Application Note

Command Line Interface for S1 Routers

Advantech Czech s.r.o., Sokolska 71, 562 04 Usti nad Orlici, Czech Republic
Document No. APP-0128-EN, revised on April 23, 2026.

Used symbols

Important

Important — Indicates a risk to personal safety or potential damage to the router. Follow these instructions precisely to prevent injury or equipment damage.

Warning

Warning — Highlights conditions that may cause malfunction, loss of data, or unexpected behavior in specific situations. Read carefully before proceeding.

Info

Info — Provides helpful tips, context, or references that improve understanding but are not strictly required to complete the task.

Firmware Version

This manual applies to firmware version **6.6.1 (April 24, 2026)**. Features introduced after this version may not be covered.

Contents

List of Tables

1. Document Introduction

1.1 Document Contents

This document is intended for **S1 Routers only.** The command set available on S1 routers represents a reduced subset of the standard router CLI. Commands are grouped into the following categories based on their usage:

- **HW Control Commands** — see Chapter 2.1.
- **File/Directory Management Commands** — see Chapter 2.2.
- **System Commands** — see Chapter 2.3.
- **Network Commands** — see Chapter 2.4.
- **Scripting/Shell Commands** — see Chapter 2.4.

Warning

You may find some commands available on the system that are not documented in this manual. These commands are not intended for regular user operations, and their incorrect use may cause system malfunction.

Info

For a comprehensive list of all commands, please refer to Appendix *Command Index*.

1.2 Command Line Access

The command-line interface (CLI) is a non-GUI alternative that allows you to manage the router. It provides an interactive interface enabling you to perform advanced configurations and troubleshoot the device directly.

SSH Connection

You can use an SSH (Secure Shell) connection to access the router's console. A commonly used application for this is *PuTTY*. To connect securely without requiring credentials, you can use Passwordless Console Login with a public key. For detailed instructions, refer to the Configuration Manual, Chapter *Administration* → *Modify User* → *Passwordless Console Login*.

Note: Ensure that SSH is enabled under *Configuration* → *Services* → *SSH*.

Web Terminal Router App

The *Web Terminal* Router App allows you to access the router's console directly from the web GUI. This Router App is available for free on the *Engineering Portal*.

1.3 Permissions Notes

Please note that only a user with the `admin` role is permitted to access the router's console; a user with the `user` role does not have this access.

Commands that modify device status or configuration require privileged mode. When you log in to the console on the S1 Routers, privileged mode is prohibited, and you must use the `sudo` command to execute commands with elevated privileges. Each command that supports privileged mode on the S1 Routers, requiring the use of the `sudo` command, is individually noted.

Some commands have read-only permissions, meaning they can read configuration settings but cannot modify them. This is also noted for the relevant commands.

2. Commands

In the ecosystem of Unix-like operating systems, commands are the essential tools through which users interact with the system, automate tasks, manage resources, and configure services. Whether you are a system administrator, a network engineer, or a software developer, understanding these commands is key to harnessing the full potential of the system.

This chapter covers an array of commands used in daily operations and specialized tasks alike. Scripting and shell command essentials form the backbone of system automation and task scheduling; administration commands ensure system security and integrity; network commands enable configuring, analyzing, and securing network communications; and file and directory management commands provide the means to navigate and manipulate the filesystem. The text processing and analysis commands section demonstrates the power of Unix-like systems in handling textual data — searching, editing, and transforming text in various ways.

Each command entry includes a synopsis, a description of available options, and practical usage examples, catering to both users seeking foundational knowledge and those looking to refine their expertise.

2.1 HW Control Commands

These commands are specific to router and network device management, offering functionalities to configure interfaces, manage routing tables, and control device-specific features.

2.1.1 gsminfo

Info

Execution with `sudo` is required.

This command retrieves and displays detailed information about the cellular module's status, including signal quality, network registration, and connection details. This tool is invaluable for diagnosing connectivity issues and optimizing signal reception.

Synopsis:

`gsminfo`

Description:

`gsminfo` provides a concise overview of the current cellular module's status, including the signal strength, network operator, and the communication channel. This information aids in assessing the quality of the connection and troubleshooting network problems.

Output Fields:

Field	Description
`Manufacturer`	Cellular module manufacturer.
`Model`	Cellular module model designation.
`Revision`	Firmware revision of the cellular module.
`IMEI`	International Mobile Equipment Identity number.
`ICCID`	Integrated Circuit Card Identifier of the active SIM card.
`IMSI`	International Mobile Subscriber Identity of the active SIM card.
`Registration`	Current network registration status.
`Operator`	Name of the mobile network operator.
`Technology`	Active cellular technology (e.g., LTE, 5G).
`PLMN`	Public Land Mobile Network code of the operator.
`Cell`	Identifier of the currently connected cell.
`Channel`	Channel number used for communication.
`Band`	Active cellular frequency band.
`Signal Strength`	Signal strength of the connected cell (dBm).
`Signal Quality`	Signal quality of the connected cell (dB).
`RSSI`	Received Signal Strength Indicator (dBm).
`RSRP`	Reference Signal Received Power (dBm).
`RSRQ`	Reference Signal Received Quality (dB).
`CSQ`	Signal strength number (0 to 31).

Table 1: gsminfo output fields

Examples:

Display the current cellular module status:

`sudo gsminfo`

Output:

```
Manufacturer       : Quectel
Model              : EC25-EUX
Revision           : EC25EUXGAR08A05M1G
Firmware Release   : EC25EUXGAR08A05M1G_01.001.01.001
IMEI               : 86584__________

ICCID              : 89420310__________
IMSI               : 23003__________
SMS Center         : +420__________
Registration       : Home Network
Operator           : Vodafone CZ
Technology         : LTE
PLMN               : 23003
Cell               : 10A804
TAC                : 947C
Channel            : 1849
Band               : B3
Signal Strength    : -90 dBm
Signal Quality     : -12 dB

RSSI               : -57 dBm
RSRP               : -90 dBm
RSRQ               : -12 dB
CSQ                : 11
```

2.1.2 hwclock

Info

Execution with `sudo` is required.

This command queries and sets the hardware clock (RTC).

Synopsis:

```
hwclock [-swul] [--systz] [-f DEV]
```

Options:

Option	Description
`-s`	Set the system time from the hardware clock.
`-w`	Set the hardware clock to the current system time.
`–systz`	Set the in-kernel timezone and correct the system time if the RTC is kept in local time.
`-f DEV`	Use the specified RTC device (e.g., `/dev/rtc2`).
`-u`	Assume the hardware clock is kept in UTC.
`-l`	Assume the hardware clock is kept in local time. If neither `-u` nor `-l` is specified, the setting is read from */etc/adjtime*.

Table 2: hwclock options

Examples:

Display the current hardware clock:

```
sudo hwclock
```

Example output:

```
Wed Apr 22 08:24:52 2026 0.000000 seconds
```

Set the hardware clock to the current system time, assuming UTC:

```
sudo hwclock -w -u
```

Set the system time from the hardware clock:

```
sudo hwclock -s
```

2.1.3 io

Info

Execution with `sudo` is required.

This command reads binary inputs and controls binary outputs of the router. If an expansion board is installed, it also supports the router's expansion ports.

Synopsis:

```
io [get <pin>] | [set <pin> <value>]
```

Options:

Option	Description
`get`	Get the state of the specified input pin.
`set`	Set the state of the specified output pin.

Table 3: io options

Examples:

Get the state of digital input BIN0.

```
sudo sudo io get bin0
```

Get the state of analog input AN1 on expansion port XC-CNT.

```
sudo sudo io get an1
```

Get the state of counter input CNT1 on expansion port XC-CNT.

```
sudo sudo io get cnt1
```

Set the state of binary output OUT0 to 1.

```
sudo sudo io set out0 1
```

2.1.4 led

Info

This command is not supported by routers of *v1* production line.

Info

Execution with `sudo` is required.

This command controls the USR or PWR LED of the router.

Synopsis:

```
led [-p] [-u] <command>
```

Options:

Option	Description
`-p`	Control of PWR LED
`-u`	Control of USR LED (default)

Table 4: led options

Commands:

Command	Description
`on`	Power on the LED.
`off`	Power off the LED.
`slow`	Start blinking the LED slowly.
`fast`	Start blinking the LED fast.

Table 5: led commands

Examples:

Turn on the USR LED:

```
sudo led on
```

Start blinking the USR LED slowly:

```
sudo led slow
```

Control the PWR LED — turn it off:

```
sudo led -p off
```

2.1.5 lpm

Info

This command is not supported by routers of *v1*, v2, ICR-2000, ICR-2400, ICR-2500, and ICR-2600 production lines.

Info

Execution with `sudo` is required.

This command switches the router into Low Power Mode (LPM) to minimize power consumption. The router can be woken from this state by one of two events: the expiration of a specified time interval or a signal change on a binary input. If both wake-up conditions are configured, the router will wake up as soon as the first event occurs.

Synopsis:

```
lpm [-b] [-i <interval>]
```

Options:

Option	Description
`-b`	Wakes up the router upon activation of a binary input. The specific input used depends on the router platform: • *BIN1* for *SmartFlex*, *SmartMotion*, *ICR-2800*, *ICR-4200*, and *ICR-4400* platforms. • *BIN0* for *SmartStart* and *ICR-3200* platforms. **Note:** This option is not supported by routers of the *ICR-2700* production line.
`-i`	Wakes up the router after the specified time interval has elapsed. The interval is defined in seconds, ranging from 1 to 16,777,215.

Table 6: lpm options

Examples:

Put the router to sleep for five minutes.

```
sudo lpm -i 300
```

Put the router to sleep, to be woken up by the activation of the binary input.

```
sudo lpm -b
```

Put the router to sleep for a maximum of five minutes. It will wake up sooner if the binary input is activated.

```
sudo lpm -b -i 300
```

2.1.6 mac

This command displays the MAC address of the `eth0` interface.

Synopsis:
`mac [<separator>]`

Examples:
Display the MAC address of eth0 using "-" as the separator instead of the default ":".
`mac -`

2.1.7 reboot

Info

Execution with `sudo` is required.

This command reboots the router.

Synopsis:
`reboot [-d <delay>] [-n] [-f]`

Options:

Option	Description
`-d`	Delay interval in seconds before rebooting.
`-n`	Do not call `sync()` before rebooting.
`-f`	Force reboot without calling shutdown.

Table 7: reboot options

Examples:
Reboot the router immediately:
`sudo sudo reboot`

Reboot the router after a 10-second delay:
`sudo sudo reboot -d 10`

2.1.8 report

Info

Execution with `sudo` is required.

This command generates and displays a diagnostic report for the router.

Warning

Sensitive data from the report are filtered out for security reasons.

Synopsis:

`report [<options>]`

Options:

Option	Description
no option	Report all sections except the configuration one.
`-a`	Report all sections.
`-s`	Report status section.
`-m`	Report router apps section.
`-l`	Report log section.
`-c`	Report configuration section.

Table 8: report options

Examples:

Generate a full diagnostic report (all sections except configuration):

```
sudo report
```

Generate a report including the configuration section:

```
sudo report -a
```

Generate only the status and log sections:

```
sudo report -s -l
```

2.1.9 sms

Info

Execution with `sudo` is required.

This command is used to send an SMS message from the router's cellular module.

Synopsis:

```
sms <phone_number> "<text>"
```

Arguments:

Argument	Description
`<phone_number>`	The recipient's telephone number. It is recommended to use the international format (e.g., +420123456789).
`<text>`	The content of the SMS message. The text must be enclosed in double quotes if it contains spaces or special characters.

Table 9: sms arguments

Examples:

Send the SMS `Hello world` to the phone number +420123456789:

```
sudo sms +420123456789 "Hello world"
```

2.1.10 status

Info

Execution with `sudo` is required.

This command displays the status of the router's interfaces and system components. It provides information equivalent to what is available on the *General Status* and *Mobile WAN Status* pages in the router's web administration interface.

Synopsis:

```
status [-h] [-v] [eth | geoloc | gnss | vlan | mobile | module | mwan | sim | bard
| ports | security | sys | hw | wifi ap | wifi sta]
```

Options:

Item	Description
`-h`	Generates HTML output, which is used when the command is called by the web interface.
`-v`	Enables verbose mode, which provides more detailed information. Data amounts are shown in bytes only, not in dynamic units (e.g., KB, MB).
`eth`	Displays the status of Ethernet interfaces (e.g., `eth0`, `eth1`), including link state, speed, and data statistics.
`geoloc`	Displays the router's last known geographical coordinates (latitude and longitude), determined by the GNSS receiver.
`gnss`	Shows the current status of the GNSS receiver, including the UTC time, fix type (e.g., 2D, 3D), dilution of precision (HDOP), and the number of satellites in use versus in view.
`vlan`	Lists the status of configured Virtual LAN (VLAN) interfaces, including their names, associated physical interfaces, and VLAN IDs.
`mobile`	Shows the status of the mobile radio connection, including registration status, operator, network technology (e.g., LTE, 5G), and signal quality metrics.
`module`	Displays the status of the cellular module(s), which can be `module 1`, `module 2`, etc., if available.
`mwan`	Provides the status of the Mobile WAN network interface, including IP address, DNS servers, and connection uptime.
`status sim` or `status sim 1`	Reports daily statistics for the first SIM card, including data usage (RX/TX), connection count, signal history, cell changes, and network availability. `status sim` is an alias for `status sim 1`.
`status sim 2`	Reports the same daily statistics as above, but for the second SIM card.
`status sim 1 full` or `status sim 2 full`	Provides a comprehensive statistical report for the specified SIM card (1 or 2). This extended view includes data for multiple time intervals: Today, Yesterday, This Week, Last Week, This Period, and Last Period.
`bard`	Displays the status of the Backup and Recovery Daemon (BARD), indicating whether it is active and monitoring backup routes.
`ports`	Shows the status of available peripheral ports (e.g., serial ports, USB).

Table 10: status options

Item	Description
`security`	Shows recent user login activity, including the last successful login, the count of failed login attempts, and the source IP address for each event.
`sys`	Provides general system information, such as uptime, memory usage, and supply voltage.
`hw`	Displays hardware capabilities and interfaces, including the presence of wireless modules (Mobile WAN, GNSS, Wi-Fi, Bluetooth), physical ports (Serial, I/O, PoE), and the product's designated region.
`wifi ap`	Displays the status of the WiFi Access Point.
`wifi sta`	Displays the status of the WiFi Station (client) connection.

Table 10: status options (continued)

Examples:

Show the verbose status of the mobile connection:

```
sudo status -v mobile
```

Output:

```
Registration       : Home Network
Operator           : Vodafone CZ
Technology         : LTE
PLMN               : 23003
Cell               : 10A804
TAC                : 947C
Channel            : 1849
Band               : B3
Signal Strength    : -90 dBm
Signal Quality     : -7 dB

RSSI               : -63 dBm
RSRP               : -90 dBm
RSRQ               : -7 dB
SINR               : 18 dB
CSQ                : 11
```

Show the system information:

```
sudo status sys
```

Output:

```
Part Number        : ICR-3211B
Firmware Version   : 6.6.0 (2025-12-17)
Serial Number      : ACZ1100000011434
Product Revision   : N/A
Profile            : Standard
Free Space         : 696.6 MiB for apps, 450.2 MiB for data
RTC Battery        : Ok
Supply Voltage     : 23.8 V
Temperature        : 37 °C
Time               : 2026-04-13 11:48:30
Uptime             : 112 days, 16 hours, 49 minutes
```

2.1.11 stty

This command prints or changes terminal line settings.

Synopsis:

```
stty [-a|g] [-F DEVICE] [SETTING]...
```

Options:

Option	Description
`-F DEVICE`	Open device instead of stdin
`-a`	Print all current settings in human-readable form
`-g`	Print in stty-readable form
`[SETTING]`	See manpage

Table 11: stty options

Examples:

Get the current parameters of the first UART serial port:

```
stty -F /dev/ttyS0
```

To only get actual speed of the second UART serial port.

```
stty -F /dev/ttyS1 speed
```

To set parameters of the first UART serial port to:

- speed to 1200 bps
- character size to 7 bits
- 2 stop bits
- disable software output flow control
- reset parameters to system default raw mode

```
stty -F /dev/ttyS0 1200 cs7 cstopb -ixon raw
```

2.1.12 tpm2

Info

TPM features are available only on products that are equipped with a TPM module. TPM support can be verified either in the Hardware Manual or directly in the router console, as described below.

This command provides tools for working with the TPM 2.0 (Trusted Platform Module) chip mounted directly on the router mainboard.

Note: To verify whether TPM functionality is supported on your device, use one of the following procedures:

1. In the router console, execute the following command: `ls /dev/tpm0`
   - If the response is `No such file or directory`, the TPM chip is not available.
   - If the response is `/dev/tpm0`, the TPM chip is available.
2. In the router console, execute the TPM command to display fixed TPM properties: `tpm2 getcap properties-fixed`
   - If multiple error messages are displayed, the TPM chip is not available.
   - If a list of TPM properties is displayed, the TPM chip is available.

