

Application Note

Command Line Interface for S1 Routers



© 2025 Advantech Czech s.r.o. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photography, recording, or any information storage and retrieval system without written consent. Information in this manual is subject to change without notice, and it does not represent a commitment on the part of Advantech.

Advantech Czech s.r.o. shall not be liable for incidental or consequential damages resulting from the furnishing, performance, or use of this manual.

All brand names used in this manual are the registered trademarks of their respective owners. The use of trademarks or other designations in this publication is for reference purposes only and does not constitute an endorsement by the trademark holder.

Used symbols



Danger – Information regarding user safety or potential damage to the router.



Attention – Problems that can arise in specific situations.



Information – Useful tips or information of special interest.

Firmware Version

This manual is compatible with firmware version 6.5.3 (May 28, 2025).

Contents

1. Document Introduction	1
1.1 Document Content	1
1.2 Command Line Access	2
1.3 Permissions Notes	2
2. Commands	3
2.1 HW Control Commands	4
2.1.1 gsminfo	4
2.1.2 hwclock	5
2.1.3 io	6
2.1.4 led	7
2.1.5 lpm	8
2.1.6 mac	9
2.1.7 reboot	9
2.1.8 report	10
2.1.9 sms	10
2.1.10 status	11
2.1.11 stty	12
2.1.12 tpm2	13
2.2 File/Directory Management Commands	15
2.2.1 basename	15
2.2.2 cat	15
2.2.3 cd	16
2.2.4 chdir	16
2.2.5 cmp	16
2.2.6 cp	17
2.2.7 cut	18
2.2.8 dd	19
2.2.9 decode	19
2.2.10 dirname	20
2.2.11 find	20
2.2.12 grep	21
2.2.13 gunzip	22
2.2.14 gzip	22
2.2.15 head	23
2.2.16 jq	24
2.2.17 less	25
2.2.18 ln	26
2.2.19 ls	27
2.2.20 mkdir	28
2.2.21 mv	28
2.2.22 pwd	29
2.2.23 readlink	29
2.2.24 realpath	30
2.2.25 rm	30
2.2.26 rmdir	31

2.2.27	sed	31
2.2.28	shred	32
2.2.29	split	33
2.2.30	sync	33
2.2.31	tail	34
2.2.32	tar	35
2.2.33	touch	36
2.2.34	vi	37
2.2.35	wc	37
2.2.36	xxd	38
2.2.37	zcat	39
2.3	System Commands	40
2.3.1	backup	40
2.3.2	chmod	40
2.3.3	chown	41
2.3.4	clog	41
2.3.5	date	41
2.3.6	defaults	42
2.3.7	df	42
2.3.8	dmesg	43
2.3.9	faillock	44
2.3.10	free	45
2.3.11	fwupdate	45
2.3.12	id	46
2.3.13	kill	47
2.3.14	killall	47
2.3.15	klog	48
2.3.16	logger	48
2.3.17	openssl	49
2.3.18	passwd	50
2.3.19	pidof	51
2.3.20	ps	51
2.3.21	restore	51
2.3.22	rlog	52
2.3.23	slog	52
2.3.24	service	52
2.3.25	su	53
2.3.26	sudo	54
2.3.27	sysext	56
2.3.28	times	56
2.3.29	top	57
2.3.30	umask	58
2.3.31	umupdate	59
2.3.32	whoami	59
2.4	Network Commands	60
2.4.1	arp	60
2.4.2	brctl	61
2.4.3	curl	62
2.4.4	email	62
2.4.5	ethtool	63

