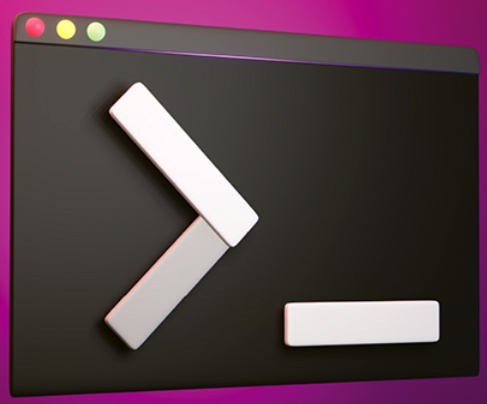
cat, ln, nan
stat, tee, a
bunzip2, he
iconv, attr
getcap, con
at, batch, z
iotop, addg
usermod, ba



```
user$ rdfind verz1 verz2
Now scanning "verz1", found 211 files.
Now scanning "verz2", found 215 files.
Now have 426 files in total.
Removed 0 files due to nonunique device
Total size is 85754591 bytes or 82 MiB
Removed 103 files due to unique sizes fr
Now eliminating candidates based on firs
    removed 4 files from list. 319 files left.
Now eliminating candidates based on las
    removed 0 files from list. 319 files left.
Now eliminating candidates based on sha
    removed 2 files from list. 317 files left.
It seems like you have 317 files that are n
Totally, 39 MiB can be reduced. Now mak
```

Linux Command Reference

Shell Commands from A to Z

Michael Kofler

Rheinwerk Computing

Overview of Commands Sorted by Topic

Managing Files

Finding Files

Compressing and Archiving Files

Processing Text Files

Access Control Lists (ACLs) and Extended Attributes (EAs)

Converters

Managing Processes

Managing Users and Groups

Administrating the File System

Logical Volume Manager (LVM)

SELinux and AppArmor

Package Management

Network Administration

Hacking and Security

Printer, Database, and Server Administration

Audio Functions and Hardware Management

Bluetooth

Kernel

System Start and Stop, init System, Logging, and GRUB

Virtualization, Containers, Cloud

Terminal and Text Console

Online Help

Graphics System and Gnome

Miscellaneous

"bash" Programming

"bash" Variable Management

Additional "bash" Commands and Special Characters

Configuration Files

Keyboard Shortcuts

Introduction

This book contains short descriptions of the most important Linux commands for managing the file system, for starting and terminating processes, for editing text files, for other administrative tasks, and for `bash` programming. The book also summarizes the syntax of basic configuration files and contains a keyboard shortcut reference for the `emacs`, `nano`, and `vi` editors, as well as some other interactive commands such as `less` or `info`.

The aim of this book is to provide a compact reference work for using Linux in the terminal. The basic principle of this book is: *less is more*. The book cannot and is not intended to replace the `man` and `info` pages of complex commands! Thus, you will still have to look up or research exotic options yourself.

With this book, however, I'm trying to relieve you of the work of searching through the often dozens of pages of original documentation for options for everyday use. Numerous examples also show the basic use of a command at a glance.

Sometimes, you're looking for a command for a specific task, but don't know its name or have just forgotten it. The thematically organized table of contents is intended for these cases.

Depending on which distribution you are using, some commands aren't available by default and must be installed separately. There are also distribution-specific commands that are only available in certain distributions, such as the package management commands `dpkg` and `apt` (Debian, Ubuntu), `rpm` and `dnf` or `yum` (Fedora, Red Hat), and `zypper` (SUSE). This is pointed out in the respective command description.

What Is a Command?

Linux doesn't distinguish between commands (as described in this book) and programs such as Firefox, LibreOffice, or GIMP. Here, "command" refers to programs without a graphical user interface, which are usually executed in a terminal window.

In this book, I also describe some commands that aren't real programs at all, but only commands of the currently active shell. I assume that you're using the `bash` (Bourne Again Shell) or the largely compatible `zsh`. These shells are used in most Linux distributions for the interactive execution of commands. An example of a shell command is `cd` to change the current directory.

Options

Most of the commands described in this book are controlled by options. The options are specified *before* all other parameters. There are two ways of notation for many commands: `-x` for short options (one letter) and `--xyz` for long options (multiple letters).

Thus, the following two `ls` commands are equivalent and display all files and directories in the `/usr` directory:

```
user$ ls -l -A /usr
user$ ls --format=long --almost-all /usr
```

For some commands, multiple options can be specified as a group (i.e., `-ab` instead of `-a -b`). Some commands also work with options that are specified after the actual parameter(s). However, this shouldn't lead you to the conclusion that this applies to all commands!

```
user$ ls -lA /usr
user$ ls /usr -lA
```

For a few commands, the order of the parameters has an influence on how the command is executed. If options are specified that are logically mutually exclusive, the last option specified applies.

man, info, and help

To avoid making this book unnecessarily long, I'll only describe the most important options. A complete overview of all options for the majority of commands is provided by `command name --help`. More detailed information is usually contained in the manual pages, which you can access via `man` *`name`* or `man 1` *`name`*. For some commands, the `man` pages only contain a reference to the `info` texts, which are displayed accordingly via `info` *`name`*.

