Cheat sheet

Linux commands

This cheat sheet presents a list of command-line executables that are frequently used by developers working on a computer running the Linux operating system. The commands are organized by category.

Application management commands

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

which

```
which <command>
```

Describes the location of an application in the computer's file system. If an application is not installed on the computer and its parent directory is not part of the system's `$PATH`, `which` will report an error.

Example:

The following example invokes the `which` command against the command `clear` and shows the result of the `which` command:

```
$ which clear
/usr/bin/clear
```

yum

```
yum
```

The application installation and removal tool for Fedora, CentOS, and Red Hat Enterprise Linux (RHEL).

Example:

The following command installs the `net-tools` application, which has many handy utilities such as `netstat`:

```
sudo yum -y install net-tools
```

Console and output management commands

Commands in this section apply to working with data sent to stdout or displayed in a computer's terminal window.

cat

```
cat <path/to/filename>
```

Displays the contents of `<path/to/filename>` to standard out.

Example:

The following command displays the contents of the `system-release` file that describes the version of RHEL, CentOS, or Fedora that's running (`$` is the command-prompt symbol):

```
$ cat /etc/system-release
Red Hat Enterprise Linux release 8.5 (Ootpa)
```

clear

Clears the terminal screen.

Example:

```
$ clear
```

echo

```
echo <string>
```

Displays a string to stdout or to a file.

Example:

This example demonstrates how to invoke `echo` to display a string in the console:

```
$ echo "Hello World"
Hello World
```

This example demonstrates how to invoke `echo` and save the output to a file named `data.txt`:

```
$ echo "Hello World" > data.txt
```

top

```
echo <string>
```

Displays information about the running Linux processes.

Example:

The following command displays the `top` command with the result piped to the `more` command in order to view the first portion of the output:

```
$ top | more
top - 12:02:29 up 5 days, 20:20,  2 users,  load average: 0.01, 0.02, 0.00
Tasks: 201 total,   2 running, 199 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  6.2 sy,  0.0 ni,93.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
st
MiB Mem :   7770.8 total,   5409.8 free,   1240.8 used,   1120.2 buff/cache
MiB Swap:   8092.0 total,   8092.0 free,      0.0 used.   6205.6 avail Mem
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+
COMMAND
  82399 guest  20   0   65584   5120   4212 R   5.9   0.1   0:00.02 top
      1 root      20   0  175932  14212   8924 S   0.0   0.2   0:06.21
systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.13
kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00
rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00
rcu_par_gp
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00
kworker/0:0H-events_highpri
      9 root       0 -20       0      0      0 I   0.0   0.0   0:00.00
mm_percpu_wq
     10 root      20   0       0      0      0 S   0.0   0.0   0:02.73
ksoftirqd/0
     11 root      20   0       0      0      0 R   0.0   0.0   0:01.10
rcu_sched
     12 root      rt   0       0      0      0 S   0.0   0.0   0:00.00
migration/0
     13 root      rt   0       0      0      0 S   0.0   0.0   0:00.04
watchdog/0
     14 root      20   0       0      0      0 S   0.0   0.0   0:00.00
cpuhp/0
     16 root      20   0       0      0      0 S   0.0   0.0   0:00.00
kdevtmpfs

--More--
```

Environment variables commands

Commands in this section apply to working with a Linux computer's environment variables.

env

```
env
```

Displays all environment variables running on the system.

Example:

The following example invokes `env` to get a system's environment variables and then pipes the result to the `more` command to show a portion of the environment variables (note that the environment variable `LS_COLORS` has been omitted because its value is quite lengthy):

```
$ env | more
SSH_CONNECTION=192.168.86.20 54276 192.168.86.34 22
LANG=en_US.UTF-8
HISTCONTROL=ignoredups
HOSTNAME=localhost.localdomain
which_declare=declare -f
XDG_SESSION_ID=11
USER=reselbob
SELINUX_ROLE_REQUESTED=
PWD=/home/reselbob
SSH_ASKPASS=/usr/libexec/openssh/gnome-ssh-askpass
HOME=/home/reselbob
SSH_CLIENT=192.168.86.20 54276 22
--More--
```

export

```
export <env var name>=<env var value>
```

Creates an environment variable with a value and then exports the environment variable or value pair to the system.