2.4.6	ifconfig	64
2.4.7	ip	65
2.4.8	ipcalc	67
2.4.9	iw	68
2.4.10	nc	69
2.4.11	net-snmpinform	70
2.4.12	net-snmptrap	71
2.4.13	netstat	72
2.4.14	ntpdate	72
2.4.15	ping	73
2.4.16	ping6	74
2.4.17	route	75
2.4.18	scp	76
2.4.19	sipcalc	78
2.4.20	snmpget	79
2.4.21	snmpset	80
2.4.22	snmptrap	81
2.4.23	ssh	82
2.4.24	tc	84
2.4.25	tcpdump	85
2.4.26	traceroute	86
2.4.27	traceroute6	87
2.4.28	vconfig	88
2.4.29	wget	89
2.5	Scripting/Shell Commands	90
2.5.1	awk	90
2.5.2	break	91
2.5.3	continue	91
2.5.4	echo	91
2.5.5	eval	92
2.5.6	exec	92
2.5.7	exit	92
2.5.8	export	92
2.5.9	hash	93
2.5.10	inc	94
2.5.11	let	95
2.5.12	local	96
2.5.13	nohup	96
2.5.14	printf	97
2.5.15	read	98
2.5.16	readonly	98
2.5.17	return	98
2.5.18	shlock	98
2.5.19	set	99
2.5.20	shift	99
2.5.21	sleep	99
2.5.22	source	100
2.5.23	test	101
2.5.24	trap	102
2.5.25	type	102

2.5.26 unset	103
2.5.27 wait	103
2.5.28 xargs	103

3. Related Documents	104
-----------------------------	------------

Appendix: Command Index	105
--------------------------------	------------

List of Figures

List of Tables

1	Description of GSM information	4
2	hwclock options	5
3	io options	6
4	led options	7
5	led commands	7
6	lpm options	8
7	reboot options	9
8	report options	10
9	status options	11
10	stty options	12
11	tpm2 subcommands	14
12	cat options	15
13	cd options	16
14	cmp options	16
15	cp options	17
16	cut options	18
17	dd options	19
18	find expressions	20
19	grep options	21
20	gunzip options	22
21	gzip options	22
22	head options	23
23	jq options	24
24	ln options	26
25	ls options	27
26	mkdir options	28
27	mv options	28
28	rm options	30
29	sed options	31
30	shred options	32
31	split options	33
32	sync options	33
33	tail options	34
34	tar options	35

35	touch options	36
36	vi options	37
37	wc options	37
38	xxd options	38
39	backup options	40
40	chmod options	40
41	date options	41
42	df options	42
43	dmesg options	43
44	faillock options	44
45	fwupdate options	45
46	id options	46
47	kill options	47
48	killall options	47
49	logger options	48
50	passwd options	50
51	pidof options	51
52	slog options	52
53	su options	53
54	sudo options	54
55	sysexr options	56
56	touch options	57
57	touch keys	57
58	umupdate options	59
59	arp options	60
60	brctl commands	61
61	email options	62
62	ifconfig options	64
63	ip options	65
64	ip objects	65
65	iw options	68
66	iw commands	68
67	nc options	69
68	net-snmpinform options	70
69	net-snmptrap options	71
70	netstat options	72
71	ntpdate options	72
72	ping options	73
73	route options	75
74	scp options	76
75	snmpget options	79
76	snmpset options	80
77	Type options	80
78	snmptrap options	81
79	ssh options	82
80	traceroute options	86
81	vconfig commands and options	88
82	wget options	89
83	awk options	90
84	echo options	91

85	nohup options	96
86	printf format specifiers	97
87	xargs options	103

1. Document Introduction

1.1 Document Content

This document is intended for **S1 Routers only**. It provides a complete list of available **console commands** for Advantech routers. 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](#).



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.



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 routers 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 vast and complex ecosystem of Unix-like operating systems, commands serve as 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 and mastering these commands is crucial to effectively harnessing the full potential of the system.

This chapter delves into an array of commands integral to daily operations and specialized tasks alike. From scripting and shell command essentials that form the backbone of system automation and task scheduling, to administration commands that ensure system security and integrity, each command is a piece of the larger puzzle of system management. Network commands provide the keys to configuring, analyzing, and securing network communications, while router management commands focus on the specifics of networking device configuration. Additionally, file and directory management commands offer the means to navigate and manipulate the filesystem, ensuring data organization and accessibility. Lastly, the text processing and analysis commands section reveals the power of Unix-like systems in handling textual data, enabling the user to edit, search, and process text in various complex ways.

Structured to cater to both novice users seeking foundational knowledge and advanced users aiming to refine their expertise, this chapter aims to equip you with the understanding and practical know-how needed to navigate the intricacies of Unix-like operating systems with confidence. Through concise descriptions, usage examples, and categorized presentation, we invite you on a journey to mastery over your system's capabilities, one command at a time.