For commands that are directly integrated into the `bash` (e.g., `cd`), `man` *`name`* leads to the `man` page of the `bash`. The command is actually described there, but searching through the very long documentation is tedious. In this context, it's more helpful to use `help` *`name`*.

The help texts from `man` and `info` sometimes overshoot the mark. If you're just looking for a few examples of how to use a command, you should use either this book or the `tldr` command. Incidentally, the command name stands for *Too long, didn't read.* :-)

ncdu [options] [directory]

ncdu is an interactive variant of the du command. Usually, it's simply called without options and parameters. It then displays the subdirectories in the current directory that take up the most space.

You can now navigate through the subdirectories using the cursor keys and Enter. Various other functions are accessible via keyboard shortcuts. An overview of the most important shortcuts can be obtained by entering ? But be careful—D deletes the current file or directory after a query.

- -r
  Runs the command in read-only mode. Accidental deletion of files or directories is therefore impossible.
- -x
  Remains in the current file system, that is, doesn't take any mount directories into account.

Example

The space required by the directories in the home directory is also symbolized by bars:

```
user$ ncdu
--- /home/kofler ---
     1.5 GiB [##########] /Nextcloud
     1.0 GiB [######    ] /Downloads
   528.5 MiB [###       ] /Images
   209.4 MiB [#         ] /Documents
   ...
```

needs-restarting [options]

The needs-restarting command available on Fedora, Red Hat, and related distributions specifies whether the computer or individual programs need to be restarted due to an update of the kernel, a basic library, or a firmware file.

The command isn't available for distributions based on Debian or Ubuntu. There, the /var/run/reboot-required file indicates that a reboot is required.

- -r
  Indicates whether and why a computer restart is required.
- -u
  Lists processes of the current user that need to be restarted. This option applies by default.

Example

The following commands show the status of a computer that should be restarted:

```
root# needs-restarting -r
Core libraries or services have been updated since boot-up:
  * glibc
Reboot is required to fully utilize these updates.
More information: https://access.redhat.com/solutions/27943
root# needs-restarting | sort -n
782 : /usr/lib/polkit-1/polkitd --no-debug
783 : /usr/libexec/power-profiles-daemon
784 : /usr/bin/qemu-ga --method=virtio-serial ...
787 : /usr/libexec/accounts-daemon
...
```

netplan [command]

Netplan (*https://netplan.io*) is a framework developed by Ubuntu that configures and integrates other network backends such as the NetworkManager and networkd (a systemd component, see `networkctl`). It's used in Ubuntu and evaluates the configuration files located in `/etc/netplan` (see `/etc/netplan/netplan.yaml`).

- `apply`
  Runs `netplan generate` and then activates the changed configuration.
- `generate`
  Reads the Netplan configuration from the YAML files in `/etc/netplan`, `/lib/netplan` and `/var/netplan`, and then generates the corresponding configuration files for the network backends. The new files end up in `/run/NetworkManager` or in `/run/systemd/network`.

  The new configuration isn't activated! The changed settings only take effect when the respective backend is requested to evaluate the files (or at the next restart).
- `ip leases` *interface*
  Outputs the current DHCP lease data for the relevant interface. This only works if the networkd backend and a DHCP configuration are in use. Otherwise, the command will return misleading error messages.

netstat [options]

`netstat` provides information about the network activity on the local computer. If the command is called without options, it returns a list of all open internet connections and sockets.

- `-a`
  Also takes into account inactive sockets, that is, services that are waiting for a connection on a network port (`LISTEN` state).
- `-e`
  Also specifies the user for internet connections.
- `-n`
  Provides numerical addresses and port numbers instead of host and network service names.
- `-p`
  Also specifies the process number (PID) and the process name of the program responsible for the connection. `netstat` requires `root` permissions so that it can provide information about nonnative processes.
- `-t` / `-u` / `-w` / `-X`
  Limits the output to connections that use the TCP, UDP, Raw, or Unix protocol.

Example

The following command lists all active connections (`established`) or monitored ports (`LISTEN`). The output is heavily abridged due to space limitations.

```
root# netstat -atupe
Active Internet connections (servers and established)
Proto Local Address           Foreign Addr   State    User     PID/Prog name
tcp   *:nfs                   *:*            LISTEN   root     -
tcp   *:ldap                  *:*            LISTEN   root     5842/slapd
tcp   localhost:mysql         *:*            LISTEN   mysql    5785/mysqld
tcp6  [::]:ssh                [::]:*         LISTEN   root     5559/sshd
udp   *:nfs                   *:*                     root     -
```

networkctl [command]

`networkctl` is one of the administration commands from the systemd family. If networkd is used as the network backend, `networkctl` helps to analyze the network status.

Although networkd is installed by default on many distributions, it's only active on a few of them. One of the exceptions is Ubuntu Server, where networkd serves as a backend for Ubuntu's own Netplan system (see `netplan`). networkd evaluates configuration files from the `/etc/systemd/network`, `/lib/systemd/network`, and `/run/systemd/network` directories (see `/etc/systemd/network/networkd.network`).

