# Advanced Linux Commands Cheat Sheet

This cheat sheet presents a collection of Linux commands and shows how they are used by developers in advanced programming scenarios. The commands are organized by category.

## APPLICATION MANAGEMENT COMMANDS

Commands in this section apply to advanced operations when working with a computer's applications and executables.

### systemctl

systemctl [options] <subcommand>
Used to inspect and control the state of the systemd suite of Linux system components.
**Examples:**
This example shows the invocation and result of a systemctl status command set that discovers the status of the Apache web server (note that $ is the command-line prompt symbol):

```
$ systemctl status httpd
 httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disable>
   Active: inactive (dead)
     Docs: man:httpd.service(8)
```

This example shows the invocation of the systemctl start command set that starts the mariadb database:
systemctl start mariadb
This example shows the invocation of the systemctl enable command set so that an application starts automatically when the computer reboots:
systemctl enable mariadb

### yum

The application installation and removal tool for Fedora, CentOS, and RHEL.
**Example:**
The following command installs the net-tools application, which has many handy utilities such as netstat:
sudo yum -y install net-tools

## AUDIT COMMANDS

The commands in this section are used to work with the audit capabilities provided by the Linux operating system.

### ausearch

ausearch [options] <application/process>
Used to query the audit logs according to events defined by different search criteria.
**Example:**
The following example shows how to use ausearch to get information about the user named root from the audit logs. The option -ui indicates that the search should be conducted according to a particular user ID, in this case root. The result is piped to the more command that uses the option -10, which indicates to show the first 10 lines of output.
Note that the ausearch command must be invoked as sudo in order to access audit logs:

```
$ sudo ausearch -ui root | more -10
----
time->Fri Jan  7 15:42:52 2022
type=SERVICE_START msg=audit(1641598972.323:5): pid=1 uid=0 auid=4294967295 ses=4294967295 subj=system_u:system_r:init_t:s0
msg='unit=rpcbind comm="systemd" exe="/usr/lib/systemd/systemd" hostname=? addr=? terminal=? res=success'
----
time->Fri Jan  7 15:42:52 2022
```

```
type=PROCTITLE msg=audit(1641598972.761:6):
proctitle=2F7362696E2F617564697463746C002D52002F6574632F61756469742F61756469742E72756C6573
type=SYSCALL msg=audit(1641598972.761:6): arch=c000003e syscall=44 success=yes exit=56 a0=3 a1=7fff0ba65240 a2=38 a3=0 items=0 ppid=843
pid=858 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="auditctl"
exe="/usr/sbin/auditctl" subj=system_u:system_r:unconfined_service_t:s0 key=(null)
type=CONFIG_CHANGE msg=audit(1641598972.761:6): op=set audit_backlog_limit=8192 old=64 auid=4294967295 ses=4294967295
subj=system_u:system_r:unconfined_service_t:s0 res=1
----
--More--
```

## DISK MANAGEMENT COMMANDS

Commands in this section apply to working with disks, devices, and volumes on a computer running the Linux operating system.

### df

df [options] <file name>

Shows the amount of disk space used and available according to the file system that represents a particular disk device mount. If no file name is given, the space available on all mounted file systems is displayed.

**Example:**

The following example shows the invocation and result of df displaying all mounted file systems. Disk space is shown in 1K blocks:

```
$ df
Filesystem          1K-blocks    Used Available Use% Mounted on
devtmpfs              3949180       0   3949180   0% /dev
tmpfs                 3978636       0   3978636   0% /dev/shm
tmpfs                 3978636    9464   3969172   1% /run
tmpfs                 3978636       0   3978636   0% /sys/fs/cgroup
/dev/mapper/rhel-root  50065528 5588744  44476784  12% /
/dev/mapper/rhel-home  24445276  228104  24217172   1% /home
/dev/sda1              1038336  262796    775540  26% /boot
tmpfs                  795724      64    795660   1% /run/user/1000
```

### fdisk

Used for working with disk partitions on a computer's hard drive. You can use fdisk to format and partition a new drive. Also, you can use fdisk to get information about existing drives.