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

Execution with `sudo` is required.

This command is designed to retrieve and display detailed information about the GSM module's signal quality and connection status. This tool is invaluable for diagnosing connectivity issues, optimizing signal reception, and ensuring reliable GSM network services.

Synopsis:

```
gsminfo
```

Description:

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

Options:

Option	Description
PLMN	Displays the code of the operator to which the GSM module is currently connected.
Cell	Shows the identifier of the cell to which the router is currently connected.
Channel	Indicates the channel number used for communication by the GSM module.
Level	Provides an assessment of the signal quality from the connected cell.
Neighbours	Lists signal quality information from neighboring cells that are within hearing range.
Uptime	Reports the duration since the PPP (Point-to-Point Protocol) connection was established.

Table 1: Description of GSM information



Examples:

Check the signal quality of the GSM module:

```
gsminfo
```

This example would display various GSM module information including signal quality, operator code, and more, directly in the console output.

2.1.2 hwclock

Execution with `sudo` is required.

This program can be used to query and set the hardware clock (RTC).

Synopsis:

```
hwclock [-r] [-s] [-w] [-u] [-l]
```

Options:

Option	Description
-r	Read hardware clock and print result
-s	Set the System Time from the Hardware Clock
-w	Set the Hardware Clock to the current System Time
-u	The hardware clock is kept in coordinated universal time
-l	The hardware clock is kept in local time

Table 2: hwclock options



Examples:

Set the hardware clock to the current system time.

```
hwclock -w -u
```

2.1.3 io

Execution with `sudo` is required.

This program can be used to read binary inputs and to control binary outputs of the router. If installed, it also supports an expansion ports of the router.

Synopsis:

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

Options:

Option	Description
get	Get the state of input
set	Set the state of output

Table 3: io options



Examples:

Get the state of digital input BIN0.

```
io get bin0
```

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

```
io get an1
```

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

```
io get cnt1
```

Set the state of binary output OUT0 to 1.

```
io set out0 1
```

2.1.4 led

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

Execution with `sudo` is required.

This program can be used to control 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 LEDn
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.

```
led on
```

Start blinking slowly with the USR LED.

```
led slow
```


2.1.5 lpm



This command is not supported by routers of [v1](#), v2, ICR-2000, ICR-2400, ICR-2500, and ICR-2600 production lines.

Execution with `sudo` is required.

This program can be used to switch the router into the Low Power Mode. The router then can be awoken up by an event according to the specified parameters. The first option is to wake up the router at the time the specified time period has expired and the second option is to wake up the router by activating the binary input. If both parameters were specified, the router will wake up at the time when the first event comes.

Synopsis:

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

Options:

Option	Description
-b	Wake up the router by activating the binary input. For <i>SmartFlex</i> , <i>SmartMotion</i> , <i>ICR-2800</i> , <i>ICR-4200</i> and <i>ICR-4400</i> router platforms use input <i>BIN1</i> . For <i>SmartStart</i> and <i>ICR-3200</i> router platforms use input <i>BIN0</i> .
-i ¹	Wake up the router at the time when the specified time period has expired. The time interval is specified in seconds.

Table 6: lpm options



Examples:

Sleep the router for five minutes.

```
lpm -i 300
```

Sleep the router and wake up by activating the binary input.

```
lpm -b
```

Sleep the router for five minutes or less if the binary input is activated within five minutes.

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

¹ICR-2700 production line routers support just `-i` option, `-b` option is not supported.

2.1.6 mac

This program can be used to display the MAC address of eth0.

Synopsis:

```
mac [<separator>]
```



Examples:

Display the MAC address of eth0. Will be used as the separator character "-" instead of ":".

```
mac -
```

2.1.7 reboot

Execution with `sudo` is required.

This program can be used to reboot the router.

Synopsis:

```
reboot [-d <delay>] [-n <nosync>] [-f <force>]
```

Options:

Option	Description
-d	Delay interval for rebooting
-n	No call to sync()
-f	Force reboot, do not call shutdown

Table 7: reboot options



Examples:

Reboot router after 10 second.

```
reboot -d 10
```

2.1.8 report

Execution with `sudo` is required.