- `list`
  Lists all network interfaces. The `SETUP` column of the result shows whether the interface is controlled by networkd (`configured`) or not (`unmanaged`).

- `lldp`
  Shows other devices in the network that were discovered via the *Link Layer Discovery Protocol* (LLDP).
- `status` [*interface*]
  Provides detailed information about the status of a network interface. If no interface is specified, the command attempts to summarize the entire network status.

Examples

The following outputs of `networkctl` were created on an Ubuntu server in a virtual machine:

```
root# networkctl list
IDX  LINK      TYPE       OPERATIONAL   SETUP
  1  lo        loopback   carrier       unmanaged
  2  ens3      ether      routable      configured
root# networkctl status ens3
      Link File: /lib/systemd/network/99-default.link
   Network File: /run/systemd/network/10-netplan-ens3.network
           Type: ether
          State: routable (configured)
            ...
     HW Address: 52:54:00:0a:8d:fc
        Address: 138.201.20.182
                 fe80::5054:ff:fe0a:8dfc
        Gateway: 138.201.20.176 (Fujitsu Technology Solutions GmbH)
            DNS: 213.133.100.100
                 213.133.98.98
   Search Domains: ubuntu-buch.info
```

The second listing shows the status summary of another server installation with IPv6 configuration:

```
root# networkctl status
        State: routable
 Online state: online
      Address: 168.119.33.110 on eth0
               172.17.0.1 on docker0
               2a01:4f8:242:1f88::4 on eth0
               fe80::5054:ff:fe4a:7321 on eth0
      Gateway: 168.119.33.119 on eth0
               2a01:4f8:242:1f88::2 on eth0
          DNS: 213.133.100.100
```

```
213.133.99.99
2a01:4f8:0:1::add:1010
2a01:4f8:0:1::add:9999
```

newaliases

The /etc/aliases file contains an alias list for the email server, which ensures, for example, that all emails addressed to postmaster are forwarded to root. For changes to this file to be taken into account by the mail server, you must run newaliases.

newgrp [group name]

The newgrp command determines the currently active group of a user who belongs to several groups. The active group determines which group newly created files belong to. The groups available for selection can be determined via groups. If no group name is specified, the primary group will be used. This group is also automatically considered an active group after a login.

Example

In the following example, the newgroup command makes docuteam the active group of the user kofler. As a result, the newly created newfile file is assigned to the docuteam group and can be edited by other members of the documentation team.

```
user$ groups
kofler docuteam wheel
user$ newgroups docuteam
user$ touch newfile
```

newusers file

newusers reads a text file and creates a new user for each line. The text file basically has the same format as /etc/passwd. However, the passwords must be entered unencrypted in the second column. Most of the other parameters (UID, GID, etc.) are optional. newusers creates a new account for each specified user, whereby the associated groups are also created if required. For missing parameters, newusers selects suitable default values, taking into account the settings in /etc/login.defs.

Note that newusers creates new home directories if their location is specified in the sixth column. However, the command doesn't take care of copying the contents of /etc/skel there.

Example

The following lines show the minimalist text file `users.txt`, which corresponds to the requirements of `newusers`. As `users.txt` contains passwords in plain text, you must make sure that no one other than `root` can read the file or that you delete the file again after running `newusers`.

```
hawks:secret1:::Howard Hawks:/home/hawks:/bin/bash
moore:secret2:::Grace Moore:/home/moore:/bin/bash
smith:secret3:::Peter Smith:/home/smith:/bin/bash
```

`newusers` now creates the three new users `hawks`, `moore`, and `smith`, as well as primary groups of the same name. `newusers` itself decides on suitable UIDs and GIDs.

```
root# newusers users.txt
```

nft command [options]

Most current Linux distributions now use the new *nftables* firewall system, which replaces its predecessor, *Netfilter*. However, the old `iptables` command is often still used to configure the firewall, which continues to work thanks to a compatibility layer.

However, new or nftables-specific functions can only be controlled using the `nft` command. The following description provides a rough overview of the most important subcommands and options of this command. You can find a lot more details about the functionality of nftables and the syntax of `nft` in `man nft` and the following wiki:

https://wiki.nftables.org

Before you call `nft`, you should make sure that a firewall isn't already running on your distribution. On Fedora, RHEL, and SUSE, for example, *Firewalld* is active (see `firewall-cmd`). Custom-defined firewall rules then often result in conflicts with the firewall of your distribution.

nft Terminology

A separate terminology applies to `nft`: Firewall *rules* are always part of a *chain*. *Tables*, in turn, consist of multiple chains. Unlike `iptables`, there are neither predefined tables nor predefined chains. Thus, you can compose as many custom tables as you like from chains also defined by yourself.

Tables, chains, and rules apply to different types of network packets, which are assigned to the following *families*:

- `ip`
  Rules for IPv4 only.