Synopsis:

`tpm2 <subcommand> [<options>...]`

The table below lists the most important `tpm2` subcommands. For more details about the subcommands, see https://github.com/tpm2-software/tpm2-tools/tree/5.1.X/man.

tpm2 subcommands

Subcommand	Description
`getcap`	Display TPM capabilities in a human readable form.
`create`	Create a child object.
`createek`	Generate TCG profile compliant endorsement key.
`createak`	Generate attestation key with given algorithm under the endorsement hierarchy.
`createprimary`	Create a primary key.
`load`	Load an object into the TPM.
`loadexternal`	Load an external object into the TPM.
`readpublic`	Read the public area of a loaded object.
`evictcontrol`	Make a transient object persistent or evict a persistent object.
`clear`	Clear lockout, endorsement and owner hierarchy authorization values.
`getrandom`	Retrieve random bytes from the TPM.
`hash`	Perform a hash operation with the TPM.
`hmac`	Perform an HMAC operation with the TPM.
`sign`	Sign a hash or message using the TPM.
`verifysignature`	Validate a signature using the TPM.
`encryptdecrypt`	Perform symmetric encryption or decryption.

Table 12: tpm2 subcommands

Subcommand	Description
ecdhzgen	Recover the shared secret value (Z) from a public point and a specified private key.
nvdefine	Define a TPM Non-Volatile (NV) index.
nvundefine	Delete a Non-Volatile (NV) index.
nvread	Read the data stored in a Non-Volatile (NV)s index.
nvwrite	Write data to a Non-Volatile (NV) index.

Table 12: tpm2 subcommands (continued)

2.2 File/Directory Management Commands

Essential for navigating and manipulating the filesystem, these commands allow users to create, list, modify, and remove files and directories, making them indispensable for day-to-day operations on a Unix-like system.

2.2.1 basename

This command strips the directory path and an optional suffix from a filename.

Synopsis:
`basename FILE [SUFFIX]`

Examples:
Strip the directory from the path `/home/myfile.txt`:
`basename /home/myfile.txt`
Output:
`myfile.txt`

Strip the directory and the `.txt` extension from the path `/home/myfile.txt`:
`basename /home/myfile.txt .txt`
Output:
`myfile`

2.2.2 cat

This command concatenates files and prints to standard output.

Synopsis:
`cat [-u] [<file>] ...`

Options:

Option	Description
-u	Ignored since unbuffered I/O is always used.

Table 13: cat options

Examples:
View the contents of the file `/proc/tty/driver/atmel_serial` (serial port information on v2i routers):
`cat /proc/tty/driver/atmel_serial`
Output:

```
serinfo:1.0 driver revision:
0: uart:ATMEL_SERIAL mmio:0xF8028200 irq:38 tx:0 rx:0 DSR|CD|RI
1: uart:ATMEL_SERIAL mmio:0xF0004200 irq:37 tx:0 rx:0 DSR|CD|RI
5: uart:ATMEL_SERIAL mmio:0xFFFFF200 irq:36 tx:0 rx:0 CTS|DSR|CD|RI
```

Merge all router configuration files into a single file `/tmp/my.cfg`:
`cat /etc/settings.* > /tmp/my.cfg`

2.2.3 cd

This command changes the current working directory.

Synopsis:
`cd [-P] [-L] [<directory>]`

Options:

Option	Description
`-P`	Do not follow symbolic links.
`-L`	Follow symbolic links (default).

Table 14: cd options

Examples:
Change to the home directory (`/tmp`):
`cd`

Change to the directory `/mnt`:
`cd /mnt`

2.2.4 chdir

The `chdir` command is an alias for `cd`. See the *2.2.3 cd* command for full documentation.

2.2.5 cmp

The `cmp` utility compares two files of any type and writes the results to the standard output.

Synopsis:

```
cmp [-l] [-s] <file1> <file2> [<skip1> [<skip2>]]
```

Options:

Option	Description
`-l`	Print the byte number (decimal) and the differing byte values (octal) for each difference.
`-s`	Print nothing for differing files; return exit status only.

Table 15: cmp options

By default, `cmp` is silent if the files are the same; if they differ, the byte and line number at which the first difference occurred is reported. Bytes and lines are numbered beginning with one. If <*file2*> is not specified, standard input is used instead.
The optional arguments <*skip1*> and <*skip2*> are the byte offsets from the beginning of <*file1*> and <*file2*> respectively, where the comparison will begin. The offset is decimal by default, but may be expressed as a hexadecimal or octal value.

Examples:

Compare two files and report the first difference:

```
cmp file1.txt file2.txt
```

Check silently whether two files are identical (useful in scripts):

```
cmp -s file1.bin file2.bin && echo "identical" || echo "differ"
```

2.2.6 cp

This command copies files and directories.

Synopsis:

```
cp [<option>] <source> <dest>
```

Options:

Option	Description
`-a`	Preserve the all attributes.
`-R, -r`	Copy directories recursively.
`-d, -P`	Never follow symbolic links.
`-H, -L`	Follow command-line symbolic links.
`-p`	Preserve the mode, ownership, timestamps attributes.
`-f`	If an existing destination file cannot be opened, remove it and try again.
`-i`	Prompt before overwrite.
`-n`	Don't overwrite.
`-l, -s`	Create (sym)links
`-T`	Refuse to copy if DEST is a directory
`-t DIR`	Copy all SOURCEs into DIR
`-u`	Copy only newer files

Table 16: cp options

Examples:

Creat copy of `myfile.txt` file:

```
cp myfile.txt myfile-copy.txt
```

2.2.7 cut

This command prints selected fields from each input FILE to standard output.

Synopsis:

```
cut [OPTIONS] [FILE]...
```

Options:

Option	Description
`-b LIST`	Output only bytes from LIST
`-c LIST`	Output only characters from LIST
`-d CHAR`	Field delimiter for input (default -f TAB, -F run of whitespace)
`-O CHAR`	Field delimiter for output (default = -d for -f, one space for -F)
`-D`	Don't sort/collate sections or match -fF lines without delimiter
`-f LIST`	Print only these fields (-d is single char)
`-s`	Output only the lines containing delimiter
`-n`	Ignored

Table 17: cut options

Examples:

Display the first field of each line, using tab as the field separator:

```
cut -f1 file.txt
```

Display the second field of each line, using colon as the field separator:

```
echo a:b | cut -d: -f2
```

Output:

```
b
```

Display the second and all subsequent fields:

```
echo a b c d | cut -d" " -f2-
```

Output:

```
b c d
```

Display the third and fourth characters:

```
echo abcd | cut -c3,4
```

Output:

```
cd
```

2.2.8 dd

This command copies a file and optionally converts the data format according to the specified operands.

Synopsis:
`dd [if=FILE] [of=FILE] [bs=N] [count=N] [skip=N] [seek=N]`

Options:

Option	Description
`if=FILE`	Read from FILE instead of stdin
`of=FILE`	Write to FILE instead of stdout
`bs=N`	Read and write N bytes at a time
`count=N`	Copy only N input blocks
`skip=N`	Skip N input blocks
`seek=N`	Skip N output blocks
`N`	May be suffixed by c (1), w (2), b (512), kB (1000), k (1024), MB, M, GB, G

Table 18: dd options

Examples:
Erase the microSD card available as `/dev/sda`:
`dd if=/dev/zero of=/dev/sda bs=4k`

Create a backup image of a device:
`dd if=/dev/sda of=/mnt/backup.img bs=1M`

2.2.9 decode

This command decodes Base64-encoded values. The decoded result is printed to standard output.

Synopsis:
`decode <base64>`

Description:
The `decode` command takes a single Base64-encoded string as its argument and prints the decoded value. This is particularly useful for reading encoded values from configuration files.

Examples:
Decode a Base64-encoded string:
`decode SGVsbG8sIFdvcmxkIQ==`
Output:
`Hello, World!`

2.2.10 dirname

This command strips the non-directory suffix from a filename.

Synopsis:
`dirname FILENAME`

Examples:
Strip the filename from the path `/home/MyFolder/myfile.txt`:
`dirname /home/MyFolder/myfile.txt`
Output:
`/home/MyFolder`

2.2.11 find

This command searches for files in a directory hierarchy.

Synopsis:
`find [<path> ...] [<expression>]`

Options:
The default path is the current directory and the default expression is `-print`. Expression may consist of:

Option	Description
`-follow`	Dereference symbolic links.
`-name <pattern>`	File name (leading directories removed) matches <*pattern*>.
`-print`	Print the found path (default).
`-type X`	Filetype matches X (where X is one of: `f`, `d`, `l`, `b`, `c`, ...).
`-perm <perms>`	Permissions match any of (`+NNN`), all of (`-NNN`), or exactly (`NNN`).
`-mtime <days>`	Modified time is greater than (`+N`), less than (`-N`), or exactly (`N`) days.
`-mmin <mins>`	Modified time is greater than (`+N`), less than (`-N`), or exactly (`N`) minutes.
`-exec <cmd>`	Execute command with all instances of `{}` replaced by the files matching <*expression*>.

Table 19: find expressions

Examples:
Search for files in the home directory modified in the last 24 hours:
`find $HOME -mtime 0`

Search for files with read and write permission for owner and group, but read-only for others:
`find . -perm 664`

2.2.12 grep

The `grep` command searches the named input files (or standard input) for lines containing a match to a given pattern. By default, it prints the matching lines. It is an essential tool for parsing logs and filtering command outputs.

Synopsis:

```
grep [options] {PATTERN | -e PATTERN... | -f FILE...} [FILE]...
```

Options:

Option	Description
`-H` / `-h`	Print (`-H`) or suppress (`-h`) the filename prefix for each match.
`-n`	Prefix each line of output with its line number.
`-l` / `-L`	Show only the names of files that match (`-l`) or do not match (`-L`).
`-c`	Show only a count of matching lines per file.
`-o`	Show only the matching part of the line, not the entire line.
`-q`	Quiet mode. Do not write anything to standard output. Returns 0 if found.
`-v`	Invert the match, selecting non-matching lines.
`-s`	Suppress error messages about nonexistent or unreadable files.
`-r` / `-R`	Recurse through directories (`-R` also dereferences symlinks).
`-i`	Ignore case distinctions.
`-w` / `-x`	Match whole words only (`-w`) or whole lines only (`-x`).
`-F`	Interpret the pattern as a literal fixed string (not a regular expression).
`-E`	Interpret the pattern as an extended regular expression.
`-m N`	Stop reading a file after `N` matches.
`-A N`	Print `N` lines of trailing context (after the match).
`-B N`	Print `N` lines of leading context (before the match).
`-C N`	Print `N` lines of output context (same as `-A N -B N`).
`-e PTRN`	Use `PTRN` as the pattern. Useful if the pattern begins with a hyphen.
`-f FILE`	Obtain patterns from `FILE`, one per line.

Table 20: grep options

Examples:

Find all lines in the system log containing the word `error`:

```
grep error /var/log/messages
```

Search the process list for processes containing the string `ppp`:

```
ps | grep ppp
```

Find the word `error` and print 2 lines of context before and after each match:

```
grep -C 2 "error" /var/log/messages
```

2.2.13 gunzip

This command decompresses a file. If the filename is `--`, data is read from standard input.

Synopsis:
`gunzip [-c] [-f] [-t] <filename>`

Options:

Option	Description
`-c`	Write output on standard output
`-f`	Force decompression even if the file has multiple links or the corresp. file already exists, or if the compressed data is read from or written to a terminal.
`-t`	Test. Check the compressed file integrity.

Table 21: gunzip options

Examples:
Decompress the file `test.tar.gz` (creates `test.tar`):
`gunzip test.tar.gz`

Decompress and write output to standard output (useful in pipes):
`gunzip -c test.tar.gz | tar -xf -`

2.2.14 gzip

This command compresses a file using maximum compression.

Synopsis:
`gzip [-c] [-d] [-f] <filename>`

Options:

Option	Description
`-c`	Write output on standard output
`-d`	Decompress
`-f`	Force compression even if the file has multiple links or the corresponding file already exists, or if the compressed data is read from or written to a terminal

Table 22: gzip options

Examples:
Compress the file `test.tar` (creates `test.tar.gz`):
`gzip test.tar`

Decompress a file using `gzip`:
`gzip -d test.tar.gz`

2.2.15 head

This command prints the first 10 lines of each file to standard output. With more than one file, each output is preceded by a header with the file name. With no file, or when the file is a dash (`-`), read standard input.

Synopsis:

```
head [<option(s)>] [<file(s)>]
```

Options:

Option	Description
`-n NUM`	Print the first NUM lines instead of the first 10.
`-c NUM`	Output the first NUM bytes.
`-q`	Never output headers giving file names.
`-v`	Always output headers giving file names.

Table 23: head options

Examples:

Display the first 5 lines of the system log:

```
head -n 5 /var/log/messages
```

Display the first 100 bytes of a file:

```
head -c 100 /var/log/messages
```

2.2.16 jq

Info

This command is not supported by routers of *v1* and v2 production lines.

This command is a powerful tool for processing JSON inputs, applying filters to JSON text, and producing the output as JSON. It can manipulate, filter, and transform structured data.

Synopsis:

```
jq [options] <jq filter> [file...]
jq [options] --args <jq filter> [strings...]
jq [options] --jsonargs <jq filter> [JSON_TEXTS...]
```

Options:

Option	Description
`-r`	Output raw strings, not JSON texts.
`-c`	Produce compact output instead of pretty-printed output.
`-f`	Load a jq program from a file.
`–arg name value`	Pass argument to the filter.

Table 24: jq options

Examples:

Output a JSON object unmodified with pretty-printing:

```
echo '{"foo": 0}' | jq .
```

Extract the value of the key `bar`:

```
echo '{"foo": 0, "bar": 1}' | jq '.bar'
```

Output:

```
1
```

Apply a filter to each element in a JSON array:

```
echo '[{"foo": 0}, {"foo": 1}]' | jq '.[] | .foo'
```

2.2.17 less

This command is a terminal pager that displays the contents of a text file one screen at a time. Unlike most pager programs, `less` allows backward as well as forward movement through the file.

Synopsis:
`less [options] file ...`

Options:
Some commonly used options include:

Option	Description
`-N`	Display line numbers at the beginning of each line.
`-i`	Ignore case when searching.
`-F`	Exit if the content fits on one screen.
`-R`	Display ANSI color escape sequences in raw form.

Table 25: less options

Examples:
Display the contents of `file.txt`:
`less file.txt`

Display `file.txt` with line numbers:
`less -N file.txt`

Pipe grep output into `less` with color formatting:
`grep 'search_term' file.txt | less -R`

Follow the end of a log file in real time (similar to `tail -f`):
`less +F /var/log/messages`

2.2.18 ln

This command creates hard or symbolic links between files.

Synopsis:

```
ln [option] <target> ... <link_name> | <directory>
```

Options:

Option	Description
`-s`	Make symbolic links instead of hard links.
`-f`	Remove existing destination files.
`-n`	Do not dereference symlinks — treat like a normal file.
`-b`	Make a backup of the target (if it exists) before the link operation.
`-S`	Use the specified suffix instead of ~ when making backup files.

Table 26: ln options

Examples:

Create a symbolic link named `my.log` pointing to `/var/log/messages`:

```
ln -s /var/log/messages my.log
```

Create a hard link:

```
ln /var/log/messages /tmp/messages.lnk
```

2.2.19 ls

This command lists directory contents.

Synopsis:

```
ls [ option ] < filename > ...
```

Options:

Option	Description
-1	List files in a single column
-A	Do not list implied . and ..
-a	Do not hide entries starting with .
-C	List entries by columns
-c	With -l: show ctime
-d	List directory entries instead of contents
-e	List both full date and full time
-i	List the i-node for each file
-l	Use a long listing form
-n	List numeric UIDs and GIDs instead of names
-L	List entries pointed to by symbolic links
-r	Sort the listing in reverse order
-S	Sort the listing by file size
-s	List the size of each file, in blocks
-t	With -l: show modification time
-u	With -l: show access time
-v	Sort the listing by version
-x	List entries by lines instead of by columns
-X	Sort the listing by extension

Table 27: ls options

Examples:

List the contents of the `/mnt` directory in long format:
`ls -l /mnt`

List all files including hidden ones in the current directory:
`ls -a`

2.2.20 mkdir

This command creates directories.

Synopsis:
`mkdir [<option>] directory ...`

Options:

Option	Description
`-m`	Set permission mode (as in `chmod`).
`-p`	No error if the directory already exists; create parent directories as needed.

Table 28: mkdir options

Examples:
Create the directory `/tmp/test/example`, including any missing parent directories:
`mkdir -p /tmp/test/example`

2.2.21 mv

This command moves or renames files.

Synopsis:
`mv [-f] [-fin] <source> ... <dest>`

Options:

Option	Description
`-f`	Don't prompt before overwriting
`-i`	Interactive, prompt before overwrite
`-n`	Don't overwrite an existing file
`-T`	Refuse to move if DEST is a directory
`-t DIR`	Move all SOURCEs into DIR

Table 29: mv options

Examples:
Rename the file `abc.txt` to `def.txt`:
`mv abc.txt def.txt`

Move all `.txt` files to the directory `/mnt`:
`mv *.txt /mnt`

2.2.22 pwd

This command prints the path of the current working directory.

Synopsis:
`pwd`

Examples:
Print the path of the current directory, which is `/home/httpd` in this case.
`pwd`
Output:
`/home/httpd`

Store the current directory path in a variable for later use.
`DIR=$(pwd)`
`echo "Working in: $DIR"`
Output:
`Working in: /home/httpd`

2.2.23 readlink

The `readlink` command is used to display the target of a symbolic link.

Synopsis:
`readlink FILE`

Description:
When given a symbolic link as an argument, `readlink` displays the path to which the symlink points. If the specified file is not a symbolic link, `readlink` produces no output.