This command can be used to list the router report from the command line.



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

Synopsis:

```
report [<options>]
```

Options:

Option	Description
<i>no option</i>	Report all sections except the configuration one.
<code>-a</code>	Report all sections.
<code>-s</code>	Report status section.
<code>-m</code>	Report router apps section.
<code>-l</code>	Report log section.
<code>-c</code>	Report configuration section.

Table 8: report options

2.1.9 sms

Execution with `sudo` is required.

This program can be used to send SMS message.

Synopsis:

```
sms <phone number> <text>
```



Examples:

Send SMS "Hello word" on telephone number +420123456789.

```
sms +420123456789 "Hello word"
```

2.1.10 status

Execution with `sudo` is required.

This program writes out the status of the routers interfaces or system. It is equivalent to information available at *General Status* and *Mobile WAN Status* in the routers web administration.

Synopsis:

```
status [-hv] [lan | mobile | module | mwan | ports | sys | wifi ap | wifi sta]
```

Options:

Option	Description
-h	Generates html output (used when called by web interface).
-v	The verbose mode that writes out more detailed information. Units of data amount are just in bytes , not in dynamic units (KB, MB, etc.).
lan	Status of primary LAN. Can be lan 1, lan 2, etc. if available.
mobile	Status of mobile WAN.
module	Status of mobile module. Can be module 1, module 2, etc. if available.
mwan	Status of mobile connection.
ports	Status of available peripheral ports.
sys	System information.
wifi ap	Status of WiFi access point.
wifi sta	Status of WiFi station.

Table 9: status options



Examples:

Show verbose status of mobile connection:

```
# status -v mobile
Registration      : Home Network
Operator         : Vodafone CZ
Technology       : LTE
PLMN             : 23003
Cell             : 10A802
LAC              : 947C
Channel          : 275
Signal Strength  : -90 dBm
Signal Quality   : -12 dB
CSQ              : 11
```

2.1.11 stty

This program can be used to print or to change terminal characteristics.

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 10: stty options



Examples:

To get 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

 Only the ICR-4400 family products support TPM features.

This program can be used to work with the TPM 2.0 (Trusted Platform Module) chip mounted directly onto the routers mainboard.

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

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

Continued on the next page

Continued from previous page

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 11: tpm2 subcommands

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 can be used to strip directory path and .SUFFIX from FILE.

Synopsis:

```
basename FILE [SUFFIX]
```



Examples:

Strip directory from `/home/myfile.txt` path, including `.txt` extension.

```
basename /home/myfile.txt .txt
```

```
myfile
```

2.2.2 cat

This command concatenates files and print on the standard output.

Synopsis:

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

Options:

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

Table 12: cat options



Examples:

View the contents of file `/proc/tty/driver/spear_serial` (info about serial ports of v2 routers).

```
cat /proc/tty/driver/spear_serial
```

Copy the contents of the router configuration files in `/tmp/my.cfg`.

```
cat /etc/settings.* > /tmp/my.cfg
```


2.2.3 cd

This command can be used to change the current working directory.

Synopsis:

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

Options:

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

Table 13: cd options



Examples:

Move to home directory (/root).

```
cd
```

Move to directory /mnt.

```
cd /mnt
```

2.2.4 chdir

The `chdir` command, also known as `cd` in many shell environments, is utilized to change the current working directory of the shell session.

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 14: 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 an hexadecimal or octal value.

2.2.6 cp

This command can be used to copy 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 15: cp options



Examples:

Copy the system log to directory /mnt.

```
cp /var/log/messages* /mnt
```

Copy configuration profile "Alternative 1" to profile "Standard".

```
cp -r /etc/alt1/* /etc
```

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 16: cut options



Examples:

Display the 1st field of each line, using tab as the field separator.

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

Displays the 2nd field of each line, using colon as the field separator.

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

```
b
```

Display the 2nd and every later field.

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

```
b c d
```

Instead of fields, treats characters. Display the third and fourth character.

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

```
cd
```

2.2.8 dd

This command can be used to copy a file and convert the data format in the process, according to the operands specified.