- `ip6`
  Rules for IPv6 only.
- `inet`
  Rules that apply equally to IPv4 and IPv6.
- `arp`
  Level 2 rules, which are analyzed before Level 3 rules.
- `bridge`
  Switching rules.
- `netdev`
  Low-level rules that are analyzed before all other rules and enable, for example, a particularly efficient defense against DDOS attacks.

nft Options

- `-a`
  Integrates *handles* into the output of `nft list`. These are numbers that uniquely identify each object within a group (e.g., a rule within a chain). Handles make it possible to delete or replace individual objects or to specify the position where new objects are to be inserted.
- `-f` *`fname`*
  Reads the commands to be executed from the specified file. Two syntax variants are permitted: the file can either contain ordinary `nft` commands line by line or a hierarchically structured sequence of tables, chains, and rules. The structure of the file for the second variant corresponds to the output of `nft list ruleset`.
- `-S`
  Uses the names of network services specified in `/etc/services` instead of port numbers (e.g., `ssh` instead of 22).
- `-v`
  Shows the version number.

nft Commands

- `add chain` *`family tname cname`*
  Creates a new chain named *`cname`* in the specified table.
- `add rule` *`family tname cname matches statements`*
  Adds a new rule to the specified *`cname`* chain. Here, *`matches`* specifies the packets to which the rule applies. *`statements`* describes what should happen to the respective packets.

Instead of `add`, the `insert` and `replace` keywords are also permitted to insert a rule at a specific position in the rule list or to replace an existing rule with a new one.

- `add table` *`family tname`*
  Creates a new table for the specified family.
- `delete/flush chain` *`family tname cname`*
  Removes the specified chain from the table. `flush` deletes all rules of the chain, but not the chain itself.
- `delete rule` *`family tname cname`* `handle` *`came`*
  Deletes the rule specified by *`handle`* from the *`cname`* chain.
- `delete/flush table` *`family tname`*
  Deletes the specified table. With `flush`, all chains and rules in the table are deleted, but the now empty table is retained.
- `flush ruleset`
  Deletes all tables, including all chains and rules contained therein. This corresponds to a complete reset of the firewall. The firewall now accepts every packet.
- `list tables`
  `list table family name`
  `list chain` *`family tname cname`*
  Lists tables, chains, and rules.
- `list ruleset`
  Lists all tables, chains, and rules. The output structured by curly brackets fulfills the syntax rules for `nft -f` and can therefore be used as the basis for a new rule file. With the additional `-j` option, the output is in JSON format.
- `monitor [filter criteria]`
  Displays nftables events (e.g., as debugging help).

Examples

The first two commands show which firewall tables are defined on the test computer (Fedora 32) and which chains and rules apply to the `firewalld` table within the `ip` family. (nftables allows the use of matching table names for different families. For this reason, the family must also be specified in the second `list` command.)

```
root# nft list tables
table bridge filter
table bridge nat
table inet firewalld
table ip firewalld
table ip6 firewalld
root# nft list table ip firewalld
table ip firewalld {
```

```
   chain nat_PREROUTING {
     type nat hook prerouting priority dstnat + 10; policy accept;
     jump nat_PREROUTING_ZONES
   }
   chain nat_PREROUTING_ZONES {
      iifname "enp1s0" goto nat_PRE_FedoraWorkstation
      goto nat_PRE_FedoraWorkstation
   }
   ...
```

`systemctl` now deactivates the default Fedora firewall daemon:

```
root# systemctl disable --now firewalld
```

Now, you can create your own firewall using `nft`. It's not usual to define the rules by repeatedly calling `nft`. It's much more elegant to formulate the rules in a text file and run this file like a script. To do this, you must enter `nft -f` as a shell interpreter in the line.

```
#!/sbin/nft -f
# delete existing firewall
flush ruleset
# new table for IPv4 and IPv6
table inet myfilter {
   chain input {
      type filter hook input priority 0; policy drop;
      # block faulty packets
      ct state invalid drop
      # always accept packets from self-generated connections
      ct state {established, related} accept
      # accept internal network traffic
      iif lo accept
      # Block packets to loopback but from external addresses
      iif != lo ip daddr 127.0.0.1/8 drop
      iif != lo ip6 daddr ::1/128 drop
      # accept ICMP packets
      ip protocol icmp accept
      ip6 nexthdr icmpv6 accept
      # accept SSH connection from outside
      tcp dport 22 accept
      # for server configuration: more rules for
      # HTTP + HTTPS (Port 80 + 443), Samba etc.
      # ...
```

```
    }
    # no forwarding
    chain forward {
       type filter hook forward priority 0; policy drop;
    }
}
```

Now you can run the script:

```
root# chmod +x myfirewall
root# ./myfirewall
```

`nft -f` first performs a syntax check. If it detects errors, it aborts the process with an error message. In this case, the previous firewall is retained.

The preceding script has been slightly modified from the Gentoo wiki. You can find more examples there and in the ArchLinux Wiki:

https://wiki.gentoo.org/wiki/Nftables/Examples

https://wiki.archlinux.org/title/Nftables

ngrep [options] [grep-search-expression] [pcap-filter-expression]

`ngrep` is referred to as a *packet sniffer*. The command reads the network traffic of a port and filters it. `ngrep` provides similar functions to the `tcpdump` command and, like this, uses the `pcap` library to read the network packets. Unlike `tcpdump`, however, `ngrep` also takes the contents of the packages into account. Naturally, this only works for nonencrypted protocols, such as HTTP or FTP.

