Monitor and manage Linux processes

Linux #redhat #processes #daemons

Applications, Processes, Jobs etc.

- **Application = Service** - is like a program that runs in your computer
  - In Windows you have Office Word, Powerpoint, Firefox, etc.
  - In Linux you have programs or applications like NTP, NFS, rsyslog, Apache, etc.
- **Script** - Shell scripts or Commands are list of instructions. Something that is written in a file and then packaged it in a way that it will execute.
  - `adduser`, `cd`, `pwd`, etc.
  - Any application that is running in your computer, for example, Apache, you would have to run that as a script and that will run in the background. All other commands in Linux are actually scripts.
- **Process** - When you run an application it actually generates processes with its process ID. Now processes could be one associated to that application or it could be multiple processes
  - Service --> Process1
    - Process2
    - Process3
- **Daemon** - Something that continuously runs in the background and runs until interrupted
  - When you run a daemon, which is also a process, but it keeps on running in the background and it keeps listening to the incoming or outgoing traffic.
- **Threads** - Every process could have multiple threads associated with it. An application that is running in your background, let's say NFS, when you run that application in your Linux machine it could have many multiple threads. If a remote machine tried to connect to you through NFS, it will generate one thread for it and it will generate a second thread or third thread for other computers that are attaching to it.
  - Service --> Process --> thread1
    - thread2
    - thread3
    - thread4
- **Job** - Job or Workorder = Run a service or process at a schedule time

Monitor and manage system resources

- When an operating system boots up many programs get loaded into system memory. These processes or programs need to be managed and monitored because they consume mainly 3 system resources like CPU, memory and disk space. If they are mismanaged then system can stress out and shut down on you an application, and your production applications can actually break.
- Here are a few monitoring commands to manage system resources
  - `df` - Gives you the statistics on your file system.
  - `du` - Stands for "disk usage" and gives you how much usage of the disk is being utilized by each file
  - `uptime` - Tells you the system how long its been up for.
  - `top`
  - `free`
  - `lsof` - Stands for "list open file system"
  - `tcpdump`

- `netstat`
- `ps`
- `kill`
- Some other commands are `vmstat`, `iostat`, `iftop`, etc.

The `df` command

- How do we check how much disk space that is utilized in the file system?

Example using `df` command:

```
[user@localhost ~]$ df
```

Output:

```
Filesystem            1K-blocks    Used Available Use% Mounted on
devtmpfs                   4096       0      4096   0% /dev
tmpfs                   5294164       0   5294164   0% /dev/shm
tmpfs                   2117668    9376   2108292   1% /run
/dev/mapper/rhel-root  17756160 7871500   9884660  45% /
/dev/sda1                983040  310840    672200  32% /boot
tmpfs                   1058832     100   1058732   1% /run/user/1000
```

Example using `df` command:

```
[user@localhost ~]$ df -h
```

- The `-h` option stands for "human readable" and specifies to change the units to be more readable to humans.

Output:

```
Filesystem             Size  Used Avail Use% Mounted on
devtmpfs               4.0M     0  4.0M   0% /dev
tmpfs                  5.1G     0  5.1G   0% /dev/shm
tmpfs                  2.1G  9.2M  2.1G   1% /run
/dev/mapper/rhel-root   17G  7.6G  9.5G  45% /
/dev/sda1              960M  304M  657M  32% /boot
tmpfs                  1.1G  100K  1.1G   1% /run/user/1000
```

- For example for the size, instead of Kilobytes we can see Megabytes and Gigabytes and we know how much that actually is.
- The 'devtmpfs' and 'tmpfs' are file system for your swap space.
  - Our swap space has been carved out for AROUND 7 Gigabytes.
- The '/dev/mapper/rhel-root' is actually the filesystem for all the data in the machine.
- The '/dev/sda1' is the filesystem that is given to our boot.
  - Whenever you boot up your Linux system there are certain files that are located on /boot, so the operating system will go into /boot and find those files, and finally boot up your system.
- If you see that the usage of any filesystem is around 98% or 100% it means your filesystem is getting full and you need to find out why it is getting full. (See next command)\

Example using `df` command:

```
[user@localhost ~]$ df -T
```

- The `-T` option is set to specify which type of file system is each of the file systems listed. E.g. ext4, exfs, etc.