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 17: dd options



Examples:

Erase the microSD card, which is available as `/dev/sda`.

```
dd if=/dev/zero of=/dev/sda bs=4k
```

2.2.9 decode

This command is designed for decoding Base64-encoded values, making it an essential tool for working with configuration files or other data that utilize Base64 encoding for obfuscation or storage efficiency. The decoded value is printed to the standard output, allowing for direct observation or redirection to a file for further use.

Synopsis:

```
decode <base64>
```

Description:

The *decode* command takes a single argument, a Base64-encoded string, and prints its decoded value. This functionality is particularly useful in scenarios where configuration files or data are encoded to prevent manipulation or ensure compatibility.

Options: This proprietary command is straightforward and does not require additional options for its operation.



Examples:

Decode a Base64-encoded string.

```
decode SGVsbG8sIFdvcmxkIQ==
```

This example will output the decoded string of the provided Base64-encoded value, in this case, "Hello, World!".

2.2.10 dirname

This command strips non-directory suffix from FILENAME.

Synopsis:

```
dirname FILENAME
```



Examples:

Strip non-directory suffix from /home/MyFolder/myfile.txt path.

```
dirname /home/MyFolder/myfile.txt
```

```
/home/MyFolder
```

2.2.11 find

Command to search for files in a directory hierarchy.

Synopsis:

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

Options:

The default path is the current directory, default expression is '-print'. Type `find ---help` for help or look up online man page for more detailed description. Expression may consist of:

Option	Description
-follow	Dereference symbolic links
-name <pattern>	File name (leading directories removed) matches <pattern>
-print	Print (default and assumed)
-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 18: find expressions



Examples:

Search for files in your home directory which have been modified in the last twenty-four hours.

```
find $HOME -mtime 0
```

Search for files which have read and write permission for their owner, and group, but which other users can read but not write to.

```
find . -perm 664
```

2.2.12 grep

This program searches the named input FILES (or standard input if no files are named, or the file name – is given) for lines containing a match to the given PATTERN. By default, grep prints the matching lines.

Synopsis:

```
grep [<options> ...] <pattern> [<file> ...]
```

Options:

Option	Description
-H	Print the filename for each match
-h	Suppress the prefixing of filenames on output when multiple files are searched
-n	Prefix each line of output with the line number within its input file
-l	Suppress normal output; instead print the name of each input file from which output would normally have been printed
-L	Suppress normal output; instead print the name of each input file from which no output would normally have been printed
-c	Suppress normal output; instead print a count of matching lines for each input file
-o	Show only the matching part of line
-q	Quiet; do not write anything to standard output. Exit immediately with zero status if any match is found, even if an error was detected. Also see the -s or --no-messages option.
-v	Invert the sense of matching, to select non-matching lines
-s	Suppress error messages about nonexistent or unreadable files
-r	Recurse
-R	Recurse and dereference symlinks
-i	Ignore case distinctions
-w	Match whole words only
-x	Match whole lines only
-F	Interpret PATTERN as a list of fixed strings, separated by new lines, any of which is to be matched
-E	PATTERN is an extended regexp
-m N	Match up to N times per file
-e PTRN	Use PATTERN as the pattern; useful to protect patterns beginning with –
-f FILE	Obtain patterns from FILE, one per line

Table 19: grep options



Examples:

See all lines of system log in which occurs the word "error".

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

View all processes whose name the contents of the string "ppp".

```
ps | grep ppp
```

2.2.13 gunzip

This program can be used to decompress FILE (or standard input if filename is '-').

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 20: gunzip options



Examples:

Decompression of file test.tar.gz (creates file test.tar).

```
gunzip test.tar.gz
```

2.2.14 gzip

This program can be used to compress FILE with 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 21: gzip options



Examples:

Compression of file test.tar (creates file test.tar.gz).

```
gzip test.tar
```

2.2.15 head

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

Synopsis:

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

Options:

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

Table 22: head options

2.2.16 jq



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 with its directive.

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 23: jq options

**Examples:**

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

Outputs the JSON object unmodified, with formatting.

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

Extracts the value of the key `bar` from the JSON input.

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

Applies the filter to each element in the JSON array, outputting the value of the `foo` key.

```
cat data.json | jq --arg keyName "myKey" --arg keyValue "myValue" '. +
```

```
{ $keyName: $keyValue }'
```

Adds a new key-value pair to the JSON object read from `data.json`.