Example:

The following command creates an environment variable named `WEB_PAGE` and sets the value to `https://www.redhat.com/en`. The `echo` command confirms the value of the environment variable:

```
$ export WEB_PAGE="https://www.redhat.com/en"
$ echo $WEB_PAGE
https://www.redhat.com/en
```

printenv

```
printenv <environment_variable_name>
```

Prints a particular environment variable to the console.

Example:

The following example uses `printenv` to print the value of the environment variable `HOSTNAME` and then displays the result:

```
$ printenv HOSTNAME
localhost.localdomain
```

source

```
ource </path/to/filename>
```

Executes commands stored in a file from within the current shell, and can also be used to refresh environment variables.

By default a new shell is launched to run a script. Therefore, changes to environment variables are not visible in the current shell.

Example:

The following example uses the `cat` command to create a file named `new_vars.sh`. The file contains an export instruction set for the environment variable `ALT_USER=barry`. Then, the `source` command is called to set the environment variable into the system. The `echo` command verifies that the environment variable `ALT_USER` is in force:

```
$ cat <<EOF > new_vars.sh
> #!/bin/bash
> export ALT_USER=barry
> EOF
$ source ./new_vars.sh
$ echo $ALT_USER
barry
```

File and directory management

Commands in this section apply to working with the files and directories on Linux computers.

cd

```
cd </path/to/directory>
```

Change to another current directory.

Example:

The following example changes the current directory to the user's home directory:

```
cd ~/
```

cp

```
cp </path/to/source/filename> </path/to/target/filename>
```

Copies the contents of the source directory or file to a target directory or file.

Example:

The following example copies the contents of the file `helloworld.txt` to the file named `helloworld.bak`. It then executes the `cat` command to verify that the file and its contents have been copied:

```
$ cp helloworld.txt helloworld.bak
$ cat helloworld.bak
Hello World!
```

find

```
sudo find <starting/directory> -name <file/directory name>
```

Finds a file or directory by name.

Example:

The following command finds a file named `hostname` starting from the root (`/`) directory of the computer's file system. Note that the command starts with `sudo` in order to access files restricted to the `root` user:

```
$ sudo find / -name hostname
/proc/sys/kernel/hostname
/etc/hostname
/var/lib/selinux/targeted/active/modules/100/hostname
/usr/bin/hostname
/usr/lib64/gettext/hostname
/usr/share/licenses/hostname
/usr/share/doc/hostname
/usr/share/bash-completion/completions/hostname
/usr/share/selinux/targeted/default/active/modules/100/hostname
/usr/libexec/hostname
```

grep

```
grep <search_expression> <input>
```

```
$ less -N ~/.bashrc
```

Here's the result:

```
1 # .bashrc
2
3 # Source global definitions
4 if [ -f /etc/bashrc ]; then
5         . /etc/bashrc
6 fi
7
8 # User specific environment
9 if ! [[ "$PATH" =~ "$HOME/.local/bin:$HOME/bin:" ]]
10 then
11     PATH="$HOME/.local/bin:$HOME/bin:$PATH"
12 fi
13 export PATH
14
15 # Uncomment the following line if you don't like systemctl's auto-paging
feature:
16 # export SYSTEMD_PAGER=
17
18 # User specific aliases and functions
19 PS1="$ "
```

ls

```
ls [options] </path/to/directory>
```

Lists the contents of a directory. Defaults to the current directory.