**Examples:**

This example invokes fdisk to display a print out of partition tables of all devices on the computer, as listed in the /proc/partitions file:

```
$ sudo fdisk -l
Disk /dev/sda: 80 GiB, 85899345920 bytes, 167772160 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x565c4e2e

Device     Boot   Start      End   Sectors Size Id Type
/dev/sda1  *       2048   2099199   2097152   1G 83 Linux
/dev/sda2       2099200 167772159 165672960  79G 8e Linux LVM

Disk /dev/mapper/rhel-root: 47.8 GiB, 51292143616 bytes, 100179968 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mapper/rhel-swap: 7.9 GiB, 8485076992 bytes, 16572416 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
```

```
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mapper/rhel-home: 23.3 GiB, 25044189184 bytes, 48914432 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

This example uses fdisk to start the process of formatting and partitioning a drive at /dev/sda1. This invocation of fdisk will open a dialog in the terminal window that walks the developer through the formatting and partitioning process:

```
fdisk /dev/sda1
```

**Note:** Using fdisk in this manner erases all data on the disk.

## mount

mount [options] <device_directory>`
Shows or attaches a device filesystem to a Linux operating system's master file tree.

**Examples:**

This example shows the mount command using the -l option to list all mounted file systems. The result of mount is piped to the more command. The more command uses the -10 option to display the first 10 lines of output:

```
$ mount -l | more -10
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime,seclabel)
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
devtmpfs on /dev type devtmpfs (rw,nosuid,seclabel,size=3949180k,nr_inodes=987295,mode=755)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,seclabel)
devpts on /dev/pts type devpts (rw,nosuid,noexec,relatime,seclabel,gid=5,mode=620,ptmxmode=000)
tmpfs on /run type tmpfs (rw,nosuid,nodev,seclabel,mode=755)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,seclabel,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,seclabel,xattr,release_agent=/usr/lib/systemd/systemd-cgroups-ag
ent,name=systemd)
--More--
```

This example uses the mount command to mount the file system for a hard drive that is represented as /dev/sda1:

```
$ sudo mount /dev/sda1
```

## xfs_repair

xfs_repair [options] <drive>
Inspects and optionally repairs a hard drive on a computer running RHEL Linux.

**Note:** You must unmount the drive using the umount command before invoking xfs_repair.

**Example:**

The following example unmounts a hard drive at /dev/sda1 and then runs the command xfs_repair against that drive:

```
$ sudo umount /dev/sda1

$ sudo xfs_repair /dev/sda1
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - zero log...
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan and clear agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
        - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
```

```
        - check for inodes claiming duplicate blocks...
        - agno = 0
        - agno = 1
        - agno = 2
        - agno = 3
    Phase 5 - rebuild AG headers and trees...
        - reset superblock...
    Phase 6 - check inode connectivity...
        - resetting contents of realtime bitmap and summary inodes
        - traversing filesystem ...
        - traversal finished ...
        - moving disconnected inodes to lost+found ...
    Phase 7 - verify and correct link counts...
    done
```

## FILE MANAGEMENT COMMANDS

Commands in this section apply to working with files and directories on a computer running the Linux operating system.

### chmod

chmod [options] <file or directory>
Changes the permissions granted to a file or directory.

**Examples:**

This example applies read-only permissions for all users to the file named sample.txt. Then, the command ls -l is executed to verify that the file is indeed read-only. The output is also displayed:

```
$ chmod a-w+r sample.txt
$ ls -l sample.txt
-r--r--r--. 1 guest guest 32 Jan 17 10:34 sample.txt
```

This example applies read and write permissions for all users to the file named sample.txt. Then, the command ls -l is executed to verify that the file is indeed read-only. The output is also displayed:

```
$ chmod a+w+r sample.txt
$ ls -l sample.txt
-rw-rw-rw-. 1 guest guest 32 Jan 17 10:34 sample.txt
```

### chown

chown [options] <owner><:<:<group>> <file>
Changes the owner of a file according to user and/or group.

**Example:**

This example lists the files in the current directory by using the ls -l command and option to display details about the files, including the user: group pair that owns each file. Then, the owner of the file two.txt is changed using the chown command, assigning the user named lennonjohn who is in the group beatles as the new file owner.
The result of the change in file ownership is displayed using the command ls with the option -l again:

```
$ ls -l
total 0
-rw-rw-r--. 1 guest guest 0 Jan 19 10:17 one.txt
-rw-rw-r--. 1 guest guest 0 Jan 19 10:18 three.txt
-rw-rw-r--. 1 guest guest 0 Jan 19 10:17 two.txt

$ sudo chown lennonjohn:beatles two.txt

$ ls -l
total 0
-rw-rw-r--. 1 guest   guest 0 Jan 19 10:17 one.txt
-rw-rw-r--. 1 guest   guest 0 Jan 19 10:18 three.txt
-rw-rw-r--. 1 lennonjohn beatles  0 Jan 19 10:17 two.txt
```

## JOB COMMANDS

Commands in this section apply to working with jobs running under the Linux operating system. A *job* is a process that is invoked from a terminal window process and is considered a child of the terminal window.

### bg

bg<job_id>
Sends a job to the background.

**Example:**

The following example creates a bash script named demo.sh that outputs the string hi there to the console and then sleeps for two seconds. The bash script is invoked as a foreground job.
Then, the foreground job is stopped by striking the ctrl+Z keys. The job is started again in the background using the bg command along with the job id %1:

```
$ echo "while true; do echo hi there; sleep 2; done" > demo.sh
$ sh demo.sh
hi there
hi there
hi there
^Z
[1]+  Stopped                 sh demo.sh
$ bg %1
[1]+ sh demo.sh &
```

### fg

fg <job_id>
Sends a job to the foreground.

**Example:**

The following example creates a bash script named demo.sh that outputs the string hi there to the console and then sleeps for two seconds. The script is invoked as a job.
Then, the foreground job is stopped by striking the ctrl+Z keys. The job is started again in the foreground using the fg command, along with the job id %1:

```
$ echo "while true; do echo hi there; sleep 2; done" > demo.sh
$ sh demo.sh
hi there
hi there
hi there
^Z
[1]+  Stopped                 sh demo.sh
$ fg %1
sh demo.sh
```

### jobs

jobs [options]
Lists the jobs invoked from the process window

**Example:**

The following example uses the jobs command to list all the jobs and the status of jobs that were started from the current terminal window. Notice that there are three jobs in force. All the jobs are running the same bash script named demo.sh. Jobs %1 and %3 are stopped. Job %2 is running in the background, as denoted by the symbol &:

```
$ jobs -l
[1]+  6265 Stopped              sh demo.sh
[2]   6262 Running              sh demo.sh &
[3]+  6265 Stopped              sh demo.sh
```

## MEMORY MANAGEMENT COMMANDS

The following command applies to working with memory on a computer running the Linux operating system.

### free

This command reports information about memory status on the local computer or virtual machine.

**Example:**

The following example invokes the free command with the -w option. The -w option reports additional information about memory status:

```
$ free -w
              total        used        free      shared     buffers       cache   available
Mem:        7957276     1653404     4563456       35032        4320     1736096     5965440
Swap:       8286204           0     8286204
```

## NETWORK COMMANDS

Commands in this section apply to working with or on a network.

### curl

curl [options] <url>

Gets or posts a file to or from the Internet according to a URL.

**Examples:**

This example downloads a web page from the Red Hat Developer website implementing the -o option to save the page to the file named article.html:

```
$ curl https://developers.redhat.com/articles/2022/01/11/5-design-principles-microservices -o article.html
```

This example uses the curl command to upload a file named data.txt to the URL https://example.com/api/data. Notice the use of the -X option to tell curl to use the HTTP POST method, the -H option to set the content type header in the request, and the -d option to define the file to upload:

```
$ curl -X POST -H "Content-Type: text/plain" -d "data.txt" https://example.com/api/data
```

### host

host [options] <domain_name>

Reports information about a given domain name.

**Example:**

The following example uses host to report the default information about the domain name redhat.com:

```
$ host redhat.com
redhat.com has address 209.132.183.105
redhat.com mail is handled by 10 us-smtp-inbound-1.mimecast.com.
redhat.com mail is handled by 10 us-smtp-inbound-2.mimecast.com.
```

### hostname

hostname

Sets or gets the hostname of the computer or virtual machine.

**Examples:**

This example displays the current hostname of the machine:

```
$ hostname
localhost.localdomain
```

This example renames the current hostname of the machine to newhostname.localdomain and then verifies the current hostname of the machine:

```
$ sudo hostname newhostname.localdomain
$ hostname
newhostname.localdomain
```

### hostnamectl

hostnamectl [options] <command>
The command hostnamectl is similar to the command hostname but with added capabilities.
**Example:**
The following example invokes and displays the result of the hostnamectl status command set to report the hostname and additional information:

```
$ hostnamectl status
   Static hostname: localhost.localdomain
Transient hostname: tempvm.localdomain
         Icon name: computer-vm
           Chassis: vm
        Machine ID: 7080e8d7b18547fa90aa06416ce6a1cf
           Boot ID: 7d0c3ed3f773457a8045602e66e2581f
    Virtualization: oracle
  Operating System: Red Hat Enterprise Linux 8.5 (Ootpa)
       CPE OS Name: cpe:/o:redhat:enterprise_linux:8::baseos
            Kernel: Linux 4.18.0-348.el8.x86_64
      Architecture: x86-64
```

## iptables

iptables [options]
Sets and monitors network access to a given computer.
**Note:** The command iptables must be run with administrator permissions under sudo.
**Examples:**
This example allows incoming TCP traffic accessing the computer via port 22:

```
$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

This example rejects any network traffic coming in from a computer running from the IP address 192.168.86.11:

```
$ sudo iptables -A INPUT -s 192.168.86.11 -j DROP
```

## nmcli

nmcli [options] <object> <command>
The application nmcli is the command-line interface for working with the Linux NetworkManager.
**Examples:**
This example uses nmcli to display the overall connection status of the computer using the general object and the status command:

```
$ nmcli general status
STATE      CONNECTIVITY  WIFI-HW  WIFI     WWAN-HW  WWAN
connected  full          enabled  enabled  enabled  enabled
```

This example uses nmcli to display the connection status of the network interface devices using the device object and the status command:

```
$ sudo nmcli device status
DEVICE      TYPE      STATE                CONNECTION
enp0s3      ethernet  connected            enp0s3
```

This example uses nmcli to display the connection profiles of network devices on a computer in a report-like format by using the option -p (pretty) against the connection object:

```
$ nmcli -p connection
=====================================
  NetworkManager connection profiles
=====================================
NAME    UUID                                  TYPE      DEVICE
-------------------------------------------------------------------------
enp0s3  c68cddff-4883-4efb-bf7a-8b746fe6b26d  ethernet  enp0s3
virbr0  49bf2d57-cf45-41a8-85d2-cd43a59f0e1c  bridge    virbr0
```

## PROCESS COMMANDS

Commands in this section apply to working with Linux processes from the command line.

## &&

<command> && <command>
Executes commands in a sequence.

**Example:**

The following command changes the current directory to /etc, then executes the command ls to list the contents of the directory:

```
$ cd /etc && ls
```

## iotop

iotop [options]
iotop is a system monitoring program. It does not ship by default with Red Hat Enterprise Linux and must must be installed using the command set sudo yum install itop.
Note that this command requires administrator access and must be invoked as sudo.