2.2.17 less

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

Synopsis:

```
less [options] file ...
```

Options:

For a complete list of options, refer to the [less\(1\) Linux manual page](#).

Some commonly used options include:

- `-N`: Displays line numbers at the beginning of each line.
- `-i`: Ignores the case when searching.
- `-F`: Exits if the content can be displayed on one screen.
- `-R`: Displays ANSI color escape sequences in raw form.



Examples:

```
less file.txt
```

Displays the contents of `file.txt`, allowing navigation through the file.

```
less -N file.txt
```

Displays `file.txt` with line numbers.

```
grep 'search_term' file.txt | less -R
```

Searches for `search_term` in `file.txt` and pipes the output to `less`, allowing the user to view the search results with ANSI color formatting.

```
less +F log.txt
```

Opens `log.txt` in `less`, automatically following the end of the file as new lines are added, similar to `tail -f`.

2.2.18 ln

This program can be used to make 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	No dereference symlinks – treat like normal file
-b	Make a backup of the target (if exists) before link operation
-S	Use suffix instead of ~ when making backup files

Table 24: ln options



Examples:

Creating a symbolic link to file /var/log/messages called my.log.

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

2.2.19 ls

This program can be used to list 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 25: ls options



Examples:

View detailed content of directory /mnt.

```
ls -l /mnt
```

View list contents of actually directory.

```
ls
```

2.2.20 mkdir

This program used to make directories.

Synopsis:

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

Options:

Option	Description
-m	Set permission mode (as in chmod), not rwxrwxrwx – umask
-p	No error if existing, make parent directories as needed

Table 26: mkdir options



Examples:

Create directory /tmp/test/example.

```
mkdir -p /tmp/test/example
```

2.2.21 mv

This program can be used to move or rename 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 27: mv options



Examples:

Rename file abc.txt na def.txt.

```
mv abc.txt def.txt
```

Move all files with the extension txt to the directory /mnt.

```
mv *.txt /mnt
```

2.2.22 pwd

This program can be used to print the path of the current directory.

Synopsis:

```
pwd
```



Examples:

Print the path of the current directory, which is /home/httpd in this case.

```
pwd /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 given file name.

Synopsis:

```
realpath FILE
```



Examples:

Return the absolute pathname of `myfile.txt` located at current directory, here in `/home/MyFolder/`.

```
realpath myfile.txt
/home/MyFolder/myfile.txt
```

2.2.25 rm

This program can be used to remove 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 28: rm options



Examples:

Remove all files with extension `txt` in the current directory.

```
rm *.txt
```

Remove directory `/tmp/test` and all subdirectories.

```
rm -rf /tmp/test
```

2.2.26 rmdir

This program can be used to remove empty directories.

Synopsis:

```
rmdir <filename>
```



Examples:

Remove empty directory /tmp/test.

```
rmdir /tmp/test
```

2.2.27 sed

This program can be used for filtering and transforming 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 29: sed options



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 parameter PPP_APN in file /etc/settings.ppp to value "internet".

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


2.2.28 shred

This program can be used to delete a file completely from a non-volatile memory. This command overwrites the contents of a file multiple times, using patterns chosen to maximize the destruction of the residual data, making it harder for even very expensive hardware probing to recover it.

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 30: shred options



Examples:

Overwrite the data of file1.txt, and file2.txt using the default shredding methods.

```
shred file1.txt file2.txt
```

Overwrite the data of file1.txt using the default shredding methods and delete the file.

```
shred -u file1.txt
```

Overwrite the data of file1.txt 10 times.

```
shred -n 10 file1.txt
```

2.2.29 split

This program splits a single file (INPUT) into multiple files. A custom PREFIX for the name of the output files can be specified.

Synopsis:

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

Options:

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

Table 31: split options



Examples:

Split file *file.img* into files (*xaa*, *xab*, *xac*, ...) every with size of 50 kB:

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

Split file *file.txt* into files (*file_a*, *file_b*, *file_c*, ...) every containing of 200 lines:

```
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 FILES to the disk.

Synopsis:

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

Options:

Option	Description
-d	Avoid syncing metadata.
-f	Sync filesystems underlying FILEs.