Examples:

Display the target of the symbolic link `/usr/bin/report`:
`readlink /usr/bin/report`
`/usr/bin/abox`

2.2.24 realpath

This command returns the absolute pathname of the given filename.

Synopsis:
`realpath FILE`

Examples:
Return the absolute pathname of `myfile.txt` located in the current directory (here `/home/MyFolder/`):
`realpath myfile.txt`
Output:
`/home/MyFolder/myfile.txt`

2.2.25 rm

This command removes files or directories.

Synopsis:
`rm [-i] [-f] [-r] <file> ...`

Options:

Option	Description
`-i`	Always prompt before removing each destination
`-f`	Remove existing destinations, never prompt
`-r`	Remove the contents of directories recursively

Table 30: rm options

Examples:
Remove all files with the `.txt` extension in the current directory:
`rm *.txt`

Remove the directory `/tmp/test` and all its contents recursively:
`rm -rf /tmp/test`

2.2.26 rmdir

This command removes empty directories.

Synopsis:

```
rmdir [OPTIONS] DIRECTORY...
```

Options:

Option	Description
`-p`	Remove the directory and its parent directories if they become empty as a result.
`–ignore-fail-on-non-empty`	Do not report an error if the directory is non-empty; silently ignore the failure.

Table 31: rmdir options

Examples:

Remove empty directory `/tmp/test`.

```
rmdir /tmp/test
```

Remove directory `/tmp/test` and its parent `/tmp` if both are empty.

```
rmdir -p /tmp/test
```

2.2.27 sed

This command filters and transforms text.

Synopsis:

```
sed [ -e ] [ -f ] [ -i ] [ -n ] [ -r ] pattern [ -files ]
```

Options:

Option	Description
`-e`	Add the script to the commands to be executed
`-f`	Add script-file contents to the commands to be executed
`-i`	Edit files in place (makes backup if extension supplied)
`-n`	Suppress automatic printing of pattern space
`-r`	Use extended regular expression syntax

Table 32: sed options

Info

If no `-e` or `-f` is given, the first non-option argument is taken as the sed script to interpret. All remaining arguments are names of input files; if no input files are specified, then the standard input is read. Source files will not be modified unless -i option is given.

Examples:

Change the parameter `PPP_APN` in the file `/etc/settings.ppp` to the value `internet`:

```
sed -e "s/\(PPP_APN=\).*/\1internet/" -i /etc/settings.ppp
```

Delete all blank lines from a file:

```
sed '/^$/d' file.txt
```

2.2.28 shred

This command securely deletes a file from non-volatile memory by overwriting its contents multiple times with patterns designed to maximize data destruction, making recovery difficult even with specialized hardware.

Synopsis:

`shred [OPTIONS] [FILE]`

Options:

Option	Description
`-f`	Change permissions to allow writing if necessary.
`-n N`	Overwrite N times instead of the default (default is 3 time).
`-z`	Add a final overwrite with zeros to hide shredding.
`-u`	Truncate and remove file after overwriting.

Table 33: shred options

Examples:

Overwrite the data of `file1.txt` and `file2.txt` using the default shredding method:
`shred file1.txt file2.txt`

Overwrite and then delete `file1.txt`:
`shred -u file1.txt`

Overwrite `file1.txt` 10 times:
`shred -n 10 file1.txt`

2.2.29 split

This command splits a single file (INPUT) into multiple smaller files. A custom PREFIX for the output filenames can be specified.

Synopsis:

```
split [OPTIONS] [INPUT [PREFIX]]
```

Options:

Option	Description
`-b N[k|m]`	Split the input file into chunks of N (kilo|mega)bytes.
`-l N`	Split the input file by N lines (default is 1000 lines).
`-a N`	Use N letters as the suffix for output filenames (default is 2 letters).

Table 34: split options

Examples:

Split `file.img` into 50 kB chunks (`xaa`, `xab`, `xac`, ...):

```
split -b 50k file.img
```

Split `file.txt` into files of 200 lines each (`file_a`, `file_b`, ...):

```
split -l 200 -a 1 file.txt file_
```

2.2.30 sync

This command forces an immediate transfer of buffered data blocks in memory (or in specified files) to disk.

Synopsis:

```
sync [OPTIONS] [FILEs]...
```

Options:

Option	Description
`-d`	Avoid syncing metadata.
`-f`	Sync the filesystems underlying the specified files.

Table 35: sync options

Examples:

Flush all pending writes to disk:

```
sync
```

Sync only the data of a specific file without updating metadata:

```
sync -d /var/log/messages
```

2.2.31 tail

This command outputs the last part of files.

Synopsis:

```
tail [-n <number>] [-f] [file ...]
```

Options:

Option	Description
-n	Print the last N lines instead of the last 10.
-f	Output appended data as the file grows.

Table 36: tail options

Examples:

Show the last 30 lines of the system log:

```
tail -n 30 /var/log/messages
```

Follow the system log in real time:

```
tail -f /var/log/messages
```

2.2.32 tar

This command creates, extracts, or lists files in a tar archive.

Synopsis:

```
tar -[czxtv0] [-f tarfile] [-C dir] [file] ...
```

Options:

Option	Description
c	Create an archive.
x	Extract files from an archive.
t	List the contents of an archive.
-f FILE	Name of the archive file, or `-` for stdin.
-C DIR	Change to directory DIR before performing the operation.
-v	Verbosely list files processed.
-O	Extract to stdout.
-o	Do not restore user:group ownership.
-k	Do not replace existing files.
-z	(De)compress using gzip.
-a	(De)compress based on file extension.
-h	Follow symlinks.

Table 37: tar options

Examples:

Create a `log.tar` archive from the directory `/var/log`:

```
tar -cf log.tar /var/log
```

Extract all files from the archive `log.tar`:

```
tar -xf log.tar
```

Create a compressed archive using gzip:

```
tar -czf log.tar.gz /var/log
```

2.2.33 touch

This command updates the timestamp of a file, or creates an empty file if it does not exist.

Synopsis:

```
touch [-c] [-h] <file> [<file> ...]
```

Options:

Option	Description
`-c`	Do not create any files.
`-h`	Do not follow symbolic links.

Table 38: touch options

Examples:

Create an empty file or update the timestamp of `/tmp/test`:

```
touch /tmp/test
```

Update the timestamps of multiple files at once:

```
touch file1.txt file2.txt file3.txt
```

2.2.34 vi

This command opens a text editor for creating or modifying files.

Synopsis:

```
vi [-R] [<file> ...]
```

Options:

Option	Description
`-R`	Open the file in read-only mode.

Table 39: vi options

Examples:

Open the file `/etc/rc.local` for editing:

```
vi /etc/rc.local
```

Open a file in read-only mode:

```
vi -R /etc/settings.ppp
```

2.2.35 wc

This command prints newline, word, and byte counts for each file, and a total line if more than one file is specified. With no file, or when the file is a dash (`-`), read standard input.

Synopsis:

```
wc [<option(s)>] [<file(s)>]
```

Options:

Option	Description
`-c`	Print the byte counts.
`-l`	Print the newline counts.
`-L`	Print the length of the longest line.
`-w`	Print the word counts.

Table 40: wc options

Examples:

Count the number of words in a file:

```
wc -w file.txt
```

2.2.36 xxd

This is a program is a command-line utility that creates a hex dump of a given file or standard input. It can also convert a hex dump back to its original binary form. This tool is commonly used for debugging, examining binary files, and performing binary file analysis.

Synopsis:

```
xxd [-pri] [-g N] [-c N] [-l LEN] [-s OFS] [-o OFS] [FILE]
```

Options:

Option	Description
`-g N`	Bytes per group
`-c N`	Bytes per line
`-p`	Show only hex bytes, assumes -c30
`-i`	C include file style
`-l LENGTH`	Show only first LENGTH bytes
`-s OFFSET`	Skip OFFSET bytes
`-o OFFSET`	Add OFFSET to displayed offset
`-r`	Reverse (with -p, assumes no offsets in input)

Table 41: xxd options

Examples:

Generate a hex dump of `file.txt`:

```
xxd file.txt
```

Create a plain hex dump of a binary file:

```
xxd -p file.bin > file.hex
```

Convert a hex dump back to its original binary form:

```
xxd -r file.hex > file.bin
```

2.2.37 zcat

This command displays the contents of gzip-compressed files without explicitly decompressing them first. It is equivalent to running `gunzip -c`.

Synopsis:

```
zcat [file ...]
```

Description:

`zcat` concatenates the uncompressed contents of compressed files to standard output. When no files are specified, or when the filename `-` is used, `zcat` reads from standard input. Since `zcat` is a symlink to `gzip`, additional `gzip` options may be passed.

Examples:

View the contents of a compressed file:

```
zcat file.gz
```

Concatenate and view multiple compressed files:

```
zcat file1.gz file2.gz file3.gz
```

Pipe the contents of a compressed file into `grep`:

```
zcat file.gz | grep 'search_pattern'
```

2.3 System Commands

System administration commands provide the necessary tools for managing users, system services, and hardware settings. They are crucial for maintaining the system's integrity, security, and performance.

2.3.1 backup

Info

Execution with `sudo` is required.

This program is used to create a backup of the router configuration. The backup data is written to the standard output (`stdout`) and can be redirected to a file using shell redirection, as shown in the examples below. A previously saved configuration can be restored using the `restore` command; see Chapter *2.3.20 restore* for details.

The backup output is generated in a plain-text, key-value format that is suitable for storage, transfer, and later restoration. Sensitive data can optionally be filtered out or encrypted.

Synopsis:

```
backup [<options>] [<filename>]
```

Note: The optional `<filename>` is not passed as a command argument. The backup file is specified using shell output redirection (`>`), as shown in the examples.

Options:

Option	Description
`-c`	Back up the configuration excluding user account data (default if no option is specified).
`-u`	Back up only user account data.
`-a`	Back up the complete configuration, including user accounts, scripts, and Router Apps settings.
`-f`	Filter sensitive information (passwords, keys, secrets, and passphrases are replaced by `###REMOVED###`).
`-p <password>`	Encrypt the backup using AES-256; the output is Base64 encoded.

Table 42: backup options

Examples:

Back up the complete configuration to a file:

```
sudo backup -a > /tmp/my.cfg
```

Back up only the configuration (without user accounts):

```
sudo backup -c > /tmp/config-only.cfg
```

Back up only user accounts:

```
sudo backup -u > /tmp/users.cfg
```

Back up the configuration while filtering sensitive data:

```
sudo backup -a -f > /tmp/support-safe.cfg
```

Create an encrypted backup:

```
sudo backup -a -p MySecretPass > /tmp/backup.enc
```

View the backup directly on the console:

```
sudo backup -c
```

2.3.2 chmod

This command is used to change file and directory permissions (mode bits) in a Unix-like system. It allows you to control who can read, write, or execute a file or directory.

For a detailed explanation of permission modes and their numeric or symbolic notation, see: chmod(1) — Linux manual page

Synopsis:

```
chmod [-R] <mode> <filename>
```

Options:

Option	Description
`-R`	Change permissions of files and directories recursively.

Table 43: chmod options

Examples:

Make a script executable by its owner and readable by others:

```
chmod 755 /tmp/script.sh
```

Allow full access for the owner and read-only access for others:

```
chmod 744 /tmp/config.cfg
```

Change permissions for all files in a directory recursively:

```
chmod -R 755 /opt/custom
```

2.3.3 chown

This command changes the user and/or group ownership of files and directories. It is commonly used by administrators to manage access rights and file control in Unix-like systems.

For detailed usage and syntax, see: chown(1) — Linux manual page

Synopsis:

```
chown [-R] <user>[:<group>] <filename>
```

Options:

Option	Description
`-R`	Change ownership of files and directories recursively.

Table 44: chown options

Examples:

Change the owner of a file to user `root`:

```
chown root /tmp/config.cfg
```

Change both owner and group of a directory recursively:

```
chown -R admin:admin /opt/custom
```

Change only the group of a file:

```
chown :network /tmp/log.txt
```

2.3.4 clog

This command prints the connection log.

Synopsis:
`clog`

Examples:
Display the connection log:
`clog`

2.3.5 date

Info

Execution with `sudo` is required.

This command displays the current date and time in the specified format, or sets the system date and time.

Synopsis:
`date [-R] [-d <string>] [-s] [-r <file>] [-u] [MMDDhhmm[[CC]YY][.ss]]`

Options:

Option	Description
`-R`	Output date and time in RFC 2822 format.
`-d <string>`	Display the time described by STRING, not the current time.
`-s`	Set the time described by STRING.
`-r <file>`	Display the last modification time of FILE.
`-u`	Print or set Coordinated Universal Time (UTC).

Table 45: date options

Examples:
Display the current date and time:
`date`

Set the date and time to December 24, 2011 at 20:00:
`date 122420002011`

2.3.6 df

This command reports file system disk space usage.

Synopsis:

```
df [-PkT] [-t TYPE] [FILESYSTEM ...]
```

Options:

Option	Description
`-P`	POSIX output format.
`-k`	Print sizes in kilobytes.
`-T`	Print the filesystem type.
`-t TYPE`	Print only mounts of this type.

Table 46: df options

Examples:

Display disk usage for all mounted filesystems:

```
df
```

Display disk usage including filesystem type:

```
df -T
```

2.3.7 dmesg

Info

Execution with `sudo` is required.

This command displays kernel log messages.

Synopsis:

```
dmesg [-R] [-T] [-c]
```

Options:

Option	Description
`-R`	Display relative time since the router booted in *sec.nanosec* format.
`-T`	Display human-readable timestamp in *YYYY-MM-DD hh:mm:ss* format.
`-c`	Read and clear all messages.

Table 47: dmesg options

Examples:

Display the latest kernel log messages and clear the ring buffer:

```
sudo dmesg -c
```

Display kernel log messages with human-readable timestamps:

```
sudo dmesg -T
```

2.3.8 faillock

Info

This command is not supported by routers of *v1* production line.

Info

Execution with `sudo` is required.

This program is used to handle user login failures. It allows listing failures for each user, resetting failures, or eventually resetting locked users. Note that this feature is associated with account locking after exceeding the allowed number of unsuccessful login attempts and is not related to the administrative account locking function in the GUI.

Synopsis:

`faillock [--user username] [--reset] [--legacy-output]`

Options:

Option	Description
`-user username`	Apply the command to the specified user.
`-reset`	Reset the failure records. This can be used to manually unlock accounts or reset the failure count.
`-legacy-output`	Display output in the legacy format.

Table 48: faillock options

Examples:

View the authentication failure records for all users:
`sudo faillock`

View the authentication failure records for the user `john`:
`sudo faillock --user john`

```
john:
When                Type    Source          Valid
2024-07-25 12:31:22 RHOST   10.64.0.1           V
2024-08-22 11:30:30 RHOST   10.64.0.25          V
```

Reset the failure records for the user `john`:
`sudo faillock --user john --reset`

2.3.9 free

The `free` command displays information about free and used memory, reported in kilobytes (kB).

Synopsis:

`free`

Examples:

Display current memory usage:

`free`

```
               total        used        free      shared  buff/cache   available
Mem:         1008588      141496      834472         188       32620      854040
Swap:              0           0           0
```

The output columns have the following meaning:

Column	Description
`total`	Total physical memory (in kB).
`used`	Memory currently used by running processes.
`free`	Memory that is completely unallocated.
`shared`	Memory used by temporary shared memory segments.
`buff/cache`	Memory used by the kernel for buffers and caches. Can be quickly reclaimed when needed.
`available`	Estimate of memory available for new applications without swapping. Includes free memory and reclaimable buff/cache.

Table 49: free output columns

Difference Between Free and Available Memory

- **Free Memory**: This refers to the portion of RAM that is not being used at all. It is completely idle and unallocated. However, relying solely on this metric can be misleading because modern Linux systems deliberately use much of the free memory for caching to speed up system performance.
- **Available Memory**: This is a more meaningful metric for determining how much memory is truly available for applications. It not only accounts for the free memory but also includes the memory used for caching and buffers that can be quickly reclaimed. Therefore, even if the "free" memory appears low, a high "available" memory indicates that the system can still allocate memory for new processes without resorting to swap space.

2.3.10 fwupdate

Info

Execution with `sudo` is required.

This command updates the router's ICR-OS firmware.

Synopsis:

```
fwupdate [-i <filename> [-h] [-n]] [-f] [-c]
```

Options:

Option	Description
`-i`	Path to the new firmware file.
`-h`	HTML output (used when called from the web interface).
`-n`	Do not reboot after the firmware update.
`-f`	Finish update procedures (restore configuration, complete firmware switch).
`-c`	Check firmware update status.

Table 50: fwupdate options

Examples:

Update the firmware from a file on the router:

```
sudo fwupdate -i /tmp/firmware.bin
```

Check the current firmware update status:

```
sudo fwupdate -c
```

2.3.11 id

This command displays user and group information for the specified user, or for the current user if no user is specified.

Synopsis:

```
id [OPTIONS] [USER]
```

Options:

Option	Description
`-u`	Print the user ID (UID).
`-g`	Print the primary group ID (GID).
`-G`	Print all supplementary group IDs.
`-n`	Print the name instead of the numeric ID (used with `-u`, `-g`, or `-G`).
`-r`	Print the real ID instead of the effective ID.

