to list jobs you started.
7. Use the fg command to foreground the disk utilization command you previously suspended.
8. Suspend the disk utilization command again.
9. Use the watch command in conjunction with the df command to watch disk utilization again, but this time start it in the background.
10. Use the top command to see where the watch job you started ranks in terms of memory usage.
11. Use the ps command to find the process id of the watch job you started.
12. Use the kill command to end the watch job you started.

If you remember nothing else...

The lower the niceness, the higher the priority a process has. Kill -9 is a less graceful way of ending processes than kill by itself.

Answer Key

1. List all running proccess using ps aux

```
# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.8  0.2  19232  1492 ?        Ss   15:38   0:00 /sbin/init
root         2  0.0  0.0      0     0 ?        S    15:38   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    15:38   0:00 [migration/0]
root         4  0.0  0.0      0     0 ?        S    15:38   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S    15:38   0:00 [stopper/0]
root         6  0.0  0.0      0     0 ?        S    15:38   0:00 [watchdog/0]
...
```

2. List the running processes that aren't owned by root.

```
# ps -U root -N
  PID TTY          TIME CMD
 1050 ?        00:00:00 pickup
 1051 ?        00:00:00 qmgr
```

3. Run an update with highest priority:

```
# nice -20 yum update -y (or apt-get upgrade -y or zypper up)
Loaded plugins: fastestmirror
Setting up Update Process
Determining fastest mirrors
...
```

4. Watch disk utilization:

```
# watch df -h
Every 2.0s: df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root
                       19G  792M   17G   5% /
tmpfs                 246M     0  246M   0% /dev/shm
/dev/vda1             477M   46M  406M  11% /boot
```

Press "ctrl+z" to suspend the watch job.

Use the jobs command:

```
# jobs
[1]+  Stopped                 watch df -h
```

Use the fg command to foreground the watch job:

```
#fg
Every 2.0s: df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root
                       19G  792M   17G   5% /
tmpfs                 246M     0  246M   0% /dev/shm
/dev/vda1             477M   46M  406M  11% /boot
```

5. Press “ctrl+z” to suspend the watch job.

6. Use the jobs command:
jobs
[1]+ Stopped watch df -h

7. Use the fg command to foreground the watch job:
#fg
Every 2.0s: df -h
Filesystem Size Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root
19G 792M 17G 5% /
tmpfs 246M 0 246M 0% /dev/shm
/dev/vda1 477M 46M 406M 11% /boot

8. Press “ctrl+z” to suspend the watch job.

9. Start the watch job again, but in the background:
watch df -h &

10. Use the top command, and sort by memory:
top
“shift+m”
top - 15:53:32 up 14 min, 2 users, load average: 0.00, 0.00, 0.00
Tasks: 74 total, 1 running, 72 sleeping, 1 stopped, 0 zombie
Cpu(s): 0.0%us, 0.3%sy, 0.0%ni, 99.7%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 502096k total, 148812k used, 353284k free, 7184k buffers
Swap: 1015804k total, 0k used, 1015804k free, 52476k cached

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1152	root	20	0	99968	4072	3096	S	0.3	0.8	0:00.34	sshd
1051	postfix	20	0	81008	3408	2536	S	0.0	0.7	0:00.00	qmgr
1040	root	20	0	80860	3404	2508	S	0.0	0.7	0:00.01	master
1050	postfix	20	0	80940	3376	2504	S	0.0	0.7	0:00.00	pickup
1083	root	18	-2	12376	2612	540	S	0.0	0.5	0:00.00	udevd
1084	root	18	-2	12376	2612	540	S	0.0	0.5	0:00.00	udevd
1156	root	20	0	105m	1896	1536	S	0.0	0.4	0:00.09	bash
1088	root	20	0	105m	1852	1496	S	0.0	0.4	0:00.07	bash
1308	root	20	0	105m	1808	820	T	0.0	0.4	0:00.00	watch

11. Use the ps command to get the process id of your watch job:
ps aux | grep watch

```
root         6  0.0  0.0      0     0 ?        S    15:38   0:00 [watchdog/0]
root        27  0.0  0.0      0     0 ?        S    15:38   0:00 [linkwatch]
root      1308  0.0  0.3 108172  1808 pts/0    T    15:53   0:00 watch df -h
root      1311  0.0  0.1 103248   872 pts/0    S+   15:54   0:00 grep watch
```

12. Kill the watch job:

```
[root@localhost ~]# kill -9 1308
[1]+  Killed                  watch df -h
```

LINUX FOUNDATION

Get Certified!

Get more information at http://training.linuxfoundation.org/certification/lfcs