Table 32: sync options

2.2.31 tail

This program can be used to output the last part of files.

Synopsis:

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

Options:

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

Table 33: tail options



Examples:

Show last 30 lines of /var/log/messages.

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

2.2.32 tar

This program can be used to create, extract or list files from a tar file.

Synopsis:

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

Options:

Option	Description
c	Create
x	Extract
t	List
-f FILE	Name of TARFILE or "-" for stdin
-C DIR	Change to directory DIR before operation
-v	Verbosely list files processed
-O	Extract to stdout
-o	Don't restore user:group
-k	Don't replace existing files
-z	(De)compress using gzip
-a	(De)compress based on extension
-h	Follow symlinks

Table 34: tar options



Examples:

Creating log.tar archive that contains files from the directory /var/log.

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

Extract files from the archive log.tar.

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

2.2.33 touch

This program can be used to update timestamp of file.

Synopsis:

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

Options:

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

Table 35: touch options



Examples:

Create a file, respectively update timestamp of file /tmp/test.

```
touch /tmp/test
```

2.2.34 vi

This program can be used to edit and read text file.

Synopsis:

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

Options:

Option	Description
-R	Read only, do not write to the file

Table 36: vi options



Examples:

Open file /etc/rc.local in the text editor vi.

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

2.2.35 wc

Print newline, word, and byte counts for each file, and a total line if more than one file is specified. With no file, or when 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 37: wc options

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 38: xxd options



Examples:

```
xxd file.txt
```

This command generates a hex dump of `file.txt`, displaying both the hexadecimal values and their corresponding ASCII characters. It's commonly used for inspecting the contents of a file, especially in debugging or when dealing with binary data.

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

In this scenario, the `-p` option is used to create a plain hex dump of a binary file (`file.bin`). The output is redirected to a new file (`file.hex`). This format is easier to read and manipulate, particularly useful in scenarios involving binary data analysis or modification.

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

This example shows how to reverse the process: converting a hex dump (`file.hex`) back into its original binary form (`file.bin`). The `-r` option is used for this reverse operation. It's particularly useful when you've made changes to the hex representation of a file and need to revert it to its binary format.

2.2.37 zcat

This command is a utility in Unix-like operating systems that allows users to view the contents of gzip-compressed files directly, without the need to explicitly decompress them first. It is equivalent to running the `gunzip -c` command and is useful for quickly inspecting the contents of compressed files or for piping the output of compressed files into other commands or programs for further processing.

Synopsis:

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

Description:

Zcat concatenates the uncompressed contents of compressed files to standard output. When no files are specified, or when the file name `-` is used, *zcat* reads from standard input. *Zcat* will decompress all specified files in order, making it convenient for viewing multiple compressed files.

Options:

- `-h`, `-help`: Display a help message and exit.
- `-V`, `-version`: Display version information and exit.
- Additional options available in *zcat* are generally passed to the *gzip* command, as *zcat* is often a symlink to *gzip*.



Examples:

View the contents of a compressed file:

```
zcat file.gz
```

Concatenate 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'
```

The `zcat` command's ability to directly read compressed files makes it an invaluable tool for quickly accessing or processing data within gzip-compressed files without the overhead of decompression.

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

Execution with `sudo` is required.

This program can be used to backup configuration of the router. The configuration is written to the standard output and can be redirected to a file, see example below. The stored configuration can be restored from the file by *restore* command, see Chapter 2.3.21.

Synopsis:

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

Options:

Option	Description
-c	Backup configuration excluding data of user accounts.
-u	Backup configuration data of user accounts only.
-a	Backup the whole configuration.

Table 39: backup options



Examples:

Backup the whole configuration to the file */tmp/my.cfg*.

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

2.3.2 chmod

This command can be used to change file mode bits.

Synopsis:

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

Options:

Option	Description
-R	Change files and directories recursively

Table 40: chmod options



Examples:

Settings rights (permit execution) of script */tmp/script*.