Output:

```
Filesystem            Type     1K-blocks    Used Available Use% Mounted on
devtmpfs              devtmpfs      4096       0      4096   0% /dev
tmpfs                 tmpfs      5294168       0   5294168   0% /dev/shm
tmpfs                 tmpfs      2117668    9376   2108292   1% /run
/dev/mapper/rhel-root xfs       17756160 7871972   9884188  45% /
/dev/sda1             xfs         983040  310840    672200  32% /boot
tmpfs                 tmpfs      1058832      96   1058736   1% /run/user/1000
```

The `du` command

- Use when you want to know which individual file is actually causing or is the culprit that you could delete to re-claim your space.
- `du` stands for "disk usage"

Example using `du` command:

```
[root@localhost ~]# du /
```

- Targeting each file in OS
- For this operation the user has to become root because there are certain files that only has permissions to root user.

Output:

```
...
0       /usr/share/locale/sat@deva
0       /usr/share/locale/sat
0       /usr/share/locale/sas/LC_MESSAGES
0       /usr/share/locale/sas
0       /usr/share/locale/sam/LC_MESSAGES
```

- You'll see each and individual file that is in my operating system with its size and it goes so fast so it is very hard to read.

Example using `du` command:

```
[root@localhost ~]# du -k
```

- The `-k` option stands for "kilobyte" spefies to show content size in Kilabytes

Output:

```
...
36      /var/lib/flatpak/repo/refs/remotes
68      /var/lib/flatpak/repo/refs
3284    /var/lib/flatpak/repo/objects/eb
9864    /var/lib/flatpak/repo/objects/9d
```

- Still the output is too fast and very long

Example using `du` command:

```
[root@localhost ~]# du -k / | sort -nr | more
```

- The `du` command is being used to check for name and size of the entirety of the `/` directory.
- The `-k` option specifies that size will be shown in Kilobytes
- This command is being piped with the `sort` command which sorts the content of a file.
  - The `-n` stands for "numeric" option specifies that a numeric sort is going to take place (Compare according to string numerical value). In this case it will arrange the files in order of decreasing size (from higher to lowest size).
  - The `-r` option stands for "reverse" specifies that the sorted content will be in reverse order.
- This command is being piped again with a `more` command to show the content of a file one page at a time.

Output:

```
...
66908   /var/tmp/flatpak-cache-F5IHP2
65112   /usr/lib/modules/5.14.0-427.18.1.el9_4.x86_64/kernel
64704   /home/mmarin/.var/app/md.obsidian.Obsidian
64704   /home/mmarin/.var/app
--More--
```

Example using `du` command:

```
[user@localhost ~]$ du -h / | sort -nr | more
```

- The `du` command is also applicable with the `-h` option which makes units more readable fur humans.

Output:

```
1020K   /var/lib/flatpak/repo/objects/06
1012K   /usr/share/microcode_ctl/ucode_with_caveats/intel-06-8e-9e-0x-0xca
1012K   /usr/lib/modules/5.14.0-427.18.1.el9_4.x86_64/kernel/drivers/net/wireless/realtek/rtlwifi
1004K   /usr/lib/firmware/rtw89
```

- Note the size unit is displayed.

After identifying which file is taking up a lot space you can go ahead and delete the file to re-claim that space but first make sure you check with the file owner or make a backup copy on another server, in this way you do not delete any important file accidentaly.