Table 51: id options

Examples:

Print the current user's UID, GID, and supplementary groups:

```
id
```

Print the UID of the current user:

```
id -u
```

Print the primary group name of the current user:

```
id -gn
```

2.3.12 kill

This command sends a signal to a running process, by default SIGTERM which requests graceful termination.

Synopsis:

```
kill [-<signal>] <PID> [<PID> ...]
kill -l
```

Options:

Option	Description
`-l`	Print a list of available signal names.

Table 52: kill options

Examples:

Terminate the process with PID 1234 gracefully (SIGTERM):

```
kill 1234
```

Force-kill the process with PID 1234 (SIGKILL):

```
kill -9 1234
```

2.3.13 killall

This command sends a signal to all processes matching the specified name, by default SIGTERM.

Synopsis:

```
killall [-q] [-<signal>] <process-name> [<process-name> ...]
```

Options:

Option	Description
`-l`	Print a list of available signal names.
`-q`	Do not report an error if no processes were found.

Table 53: killall options

Examples:

Terminate all `pppd` processes gracefully (SIGTERM):

```
killall pppd
```

Force-kill all `pppd` processes (SIGKILL):

```
killall -9 pppd
```

2.3.14 klog

Info

Execution with `sudo` is required.

This command prints kernel log messages.

Synopsis:
`klog`

Examples:
Display the kernel log:
`sudo klog`

2.3.15 logger

This program makes entries in the system log. It provides a shell command interface to the system log module.

Synopsis:
`logger [OPTIONS] [MESSAGE ...]`

Options:

Option	Description
`-i`	Log the process ID of the logger process with each line.
`-s`	Log the message to standard error, as well as the system log.
`-f <file>`	Log the specified file.
`-p <priority>`	Enter the message with the specified priority. The priority may be specified numerically or as a `facility.level` pair.
`-t <tag>`	Mark every line in the log with the specified tag.
`-u <socket>`	Write to the specified socket instead of using the built-in syslog routines.
`-d`	Use a datagram instead of a stream connection to the socket.

Table 54: logger options

Examples:
Send the message `System rebooted` to the syslog daemon.
`logger "System rebooted"`

Send the message `System going down immediately!!!` to the syslog daemon at the `emerg` level and `user` facility.
`logger -p user.emerg "System going down immediately!!!"`

2.3.16 openssl

The `openssl` program is a command-line tool for using the various cryptography functions of the OpenSSL library. It can be used for:

- Creation of RSA, DH, and DSA key parameters
- Creation of X.509 certificates, CSRs, and CRLs
- Calculation of message digests
- Encryption and decryption with ciphers
- SSL/TLS client and server tests
- Handling of S/MIME signed or encrypted mail

Synopsis:

`openssl [<option> ...]`

Options:

Info

For a detailed description of this command, refer to the Linux manual pages.

Examples:

Generate a new RSA key for the SSH server:

```
openssl genrsa -out /etc/certs/ssh_rsa_key 512
```

Generate a new self-signed certificate for the HTTPS server:

```
openssl req -new -out /tmp/csr -newkey rsa:1024 -nodes -keyout /etc/certs/https_key
openssl x509 -req -setstart 700101000000Z -setend 400101000000Z \
  -in /tmp/csr -signkey /etc/certs/https_key -out /etc/certs/https_cert
```

2.3.17 passwd

This command changes user passwords. A user with the *User* role can change only their own password; a user with the *root* role can change passwords for all users.

Synopsis:

`passwd [options] [LOGIN]`

Options:

Option	Description
`-a, -all`	Report password status on all accounts
`-d, -delete`	Delete the password for the named account
`-e, -expire`	Force expire the password for the named account
`-h, -help`	Display this help message and exit
`-k, -keep-tokens`	Change password only if expired
`-i, -inactive INACTIVE`	Set password inactive after expiration to INACTIVE
`-l, -lock`	Lock the password of the named account
`-n, -mindays MIN_DAYS`	Set minimum number of days before password change to MIN_DAYS
`-q, -quiet`	Quiet mode
`-r, -repository REPOSITORY`	Change password in REPOSITORY repository
`-R, -root CHROOT_DIR`	Directory to chroot into
`-P, -prefix PREFIX_DIR`	Directory prefix
`-S, -status`	Report password status on the named account
`-u, -unlock`	Unlock the password of the named account
`-w, -warndays WARN_DAYS`	Set expiration warning days to WARN_DAYS
`-x, -maxdays MAX_DAYS`	Set maximum number of days before password change to MAX_DAYS
`-s, -stdin`	Read new token from stdin

Table 55: passwd options

Examples:

Change the password for the currently logged-in user `john`:

`passwd`

```
Changing password for john.
Current password:
New password:
Retype new password:
passwd: password updated successfully
```

Administrator changes the password for user `john` (requires *root* role):

`passwd john`

```
New password:
Retype new password:
passwd: password updated successfully
```

2.3.18 pidof

This command lists the PIDs of all processes whose names match those specified on the command line.

Synopsis:

```
pidof <process-name> [<option>] [<process-name> ...]
```

Options:

Option	Description
-s	Display only a single PID.

Table 56: pidof options

Examples:

Find the PID of the `sshd` process:

```
pidof sshd
```

Find a single PID of the `dnsmasq` process:

```
pidof -s dnsmasq
```

2.3.19 ps

This command displays information about currently running processes on the system.

Synopsis:

```
ps [options]
```

Options:

Option	Description
-w	Wide output (do not truncate long lines).
-a	Show processes from all terminals.
-e	Show all processes.

Table 57: ps options

Examples:

Display all running processes:

```
ps
```

Display all processes with full-width output:

```
ps -ww
```

2.3.20 restore

Info

Execution with `sudo` is required.

This program restores a previously created router configuration from a backup file generated by the `backup` command; see Chapter *2.3.1 backup*. The backup file may be either plain text or encrypted.
During restoration, the configuration is validated, optionally decrypted, and then applied to the device.

Synopsis:

```
restore [-p <password>] <filename>
```

Options:

Option	Description
`-p <password>`	Decrypt the backup file before restoration (required if the backup was created with `backup -p`).

Table 58: restore options

Notes:

- The `<filename>` must be provided as a normal command argument (unlike `backup`, which uses output redirection).
- If the backup is encrypted, the same password used during backup must be supplied with the `-p` option.
- When the configuration is changed, the system generates an internal *configuration changed* event.

Examples:

Restore a plain-text backup:

```
sudo restore /tmp/my.cfg
```

Restore an encrypted backup:

```
sudo restore -p MySecretPass /tmp/backup.enc
```

Typical output messages returned by `restore`:

- `Configuration remains unchanged.`
- `File not found.`
- `Decryption of configuration failed.`
- `Configuration successfully updated.`

2.3.21 rlog

Info

Execution with `sudo` is required.

This command prints emergency log messages.

Synopsis:

`rlog`

Examples:

Display the emergency log:

`sudo rlog`

2.3.22 slog

Info

Execution with `sudo` is required.

This command prints the system log (file `/var/log/messages`).

Synopsis:

`slog [-n <number>] [-f]`

Options:

Option	Description
`-n`	Print the last N lines instead of the last 10.
`-f`	Follow the log output as the file grows.

Table 59: slog options

Examples:

Follow the system log in real time:

`sudo slog -f`

Display the last 50 lines of the system log:

`sudo slog -n 50`

2.3.23 service

Info

Execution with `sudo` is required.

This command starts, stops, or restarts a specified service. All available services can be listed with `ls /etc/init.d/`.

Synopsis:

```
service <service_name> <start | stop | restart>
```

Examples:

Start the `cron` service:

```
sudo service cron start
```

Restart the `syslog` service:

```
sudo service syslog restart
```

Stop the `ppp` service:

```
sudo service ppp stop
```

2.3.24 su

This program can be used to change user id or become root.

Synopsis:

```
su [OPTIONS] [-] [username]
```

Options:

Option	Description
`-p, -m`	Preserve environment
`-c CMD`	Command to pass to `sh -c`
`-s SH`	Shell to use instead of default shell

Table 60: su options

Examples:

Change to the root user while preserving the environment.

```
su -p
```

Execute a command as the root user.

```
su -c "whoami"
```

Change to a specific user and use a specific shell.

```
su -s /bin/bash username
```

2.3.25 sudo

The `sudo` command allows permitted users to execute a command as the superuser or another user, as specified by the system's security policy.

Synopsis:

```
sudo -h | -K | -k | -V
sudo -v [-ABkNnS] [-g group] [-h host] [-p prompt] [-u user]
sudo -l [-ABkNnS] [-g group] [-h host] [-p prompt] [-U user] [-u user]
    [command [arg ...]]
sudo [-ABbEHkNnPS] [-C num] [-D directory] [-g group] [-h host] [-p prompt]
    [-R directory] [-T timeout] [-u user] [VAR=value] [-i | -s] [command [arg ...]]
sudo -e [-ABkNnS] [-C num] [-D directory] [-g group] [-h host] [-p prompt]
    [-R directory] [-T timeout] [-u user] file ...
```

Options:

Option	Description
`-A`	Use a helper program for password prompts.
`-B`	Enable background execution.
`-b`	Run the command in the background.
`-C num`	Close all file descriptors with a number higher than `num`.
`-D directory`	Change to `directory` before executing the command.
`-E`	Preserve the user environment when executing the command.
`-e`	Edit files as the specified user.
`-g group`	Run the command as the specified group.
`-h host`	Specify a remote host (if supported by policy).
`-i`	Run the shell as a login shell.
`-k`	Invalidate the user's cached credentials.
`-K`	Remove the user's cached credentials completely.
`-l`	List permitted or specified commands.
`-N`	Non-interactive mode; do not prompt for a password.
`-n`	Do not prompt for a password (useful in scripts).
`-p prompt`	Customize the password prompt.
`-R directory`	Change the root directory for command execution.
`-S`	Read the password from standard input.
`-T timeout`	Set a timeout for the command.
`-u user`	Run the command as *user* instead of the default target user (root).
`-U user`	When used with `-l`, list the privileges of *user* instead of the current user. Requires root privileges.
`-V`	Display the version of `sudo`.

Table 61: sudo options

Examples:

Run a command as the superuser:

```
sudo command
```

List the commands available to the current user:
`sudo -l`

Run a command as a specified user (e.g., `admin`):
`sudo -u admin command`

Edit a file with elevated permissions:
`sudo -e /path/to/file`

Run a command in non-interactive mode without prompting for a password:
`sudo -n command`

2.3.26 sysext

Info

Execution with `sudo` is required.

The `sysext` command is used to activate (merge) or deactivate (unmerge) Router Apps in the filesystem.

Synopsis:

```
sysext {merge|unmerge|refresh}
```

Options:

Option	Description
`merge`	Scans for Router Apps stored in `/var/lib/extensions` and activates (merges) them into the filesystem. These appear as read-only, typically under the `/opt` directory.
`unmerge`	Deactivates (unmerges) all merged Router Apps, resulting in an empty `/opt` directory.
`refresh`	Deactivates and then reactivates all Router Apps.

Table 62: sysext options

2.3.27 times

This shell built-in command displays the accumulated user and system times for the current shell and all its child processes. It is useful for evaluating the performance and resource usage of scripts.

Synopsis:

```
times
```

Examples:

Display accumulated shell and child process times:
`times` Output:

```
0m0.020s 0m0.070s
0m0.130s 0m0.260s
```

2.3.28 top

Warning

This command displays only free memory, not available memory. For more information, refer to Chapter *2.3.9 free*.

This program provides a dynamic real-time view of a running system. It can display system summary information, as well as a list of processes or threads currently being managed by the kernel.

Synopsis:

```
top [-b] [-nCOUNT] [-dSECONDS]
```

Options:

Option	Description
`-b`	Batch mode - could be useful for sending output from *top* to other programs or to a file. In this mode, *top* will not accept input and runs until the iterations limit you've set with the '-n' command-line option, or until killed.
`-nCOUNT`	Exit after N iterations.
`-dSECONDS`	Delay between updates in *secs* format.

Table 63: top options

Keys:

Key	Description
`n/m/p/t`	Sort by PID / memory / CPU / time.
`r`	Reverse sort.
`q, ^C`	Exit.

Table 64: top keys

Examples:

Run `top` in batch mode and exit after 5 iterations:

```
top -b -n5
```

Run `top` and update the output every 10 seconds:

```
top -d10
```

2.3.29 umask

This shell built-in command sets or displays the default file permission creation mask. The umask determines which permissions are *not* set on newly created files and directories.

Synopsis:

```
umask [OPTION]... [MODE]
```

Options:

Option	Description
-S	Display the current umask in symbolic format.
-p	Display the output in a format reusable as input.

Table 65: umask options

Examples:

Display the current umask value:

```
umask
```

Set a new umask so that new files are accessible only by the owner:

```
umask 077
```

Display the current umask in symbolic format:

```
umask -S
```

2.3.30 umupdate

Info

Execution with `sudo` is required.

This command installs, updates, or removes a Router App from the command line.

Synopsis:

```
umupdate [-a <filename>] [-d <n>]
```

Options:

Option	Description
`-a`	Install or update a Router App. Specify the path to the installation file.
`-d`	Remove an installed Router App with the specified name. The list of installed apps can be obtained with `service module list`.

Table 66: umupdate options

Examples:

Install a *Python 3* Router App from an installation file:

```
sudo umupdate -a /var/tmp/python3.v3.tgz
```

Remove an installed *Python 3* Router App:

```
~ # ls /opt
fota          log_temp.csv  python3       usrled_mgmt   zabbix_agent
iperf3        pinger        sleepmode     webTerm
~ # umupdate -d python3
```

2.3.31 whoami

This program prints the user name associated with the current effective user ID.

Synopsis:

```
whoami
```

Examples:

Print the current user's name.

```
whoami
```

Output:

```
admin
```

2.4 Network Commands

Networking commands facilitate the configuration and troubleshooting of network settings. These tools are essential for managing connections, analyzing traffic, and ensuring secure communication over the network.

2.4.1 arp

Info

This command has read-only rights.

This program displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol.

Synopsis:

```
arp [-a <hostname>][-s <hostname> <hw_addr>][-d <hostname>][-v][-n][-i <if>]
[-D <hostname>][-A ][-f <filename>]
```

Options:

Option	Description
`-a`	The entries will be displayed in alternate (BSD) style.
`-v`	Tell the user what is going on by being verbose.
`-n`	Shows numerical addresses instead of trying to determine symbolic host, port or user names.
`-i`	Select an interface.

Table 67: arp options

With no flags, the program displays the current ARP entry for hostname.

Examples:

View the ARP table without resolving IP addresses to domain names:

```
arp -n
```

2.4.2 brctl

Info

The `brctl` command is a legacy utility for managing Ethernet bridges. While retained for backward compatibility, it is considered deprecated. For new configurations, especially those involving VLANs or Distributed Switch Architecture (DSA), it is strongly recommended to use the modern `ip` and `bridge` commands. `brctl` does not support advanced features like VLAN filtering.

This command sets up, maintains, and inspects the Ethernet bridge configuration in the Linux kernel. An Ethernet bridge connects different Ethernet networks, making them appear as a single unified network to all connected devices.

Each connected network segment corresponds to a physical interface added to the bridge. These individual interfaces are bundled into one larger logical network, which corresponds to the bridge network interface itself.

Synopsis:

`brctl [<command>]`

Commands:

Command	Parameters	Description
addbr	<bridge>	Creates a new bridge instance.
delbr	<bridge>	Deletes an existing bridge.
addif	<bridge> <device>	Adds a network interface (port) to a bridge.
delif	<bridge> <device>	Removes a network interface from a bridge.
setageing	<bridge> <time>	Sets the Ethernet address ageing time (in seconds).
setbridgepri	<bridge> <priority>	Sets the bridge's priority for STP (Spanning Tree Protocol).
setfd	<bridge> <time>	Sets the bridge's forward delay (in seconds).
sethello	<bridge> <time>	Sets the time between hello packets for STP.
setmaxage	<bridge> <time>	Sets the maximum message age for STP.
setpathcost	<bridge> <port> <cost>	Sets the STP cost of a path via a specific port.
setportprio	<bridge> <port> <priority>	Sets the STP priority for a specific port.
show		Displays a list of all current bridge instances.
showmacs	<bridge>	Displays a list of MAC addresses learned by the bridge.
showstp	<bridge>	Displays the STP status for a bridge.
stp	<bridge> {on | off}	Enables or disables the Spanning Tree Protocol on a bridge.

Table 68: brctl commands

Examples:

Create a bridge named `br0` and add interfaces `eth0` and `eth1` to it:

```
brctl addbr br0
brctl addif br0 eth0
brctl addif br0 eth1
```

Display the status of all configured bridges:

```
brctl show
```

2.4.3 bridge

Info

The `bridge` command is the modern, feature-rich utility for managing Ethernet bridges and their advanced features, such as VLAN filtering. It is part of the `iproute2` suite and serves as the replacement for the legacy `brctl` command. For all new configurations, the use of `ip` and `bridge` is strongly recommended.

The `bridge` command allows for the configuration and inspection of bridge devices, ports, and VLAN settings. It works in conjunction with the `ip` command, which is used for creating the bridge device itself and for assigning interfaces to it.

Synopsis:

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

Global Options:

Option	Description
`-V, -Version`	Displays the version of the `bridge` utility.
`-s, -stats`	Displays more detailed statistics.
`-d, -details`	Provides more detailed information in the output.
`-j, -json`	Displays output in JSON format.
`-p, -pretty`	Makes JSON output more human-readable.
`-o, -oneline`	Formats output on a single line, which is useful for scripting.