**Example:**

The following example shows how to invoke iotop to read system IO. A portion of the output is displayed:

```
$ sudo iotop

Total DISK READ : 0.00 B/s | Total DISK WRITE :       0.00 B/s
Actual DISK READ:       0.00 B/s | Actual DISK WRITE:       0.00 B/s
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
69034 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.03 % [kworker/0:4-events_power_efficient]
    1 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % systemd --switched-root --system --deserialize 17
    2 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [kthreadd]
    3 be/0 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [rcu_gp]
    4 be/0 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [rcu_par_gp]
    6 be/0 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [kworker/0:0H-events_highpri]
    9 be/0 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [mm_percpu_wq]
   10 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [ksoftirqd/0]
   11 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [rcu_sched]
   12 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [migration/0]
   13 rt/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [watchdog/0]
   14 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [cpuhp/0]
   16 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [kdevtmpfs]
   17 be/0 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [netns]
   18 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [rcu_tasks_trace]
   19 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [rcu_tasks_rude_]
   20 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [kauditd]
   21 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [khungtaskd]
   22 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % [oom_reaper]
.
.
.
```

## ps

ps [options]
Displays the status of current processes.

**Example:**

The following example invokes the ps command with the options aux to display every process on the system. The result of the invocation is piped to the more command using the -10 to display the first 10 lines of results for stdout:

```
$ ps aux | more -10
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.1 175932 14212 ?        Ss   Jan07   0:06 /usr/lib/systemd/systemd --switched-root --syst
em --deserialize 18
root           2  0.0  0.0      0     0 ?        S    Jan07   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   Jan07   0:00 [rcu_gp]
```

```
root         4  0.0  0.0      0     0 ?        I<   Jan07   0:00 [rcu_par_gp]
root         6  0.0  0.0      0     0 ?        I<   Jan07   0:00 [kworker/0:0H-events_highpri]
root         9  0.0  0.0      0     0 ?        I<   Jan07   0:00 [mm_percpu_wq]
root        10  0.0  0.0      0     0 ?        S    Jan07   0:02 [ksoftirqd/0]
root        11  0.0  0.0      0     0 ?        I    Jan07   0:01 [rcu_sched]
--More--
```

## SELINUX MANAGEMENT COMMANDS

Commands in this section apply to working with Red Hat's Security-Enhanced Linux (SELinux), which provides an additional layer of system security. SELinux fundamentally answers the question: "May <subject> do <action> to <object>." For example: "May a web server access files in users' home directories?"

### getenforce

getenforce
Reports the current mode of SELinux rules enforcement. The modes are Enforcing, Permissive, or Disabled.

**Example:**

The following example invokes the command getenforce along with the results:

```
$ getenforce
Enforcing
```

### getsebool

getsebool [-a] <boolean_value>
Reports whether an applicable SELinux setting is on or off. Use the -a option to show the boolean value of all settings.

**Examples:**

This example use getsebool to show the value of the SELinux setting virt_use_xserver:

```
$ getsebool virt_use_xserver
virt_use_xserver --> off
```

This example use getsebool along with piping the result to the grep and then more commands to show all SELinux settings that have the value on. The more command uses the -10 option to show the first 10 lines of output:

```
$ getsebool -a | grep "on$" | more -10
abrt_upload_watch_anon_write --> on
auditadm_exec_content --> on
boinc_execmem --> on
cron_userdomain_transition --> on
dbadm_exec_content --> on
domain_fd_use --> on
entropyd_use_audio --> on
fips_mode --> on
gssd_read_tmp --> on
guest_exec_content --> on
--More--
```

### restorecon

restorecon [options] </path/to/dirctory_or_filename>
Used to set the security context (extended SELinux file labels) on one or more files to the default setting.