The search expression must be formulated as a regular pattern as in `grep`. The syntax summarized in `tcpdump` applies to the `pcap` filter expression.

- `-d` *`interface`*`|any`
  Determines the network interface.
- `-i`
  Ignores uppercase and lowercase in the `grep` search expression.
- `-v`
  Inverts the search. `ngrep` therefore only returns the packets in which the `grep` search pattern was *not* recognized.
- `-w`
  Interprets the `grep` search expression as a word.
- `-W byline`
  Takes line breaks into account during output, resulting in more legible output.

Example

The following example listens on all interfaces for HTTP packets containing the keywords `user`, `pass`, and so on:

```
root# ngrep -d any -i 'user|pass|pwd|mail|login' port 80
interface: any
filter: (ip or ip6) and ( port 80 )
match: user|pass|pwd|mail|login
T 10.0.0.87:58480 -> 91.229.57.14:80 [AP] POST /index.php
  HTTP/1.1..Host: ... user=name&pass=geheim&login=Login
...
```

nice [options] program

`nice` starts the specified program with a reduced or increased priority. The command can be used to start non-time-critical programs with low priority so as not to affect the rest of the system too much.

- `-n` +/-*n*
  Specifies the `nice` value. By default (i.e., without `nice`), programs are started with the `nice` value of `0`. A value of `-20` means highest priority, and a value of `+19` means lowest priority. Values less than `0` may only be specified by `root`; that is, most users can only start programs with reduced priority with `nice`. If this option is omitted, `nice` starts the program with the `nice` value of `+10`.

Note that `nice` only controls the CPU load. If you want to reduce the I/O load of a command, it's better to use `ionice`.

Example

The following command starts the `mybackup.sh` script with lower priority:

```
user$ nice -n 10 mybackup.sh
```

nl [options] file

`nl` numbers all nonempty lines of the specified text file and writes the result to the standard output. By setting the numerous options, you can have page-by-page numbering, the numbering of headers and footers, and more.

nmap [options] hostname/ip-address/ip-address-range

The `nmap` command (*network mapper*) from the package of the same name performs a port scan and attempts to determine which network services are active on the specified

computer or in the specified network. `nmap` should only be used to analyze your own computers or after consultation with the respective administrator. A port scan of other computers can be interpreted as an attempted attack!

- `-A`
  Performs an extensive ("aggressive") scan, corresponding to `-sV -O -sC --traceroute`.
- `-F`
  Considers only the 100 most important ports from `/usr/share/nmap/nmap-services` (quick scan).
- `-iL` *file*
  Scans the IP addresses specified in the file.
- `-oN` *file* / `-oG` *file* / `-oX` *file*`.xml`
  Writes the results to an ordinary text file, to a text file that can be easily processed with `grep`, or to an XML file. Without the option, `nmap` uses the standard output and the ordinary text format.
- `-O`
  Attempts to recognize the operating system; this option must be combined with a scan option, such as `-sS`, `-sT`, or `-sF`.
- `-p1-10,22,80`
  Takes only the specified ports into account.
- `-Pn`
  Doesn't perform a ping test. This means that `nmap` considers all hosts to be online and performs a scan in any case (slowly!).
- `-sL`
  Lists all ports and specifies host names assigned in the past. This is particularly fast, but provides outdated data even from devices that are currently no longer online.
- `-sP`
  Performs only a ping test (fast).
- `-sS`
  Performs a TCP SYN scan (applies by default).
- `-sU`
  Takes UDP into account as well. This option can be used together with another `-s` option.
- `-sV`
  Tries to find out which service is provided on open ports (*service version detection*) or which program is responsible for the service and in which version.
- `-T0` to `-T5`
  Selects a timing schema:
  - `-T5` is the fastest.

  - `-T3` is the default.
  - `-T0` and `-T1` are extremely slow, but minimize the risk of the scan being detected.

- `-v`
  Outputs detailed information (*verbose*).

You must choose an `-s option` when calling. Only `-sU` may be combined with other `-s` options. In general, the right choice of options is a trade-off between thoroughness and speed.

In many cases, `nmap -v -A name` is sufficient to get an initial overview of the network services of the specified computer. Advanced `nmap` users can find more details on the `man` page and on the following websites:

https://insecure.org/nmap

https://nmap.org/book

Graphical user interfaces also exist for `nmap`, such as `nmap-frontend` or `zenmap`.