Table 69: global bridge options

The `bridge` utility operates on several objects. The most common objects are `link`, `mdb`, and `vlan`.

Object: link

The `link` object is used to inspect and configure bridge ports.

Command	Description
`show`	Displays the status and configuration of all bridge ports.
`set dev <port> <args>`	Sets various parameters for a specific port. Common arguments include: • `cost <value>`: Sets the STP path cost. • `priority <value>`: Sets the STP port priority. • `state <state>`: Sets the port state (e.g., `forwarding`). • `vlan_filtering <0|1>`: Disables or enables VLAN filtering on the bridge device. Must be set on the bridge interface (e.g., `br0`), not a port.

Table 70: bridge link commands

Object: fdb

The `fdb` object manages the Forwarding Database, which contains the MAC address table for the bridge.

Command	Description
`show [dev <bridge>]`	Displays the FDB entries (MAC table) for a given bridge.
`add <MAC> dev <port>`	Adds a permanent (static) MAC address entry to the FDB, associating it with a specific port.
`delete <MAC> dev <port>`	Removes a MAC address entry from the FDB.
`append <MAC> dev <port>`	Appends a static FDB entry that is externally learned.

Table 71: bridge fdb commands

Object: vlan

The `vlan` object is used to configure VLAN filtering on bridge ports, which is a key feature not available in `brctl`. VLAN filtering must first be enabled on the bridge device itself using `bridge link set dev <bridge> vlan_filtering 1`.

Command	Description
`show [dev <port>]`	Displays the VLAN configuration for all ports or for a specific port.
`add vid <ID> dev <port> [pvid] [untagged]`	Adds a VLAN to a port. • `vid <ID>`: The VLAN ID to add. • `pvid`: Marks this VLAN as the Port VLAN ID (native VLAN). Ingress untagged traffic is assigned to this VLAN. • `untagged`: Egress traffic for this VLAN will be sent untagged.
`delete vid <ID> dev <port>`	Removes a VLAN from a port.

Table 72: bridge vlan commands

Examples:

Create a bridge, add ports, and bring it up. This uses the `ip` command, which is standard practice.
`ip link add name br0 type bridge` Create a bridge device named br0
`ip link set dev eth0 master br0` Add eth0 to the bridge
`ip link set dev eth1 master br0` Add eth1 to the bridge
`ip link set dev br0 up` Bring the bridge interface up

Enable VLAN filtering on the bridge and configure VLANs on its ports.
`bridge link set dev br0 vlan_filtering 1` Enable VLAN awareness
`bridge vlan add vid 100 dev eth1` Allow tagged VLAN 100 on eth1
`bridge vlan add vid 200 dev eth2 pvid untagged` Allow untagged VLAN 200 on eth2 (native VLAN)

Display the VLAN configuration for the bridge.
`bridge vlan show`

2.4.4 curl

`curl` (Client URL) is a versatile tool for transferring data to or from a server. It supports a wide range of protocols, including HTTP, HTTPS, FTP, and many others. It is often used as a more feature-rich alternative to `wget` (see Chapter 2.4.34).
Curl is an extremely powerful tool with hundreds of options. For complete documentation, type `curl --help` or visit the online manual: curl.1 the man page

Synopsis:
`curl [options...] <url>`

Common Options:

Option	Description
`-o <file>`	Write the output to a specified file instead of stdout.
`-O`	Write output to a local file named like the remote file we get.
`-I`	Fetch the HTTP-header only. Useful for checking if a link is alive or seeing server information.
`-L`	If the server reports that the requested page has moved to a different location (3xx error), this option will make curl redo the request on the new place.
`-k`	Allow insecure connections (ignore SSL certificate issues). Common on local networks with self-signed certificates.
`-u <user:pwd>`	Specify the user name and password to use for server authentication.

Table 73: curl options

Examples:
Download a file, keeping the remote filename:
`curl -O https://downloads.example.com/firmware.bin`

Check HTTP response headers without downloading the page:
`curl -I https://www.google.com`
Output:
```
HTTP/2 200 OK
Content-Type: text/html; charset=ISO-8859-1
```

Call a web API with authentication (e.g., for Dynamic DNS update):
`curl -u "admin:password123" "https://api.service.com/update?ip=1.2.3.4"`

2.4.5 dig

This command is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers returned by the queried servers in a human-readable format. The output includes the question section, answer section, authority section, and additional section as appropriate.

The `dig` utility supports both IPv4 and IPv6 queries, various query types (A, AAAA, MX, NS, TXT, etc.), and advanced DNS features like DNSSEC validation, EDNS options, and encrypted transports (DoH/DoT).

For complete documentation of all options including extensive debug flags (+d-opt), query classes, and advanced features, see: dig(1) — Arch Linux manual page

Synopsis:

```
dig [@global-server] [domain] [q-type] [q-class] {q-opt}
  {global-d-opt} host [@local-server] {local-d-opt}
  [ host [@local-server] {local-d-opt} [...] ]
```

Note: Global debug options (before hostname) affect all queries. Local debug options (after hostname) apply only to that specific lookup. Default query class is `in` and type is `a`.

Options:

Option	Description
`-4`	Use IPv4 query transport only.
`-6`	Use IPv6 query transport only.
`-b address[#port]`	Bind to source address/port.
`-c class`	Specify query class (in,hs,ch,...).
`-f filename`	Batch mode - read queries from filename.
`-h`	Print help and exit.
`-k keyfile`	Specify TSIG key file.
`-m`	Enable memory usage debugging.
`-p port`	Specify port number.
`-q name`	Specify query name.
`-r`	Do not read `/.digrc`.
`-t type`	Specify query type (a,mx,ns,soa,...).
`-u`	Display times in microseconds instead of milliseconds.
`-v`	Print version and exit.
`-x dot-notation`	Shortcut for reverse lookups (PTR queries).
`-y [hmac:]name:key`	Specify named base64 TSIG key.

Table 74: dig options

Examples:

Basic domain lookup (A record):
`dig example.com`

Specify nameserver and query type:
`dig @8.8.8.8 example.com MX`

IPv6-only AAAA query
`dig -6 example.com AAAA`

Reverse DNS lookup using shortcut
`dig -x 93.184.216.34`

Custom port and query class

```
dig -p 5353 chaos.example.com CH TXT
```

DNSSEC validation with trace

```
dig +trace +dnssec example.com
```

Multiple queries with different options

```
dig google.com +short A
```

DNS over HTTPS

```
dig +https example.com
```

Show statistics and memory usage

```
dig -m example.com +stats +multiline
```

2.4.6 email

The program can be used for sending email.

Warning

To work properly, this command requires the SMTP service to be configured correctly. Refer to the *Configuration Manual* of your router, chapter *Configuration → Services → SMTP*.

Synopsis:

```
email -t <to> [-s <subject>] [-m <message>] [-a <attachment>] [-r <retries>]
```

Options:

Option	Description
`-t`	E-mail address of the recipient
`-s`	Subject, enter the subject in quotation marks
`-m`	Message, enter the message in quotation marks
`-a`	Attachment file of the email
`-r`	Number of attempts to send the e-mail (default value: 2)

Table 75: email options

Examples:

Send the system log as an email attachment:

```
email -t john.doe@email.com -s "System Log" -a /var/log/messages
```

2.4.7 ether-wake

This command sends a Wake-on-LAN (WoL) Magic Packet to a network interface, which can wake up sleeping machines that support this feature. The target machine is identified by its MAC address.

Synopsis:

```
ether-wake [-b] [-i IFACE] [-p aa:bb:cc:dd[:ee:ff]] MAC
```

Options:

Option	Description
`MAC`	The MAC address of the network card of the machine to be woken up. The address must be specified in the format `00:11:22:33:44:55`.
`-b`	Broadcasts the Magic Packet to all devices on the network segment.
`-i IFACE`	Specifies the network interface through which the Magic Packet will be sent. If not specified, the default interface is *eth0*.
`-p PASSWORD`	Appends a four or six-byte password to the Magic Packet for systems that require a SecureON password. The password should be specified in a MAC-like hex format, separated by colons.

Table 76: ether-wake options

Examples:

Wake up a machine using the default `eth0` interface:
`ether-wake 00:11:22:33:44:55`

Wake up a machine, sending the packet through the *eth1* interface.
`ether-wake -i eth1 00:11:22:33:44:55`

Broadcast a Magic Packet to wake up a machine with a specific MAC address.
`ether-wake -b 00:11:22:33:44:55`

Wake up a machine that requires a six-byte SecureON password.
`ether-wake -p 11:22:33:44:55:66 00:11:22:33:44:55`

2.4.8 ethtool

Info

This command has read-only rights.

This command displays or modifies Ethernet interface settings.

Synopsis:

```
ethtool [<option> ...] <devname> [<commands>]
```

Options:

For a detailed description of this command, refer to the Linux manual pages.

Examples:

View the status of the interface `eth0`:

```
ethtool eth0
```

2.4.9 hostname

Info

- Execution with `sudo` is required.
- Go to *Configuration* → *System* → *Identification* in the router GUI if you want to change the hostname.

The `hostname` command is used to display the system's network name. In a network environment, this name helps to identify the router among other devices.

On this system, the `hostname` command is used solely for viewing the name. It finishes execution immediately and returns you to the shell prompt.

Synopsis:

`hostname [-b]`

Options:

Option	Description
`-b`	Set hostname to 'localhost' if it is otherwise unset/empty (boot default).

Table 77: hostname options

Examples:

Display the current hostname:
`sudo hostname`
Output:
`Router`

2.4.10 ifconfig

Info

This command has read-only rights.

The `ifconfig` (interface configurator) utility is used to display or configure network interfaces. It allows you to assign IP addresses, enable or disable interfaces, and manage settings like MTU or promiscuous mode.

Note: Changes made with `ifconfig` are immediate but temporary. They will be lost after a system reboot unless they are also updated in the router's permanent configuration files or via the GUI.

Synopsis:

`ifconfig [-a] <interface> [<option> ...]`

Examples:

View the status of all active interfaces:

`ifconfig`

Output:

```
eth0   Link encap:Ethernet  HWaddr 00:11:22:33:44:55
       inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       ...
```

2.4.11 ip

Info

This command has read-only rights.

The `ip` command is a powerful tool for network interface configuration, routing, and tunnel management. It serves as a modern replacement for the older `ifconfig`, `route`, and `arp` utilities.

For complete documentation of all sub-commands and advanced networking features (tunnels, xfrm, policy routing), see ip-address(8) — Arch Linux manual page.

Synopsis:

```
ip [options] <object> command | help
```

Common Options:

Option	Description
`-s`	Statistics. Output more information (e.g., packet counts). Repeat for more detail.
`-o`	Oneline. Output each record on a single line for easier parsing with `grep` or `awk`.
`-4` / `-6`	Shortcut for IPv4 or IPv6 protocol family.

Table 78: ip common options

Common Objects:

Object	Description
`link`	Network device (e.g., eth0, wlan0). Used to bring interfaces up or down.
`addr`	IP addresses assigned to devices.
`route`	Routing table entries.
`neigh`	ARP or NDISC cache entries (neighbor management).

Table 79: ip common objects

Examples:

Display all interfaces and their IP addresses:

```
ip addr show
```

Display the routing table:

```
ip route list
```

2.4.12 ipcalc

The `ipcalc` utility is used to calculate and display network settings based on an IP address and a netmask or prefix length. It is an essential tool for verifying subnetting and determining broadcast or network addresses.

Synopsis:

```
ipcalc [options] ADDRESS[/PREFIX] [NETMASK]
```

Options:

Option	Description
`-b`	Display the broadcast address for the given network.
`-n`	Calculate and display the network address.
`-m`	Display the netmask for the provided IP address/prefix.
`-p`	Show the prefix length (e.g., /24).
`-h`	Display the hostname (via reverse DNS lookup).
`-s`	Silent mode. Suppress error messages.

Table 80: ipcalc options

Examples:

Calculate the broadcast address, network address, and netmask for a /24 subnet:
`ipcalc -bnm 192.168.1.100/24`
Output:

```
NETMASK=255.255.255.0
BROADCAST=192.168.1.255
NETWORK=192.168.1.0
```

Example #2: Determine prefix length from a netmask
If you have a traditional netmask and need the CIDR prefix notation.
`ipcalc -p 192.168.1.100 255.255.255.0`
Output:

```
PREFIX=24
```

Example #3: Get only the netmask
Useful for scripts that need to extract a specific value.
`ipcalc -m 192.168.1.100/26`
Output:

```
NETMASK=255.255.255.192
```

2.4.13 iw

Info

This command has read-only rights.

This program is used for displaying and manipulating wireless devices and their configuration.

Synopsis:

`iw [options] command`

Options:

For more details see iw manual page.

Command	Description
`-debug`	enable netlink debugging
`-version`	show version

Table 81: iw options

Commands:

For more details see iw manual page.

Command	Description
`dev`	List all devices or specify a device to manipulate.
`link`	Display the status of the current link.
`station dump`	Show information about all stations.
`phy`	Show information about physical devices.

Table 82: iw commands

Examples:

Scan for available wireless networks on `wlan0`:
`iw dev wlan0 scan`

Display the status of the current wireless link on `wlan0`:
`iw dev wlan0 link`

Show detailed information about all connected stations:
`iw dev wlan0 station dump`

Show capabilities of the physical device `phy0`:
`iw phy phy0 info`

2.4.14 nc

The `nc` (`netcat`) program can be used to open a pipe to IP:port.

Synopsis:

`nc [OPTIONS] HOST PORT` : connect
`nc [OPTIONS] -l -p PORT [HOST] [PORT]` : listen

Options:

Option	Description
`-e`	PROG Run PROG after connect (must be last)
`-l`	Listen mode, for inbound connects
`-lk`	With -e, provides persistent server
`-p PORT`	Local port number
`-s ADDR`	Local address
`-w SEC`	Timeout for connects and final net reads
`-i SEC`	Delay interval for lines sent
`-n`	Don't do DNS resolution
`-u`	UDP mode
`-b`	Allow broadcasts
`-v`	Verbose
`-o FILE`	Hex dump traffic
`-z`	Zero-I/O mode (scanning)

Table 83: nc options

Examples:

Open a TCP connection to port 42 of 192.168.3.1 using port 31337 as the source port with a 5-second timeout:

```
nc -p 31337 -w 5 192.168.3.1 42
```

Listen on port 8080 and print received data:

```
nc -l -p 8080
```

2.4.15 net-snmpinform

This command is designed to send SNMP INFORM messages to a management entity. It supports SNMP versions 1, 2c, and 3, offering a reliable way to notify management systems of significant events.

Synopsis:

```
net-snmpinform [OPTIONS] AGENT TRAP-PARAMETERS
```

Options:

For more details, see the *snmpinform(1) - Linux man page*.

Option	Description
`-v 1|2c|3`	Specifies the SNMP version to use.
`-c COMMUNITY`	Sets the community string (for SNMP v1 and v2c).
`-a PROTOCOL`	Sets the authentication protocol (for SNMP v3).
`-A PASSPHRASE`	Sets the authentication protocol passphrase (for SNMP v3).
`-l LEVEL`	Sets the security level (for SNMP v3).
`-u USER-NAME`	Sets the security name (for SNMP v3).

Table 84: net-snmpinform options

Examples:

`net-snmpinform -v 2c -c public AGENT ” 0 .1.3.6.1.6.3.1.1.5.2`
Sends an INFORM message to the SNMP manager specified by AGENT using SNMP version 2c and the community string ”public”.

`net-snmpinform -v 3 -u myUser -l authPriv -a SHA -A myAuthPass -x DES -X myPrivPass AGENT ” 0 .1.3.6.1.6.3.1.1.5.3`
Sends an SNMPv3 INFORM message with authentication (SHA) and encryption (DES), using the specified usernames and passphrases.

`net-snmpinform -v 1 -c public AGENT .1.3.6.1.4.1.8072.2.3.2.1 0 .1.3.6.1.4.1.8072.2.3.2.26`
Uses SNMP version 1 to send an INFORM message with specified enterprise OID and trap parameters.

2.4.16 net-snmptrap

This command is used to send SNMP trap messages to a management entity. It supports SNMP versions 1, 2c, and 3.

Synopsis:

```
net-snmptrap [OPTIONS] AGENT TRAP-PARAMETERS
```

Options:

For more details see the *snmptrap(1) - Linux man page*.

Option	Description
`-v 1|2c|3`	Specifies SNMP version to use.
`-c COMMUNITY`	Set the community string (for SNMP v1 and v2c).
`-a PROTOCOL`	Set authentication protocol (for SNMP v3).
`-A PASSPHRASE`	Set authentication protocol pass phrase (for SNMP v3).
`-l LEVEL`	Set security level (for SNMP v3).
`-u USER-NAME`	Set security name (for SNMP v3).
`-C i`	Send an INFORM instead of a TRAP.

Table 85: net-snmptrap options

Examples:

`net-snmptrap -v 2c -c public AGENT ” 0 .1.3.6.1.4.1.8072.2.3.0.1 0 0 ”`
Sends a test trap to the SNMP manager specified by AGENT using SNMP version 2c and the community string ”public”.