**Example:**

The following example uses restorecon to restore the default labels on all files under the directory /var/www/html. The option -F is used to force a change. The option v will show changes in a file's labels. The option -R implements execution of the command recursively through all subordinate directories and files from the directory where the command is invoked:

```
$ restorecon -FvR /var/www/html
```

## semanage

semanage <object> [options]
Allows administrators to manage SELinux capabilities according to the particular object of interest. Each object will have its own set of options. Examples of semanage objects are user, login, port, and fcontext, to name a few.
Note that this command must be invoked as sudo.
Also note that the semanage program does not ship with SELinux by default: You must install the policycoreutils-python package with yum to get the semanage command.
**Examples:**
This example uses semanage to get the security settings for the user objects. The -l option is used to list the information for all users:

```
$ sudo semanage user -l

                Labeling   MLS/       MLS/
SELinux User    Prefix     MCS Level  MCS Range                SELinux Roles

guest_u         user       s0         s0                       guest_r
root            user       s0         s0-s0:c0.c1023           staff_r sysadm_r system_r unconfined_r
staff_u         user       s0         s0-s0:c0.c1023           staff_r sysadm_r unconfined_r
sysadm_u        user       s0         s0-s0:c0.c1023           sysadm_r
system_u        user       s0         s0-s0:c0.c1023           system_r unconfined_r
unconfined_u    user       s0         s0-s0:c0.c1023           system_r unconfined_r
user_u          user       s0         s0                       user_r
xguest_u        user       s0         s0                       xguest_r
```

This example uses semanage to view a list of login information:

```
$ sudo semanage login -l

Login Name           SELinux User         MLS/MCS Range        Service

__default__          unconfined_u         s0-s0:c0.c1023       *
root                 unconfined_u         s0-s0:c0.c1023       *
```

## setsebool

setsebool <setting> <value>
Applies a boolean value to an SELinux setting.
**Example:**
The following example uses setsebool to apply the value 1 (true) to the SELinux setting httpd_can_network_connect_db:

```
$ setsebool httpd_can_network_connect_db 1
```

## subscription-manager

subscription-manager <command> [options]
The subscription-manager is the command-line version of the graphical Red Hat Enterprise Linux Subscription Manager. Subscription Manager is a service that keeps track of the Red Hat products and subscriptions installed on a local computer. Subscription Manager communicates with the subscription service on the backend. Typically, backend settings are managed via the Customer Portal or an on-premise server such as Subscription Asset Manager. Subscription Manager works with content management tools such as yum.
The command subscription-manager will run as the root user; thus, a password prompt will be presented upon invocation.
**Example:**
The following example uses the subscription-manager facts command set to get facts about the local computer. The command pipes the result to the more command, which in turn uses the -15 option to display the first 15 lines of output.

```
$ sudo subscription-manager facts | more -15
Password: You are attempting to run "subscription-manager" which requires administrative
Password:

cpu.core(s)_per_socket: 1
cpu.cpu(s): 1
cpu.cpu_socket(s): 1
cpu.thread(s)_per_core: 1
```

```
cpu.topology_source: kernel /sys cpu sibling lists
distribution.id: Ootpa
distribution.name: Red Hat Enterprise Linux
distribution.version: 8.5
distribution.version.modifier: Unknown
dmi.baseboard.manufacturer: Oracle Corporation
dmi.baseboard.product_name: VirtualBox
dmi.baseboard.serial_number: 0
dmi.baseboard.version: 1.2
dmi.bios.address: 0xe0000
dmi.bios.release_date: 12/01/2006
--More--
```

### vmstat

vmstat [options]
Reports information about virtual memory usage as well as other relevant system data.

**Example:**

The following command invokes vmstat with the --stats option to display virtual memory and system information for the local virtual machine:

```
$ vmstat --stats
      7957276 K total memory
      1947752 K used memory
       999836 K active memory
      2537556 K inactive memory
      4001688 K free memory
         5248 K buffer memory
      2002588 K swap cache
      8286204 K total swap
            0 K used swap
      8286204 K free swap
        20330 non-nice user cpu ticks
         1964 nice user cpu ticks
        21780 system cpu ticks
     92624051 idle cpu ticks
        23290 IO-wait cpu ticks
        83766 IRQ cpu ticks
        38878 softirq cpu ticks
            0 stolen cpu ticks
      1077302 pages paged in
      1956245 pages paged out
            0 pages swapped in
            0 pages swapped out
     42443346 interrupts
     82932549 CPU context switches
   1641598949 boot time
       202862 forks
```

## SYSTEM INFORMATION COMMANDS

Commands in this section provide memory and hardware information about the computer in which the given command is invoked.