Example:

This example shows how to list all the directories in the current directory:

```
$ ls
code  docs  images
```

This example lists all the files and directories in the current directory using the option `-l` which denotes a long listing view of the output:

```
$ ls -l
total 0
drwxrwxr-x. 2 guest guest  6 Jan 12 11:33 code
drwxrwxr-x. 2 guest guest 25 Jan 12 11:37 docs
drwxrwxr-x. 2 guest guest  6 Jan 12 11:34 images
```

This example lists all the files and directories in the current directory along with the hidden files, using the long listing option `-l` and the show-hidden-files option `-a`:

```
$ ls -la
total 4
drwxrwxr-x. 5 guest guest 60 Jan 12 11:36 .
drwxr-xr-x. 3 guest guest 68 Jan 12 11:33 ..
drwxrwxr-x. 2 guest guest  6 Jan 12 11:33 code
drwxrwxr-x. 2 guest guest 25 Jan 12 11:37 docs
drwxrwxr-x. 2 guest guest  6 Jan 12 11:34 images
-rw-rw-r--. 1 guest guest 15 Jan 12 11:36 .secrets
```

This example lists all the files and directories in the subdirectory named docs using the long listing option `-l`:

```
$ ls -l docs
total 4
drwxrwxr-x. 2 guest guest  6 Jan 12 11:44 drafts
-rw-rw-r--. 1 guest guest 49 Jan 12 11:37 hithere.txt
-rw-rw-r--. 1 guest guest  0 Jan 12 11:45 notes.txt
```

mkdir

```
cd <directory_name>
```

Creates a directory.

Example:

Creates a new directory named `documents` in the user's home directory:

```
mkdir ~/documents
```

more

```
more [options] </path/to/filename or stdout>
```

Allows a user to view and traverse the content of a file or stdout. The command `more` invokes itself within a distinct command-line user interface. To exit the process users press the `q` key.

Example:

This example uses the more command to display the first four lines of the file `/etc/passwd`. Users can then traverse the rest of the file one line at time by striking the `<ENTER>` key:

```
$ more -4 /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
--More--(5%)
```

This example illustrates using the `more` command to process stdout data. The example pipes the result of running `ls` against the directory /etc. The command `more` displays the first four lines of the output from stdout as declared in the option `-4`. Users can traverse through the rest of stdout a line at time by pressing the `<ENTER>` key:

```
$ ls /etc | more -4
accountsservice
adjtime
aliases
alsa
--More--
```

mv

```
mv <source file/directory> <target file/directory>
```

Moves a file or directory. The `mv` command transfers all the contents from the source file or directory to the new location.

Example:

This example moves the directory `documents` to the directory `docs-bak`. When `move` is invoked, the source directory will be renamed `docs-bak`:

```
mv ./documents ./docs-bak
```

This example moves the contents of the file `hithere.txt` in the directory `documents` to a file named `new_hithere.txt` in the same directory:

```
mv ./documents/hithere.txt ./documents/new_hithere.txt
```

pwd

```
pwd
```

Displays the name of the present working directory.

Example:

The following example displays the invocation and result of using the command `pwd` in the `HOME` directory for a user named `guest`:

```
$ pwd
/home/guest
```

rm

```
rm [options] <file or directory>
```

Removes a file or directory.

Example:

This example removes the file named `hithere.txt` from the current directory (`$`indicates the command-line prompt):

```
$ rm hithere.txt
```

This example removes the directory named `documents` along with all the files and subdirectories. The options `-rf` force the removal recursively:

```
$ rm -rf ./documents
```

tar

```
tar [options] <archive filename> <file or directory to be compressed>
```

Compresses and decompresses files or directories.

Example:

This example compresses a directory named `documents`, shows the output of the `tar` command, and then invokes the `ls` command to list the contents of the current directory:

```
$ tar cvzf docs.tar.gz documents/
documents/
documents/1.txt
documents/2.txt
documents/3.txt
documents/4.txt

$ ls
docs.tar.gz  documents
```

This example extracts the contents of the compressed file `docs.tar.gz` into an existing directory named `new-docs`:

```
$ tar -xvf docs.tar.gz -C  ./new-docs
```

Help commands

The command in this section applies to working with the command-line help documentation provided on a Linux computer.

man

```
man <path/to/command>
```

Displays the internal help documentation for a given command.