`net-snmptrap -v 3 -u myUser -l authPriv -a SHA -A myAuthPass -x DES -X myPrivPass AGENT ” 0 .1.3.6.1.6.3.1.1.5.1`
Sends an SNMPv3 trap with authentication (SHA) and encryption (DES), using the specified usernames and passphrases.

`net-snmptrap -v 2c -c public -C i AGENT ” 0 .1.3.6.1.4.1.8072.2.3.0.2 0 0 ”`
Uses the `-C i` option to send an INFORM message instead of a TRAP using SNMP version 2c.

2.4.17 netstat

The `netstat` (network statistics) command is a valuable troubleshooting tool used to display active network connections (sockets), routing tables, and network interface statistics.

Synopsis:
`netstat [options]`

Options:

Option	Description
`-l`	Display only listening server sockets (wait for incoming connections).
`-a`	Display all sockets (both listening and connected).
`-e`	Display extended information.
`-n`	Do not resolve names. Show IP addresses and port numbers numerically (this significantly speeds up the output).
`-r`	Display the kernel routing table.
`-t`	Filter for TCP sockets.
`-u`	Filter for UDP sockets.
`-w`	Filter for RAW sockets.
`-x`	Filter for UNIX domain sockets.

Table 86: netstat options

Examples:
List all listening TCP and UDP ports numerically:
`netstat -tuln`

Combining the `-t`, `-u`, `-l`, and `-n` flags is the most common way to see which services are currently waiting for connections on the router, without resolving DNS names.
`netstat -tuln`
Output:
```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
udp        0      0 0.0.0.0:53              0.0.0.0:*
```

Example #2: Print the routing table
Use the `-r` flag to show the routing table. Adding `-n` prevents delays caused by DNS lookups on the gateway IP addresses.
`netstat -rn`
Output:
```
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG        0 0          0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

2.4.18 nsupdate

The `nsupdate` command is a DNS update utility that allows dynamic updates to a zone using the DNS UPDATE protocol (RFC 2136). It reads commands from files or standard input to add, delete, or modify DNS resource records on a nameserver that supports dynamic updates.

Synopsis:

```
nsupdate [-CdDi] [-L level] [-l] [-g | -o | -y keyname:secret | -k keyfile] [-p port]
         [-v] [-V] [-P] [-T] [-4 | -6] [filename]
```

Options:

Option	Description
`-C`	Check names processing.
`-d`	Debug mode.
`-D`	Debug level.
`-i`	Interactive mode.
`-L level`	Set logging level.
`-l`	Localhost mode (default server/zone).
`-g`	GSS-TSIG authentication.
`-o`	GSS-TSIG ownername mode.
`-y keyname:secret`	Direct TSIG key specification.
`-k keyfile`	Specify TSIG key file for authentication.
`-p port`	Specify destination port.
`-v`	Verbose output.
`-V`	Print version.
`-P`	Primary server only.
`-T`	Use TCP transport.
`-4`	Use IPv4 transport only.
`-6`	Use IPv6 transport only.
`[filename]`	Read update commands from specified file (or stdin).

Table 87: nsupdate options

Examples:

Update a DNS A record using a TSIG key (the `send` command is required):

```
nsupdate -k ddns-key.txt «EOF
update delete routerX.example.com A
update add routerX.example.com 200 A 1.2.3.4
send
EOF
```

Example #2: Interactive mode with verbose output
Start an interactive session to type commands manually.
`nsupdate -v`

Example #3: Update from a command file
Read a list of update commands from a text file. Ensure the file contains the `send` command at the very end.
`nsupdate -4 updates.txt`

Example #4: Localhost zone updates
Update the zone on the local server without specifying credentials.
`nsupdate -l`

2.4.19 ntpdate

Info

Execution with `sudo` is required.

This command synchronizes the system time from an NTP server.

Synopsis:

`ntpdate [-p <probes>] [-t <timeout>] <server>`

Options:

Option	Description
`-p`	Specify the number of samples to be acquired from each server as the integer samples, with values from 1 to 8 inclusive.
`-t`	Specify the maximum time waiting for a server response as the value timeout, in seconds and fraction.

Table 88: ntpdate options

Examples:

Synchronize the system time with an NTP server:
`sudo ntpdate time.windows.com`

2.4.20 ping

This command sends ICMP echo requests to a network host to test connectivity.

Synopsis:

```
ping [OPTIONS] HOST
```

Options:

Option	Description
`-4,-6`	Force IP or IPv6 name resolution.
`-c CNT`	Send only CNT pings.
`-s SIZE`	Send SIZE data bytes in packets (default = 56).
`-i SECS`	Interval.
`-A`	Ping as soon as reply is recevied.
`-t TTL`	Set TTL.
`-I IFACE/IP`	Source interface or IP address.
`-W SEC`	Seconds to wait for the first response (default 10). After all -c CNT packets are sent.
`-w SEC`	Seconds until ping exits (default:infinite). Can exit earlier with -c CNT.
`-q`	Quiet mode, only displays output at start and when finished.
`-p HEXBYTE`	Payload pattern.

Table 89: ping options

Examples:

Send a single ICMP echo request with a 500-byte payload to 10.0.0.1:

```
ping -c 1 -s 500 10.0.0.1
```

Continuously ping a host to check connectivity:

```
ping 192.168.1.1
```

2.4.21 ping6

This command is designed for diagnosing IPv6 network connections by sending ICMP ECHO_REQUEST packets to a specified host. It is a useful tool for testing the reachability of hosts on an IPv6 network and measuring the round-trip time for messages sent from the originating host to a destination computer.

Usage:

```
ping6 [OPTIONS] HOST
```

Description:

Sends ICMP ECHO_REQUEST packets to the HOST specified as an argument. By default, *ping6* sends packets until interrupted. If the host is reachable and responding, *ping6* displays the time taken for the round-trip.

Options:

Option	Description
`-c CNT`	Stop after sending `CNT` pings.
`-s SIZE`	Specifies the number of data bytes to be sent.
`-i SECS`	Wait `SECS` seconds between sending each packet.
`-A`	Ping the host as soon as the reply is received.
`-I IFACE/IP`	Use the specified interface or IP address as the source.
`-W SEC`	Time to wait for the first response before timing out.
`-w SEC`	Timeout before *ping6* exits, regardless of how many packets have been sent or received.
`-q`	Operate in quiet mode, only displaying summary lines at startup and completion.
`-p HEXBYTE`	Pattern to use for payload data.

Table 90: ping6 options

Examples:

Ping an IPv6 host 4 times:

```
ping6 -c 4 fe80::1
```

Ping with a 2-second interval and 120-byte packets:

```
ping6 -i 2 -s 120 fe80::1
```

2.4.22 route

Info

This command has read-only rights.

This command displays and manipulates the IP routing table.

Synopsis:

```
route [ -n ] [ -e ] [ -A ]
```

Options:

Option	Description
-n	Don't resolve names
-e	Display other/more information
-A	Select address family

Table 91: route options

Info

For a detailed description of this command, refer to the Linux manual pages.

Examples:

Display the routing table without resolving IP addresses:

```
route -n
```

2.4.23 scp

This program can be used for secure file transferring between hosts on a network. It uses the `ssh` protocol for data transfer with the same authentication and security.

Synopsis:

```
scp [-12346BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file]
[-l limit] [-o ssh_option] [-P port] [-S program]
[[user@]host1:]file1 ... [[user@]host2:]file2
```

Options:

Option	Description
`-1`	Forces scp to use protocol 1.
`-2`	Forces scp to use protocol 2.
`-4`	Forces scp to use IPv4 addresses only.
`-6`	Forces scp to use IPv6 addresses only.
`-B`	Selects batch mode (prevents asking for passwords or passphrases).
`-C`	Compression enable. Passes the -C flag to ssh to enable compression.
`-c cipher`	Selects the cipher to use for encrypting the data transfer. This option is directly passed to ssh.
`-F ssh_config`	Specifies an alternative per-user configuration file for ssh. This option is directly passed to ssh.
`-i identity_file`	Selects the file from which the identity (private key) for public key authentication is read. This option is directly passed to ssh.
`-l limit`	Limits the used bandwidth, specified in Kbit/s.
`-o ssh_option`	Can be used to pass options to ssh in the format used in ssh_config.
`-P port`	Specifies the port to connect to on the remote host.
`-p`	Preserves modification times, access times, and modes from the original file.
`-q`	Quiet mode: disables the progress meter as well as warning and diagnostic messages from ssh.
`-r`	Recursively copy entire directories. Note that scp follows symbolic links encountered in the tree traversal.
`-S program`	Name of program to use for the encrypted connection. The program must understand ssh options.
`-v`	Verbose mode. Causes scp and ssh to print debugging messages about their progress.

Table 92: scp options

Info

The scp utility exits 0 on success, and >0 if an error occurs.

Examples:

Copy a file from a remote host to the local machine:

```
scp root@remotehost.edu:/etc/version/myFolder
```

Copy a file from the local machine to a remote host:

```
scp /etc/version root@remotehost.edu:/
```

Copy a directory recursively to a remote host:

```
scp -r /home/user root@remotehost.edu:/tmp/bar
```

2.4.24 sipcalc

This command is an advanced console-based IP subnet calculator. It is capable of handling both IPv4 and IPv6 addressing schemes. It provides detailed information about network addresses, subnet masks, and more, making it a valuable tool for network administrators and IT professionals.

Usage:

```
sipcalc [OPTIONS]... <[ADDRESS]... [INTERFACE]... | [-]>
```

Global options

Option	Description
`-a, –all`	Display all possible information.
`-d, –resolve`	Enable name resolution for addresses.
`-h, –help`	Show help message and exit.
`-I, –addr-int=INT`	Specify an interface to add.
`-n, –subnets=NUM`	Display NUM extra subnets starting from the current subnet.
`-u, –split-verbose`	Enable verbose output for subnet splitting.
`-v, –version`	Show version information and exit.
`-4, –addr-ipv4=ADDR`	Specify an IPv4 address to add.
`-6, –addr-ipv6=ADDR`	Specify an IPv6 address to add.

Table 93: sipcalc – global options

IPv4 options

Option	Description
`-b, –cidr-bitmap`	Show CIDR bitmap.
`-c, –classful-addr`	Display classful address information.
`-i, –cidr-addr`	Display CIDR address information (default).
`-s, –v4split=MASK`	Split the current network into subnets of MASK size.
`-w, –wildcard`	Display information for a wildcard (inverse mask).
`-x, –classful-bitmap`	Show classful bitmap.

Table 94: sipcalc – IPv4 options

IPv6 Options

Option	Description
`-e, –v4inv6`	Show IPv4 compatible IPv6 information.
`-r, –v6rev`	Generate IPv6 reverse DNS output.
`-S, –v6split=MASK`	Split the current IPv6 network into subnets of MASK size.
`-t, –v6-standard`	Show standard IPv6 address information (default).

Table 95: sipcalc – IPv6 options

Examples:

Calculate and display all information for an IPv4 address:

```
sipcalc -a 192.168.1.1/24
```

Split an IPv6 network into smaller subnets:
`sipcalc -S 64 2001:db8::/32`

Address and netmask formats are flexible, supporting dotted quad, number of bits, and hex formats. The tool also supports reading arguments from stdin if `-` is used in place of an address or interface name.

For detailed usage and options, refer to the `sipcalc` manual or the help option `-h`.

2.4.25 snmpget

This is an SNMP application that uses the SNMP GET request to query for information on a network entity. One or more object identifiers (OIDs) may be given as arguments on the command line.

Synopsis:
`snmpget [OPTIONS] [-Cf] OID [OID]...`

Options:

Option	Description
`-h, –help`	Display the help.
`-H`	Display configuration file directives understood.
`-v 1|2c|3`	Specifies SNMP version to use.
`-V, –version`	Display package version number.
`-Cf`	If -Cf is not specified, some applications (*snmpdelta*, *snmpget*, *snmpgetnext* and *snmpstatus*) will try to fix errors returned by the agent that you were talking to and resend the request. The only time this is really useful is if you specified a OID that didn't exist in your request and you're using SNMPv1 which requires "all or nothing" kinds of requests.
other	*For other options see the command help.*

Table 96: snmpget options

Examples:
Retrieve the `sysDescr.0` variable from a host using the community string `public`:
`snmpget -c public zeus system.sysDescr.0`

2.4.26 snmpset

This is an SNMP application that uses the SNMP SET request to set information on a network entity. One or more object identifiers (OIDs) must be given as arguments on the command line. A type and a value to be set must accompany each object identifier.

Synopsis:

```
snmpset [OPTIONS] OID TYPE VALUE [OID TYPE VALUE]...
```

Options:

Option	Description
`-h, –help`	Display the help.
`-H`	Display configuration file directives understood.
`-v 1|2c|3`	Specifies SNMP version to use.
`-V, –version`	Display package version number.
other	*For other options see the command help.*

Table 97: snmpset options

The TYPE is a single character, one of:

Character	Description
i	integer
u	unsigned
s	string
x	hex string
d	decimal string
n	nullobj
o	objid
t	timeticks
a	ipaddress
b	bits

Table 98: type options

Examples:

Set `sysContact.0` and `ipForwarding.0`:

```
snmpset -c private -v 1 test-hub \\
  system.sysContact.0 s dpz@noc.rutgers.edu \\
  ip.ipForwarding.0 i 2
```

2.4.27 snmptrap

Info

See similar programs `snmpinform` (Chapter 2.4.15) and `snmptrap` (Chapter 2.4.16).

This command sends an SNMP trap.

Synopsis:

```
snmptrap [-c <community>] [-g <generic>] [-s <specific>] <hostname>
[<oid> <type> <value>]
```

Options:

Option	Description
`-c`	Community
`-g`	Specifies generic trap types: 0 – coldStart 1 – warmStart 2 – linkDown 3 – linkUp 4 – authenticationFailure 5 – egpNeighborLoss 6 – enterpriseSpecific
`-r`	Sends MAC address of eth0 interface
`-s`	Specifies user definition trap types in the enterpriseSpecific

Table 99: snmptrap options

Examples:

Send a trap with the state of digital input BIN0 to 192.168.1.2:

```
snmptrap 192.168.1.2 1.3.6.1.4.1.30140.2.3.1.0 u 'io get bin0'
```

Send a warm-start trap to 192.168.1.2:

```
snmptrap -g 1 192.168.1.2
```

2.4.28 ssh

This is a program for logging into a remote machine and for executing commands on a remote machine.

Synopsis:

```
ssh [-46AaCfGgKkMNnqsTtVvXxYy] [-B bind_interface] [-b bind_address]
  [-c cipher_spec] [-D [bind_address:]port] [-E log_file]
  [-e escape_char] [-F configfile] [-I pkcs11] [-i identity_file]
  [-J destination] [-L address] [-l login_name] [-m mac_spec]
  [-O ctl_cmd] [-o option] [-P tag] [-p port] [-Q query_option]
  [-R address] [-S ctl_path] [-W host:port] [-w local_tun[:remote_tun]]
  destination [command [argument ...]]
```

Options:

For more details see ssh(1) — Linux manual page.

Option	Description
`-4`	Forces to use IPv4 addresses only.
`-6`	Forces to use IPv6 addresses only.
`-i identity_file`	Specifies the file from which the identity (private key) for public key authentication is read.
`-l login_name`	Specifies the user to log in as on the remote machine.
`-p port`	Specifies the port to connect to on the remote host.
`-v`	Verbose mode. Causes ssh to print debugging messages about its progress. This is helpful in diagnosing connection, authentication, and configuration problems.
`-A and -a`	These options control the use of SSH agent forwarding, a mechanism for single sign-on.
`-X and -x`	These manage X11 forwarding, enabling the secure transmission of X11 windows from the remote machine to the local machine.
`-L address`	Specifies that connections to the given TCP port or Unix socket on the local (client) host are to be forwarded to the given host and port, or Unix socket, on the remote side.
`-C`	Requests compression of all data. This can speed up transfers over slow networks.
`-F configfile`	Specifies an alternative per-user configuration file.

Table 100: ssh options

Examples:

Log in to a remote server:
`ssh root@192.168.1.1`

Connect to a non-standard SSH port:
`ssh -p 2222 username@remote_host`

Use a specific private key for authentication:
`ssh -i /path/to/private_key username@remote_host`

Execute a single command on the remote server:
`ssh user@server 'ls -l /var/www'`

Set up SSH local port forwarding:
`ssh -L local_port:remote_host:remote_port username@ssh_server`

2.4.29 tc

The `tc` (traffic control) command in Linux is a powerful tool utilized for managing network bandwidth and handling Quality of Service (QoS). It allows administrators to control the flow of network traffic by defining policies for traffic classification, prioritization, and rate limiting, among other functionalities. This command is essential for optimizing network performance and ensuring that critical network services remain highly available and responsive.

Synopsis:

```
tc [OPTIONS] OBJECT  COMMAND | help
```

Description:

Tc operates on several objects such as qdiscs (queuing disciplines), classes, and filters, each of which plays a vital role in defining traffic control policies. By manipulating these objects, administrators can tailor the network traffic behavior to suit the specific needs of their environment. This includes creating traffic queues, setting up bandwidth limits for different types of traffic, and applying traffic filters to categorize network packets into various classes.

Options:

Option	Description
`-s`	Show detailed information. When used, *tc* displays more detailed information about the specified object.
`-d`	Show raw data. This option is useful for debugging purposes.
`-p`	Pretty print. Formats the output in a more readable form.
`-b`	Batch mode. Allows *tc* to read a series of commands from a file.

Table 101: tc options

Examples:

Show all current qdiscs:

```
tc -s qdisc show
```

Limit outbound traffic on `eth0` to 1 Mbit/s:

```
tc qdisc add dev eth0 root tbf rate 1mbit burst 32kbit latency 400ms
```

For a comprehensive guide on QoS configuration, see the Quality of Service (QoS) Application Note.

2.4.30 tcpdump

This command captures and displays network traffic.

Synopsis:

```
tcpdump [-AdDeflLnNOpqRStuUvxX] [-c <count>] [-C <file size>] [-E algo:secret]
[-F <file>] [-i <interface>] [-r <file>] [-s <snaplen>] [-T type] [-w <file>]
[-y <datalinktype>] [expression]
```

Options:

For a detailed description of this command, refer to the Linux manual pages.

Examples:

Capture traffic on interface `ppp0`:

```
tcpdump -n -i ppp0
```

View traffic on interface eth0 except protocol Telnet.