### lscpu

lscpu [options]
Displays information about the CPU architecture on the given machine.
When lscpu is run on a virtual machine, the CPU architecture information reported reflects the configuration of the guest operating system, which is typically different from the operating system on the host computer.

This command is part of the util-linux package, which you can install using the command: sudo yum install -y util-linux. In some cases, the package is part of the operating system's default installation.

**Example:**

The following example uses lscpu with the --json option to display CPU architecture information about a virtual machine in JSON format:

```
$ lscpu --json
{
   "lscpu": [
      {"field": "Architecture:", "data": "x86_64"},
      {"field": "CPU op-mode(s):", "data": "32-bit, 64-bit"},
      {"field": "Byte Order:", "data": "Little Endian"},
      {"field": "CPU(s):", "data": "1"},
      {"field": "On-line CPU(s) list:", "data": "0"},
      {"field": "Thread(s) per core:", "data": "1"},
      {"field": "Core(s) per socket:", "data": "1"},
      {"field": "Socket(s):", "data": "1"},
      {"field": "NUMA node(s):", "data": "1"},
      {"field": "Vendor ID:", "data": "GenuineIntel"},
      {"field": "CPU family:", "data": "6"},
      {"field": "Model:", "data": "142"},
      {"field": "Model name:", "data": "Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz"},
      {"field": "Stepping:", "data": "10"},
      {"field": "CPU MHz:", "data": "1991.998"},
      {"field": "BogoMIPS:", "data": "3983.99"},
      {"field": "Hypervisor vendor:", "data": "KVM"},
      {"field": "Virtualization type:", "data": "full"},
      {"field": "L1d cache:", "data": "32K"},
      {"field": "L1i cache:", "data": "32K"},
      {"field": "L2 cache:", "data": "256K"},
      {"field": "L3 cache:", "data": "8192K"},
      {"field": "NUMA node0 CPU(s):", "data": "0"},
      {"field": "Flags:", "data": "fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 syscall nx rdtscp
lm constant_tsc rep_good nopl xtopology nonstop_tsc cpuid tsc_known_freq pni pclmulqdq monitor ssse3 cx16 pcid sse4_1 sse4_2 x2apic movbe
popcnt aes xsave avx rdrand hypervisor lahf_lm abm 3dnowprefetch invpcid_single pti fsgsbase avx2 invpcid rdseed clflushopt md_clear flush_l1d"}
   ]
}
```

## lshw

lshw [options]

Displays information about a system's hardware. The command needs to be run as sudo in order to get all hardware information.