Examples

The following command performs a quick network scan on the local network (256 IP addresses). Thanks to the focus on the most important 100 ports, the job is done within about two seconds. The `nmap` outputs have been shortened for reasons of space and only show the results of two of the devices found:

```
root# nmap -F  -T4 10.0.0.0/24
Nmap scan report for imac (10.0.0.2)
  Host is up (0.00019s latency).
  PORT    STATE SERVICE
  22/tcp  open  ssh
  88/tcp  open  kerberos-sec
  445/tcp open  microsoft-ds
  548/tcp open  afp
  MAC Address: AC:87:A3:1E:4A:87 (Apple)
Nmap scan report for raspberrypi (10.0.0.22)
  Host is up (0.00038s latency).
  Not shown: 99 closed ports
  PORT   STATE SERVICE
  22/tcp open  ssh
  MAC Address: B8:27:EB:11:44:2E (Raspberry Pi Foundation)
...
Nmap done: 256 IP addresses (6 hosts up) scanned in 2.42 seconds
```

The second example shows the `nmap` result for a NAS device that is located in a different local network (again, it has been heavily shortened):

```
root# nmap -v -A 192.168.178.28
Nmap scan report for DiskStation.fritz.box (192.168.178.28)
Host is up (0.0023s latency).
...
Discovered open port 445/tcp on 192.168.178.28
Discovered open port 139/tcp on 192.168.178.28
Discovered open port 80/tcp on 192.168.178.28
Discovered open port 443/tcp on 192.168.178.28
Discovered open port 22/tcp on 192.168.178.28
Discovered open port 5001/tcp on 192.168.178.28
Discovered open port 5000/tcp on 192.168.178.28
Discovered open port 548/tcp on 192.168.178.28

PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 7.4 (protocol 2.0)
  ssh-hostkey:
    2048 5a:e7:3a:66:f4:99:9f:0a:0a:...   (RSA)
    256 06:1a:bf:9f:e9:d0:64:3a:92:49:... (ECDSA)
    256 ad:b7:7d:ab:ae:70:0a:c9:a6:0c:... (ED25519)
    ...
80/tcp   open  http        nginx
139/tcp  open  netbios-ssn Samba smbd 4.6.2
445/tcp  open  netbios-ssn Samba smbd 4.6.2
...
Nmap done: 1 IP address (1 host up) scanned in 64.19 seconds
```

nmblookup [options] workgroupname

`nmblookup` from the `samba-common-bin` (Debian/Ubuntu) or `samba-client` (Fedora/RHEL) package determines which devices or servers in the local network provide SMB shares. Previously, `smbtree` could be used for this, but this command requires the browsing functions of SMB1 and is obsolete in modern networks.

- `-S`
  Performs a status query for each server found and then displays the detailed results.
- `-T`
  Provides an ordered list of server names and IP addresses. In my experience, the results of `nmblookup` are more reliable if you run `nmblookup -S` first or repeat the command again after a few seconds. The first result is often incomplete.

Example

There are two SMB servers, a NAS device, and an appropriately configured Raspberry Pi in the local network:

```
user$ nmblookup -T WORKGROUP
DiskStation.fritz.box, 192.168.178.43  WORKGROUP<00>
pi5.fritz.box,         192.168.178.123 WORKGROUP<00>
```

nmcli [options] con|dev|nm command

Network Manager is usually controlled via a menu in the Gnome or KDE panel. You can also use the `nmcli` command to control network connections via the command line or a script.

The commands available for selection depend on which object they refer to. Options `-t`, `-p`, and so on can be used to control the form of the `nmcli` output—depending on whether the output is to be formatted properly or processed further by a script (for details, see `man nmcli`).

- `con down` *name* or `con down uuid` *n*
  Deactivates the specified connection.
- `con show`
  Lists all configured connections, specifying the name and UUID of each connection.
- `con up` *name* or `con up uuid` *n*
  Activates the specified connection.
- `dev disconnect` *name*
  Terminates the connection for the specified interface.
- `dev`
  Lists all network interfaces known to Network Manager and specifies their properties.
- `dev show`
  Displays detailed information on all network interfaces.
- `dev wifi hotspot`
  Configures the Wi-Fi adapter as a router.
- `dev wifi list`
  Provides a list of all Wi-Fi networks within range, including the signal strength. The command requires the `wpa_supplicant` service to be running.
- `general [status]`
  Displays the status of Network Manager.

- `networking [on|off]`
  Indicates whether there's a network connection or not. All network connections can be interrupted or restored by selecting `off` or `on`.
- `radio [wifi|wwan|wimax|all] [on|off]`
  Displays the status of the wireless networks or changes their status. `wifi` refers to Wi-Fi connections, `wwan` to mobile connections, and `wimax` to WiMAX technology.

Examples

The first command lists all connections known to Network Manager. The following two commands deactivate and reactivate the connection named *Wired connection 1*:

```
root# nmcli con show
NAME                    UUID          TYP             DEVICE
Wired connection 1      effd094d-...  802-3-ethernet  enp0s3
virbr0                  207d9f4f-...  bridge          virbr0
root# nmcli con down   id 'Wired connection 1'
root# nmcli con up     id 'Wired connection 1'
```

The second example filters out the IP addresses of the name server from the detailed interface information. This is particularly useful on Ubuntu, where these addresses aren't listed in `/etc/resolv.conf`. Rather, `resolv.conf` references a local name server in this distribution, that is, the address 127.0.0.1.

```
root# nmcli dev show | grep DNS
IP4.DNS[1]:  10.0.0.138
IP4.DNS[2]:  4.4.8.8
```

The third command sets up a hotspot on a Raspberry Pi connected to the LAN via ethernet cable:

```
root# device wifi hotspot ssid 'wlan-name' password 'wlan-pw'
```

nohup command

If you start a command as a background process in a shell window and then close the window, or if you start the command in a text console and then log out, the background process is automatically terminated. As a rule, this is reasonable behavior.