```
tcpdump -n not tcp port 23
```

View UDP traffic on interface eth0.

```
tcpdump -n udp
```

View HTTP traffic on interface eth0.

```
tcpdump -n tcp port 80
```

View all traffic from/to IP address 192.168.1.2.

```
tcpdump -n host 192.168.1.2
```

View traffic from/to IP address 192.168.1.2 except protocol Telnet.

```
tcpdump -n host 192.168.1.2 and not tcp port 23
```

2.4.31 traceroute

This command traces the network route to a remote host.

Synopsis:

```
traceroute,[-FIldnrv],[-f,<1st_ttl>],[-m,<max_ttl>],[-p,<port#>],[-q,<nqueries>]
[-s,<src_addr>],[-t,<tos>],[-w,<wait>],[-g,<gateway>] [-i,<iface>],[-z,<pausemsecs>]
host,[data size]
```

Options:

Item	Description
`-4,-6`	Force IP or IPv6 name resolution
`-F`	Set the don't fragment bit
`-l`	Display the TTL value of the returned packet
`-n`	Print hop addresses numerically rather than symbolically
`-r`	Bypass the normal routing tables and send directly to a host
`-f N`	First number of hops (default 1)
`-m N`	Set the max time-to-live (max number of hops)
`-q N`	Set the number of probes per hop (default is 3)
`-p N`	Set the base UDP port number used in probes (default is 33434)
`-s IP`	Use the following IP address as the source address
`-i IFACE`	Source interface
`-t N`	Set the type-of-service in probe packets to the following value (default 0)
`-w SEC`	Set the time (in seconds) to wait for a response to a probe (default 3 sec)
`-z MSEC`	Wait before each send
`-I`	Use ICMP ECHO instead of UDP datagrams
`-d`	Enable socket level debugging
`-v`	Verbose output

Table 102: traceroute options

Examples:

Trace the route to a remote host:

```
traceroute 8.8.8.8
```

```
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 46 byte packets
 1  192.168.1.1 (192.168.1.1)  1.014 ms  0.921 ms  0.878 ms
 2  10.0.0.1 (10.0.0.1)  8.342 ms  8.251 ms  8.198 ms
 3  * * *
 4  72.14.194.34 (72.14.194.34)  19.443 ms  19.381 ms  19.302 ms
 5  8.8.8.8 (8.8.8.8)  19.451 ms  19.367 ms  19.289 ms
```

Trace the route without resolving hostnames (faster output):

```
traceroute -n 8.8.8.8
```

Trace the route using ICMP instead of UDP (useful when UDP is filtered):

```
traceroute -I 8.8.8.8
```

Limit the trace to a maximum of 15 hops:

```
traceroute -m 15 192.168.100.1
```

Specify a source interface and send only 1 probe per hop:

```
traceroute -i eth0 -q 1 10.0.0.1
```

2.4.32 traceroute6

This command is a network diagnostic tool included with BusyBox that is designed to trace the path packets take to reach an IPv6 host. By sending packets with incrementally increasing hop limits (TTL, Time-To-Live), *traceroute6* determines the route packets follow to reach the target host. This command is essential for identifying network bottlenecks and routing issues in IPv6 networks.

Synopsis:

```
traceroute6 [-nrv] [-f 1ST_TTL] [-m MAXTTL] [-q PROBES] [-p PORT]
[-t TOS] [-w WAIT_SEC] [-s SRC_IP] [-i IFACE] [-z PAUSE_MSEC] HOST [BYTES]
```

Description:

`traceroute6` utilizes IPv6 packets to trace the network route from the source to the specified HOST. It provides various options to customize the trace, including setting the first and maximum number of hops, the number of probes per hop, and the source IP address or interface.

Options:

Option	Description
`-n`	Do not resolve IP addresses to their domain names, display numeric addresses.
`-r`	Bypass the normal routing tables and send directly to the host.
`-f N`	Set the initial time-to-live (hop limit) to N.
`-m N`	Specify the maximum number of hops (TTL) traceroute6 will probe.
`-q N`	Set the number of probe packets per hop.
`-p N`	Use N as the base UDP port number for probes.
`-s IP`	Use IP as the source address for the outgoing probe packets.
`-i IFACE`	Specify the interface through which traceroute should send packets.
`-t N`	Set the type-of-service in probe packets to N.
`-w SEC`	Set the time to wait for a response to a probe.
`-z MSEC`	Wait MSEC milliseconds between sending each packet.

Table 103: traceroute6 options

Examples:

Trace the route to an IPv6 host without resolving names:
`traceroute6 -n HOST`

Trace the route with a specific number of probes per hop:
`traceroute6 -q 1 HOST`

Specify a source address and maximum number of hops:
`traceroute6 -s SRC_IP -m 15 HOST`

2.4.33 vconfig

Info

This command is available with read-only permissions. Although the original utility allows creating and removing virtual Ethernet devices, these operations are not supported in this environment.

2.4.34 wget

The `wget` utility is a non-interactive network downloader used to retrieve files from the web using HTTP, HTTPS, or FTP protocols. Because it is non-interactive, it is highly suitable for use in background scripts, cron jobs, or automated provisioning.

Synopsis:

```
wget [options] <URL>
```

Options:

Option	Description
`–spider`	Only check if the URL exists (do not download the file). The exit status (`$?`) is 0 if the file exists.
`-c`	Continue retrieving a partially downloaded file (resume an aborted transfer).
`-q`	Quiet mode. Turn off all output, including error messages.
`-P <DIR>`	Save the downloaded file to the specified directory `<DIR>` instead of the current working directory.
`-S`	Print the HTTP or FTP server response headers.
`-T <SEC>`	Set the network read timeout to `<SEC>` seconds.
`-O <FILE>`	Save the downloaded document as `<FILE>`. Use `-` to write to standard output (stdout).
`-o <FILE>`	Log all messages and output to `<FILE>` instead of the terminal.
`-U <STR>`	Identify as `<STR>` in the `User-Agent` HTTP header.
`-Y on/off`	Explicitly turn the use of a proxy server on or `off`.

Table 104: wget options

Examples:

Download a file from a local HTTP server:

```
wget http://10.0.0.1/my.cfg
```

Save a downloaded file under a different name:

```
wget -O /tmp/new_config.cfg http://10.0.0.1/my.cfg
```

Check whether a file exists on a server without downloading it:

```
wget --spider http://10.0.0.1/my.cfg
```

Output:

```
0
```

2.5 Scripting/Shell Commands

This section covers commands integral to shell scripting and command-line manipulation. These commands are foundational for automating tasks, managing files, and configuring system behavior through scripts.

2.5.1 awk

This program scans each input file for lines that match any of a set of patterns specified literally in program-text or in one or more files specified as -f progfile.

Synopsis:

```
awk [-v var=val] [-F FS] [-f progfile] [<program-text>] [<file> ...]
```

Note: Program can be inline or from file (-f). Multiple -f options allowed.

Options:

Option	Description
`-v var=val`	Set variable before program execution (BEGIN block).
`-F FS`	Set input field separator (FS variable).
`-f progfile`	Read program from file (multiple allowed).

Table 105: awk options

Examples:

Show IP address of Gateway

```
route -n | awk '/^0.0.0.0/ { print $2 }'
```

Output:

```
192.168.1.1
```

2.5.2 break

This command is used within looping constructs in Bash/Shell scripting to exit from the loop prematurely. It breaks out of the nearest enclosing loop, skipping the remaining iterations of the loop.

Synopsis:

`break [n]`

Note: Optional argument `n` specifies which enclosing loop to break (default: 1 = innermost loop).

Options:

Option	Description
`[n]`	Exit from the nth enclosing loop (default: 1).

Table 106: break options

2.5.3 clear

This command clears the terminal screen/console, moving the cursor to the home position (top-left corner). It is equivalent to pressing `Ctrl+L` in most terminals. The screen content is not deleted from memory - only the display is refreshed.

Synopsis:

`clear`

2.5.4 continue

The `continue` command is used within loops in Bash/Shell scripting to skip the rest of the current loop iteration and continue with the next iteration. This command is particularly useful when a condition is met that requires prematurely proceeding to the next cycle of the loop without executing the remaining commands in the current iteration.

Synopsis:

`continue [n]`

Note: Optional argument `n` specifies which enclosing loop to continue (default: 1 = innermost loop).

Options:

Option	Description
`[n]`	Continue at the nth enclosing loop (default: 1).

Table 107: continue options

2.5.5 echo

This command prints strings to standard output. It supports basic escape sequence interpretation and newline control.

Synopsis:

```
echo [-n] [-e] [-E] [<string> ...]
```

Note: Multiple strings are concatenated with spaces. Backslash escapes are processed only with `-e`.

Options:

Option	Description
`-n`	Do not output the trailing newline.
`-e`	Enable interpretation of backslash escapes.
`-E`	Disable interpretation of backslash escapes (default).

Table 108: echo options

Examples:

Switch profile to ”Standard”.

```
echo "PROFILE=" > /etc/settings
reboot
```

Switch profile to ”Alternative 1”.

```
echo "PROFILE=alt1" > /etc/settings
reboot
```

Send a sequence of bytes 0x41,0x54,0x0D,0x0A to serial line (write data in octal).

```
echo -n -e "\101\124\015\012" > /dev/ttyS0
```

2.5.6 eval

This command is a built-in utility in Bash/Shell scripting environments used to concatenate its arguments into a single command, which is then executed by the shell. This command is particularly useful for constructing commands based on variables or processing complex expressions where commands depend on other commands' outputs or file contents.

Synopsis:

```
eval [arg ...]
```

Note: All arguments are concatenated (separated by spaces) into a single string, then parsed and executed as shell commands.

Options:

Option	Description
`[arg ...]`	Arguments to be concatenated and executed as shell command(s).

Table 109: eval options

Examples:

Execute dynamically constructed command

```
CMD="ls -l"; eval $CMD
```

Parse output from previous command

```
read var1 var2 < <(date); eval echo "Year: $var2"
```

Create function from variable content

```
FUNC="echo Hello"; eval $FUNC
```

2.5.7 exec

This command is used in shell scripting to replace the current shell process with a specified program. Unlike running a program normally, *exec* does not return to the shell once the program completes. Instead, the new program takes over the current process space, maintaining the same process ID (PID).

Synopsis:

```
exec command [arguments]
```

Note: Without arguments, exec reopens stdin/stdout/stderr from the shell's current redirections.

Options:

Option	Description
command [arguments]	Replace shell with specified command (no return).
(no arguments)	Reopen stdin/stdout/stderr using current redirections.
-a name	Set argv[0] of executed program to name.
-l	Place a dash at start of argv[0] (login shell).

Table 110: exec options

Examples:

Replace shell with vi editor

```
exec vi /etc/config
```

Replace shell with new shell instance

```
exec /bin/sh
```

Redirect and exec new program

```
exec > /tmp/log 2>&1
```

Exec with custom argv[0]

```
exec -a mysh /bin/sh
```

2.5.8 exit

This command terminates the current shell session or script execution. It allows the script or shell to exit with an optional exit status, which can be used to indicate the success or failure of the script's execution to the calling environment.

Synopsis:

```
exit [n]
```

Note: Exit status `n` should be 0-255. `0` indicates success, non-zero values indicate various errors.

Options:

Option	Description
`[n]`	Exit status code (0-255, default: last command status).

Table 111: exit options

Examples:

Exit shell with success status

```
exit 0
```

Exit script with error status

```
exit 1
```

Exit with status of last command

```
false; echo $?; exit
```

2.5.9 export

This command in shell scripting is utilized to set or export environment variables and functions to the current shell and all processes started from it. By marking an environment variable or function to be exported, it becomes available to subprocesses spawned from the shell, allowing those processes to use the variable or function.

Synopsis:

```
export [NAME[=VALUE] ...]
```

Note: Without arguments, displays all exported variables. `NAME=VALUE` sets and exports; `NAME` only exports existing variable.

Options:

Option	Description
`[NAME[=VALUE]]`	Set and/or export environment variable(s).
`(no arguments)`	List all currently exported variables.
`-p`	List in portable format (`export NAME=VALUE`).
`-n`	Remove NAME from exported list.

Table 112: export options

Examples:

Display all exported variables

```
export
```

Set and export HOME variable

```
export HOME=/root
```

Append to PATH and export

```
export PATH=$PATH:/usr/local/bin
```

2.5.10 hash

This command in shell scripting is utilized to handle the hash table of commands in memory, which tracks the full path of previously executed commands. This optimizes the shell's performance by avoiding the need to search the $PATH environment variable for command locations on subsequent executions.

Synopsis:

```
hash [options] [name ...]
```

Note: Without arguments, displays current hash table contents. With name only, adds command to hash table.

Options:

Option	Description
`-r`	Reset/clear the entire hash table.
`-l`	List hash table in portable format.
`-p pathname name`	Probe specific pathname for command name.
`-d name`	Remove specified name from hash table.
`-t name`	Show path for specific command only.
`[name]`	Add command name to hash table.
`(no arguments)`	Display all hashed commands and paths.

Table 113: hash options

Examples:

Display current hash table
`hash`

Add command to hash table
`hash myscript`

Reset hash table
`hash -r`

Remove specific command
`hash -d myscript`

Direct pathname mapping
`hash -p /usr/local/bin/myscript myscript`

2.5.11 inc

This command is a specialized, proprietary tool designed to output a number that is incremented by one from the given value. This utility can be particularly useful in scripting and automation tasks where numerical adjustments and calculations are required.

Synopsis:

`inc <value>`

Note: Accepts only numeric input. Outputs result to stdout.

Options:

Option	Description
<value>	Numeric value to increment by 1.

Table 114: inc options

Examples:

Increment value 5 to 6
`inc 5`
Output:
6

Use in arithmetic sequence
`val=10; next=$(inc $val); echo $next`
Output:
11

Increment a variable
`counter=20`
`counter=$(inc $counter)`
`echo $counter`
Output:
21

2.5.12 let

This command is a built-in shell command primarily used for evaluating arithmetic expressions. It provides a straightforward method for performing numerical calculations within shell scripts, supporting operators like addition, subtraction, multiplication, division, and bitwise operations.

Synopsis:

```
let "expression"
```

Note: Variables in expressions don't need $ prefix. Expressions must be quoted to prevent shell expansion.

Options:

Option	Description
”expression”	Arithmetic expression to evaluate (quoted).

Table 115: let options

Examples:

Increment counter

```
count=5; let "count += 1"; echo $count
```

Output:

```
6
```

Basic arithmetic

```
a=10; b=3; let "c = a * b"; echo $c
```

Output:

```
30
```

Compare and assign

```
x=15; y=25; let "z = (x > y) ? x : y"; echo $z
```

Output:

```
25
```

2.5.13 local

This command is a shell built-in that is used within functions to declare variables as having a function-local scope. It prevents variable names from conflicting with variables outside the function, making shell scripts more reliable and modular.

Synopsis:

`local [name[=value] ...]`

Note: Can only be used inside functions. Variables remain local to function scope.

Options:

Option	Description
`name[=value]`	Declare local variable with optional initial value.
`-`	Restore local variables from stack (advanced usage).

Table 116: local options

Examples:

Local variable in function

`myfunc() { local temp=10; echo $temp; }; myfunc`

Output:

```
10
```

Local variable in function

`f() { local x=5 y=10; echo $((x+y)); }`

`f`

Output:

```
15
```

Prevent global namespace pollution

`foo()  local counter=1; counter=$((counter+1)); echo $counter;`

`foo`

Output:

```
2
```

2.5.14 nohup

`nohup` is short for "No Hang-up". This command runs a program immune to hangups, with output redirected to `nohup.out` by default. The program continues running even after the user logs out.

Synopsis:

```
nohup <program> [<arguments>]
```

Note: Output is redirected to `nohup.out` unless redirected elsewhere. Exit status is returned when program completes.

Options:

Option	Description
`program [arguments]`	Program to run immune to hangups and logout.

Table 117: nohup options

Examples:

Run ping in background immune to logout

```
nohup ping -c 10 google.com &
```

Output:

```
nohup: appending output to nohup.out
```

Custom output file with nohup

```
nohup ./myscript.sh > mylog.txt 2>&1 &
```

2.5.15 printf

This command formats and prints ARGUMENT(s) according to FORMAT (C printf syntax). It provides precise control over output formatting including numbers, strings, and escape sequences.

Synopsis:

```
printf FORMAT [ARGUMENT...]
```

Note: FORMAT string uses C-style conversion specifiers. Unlike `echo`, no automatic newline is added.

Format Options:

Specifier	Description
`%d, %i`	Signed decimal integer
`%u`	Unsigned decimal integer
`%o`	Unsigned octal
`%x, %X`	Unsigned hexadecimal (lower/upper)
`%f`	Decimal floating point
`%e, %E`	Scientific notation (lower/upper)
`%g, %G`	Shortest of %e/%f or %E/%F
`%a, %A`	Hexadecimal floating point (lower/upper)
`%c`	Character
`%s`	String
`%p`	Pointer address
`%n`	Store number of characters printed
`%%`	Literal percent sign

Table 118: printf format specifiers

Examples:

Print hex number

```
printf "Output: %X\n" 10
```

Output:

```
A
```

Print system variables

```
printf "User: %s PWD: %s" $USER $PWD
```

Output:

```
User: john PWD: /var/data/john
```

Right-aligned decimal

```
printf "Value: %8d\n" 42
```

Output:

```
Value:    42
```

2.5.16 read

This command reads one line from standard input (stdin) and assigns values to shell variables.

Synopsis:

```
read [-r] [-t timeout] [name1] [name2] ...
```

Options:

Option	Description
`-r`	Raw mode: backslash does not act as escape character
`-t SEC`	Timeout in seconds
`name1`	Variable to store first word
`name2`	Variable to store second word (etc.)

Table 119: read options

Examples:

Read single line into variable
`read line;`
Input:
`Hello World`
`echo "You entered: $line"`
Output:
`You entered: Hello World`

Read into multiple variables
`read user host;`
Input:
`john server1`
`echo "$user@$host"`
Output:
`john@server1`

Read with timeout
`read -t 3 line || echo "Timeout"`
Output:
`Timeout`

Raw mode (preserve backslashes)
`read -r line;`
Input:
`path to config.ini`
`echo "$line"`
Output:
`path to config.ini`

2.5.17 readonly

Marks variables and functions as read-only (immutable).

Synopsis:

```
readonly [-p] [name[=value] ...]
```

Options:

Option	Description
name	Variable or function name

Table 120: readonly options

Examples:

Read-only variable

```
readonly VERSION="1.2.3";
VERSION="2.0"
```

Output:

```
-sh: VERSION: is read only
```

2.5.18 return

Terminates function execution and returns exit status to caller.

Synopsis:

```
return [n]
```

Options:

Option	Description
n	Exit status (0-255, default 0)

Table 121: return options

Examples:

Return success from function

```
check_file() { [ -f /etc/passwd ] && return 0; };
check_file && echo "OK"
```

Output:

```
OK
```

Return success status

```
divide() { [ $2 -eq 0 ] && return 1; return 0; };
```

Output:

(nothing)

Return error status

```
divide 10 0 && echo "OK" || echo "Error: $?"
```

Output:

```
Error: 1
```

Return error/success message

```
divide 10 1 && echo "OK" || echo "Error: $?"
```

Output:

```
OK
```

2.5.19 shlock

This command locks a specified file during script execution using `flock(LOCK_EX)`. Ensures only one script instance accesses the file at a time. If already locked, waits until released.

Synopsis:

```
shlock <filename>
```

Description:

Creates exclusive lock on specified file, preventing concurrent script execution. Other shlock instances block until the locking script terminates. Manual file access (vi/cat) remains possible. Lock automatically released when script exits.

Examples:

Multi-script coordination:
Script 1: `shlock /tmp/config.txt; sleep 10; echo "Done"`
Script 2: `shlock /tmp/config.txt` `(waits 10s until Script 1 finishes)`

Lock released automatically when script process terminates.

2.5.20 set

Shell built-in to set/unset options and positional parameters ($1, $2...).

Synopsis:

`set [--] [arg1 arg2 ...]`
`set -e`, `set -x`, `set +e`, `set +x`

Examples:

Set positional parameters
`set -- file1 file2 file3`
`echo $1 $2 $3`
Output:
```
file1 file2 file3
```

Enable exit-on-error in a script
`set -e; false; echo "This won't print"`
```
(script exits immediately, echo is not printed)
```

Enable command tracing
`set -x`
`ls /tmp`
```
+ ls /tmp
```
Output:
```
+ ls /tmp
(shows each command as executed)
```

Disable tracing
`set +x`
Output:
```
+ set +x
```

2.5.21 shift

The `shift` command shifts the positional parameters to the left. It renames the positional parameters `$N+1`, `$N+2`, ... to `$1`, `$2`, ... and so on. If `n` is specified, the parameters are shifted by `n` positions.

Synopsis:
`shift [n]`

Examples:

Example #1: Process all script arguments using a loop

`script.sh` code:

```
#!/bin/sh
# Process arguments one by one
while [ "$#" -gt 0 ]; do
  echo "Processing: $1"
  shift
done
```

Execute script:
`./script.sh apple banana cherry`

Output:
```
Processing: apple
Processing: banana
Processing: cherry
```

Example #2: Shift by 2 positions
`set -- a b c d e; echo $1 $2 $3; shift 2; echo $1 $2 $3`

Output:
```
a b c
c d e
```

Example #3: Check the number of remaining arguments
The `$#` variable holds the count of current positional parameters. Continuing from Example #2 (where 3 arguments remained):
`echo "Arguments remaining: $#"`

Output:
```
Arguments remaining: 3
```

2.5.22 sleep

The `sleep` command pauses the execution of a script or process for a specified amount of time. The time must be specified as an integer number of seconds.

Synopsis:

`sleep <seconds>`

Examples:

Example #1: Sleep for 30 seconds
`sleep 30`

Example #2: Sleep for 5 minutes (300 seconds)
`sleep 300`

Example #3: Practical usage in a script
This script checks for an internet connection every 10 seconds.

```
#!/bin/sh
while ! ping -c 1 8.8.8.8 > /dev/null; do
    echo ”Waiting for internet connection...”
    sleep 10
done
echo ”Connected!”
```

2.5.23 source

The `source` command reads and executes commands from a specified file within the **current** shell environment.

Unlike executing a script directly (which launches a new subshell process), `source` runs the commands in the existing process. This means that any variables set, functions defined, or environment changes made by the script will persist in the current shell after the script finishes.
Note: The `.` (dot) command is a synonym for `source` in POSIX shells.

Synopsis:

`source <filename> [arguments]`
or
`. <filename> [arguments]`

Examples:

Example #1: Load system-wide environment variables
It is common to use `source` to reload configuration files without rebooting or logging out.
`source /etc/profile`

Example #2: Persisting variables (The difference between execution and sourcing)
Create a file named `config.sh` with content of:
`MY_VAR="12345"`

Scenario A: Executing the script
If you run the script as an executable, the variable is set in a subshell and disappears when the script ends:
`chmod +x config.sh`
`./config.sh`
`echo "Value: $MY_VAR"`
Output:
`Value:`

Scenario B: Sourcing the script
If you use `source`, the variable is set in your current shell. Note that if the file is in the current directory, you should prefix it with `./` to ensure it is found.
`source ./config.sh`
`echo "Value: $MY_VAR"`
Output:
`Value: 12345`

2.5.24 test

The `test` command checks file types and compares values. It returns an exit status of 0 (true) or 1 (false).

This command is most commonly used as `[ EXPRESSION ]` in `if` statements or combined with logical operators `&&` and `||`. The `[` command is a synonym for `test`, but it requires a closing `]` as its last argument.

Synopsis:

`test <expression> | [ <expression> ]`

Common Operators:

Option	Description
`-f <file>`	Returns true if the file exists and is a regular file.
`-d <file>`	Returns true if the file exists and is a directory.
`-e <file>`	Returns true if the file exists (regardless of type).
`-z <string>`	Returns true if the length of the string is zero.
`-n <string>`	Returns true if the length of the string is non-zero.
`s1 = s2`	Returns true if the strings are equal.
`s1 != s2`	Returns true if the strings are not equal.
`n1 -eq n2`	Returns true if integers are equal.
`n1 -ne n2`	Returns true if integers are not equal.
`n1 -gt n2`	Returns true if *n1* is greater than *n2*.
`n1 -ge n2`	Returns true if *n1* is greater than or equal to *n2*.
`n1 -lt n2`	Returns true if *n1* is less than *n2*.
`n1 -le n2`	Returns true if *n1* is less than or equal to *n2*.
`! <expr>`	Returns true if the expression is false (negation).

Table 122: commonly used tests

Examples:

Example #1: Check if a file exists
Using the `[ ]` syntax within an if-statement:

```
if [ -f ”/etc/config/network” ]; then
    echo ”Network configuration found.”
else
    echo ”Configuration missing.”
fi
```

Example #2: Compare two strings
Note the spaces around the brackets and the equals sign.

```
STATUS=”active”
if [ ”$STATUS” = ”active” ]; then
    echo ”The service is running.”
fi
```

Example #3: Integer comparison
Check if a counter is greater than 5.

```
COUNT=10
if [ ”$COUNT” -gt 5 ]; then
    echo ”Count is greater than 5.”
fi
```

2.5.25 trap

The `trap` command allows you to catch signals and execute code when they occur. This is useful for cleaning up temporary files, ignoring specific signals, or executing custom commands when a script finishes.

Synopsis:

`trap [action] [signal...]`

To list currently defined traps:
`trap`

Common Signals:

Option	Description
SIGINT (2)	Interrupt signal (typically sent by pressing Ctrl+C).
SIGTERM (15)	Termination signal. Requests the program to end gracefully.
SIGHUP (1)	Hangup signal. Often sent when a terminal window is closed.
EXIT (0)	A pseudo-signal used to execute commands when the shell exits (regardless of the reason).

Table 123: common signals

Examples:

Example #1: Ensure cleanup on exit
This script creates a temporary file and ensures it is deleted even if the user presses Ctrl+C. Note that we trap `EXIT`, `SIGINT`, and `SIGTERM` to ensure the cleanup runs in all cases.

```
#!/bin/sh
TEMP_FILE="/tmp/data.tmp"

# Define cleanup function
cleanup() {
    echo "Cleaning up temporary files..."
    rm -f "$TEMP_FILE"
    exit
}

# Register the trap for EXIT and signals
trap cleanup EXIT SIGINT SIGTERM

touch "$TEMP_FILE"
echo "Working... (Press Ctrl+C to test cleanup)"
sleep 10
```

Example #2: Ignore a signal
To ignore a signal (e.g., to prevent a script from stopping if the terminal closes), use an empty string as the action.
`trap '' SIGHUP`

Example #3: Reset signal to default
To revert the signal handling back to the system default, use a hyphen (`-`).
`trap - SIGINT`

2.5.26 type

The `type` command indicates how the shell interprets a specific command name. It reveals whether a command is a shell builtin, an external executable file, a function, or if it does not exist at all.

This is particularly useful for debugging scripts to ensure that you are executing the specific version of a command you expect (e.g., a system binary versus a user-defined function).

Synopsis:

`type <command_name>`

Examples:

Example #1: Check a shell builtin
Commands like `cd` or `echo` are built directly into the shell.
`type cd`
Output:
`cd is a shell builtin`

Example #2: Check an executable
Commands like `cat` or `vi` are typically external binaries (or applets in BusyBox).
`type cat`
Output:
`cat is /bin/cat`

Example #3: Check if a command exists (Scripting)
You can use `type` in a script to verify that a required program is installed before trying to run it. This example checks for `cat` and displays the hosts file.

```
CHECK_CMD=”cat”

if type ”$CHECK_CMD” > /dev/null; then
    echo ”Command '$CHECK_CMD' found. Reading /etc/hosts:”
    $CHECK_CMD /etc/hosts
else
    echo ”Error: $CHECK_CMD is not installed.”
    exit 1
fi
```

Output:
```
Command ’cat’ found. Reading /etc/hosts:
127.0.0.1 localhost
```

2.5.27 unset

The `unset` command removes variables and functions from the current shell execution environment. Once unset, the variable or function is no longer accessible.

By default, `unset` tries to unset a variable. If you want to explicitly unset a function, use the `-f` option.

Synopsis:

```
unset [-v] [-f] <name>
```

Examples:

Example #1: Delete a variable
Define a variable, display it, then remove it.

```
CITY="Prague"
```

```
echo "City is: $CITY"
```

Output:

```
City is: Prague
```

```
unset CITY
```

```
echo "City is: $CITY"
```

Output:

```
City is:
```

Example #2: Delete a function
Define a function and then remove it using the `-f` flag.

```
# Define function
```

```
hello() {
```

```
echo "Hello World"
```

```
}
```

Run function

```
hello
```

Output:

```
Hello World
```

Unset function

```
unset -f hello
```

Try to run it again

```
hello
```

Output:

```
-sh: hello: not found
```

2.5.28 wait

The `wait` command pauses the script execution until a background process finishes. You can specify a Process ID (PID) to wait for a specific process. If no argument is provided, the shell waits for **all** currently active background child processes to complete.

This is essential for scripts that launch parallel tasks and need to ensure they are all finished before proceeding.

Synopsis:

`wait [PID]`

Examples:

Example #1: Wait for a specific process
Start a process in the background using `&`, capture its PID using the `$!` variable, and then wait for it to finish.
Start background process
`sleep 30 &`
Capture PID
`PID=$!`
Wait for the PID
`echo "Waiting for PID $PID..."`
`wait $PID`
`echo "Done."`
Output:

```
Waiting for PID 1045...
Done.
```

Example #2: Wait for all background processes
If you do not specify a PID, `wait` pauses until all child processes are finished. This is useful for parallel execution.
Start first job
`sleep 30 &`
Start second job
`sleep 30 &`
Wait for both
`echo "Waiting for jobs..."`
`wait`
`echo "All finished."`

Output:

```
Waiting for jobs...
All finished.
```

2.5.29 watch

The `watch` command is used to run a specified program periodically, showing its output in real-time. This is useful for monitoring status changes, such as network connections, process lists, or file sizes, without manually re-running the command.

Synopsis:

```
watch [-n SEC] [-t] PROG [ARGS]
```

Options:

Option	Description
`-n <SEC>`	Specify the update period in seconds (the default is 2 seconds).
`-t`	Do not print the header (which normally shows the interval, command name, and current time).

Table 124: watch options

Examples:

Example #1: Monitor active network connections
Every 5 seconds, display the current listening ports and active connections using `netstat`.
Run watch with 5-second interval (exit the command by pressing *Ctrl* + *c*):

```
watch -n 5 netstat -tupln
```

Output (Header included):

```
Every 5.0s: netstat -tupln            2026-02-12 12:45:01
(netstat output follows...)
```

Example #2: Monitor system memory without header
Use the `-t` option to display only the raw output of the `free` command, updated every 2 seconds.
Run watch without header (exit the command by pressing *Ctrl* + *c*):

```
watch -t free
```

Output:

```
               total        used        free      shared  buff/cache   available
Mem:         1008460      149408      802824         372       56228      845280
Swap:
```

2.5.30 xargs

The `xargs` command reads items from standard input (delimited by blanks or newlines) and executes a specified command using those items as arguments.

It is commonly used in pipelines to process lists of files generated by commands like `find` or `ls`.

Synopsis:

```
xargs [options] [command [initial-arguments]]
```

Options:

Option	Description
`-o`	Reopen stdin as `/dev/tty`. This is necessary if you want to run interactive applications (like a text editor) via xargs.
`-r`	Do not run the command if the standard input is empty (no input).
`-t`	Print the command line to standard error (stderr) before executing it (verbose mode).
`-n <n>`	Pass at most *n* arguments per command line. The command will be executed repeatedly until all input is processed.
`-s <n>`	Pass a command line of no more than *n* bytes.
`-E <str>`	Stop processing input when the string *str* is encountered.
`-e[<str>]`	Synonym for `-E`. If *str* is omitted, the end-of-file string feature is disabled.

Table 125: xargs options

Examples:

Example #1: Limit arguments per command
Use the `-n` option to process items in groups. In this example, we echo two numbers at a time.
Pipe numbers to xargs

```
printf "1 2 3 4 5 6" | xargs -n 2 echo "Pair:"
```

Output:

```
Pair: 1 2
Pair: 3 4
Pair: 5 6
```

Example #2: Delete files found by `find`
Find files named `core` in `/tmp` and delete them. The `-r` option ensures `rm` is not run if no files are found.
Note: This method may fail if filenames contain spaces or newlines.

```
find /tmp -name core -type f -print | xargs -r rm -f
```

3. Related Resources

You can obtain all product-related documents, software updates, and supplementary materials on the Advantech *Engineering Portal* at *icr.advantech.com*.

For easy access to specific resources, please refer to the following sections of the portal:

- **Router Support Materials:** To access your router's supporting documents (such as the *Hardware Manual* and *Configuration Manual*), the latest firmware, or other technical resources, navigate to *Support* → *Router Models*. Locate your specific model and select the appropriate tab under the *Documents to download* section. Available tabs include *Brochures*, *Manuals*, *Certificates*, *Firmware*, *Images/3D Models*, *PCN/SA*, and *Others*.
- **Router Apps:** To extend your router's functionality, installation packages and comprehensive manuals for various extension modules are available by navigating to *Download* → *Router Apps*.
- **Application Notes:** For detailed guides, configuration examples, and step-by-step instructions for implementing specific networking features and use cases, navigate to *Download* → *Application Notes*.
- **Development Documents:** If you are interested in custom scripting, programming your own applications, or compiling custom modules, navigate to *Development* page.

Appendix: Command Index

X

Z