**Example:**

The following example uses the lshw command along with the -short option to get an abbreviated list of system information:

```
$ sudo lshw -short
H/W path        Device     Class      Description
=====================================================
                           system     VirtualBox
/0                         bus        VirtualBox
/0/0                       memory     128KiB BIOS
/0/1                       memory     8320MiB System memory
/0/2                       processor  Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz
/0/100                     bridge     440FX - 82441FX PMC [Natoma]
/0/100/1                   bridge     82371SB PIIX3 ISA [Natoma/Triton II]
/0/100/1/0                 input      PnP device PNP0303
/0/100/1/1                 input      PnP device PNP0f03
/0/100/1.1      scsi1      storage    82371AB/EB/MB PIIX4 IDE
/0/100/1.1/0.0.0 /dev/cdrom disk      CD-ROM
/0/100/2        /dev/fb0   display    VirtualBox Graphics Adapter
/0/100/3        enp0s3     network    82540EM Gigabit Ethernet Controller
/0/100/4                   generic    VirtualBox Guest Service
/0/100/5        card0      multimedia 82801AA AC'97 Audio Controller
/0/100/6                   bus        KeyLargo/Intrepid USB
```

```
/0/100/6/1      usb2      bus       OHCI PCI host controller
/0/100/6/1/1    input5    input     VirtualBox USB Tablet
/0/100/7                  bridge    82371AB/EB/MB PIIX4 ACPI
/0/100/b                  bus       82801FB/FBM/FR/FW/FRW (ICH6 Family) USB2 EHCI Controller
/0/100/b/1      usb1      bus       EHCI Host Controller
/0/100/d        scsi2     storage   82801HM/HEM (ICH8M/ICH8M-E) SATA Controller [AHCI mode]
/0/100/d/0.0.0  /dev/sda  disk      85GB VBOX HARDDISK
/0/100/d/0.0.0/1  /dev/sda1  volume  1GiB Linux filesystem partition
/0/100/d/0.0.0/2  /dev/sda2  volume  78GiB Linux LVM Physical Volume partition
/1              input0    input     Power Button
/2              input1    input     Sleep Button
/3              input2    input     AT Translated Set 2 keyboard
/4              input4    input     ImExPS/2 Generic Explorer Mouse
/5              input6    input     Video Bus
/6              input7    input     PC Speaker
```

## USERS AND GROUPS COMMANDS

Commands in this section apply to working with users and groups as supported by the Linux operating system.

### users

users [options]
Displays the name of the users logged in to the computer.

**Example:**

The following example uses the command users to list the users logged in to the system:

```
$ users
cooluser jaggermick lennonjohn
```

### useradd

adduser [options] <username>
Adds a user to the computing environment. The command must be run as sudo in order to have administrator access.

**Example:**

The following example adds a user with the login name cooluser. The HOME directory home/cooluser is created by default. Then, the example invokes the command passwd to set a password for the new user:

```
$ sudo adduser cooluser

$ sudo passwd cooluser
Changing password for user cooluser.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

### userdel

userdel [options] <username>
Deletes a user from the computer. The command must be run as sudo in order to have administrator access.

**Example:**

The following example uses the userdel command to remove the user with the login name cooluser from the system. The -r option indicates that the user's HOME directory should also be deleted:

```
$ sudo userdel -r cooluser
```

### usermod

usermod [options] <username>
Modifies user account information. The command can be used to add a user to a group. The command must be run as sudo in order to have administrator access.
**Example:**
The following example uses the command usermod to add a user with a login name lennonjohn to a group named beatles. Then, the command groups is used to verify that the user lennonjohn is indeed assigned to the group beatles:

```
$ sudo usermod -a -G beatles lennonjohn

$ groups lennonjohn
lennonjohn : lennonjohn beatles
```

## groups

groups [options] <username>
Lists the groups to which a user belongs.
**Example:**
The following example uses the command groups to list the groups to which the user with the username lennonjohn belongs:

```
$ groups lennonjohn
lennonjohn : lennonjohn beatles
```

## gpasswd

gpasswd [options] <group>
Used to manage the configuration of a group under the Linux operating system. The command must be run as sudo in order to have administrator access.
**Example:**
The following example uses gpasswd to remove a user from a group. The -d option followed by the username indicates that a user should be deleted:

```
$ sudo gpasswd -d jaggermick beatles
Removing user jaggermick from group beatles
```

## groupadd

groupadd [options] <groupname>
Adds a group to a computer running the Linux operating system. The command must be run as sudo in order to have administrator access.
**Example:**
The following example uses the groupadd command to create a group named beatles:

```
$ sudo groupadd beatles
```

## groupdel

groupdel [options] <groupname>
Deletes a group from the computer. The command must be run as sudo in order to have administrator access.
**Example:**
The following example uses the command groupdel to delete the group named beatles from the system:

```
$ sudo groupdel beatles
```