However, sometimes you want to start a process that continues to run after you log out—and that's exactly what `nohup` is there for. The command must be specified with its full path. It can't write text outputs to the standard output. If necessary, such outputs are redirected to the `nohup.out` file in the local directory.

Example

In the following example, an administrator logs in to a server via `ssh`, starts a backup script in the background, and then logs out again. The backup script continues to run.

```
someone@localhost$ ssh user@remotehost
user@remotehost$ nohup backup-script &
user@remotehost$ exit
```

nproc

`nproc` indicates how many CPU cores are available. For CPUs with hyperthreading, the virtual cores are also included in the calculation. The `lscpu` command and the contents of the `/proc/cpuinfo` file provide more detailed information about the CPU.

nvidia-xconfig [options]

`nvidia-xconfig` helps to configure the proprietary NVIDIA graphics driver. The command is only available if this driver is installed.

If the command is executed without parameters, it creates or changes the `/etc/X11/xorg.conf` file so that the NVIDIA graphics driver will be used in the future. If the graphics system is working after restarting the computer, the `nvidia-settings` graphical user interface can be used for further configuration.

- `-c file`
  Uses the specified file instead of `/etc/X11/xorg.conf`.
- `--mode=`*W*x*H*
  Adds a graphics mode for *W*×*H* pixels to `xorg.conf`.
- `--query-gpu-infos`
  Displays details of all available graphics cards and connected monitors.
- `-t`
  Reads `xorg.conf` and displays the settings it contains in a clear tree view.

openssl command

`openssl` from the package of the same name helps with the creation and further administration of certificates as well as private and public keys. The command uses the OpenSSL library. New certificates or keys are written to the standard output if you don't specify a target file via `-out`.

- `dhparam [options] [`*n*`]`
  Generates or manipulates parameter files for the Diffie-Hellman method. The *n*

parameter specifies the key size in bits. In the simplest case, `openssl dhparam -out keyfile 512` generates a Diffie-Hellman key with a length of 512 bits.

- `enc [options]`
  Reads data from the standard input, encodes or decodes it symmetrically (option `-d`), and writes the result to the standard output. You need to specify the algorithm to be used and the key with options, for example, in the format `-aes-256-cbc -pass file:mykeyfile`. Note that `openssl enc` is only suitable for relatively small amounts of data. To encrypt large files, it's better to use the `gpg` command.

- `genpkey [options]`
  Generates a private key. To write a 2,048-bit RSA key to a file, the following options are required:

  ```
  openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 –out server.key
  ```

  If the key itself is to be encrypted, you must specify the name of the encryption algorithm with an additional option, such as `-aes256`. During the execution of the command, `openssl` asks for an encryption password twice.

- `list-standard-commands | list-message-digest-commands | list-cipher-commands | list-cipher-algorithms | list-message-digest-algorithms | list-public-key-algorithms`
  Lists the commands, algorithms, and so on that are supported by `openssl`.

- `rand [options] n`
  Writes *n* bytes of pseudo-random data to the standard output. Note that the output is binary data, not ASCII text! The `-base64` option causes `openssl` to perform Base64 encoding in ASCII format. This increases the length of the resulting character string by around a third.

- `req [options]`
  Generates a request to sign a certificate (i.e., a *certificate signing request* [CSR] file). With `-new`, you can specify that a new certificate must be set up. `-key` specifies the key file to be used. Without this option, `openssl` generates a new RSA key for this purpose, which you can save in a separate file using `-keyout`. `openssl req -new -sha256 -key server.key -out server.csr openssl` then interactively asks for the key data of the certificate, that is, the country code, location, your name, host name (common name), and more. With the additional `-x509 -days n` options, `openssl` generates a self-signed certificate with a validity of *n* days instead of a certificate request.

- `rsa [options]`
  Processes RSA keys and converts them between different forms and formats. `-in file` and `-out file` specify the file from which the original key is read and where the new key is written. If you don't specify any further options, `openssl` removes the encryption (*pass phrase*) of the key. The form of the key can be specified with `-inform` and `-outform` (by default, it's `PEM`, alternatively `DER` or `NET`). With `-in file -text`, you

can determine the properties of an existing key, such as the key length (the number of bits).

- `sclient [options]`

  Acts as an SSL/TLS client to test connections to an encrypted server. You can find an application example at the following address:

  https://www.misterpki.com/openssl-s-client/

- `speed`

  Performs multiple benchmark tests, each of which lasts three seconds. The result shows how many bytes were processed for the respective algorithm (the more there are, the faster it runs or the better optimized the hardware on which the test is performed).

- `x509 [options]`

  Displays certificate data, signs certificates, and converts certificates between different forms. Again, `-in` and `-out` specify the file names of the source and target certificates. With `-req`, `openssl` expects a certificate request as input, not the certificate itself. `-CA` specifies which CA certificate is to be used for the signature. `-CAkey` specifies the corresponding private key.