The `uptime` command

Example using `uptime` command:

```
[user@localhost ~]$ uptime
```

Output:

```
20:23:50 up 31 min,  3 users,  load average: 0.00, 0.00, 0.08
```

- After running the `uptime` command, the following fields are displayed:
  - The right existing time (20:23:50 up 31 min)
  - Number of users that's been logged in since the system has been up or logged in right now (3 users)
  - The system load average in percentage in the following time intervals respectively:
    - the last minute (0.00), 5 minutes (0.00), 15 minutes(0.08)
- Is important to check the uptime if the system is running is slower or the response time is slower.

The `top` command

- Every system administrator favorite command
- All the processes that you are running on your Linux system will be displayed by running the command `top`

Example using `top` command:

```
[user@localhost ~]$ top
```

Output:

```
top - 20:45:32 up 53 min,  3 users,  load average: 0.00, 0.00, 0.00
Tasks: 242 total,   1 running, 241 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni, 99.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  10340.2 total,   8254.4 free,   1378.2 used,   1036.2 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   8962.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   3151 root      20   0  226056   4352   3456 R   0.7   0.0   0:00.10 top
      1 root      20   0  173540  18068  10516 S   0.0   0.2   0:10.35 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.21 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
...
```

- The first line says the command, the time right now, the system has been up and running, users, and load average (The same as from the `uptime` command)
  - `top - 20:45:32 up 53 min, 3 users, load average: 0.00, 0.00, 0.00`
- The second line tells you the tasks, total tasks that we are running in our system or processes. E.g.
  - `Tasks: 242 total, 1 running, 241 sleeping, 0 stopped, 0 zombie`
  - 242 total tasks
  - 1 task running
  - 241 tasks are sleeping
  - 0 stop
  - 0 zombie

- The third line tells you the CPU usage and system stats.
  - `%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni, 99.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st`
  - If CPU is at around 90-95% then definitely means something is wrong with your system, and there might be a process here that is causing your CPU load to go up.
- Then we have memory stats
  - `MiB Mem : 10340.2 total, 8254.4 free, 1378.2 used, 1036.2 buff/cache`
- Then we have swap space stats
  - `MiB Swap: 2048.0 total, 2048.0 free, 0.0 used. 8962.0 avail Mem`
- Next is the columns which are described in the following list:
  - PID: Process ID.
  - USER: The owner of the process.\
  - PR: Process priority.
  - NI: The nice value of the process.
  - VIRT: Amount of virtual memory used by the process.
  - RES: Amount of resident memory used by the process.
  - SHR: Amount of shared memory used by the process.
  - S: Status of the process.
    - S = sleeping
    - R = running
    - D = Uninterruptible sleep
    - I = Idle
    - T = Stopped
    - t = Being stopped
    - Z = Dead process (Zombie)
  - %CPU: The share of CPU time used by the process since the last update.
  - %MEM: The share of physical memory used.
  - TIME+: Total CPU time used by the task in hundredths of a second.
  - COMMAND: The command name or command line (name + options).
- The PID is useful to identify the process and then kill it if necessary.
- You can hit "q" on your keyboard to get out of the `top` command

The `free` command

- `free` will give you the information about your memory

Example using `free` command:

```
[user@localhost ~]$ free
```

- Displays information about the memory

Output:

```
               total        used        free      shared  buff/cache   available
Mem:        10588336     1415492     8399276       22548     1110100     9172844
Swap:        2097148           0     2097148
```

- Note there are no units

Example using `free` command:

```
[user@localhost ~]$ free -m
```

- The `-m` option is used to display the memory information in Megabyte units

Output:

```
               total        used        free      shared  buff/cache   available
Mem:           10340        1382        8202          22        1084        8957
Swap:           2047           0        2047
```

- This is the physical memory we have or in case of a Virtual Machine the memory we assigned when we built our machine.
- Swap space is actually a space that is on the disk. It is not a dedicated memory like a physical memory, it is actually a space carved out on a disk.