Example:

The following example shows how to display the command-line help documentation for the command `cp` :

```
$ man cp
```

Network commands

Commands in this section apply to working with networks on and from a Linux computer.

curl

```
curl [options] <url>
```

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

Example:

This example downloads a web page from the Red Hat Developer website and implements the `-o` option to save the page to the file `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
```

ip

```
ip [ OPTIONS ] OBJECT { COMMAND | help }
```

Gets the IP information for the physical or virtual machine.

Example:

The following example returns the IP address information associated with network interfaces on the current machine:

```
$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state
UP group default qlen 1000
    link/ether 08:00:27:45:95:d3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.86.34/24 brd 192.168.86.255 scope global dynamic
noprefixroute enp0s3
       valid_lft 80971sec preferred_lft 80971sec
    inet6 fe80::a00:27ff:fe45:95d3/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state
DOWN group default qlen 1000
    link/ether 52:54:00:f7:2c:71 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
4: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0
state DOWN group default qlen 1000
    link/ether 52:54:00:f7:2c:71 brd ff:ff:ff:ff:ff:ff
```

netstat

```
netstat [options]
```

Displays information about network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.

Example:

The following example uses `netstat` to list the status of ports and the process using the particular port:

```
$ sudo netstat -anp | grep tcp
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
1/systemd
tcp        0      0 192.168.122.1:53        0.0.0.0:*               LISTEN
1987/dnsmasq
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
1084/sshd
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
1087/cupsd
tcp       32      0 192.168.86.34:57152     8.43.85.13:443
CLOSE_WAIT  6588/gnome-shell
tcp        0      0 192.168.86.34:22        192.168.86.20:56253
ESTABLISHED 82512/sshd: guest
```

```
tcp6       0      0 :::111                  :::*                    LISTEN
1/systemd
tcp6       0      0 :::22                   :::*                    LISTEN
1084/sshd
tcp6       0      0 ::1:631                 :::*                    LISTEN
1087/cupsd
tcp6       0      0 :::9090                 :::*                    LISTEN
1/systemd
```

ssh

```
ssh [options] <ip_address>
```

Secure shell is an encrypted network protocol that provides remote login and command execution capabilities. On Windows, you would use `PuTTY` and `WinSCP`. An `ssh.exe` is also available via Cygwin, as well as with a Git installation.

Example:

The following example shows how to use `ssh` to log into a remote computer that has the IP address `192.168.86.11`:

```
$ ssh 192.168.86.11
```

wget

```
wget [options] <url>
```

Downloads files from the Internet. Supports the HTTP, HTTPS, and FTP protocols. You can use `wget` as an alternative to `curl`.

Example:

The following example uses the `wget` command to download a file from the URL https://developers.redhat.com/articles/2022/01/11/5-design-principles-microservices, then uses the `-o` option to save the content to a file named `article.html`:

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

Process management commands

Commands in this section apply to working with processes running on a Linux computer.

&&

```
<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
```

kill

```
kill <process_id>
```

Removes a running process from memory.

Example:

The following example removes the process with the ID of `10` (`$` is the command-line prompt symbol):

```
$ kill 10
```

ps

```
ps [options]
```

Displays the status of the 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--
```

System control commands

Commands in this section apply to controlling the operation of a physical Linux computer.

poweroff

```
poweroff
```

Shuts down a computer. Must be run as `sudo`.

Example:

```
$ sudo poweroff
```

Note that `$` indicates the command-line prompt.

restart

```
restart
```

Restarts a computer. Must be run as `sudo`.

Example:

```
$ sudo restart
```

User management commands

The command in this section applies to working with users on a Linux computer.

whoami

```
whoami
```

Displays the user ID.

Example:

The following example shows the invocation for a user with the login ID of `jerryr`:

```
$ whoami
jerryr
```