There are separate `man` pages available for most `openssl` commands. For example, `man req` describes the countless options of `openssl req`, `man rsa` describes the options of `openssl rsa`, and so on.

Examples

If you want to have a certificate signed for your web server by an official certification authority (e.g., Thawte), you should first generate a key (if you don't already have one) and then a certificate signing request. The key isn't encrypted here because otherwise the web server would have to ask for the encryption password every time it's started.

```
root# genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -out server.key
root# chmod 400 server.key
root# openssl req -new -key server.key -out server.csr
...
Common Name (eg server FQDN or YOUR name) []: company-abc.com
```

You can take a look at the CSR file to check this:

```
root# openssl req -in server.csr -noout -text
```

Then, you need to send the CSR file to the certification authority. You will then receive the signed certificate from there for a fee. The usual file identifier is `.pem` or `.crt`. You must enter this file, your own key, and a CA certificate from the certification authority in the Apache configuration.

```
# Apache configuration file
SSLCertificateFile      /etc/apache2/server.pem
SSLCertificateKeyFile   /etc/apache2/server.key
SSLCertificateChainFile /etc/apache2/sub.class1.server.ca.pem
SSLCACertificateFile    /etc/apache2/ca.pem
```

If you don't want to pay for an official signature, you can sign your certificate yourself:

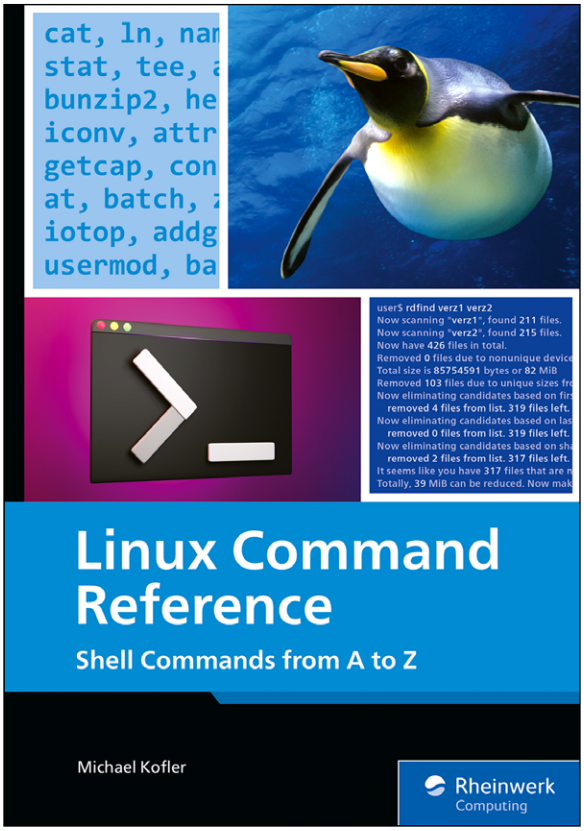
```
root# openssl x509 -req -days 1900 -in server.csr  \
         -signkey server.key -sha256 -out server.pem
Signature ok
subject=/C=DE/L=Boston/O=John Doe/CN=www.company-abc.com/
  emailAddress=webmaster@company-abc.com
Getting Private key
```

To take a look at the certificate file, you can run `openssl x509 -text`:

```
root# openssl x509 -text -in server.pem
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 12669601459972319941 (0xafd37766c36baac5)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=DE, L=Boston, O=John Doe,
                CN=www.company-abc.com/emailAddress=webmaster@company-abc.com
        Validity
            Not Before: Sep 28 14:48:03 2025 GMT
            Not After : Dec 10 14:48:03 2029 GMT     ...
```

The generation of a key, a certificate request, and self-signing can also be carried out in a single command: The following command generates a self-signed certificate for a mail server that is valid for 10 years. It's important that you enter the host name of your server as the common name when running `openssl`. Because the certificate itself is signed, your mail client will indicate during configuration that the certificate isn't trustworthy.

```
root# openssl req -new -x509 -days 3650 -nodes \
        -out    /etc/ssl/certs/postfix.pem \
        -keyout /etc/ssl/private/postfix.key
...
Common Name (eg server FQDN or YOUR name) []: company-abc.com
root# chmod 400 /etc/ssl/private/postfix.key
```



Michael Kofler

Linux Command Reference

Shell Commands from A to Z

- More than 500 Linux commands and configuration files at your fingertips
- Explore commands by topic and alphabetically
- Learn to use commands with detailed examples and instructions

rheinwerk-computing.com/6157

Dr. Michael Kofler is a programmer and Linux administrator. He has been one of the most successful and versatile computing authors in the German-speaking world for many years. His current focuses include Linux, Docker, Git, hacking and security, Raspberry Pi, and the programming languages Swift, Java, Python, and Kotlin.

ISBN 978-1-4932-2749-5 • 493 pages • 07/2025

E-book: $34.99 • **Print book:** $39.95 • **Bundle:** $49.99

Rheinwerk Publishing