The `lsof` command

- Stands for "list open file"
- When you are running a process a process most likely will go into certain files and certain configuration file to look what it needs to do. It needs some instructions.
- Sometimes when you stop a process, the files that it was looking into remain open, which can also cause a system to break or slow down.
- You could find the list of open files by running the command `lsof`.

Example using `lsof` command:

```
[user@localhost ~]$ lsof
```

Output:

```
...
NetworkMa  937                        root    22u  a_inode              0,14         0       1057
[timerfd]
NetworkMa  937                        root    24u  a_inode              0,14         0       1057
[eventpoll:25,26]
NetworkMa  937                        root    25u  a_inode              0,14         0       1057
[timerfd]
NetworkMa  937                        root    26u     pack             16099       0t0        ARP
type=SOCK_DGRAM
...
```

- When running the `lsof` command you are gonna find a bunch of files because we have so many processes running.
- Each process could have multiple files open.
    - If it's actually asking hardware to print something, asking mouse to move or asking keyboard to type, it's actually going through each file and touching that file.

The `tcpdump` command

- This command is specifically for networking.
- Which activity is coming into the system?
- Which traffic is going out of the system?
- This is the command that will tell you everything.

Example using `tcpdump` command:

```
[user@localhost ~]$ tcpdump
```

Output:

```
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
22:18:18.485778 IP localhost.localdomain.ssh > 192.168.1.58.51835: Flags [P.], seq
522908912:522909152, ack 3022170718, win 1872, length 240
22:18:18.532905 IP 192.168.1.58.51835 > localhost.localdomain.ssh: Flags [.], ack 240, win 8194,
length 0
22:18:18.577893 IP localhost.localdomain.34562 > _gateway.domain: 59098+ PTR? 58.1.168.192.in-
addr.arpa. (43)
22:18:20.193096 IP 192.168.1.58.57621 > 192.168.1.255.57621: UDP, length 44
...
```

- There is a bunch of information coming in and out of your system.
  - Each line tiles you:
    - The time
    - The IP address
    - Where is it coming from
    - What is the gateway
    - What is the broadcast information
    - etc.

How to find what is the name of your interface?

- Simply run the `ifconfig` command and the name of your network adapter will be the name of your interface.

Example using `ifconfig` command:

```
[user@localhost ~]$ ifconfig
```

Output:

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.156  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::a00:27ff:feda:ce70  prefixlen 64  scopeid 0x20<link>
...
```

- In this example, the name of the network adapter is "enp0s3". This is also the name of the interface.

Example using `tcpdump` command:

```
[user@localhost ~]$ tcpdump -i enp0s3
```

- This line uses the `tcpdump` command to see all the traffic that are flowing over this network interface. I don't want to know any of the traffic happening internally which is local or on the virtual adapter.
- The `-i` option stands for "interface" and it lets you see the traffic from a specific network adapter.
- There is still so much information going in and out.
- If you are troubleshooting a network connection, this is the best command that you could get.

The `netstat` command

- This command is related to your network statistics or anything coming in and out.
- What are the connections that's been established?
- Who's your router?
- Who's your gateway?
- etc.

Example using `netstat` command:

```
[user@localhost ~]$ netstat -rnv
```

- The `netstat` command shows useful network statistics and identifications.
- The `-r` option specifies to display the kernel routing tables
- The `-n` option specifies to show numerical addresses instead of trying to determine symbolic host, port, or user names.
- The `-v` option specifies to tell the user what is going on by being verbose. Especially print some useful information about unconfigured address families.

Output:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG        0 0          0 enp0s3
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 enp0s3
```

- In this example all the traffic that is going through our Linux machine is going through the gateway 192.168.1.1.

Example using `netstat` command:

```
[user@localhost ~]$ netstat -a
```

- The `-a` option specifies to give you all the connection information that is connected to outside.

Output:

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0    240 localhost.localdoma:ssh 192.168.1.58:51835      ESTABLISHED
udp        0      0 localhost.locald:bootpc _gateway:bootps         ESTABLISHED
Active UNIX domain sockets (w/o servers)
```

```
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    26796    /run/user/1000/systemd/notify
unix  3      [ ]         DGRAM      CONNECTED     58       /run/systemd/notify
unix  17     [ ]         DGRAM      CONNECTED     71       /run/systemd/journal/dev-log
...
```

Example using `netstat` command:

```
[user@localhost ~]$ netstat -au
```

- The `-u` option specifies to give you all the UDP traffic

Example using `netstat` command:

```
[user@localhost ~]$ netstat -at
```

- The `-t` option specifies to give you all the TPD traffic

Output (TCP):

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 localhost:ipp           0.0.0.0:*               LISTEN
tcp        0     64 localhost.localdoma:ssh 192.168.1.58:51835      ESTABLISHED
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
tcp6       0      0 localhost:ipp           [::]:*                  LISTEN
```

- Will give you all the TCP traffic that it's listening to.

The `ps` command

- Stands for "process status"
- It will give you the list of all the processes that are running in the system.
  - You could also use the `top` command for this but it will only give you the highest running processes on top, but there's so many other processes as well.
- What if you wanted to delete a process with ID 31? It won't show up on `top`

Example using `ps` command:

```
[user@localhost ~]$ ps -ef
```

- This line uses the `ps` command to see every process on the system using standard syntax
- `ps` reports a snapshot of the current processes.
- The `-e` option specifies to select all processes (Same as `-A`)
- The `-f` option specifies to do a full-format listing.

Output:

```
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 19:52 ?        00:00:10 /usr/lib/systemd/systemd rhgb --switched-root --
system --deserial
root           2       0  0 19:52 ?        00:00:00 [kthreadd]
root           3       2  0 19:52 ?        00:00:00 [rcu_gp]
root           4       2  0 19:52 ?        00:00:00 [rcu_par_gp]
...
```

- Displays all the processes that we have running

What if you wanted to find out information about a specific process?

- You can also run `ps -ef | more` to look for that process one page at a time or...
- You can run `ps -ef | grep process_name` and replace "process_name" with the actual name of the process to look for it.

Example using `ps -ef` command:

```
[user@localhost ~]$ ps -ef | grep top
```

- The `ps -ef` command is being piped with the `grep` command to look exactly for the process called "top" in the outputs of the `ps -ef` command.

Output:

```
root        3655    3465  0 23:11 pts/1    00:00:00 grep --color=auto top
```

- There isn't anything listed because the process is not running.
- It is just listing the actual `grep` command looking for the word "top".
    - As soon as it execute it becomes its own process. (It actually greps its own running process)

Output:

```
mmarin      3714    3667  0 23:12 pts/2    00:00:00 top
root        3716    3465  0 23:12 pts/1    00:00:00 grep --color=auto top
```

- This is how it would look like if a user is currently running the process called `top`

The `kill` command

- What if a process is not stopping even after closing the window? Kill it :D
- You can use the `kill` command to kill a process right away

Example using `kill` command:

```
[user@localhost ~]$ kill 3714
```

- First identify the process ID by using either `top` or `ps -ef`

    - In this example the process ID is the number "3714"
- The `kill` command will then eliminate the process with process ID "3714"

Another powerful option with `kill`:

- If you are attempting to kill a process and the process is not getting killed, then...

Example using `kill` command:

```
[user@localhost ~]$ kill -9 3714
```

- In this line the `kill` command will just go and simply kill it harshly without looking here and there the "3714" process. (It will not go through the graceful killing process.)
- It will actually kill all the associated process with that.
- The `-9` signal is used to preference to the KILL signal.

Other commands

Example using `vmstat` command:

```
[user@localhost ~]$ vmstat
```

- The `vmstat` command reports virtual memory statistics

Output:

```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 8363240   2780 1137344    0    0     8     0   14   13  0  0 100  0  0
```

- `iostat` and `iftop` commands are not included by default in RHEL 9.