```
chmod 755 /tmp/script
```

2.3.3 chown

This command changes the user and/or group ownership of specified files or directories. This command is crucial for managing permissions and access control in Unix-like operating systems.

2.3.4 clog

This command can be used to print the connection logs.

2.3.5 date

Execution with `sudo` is required.

This command can be used to display the current time in the given FORMAT, or set 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 time described by STRING, not 'now'
-s	Set time described by STRING
-r <file>	Display the last modification time of FILE
-u	Print or set Coordinated Universal Time

Table 41: date options



Examples:

Display the current date and time.

```
date
```

Setting the date and time on December 24, 2011 20:00.

```
date 122420002011
```

2.3.6 defaults



Executing this command may cause you to lose connection to your router.

Execution with `sudo` is required.

The `defaults` command resets the router's configuration to its default state, similar to a long press of the RST button. This command has no options and proceeds without any prompt or warning.

Synopsis:

```
defaults
```

2.3.7 df

This command can be used to view report 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 filesystem type
-t TYPE	Print only mounts of this type

Table 42: df options

2.3.8 dmesg

Execution with `sudo` is required.

This command can be used to display the Kernel log messages.

Synopsis:

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

Options:

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

Table 43: dmesg options



Examples:

Display latest Kernel log messages and subsequent deletion of the Kernel ring buffer.

```
dmesg -c
```

Display latest Kernel log messages including the human-readable timestamp.

```
dmesg -T
```

2.3.9 faillock

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

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
<code>-user username</code>	Apply the command to the specified user.
<code>-reset</code>	Reset the failure records. This can be used to manually unlock accounts or reset the failure count.
<code>-legacy-output</code>	Display output in the legacy format.

Table 44: faillock options

Examples:

View the authentication failure records for all users.

```
faillock
```

View the authentication failure records for the user john.

```
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
```

View the authentication failure records using the legacy output format.

```
faillock --legacy-output
```

```
Login      Failures  Latest failure      From
john       2         2024-08-22 11:30:30  10.64.0.25
root       0
faillock: Error no such user: Test_admin
faillock: Error no such user: pam_client
faillock: Error no such user: test_admin
```

Reset the failure records for the user john.

```
faillock --user john --reset
```

2.3.10 free

This command can be used to display information about free and used memory.

Synopsis:

```
free
```

2.3.11 fwupdate

Execution with `sudo` is required.

This program can be used for router's ICR-OS firmware update.

Synopsis:

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

Options:

Option	Description
-i	File of the new ICR-OS, filename has to be specified
-h	HTML output (used when called from web configuration)
-n	Do not reboot after ICR-OS update
-f	finish update procedures, called by default

Table 45: fwupdate options

2.3.12 id

This command is utilized to display user and group information for a specified user, or for the current user if no user is specified. It provides details such as user ID (UID), group ID (GID), and supplementary groups associated with the user.

Synopsis:

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

Description:

By default, without options, the command prints the UID, GID, and supplementary groups of the specified user or the current user. The output includes both the IDs and the names of the user and groups.

Options:

Option	Description
-u	Prints the user ID of the specified user or the current user
-g	Prints the primary group ID of the specified user or the current user
-G	Prints all the supplementary group IDs of the specified user or the current user
-n	When used with -u, -g, or -G, prints the name(s) of the user or group(s) instead of the numeric ID(s)
-r	Prints the real, rather than effective, user or group ID

Table 46: 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 user:

```
id -gn
```

Print all supplementary group names of the current user:

```
id -Gn
```

For a comprehensive guide on the `id` command and its options, consult the man pages or official documentation available on your system or online.

2.3.13 kill

This command can be used to terminate process.

Synopsis:

```
kill [ -<signal> ] <process-id> [ <process-id> ... ] \[1mm] kill -l
```

Options:

Option	Description
-l	Print a list of signal names. These are found in /usr/include/linux/signal.h

Table 47: kill options



Examples:

End the process with PID 1234 by sending signal SIGTERM.

```
kill 1234
```

End the process with PID 1234 by sending signal SIGKILL.

```
kill -9 1234
```

2.3.14 killall

This command can be used to kill all process with process name.

Synopsis:

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

Options:

Option	Description
-l	Print a list of signal names. These are found in /usr/include/linux/signal.h
-q	Do not complain if no processes were killed

Table 48: killall options



Examples:

End the all processes with name pppd by sending signal SIGTERM.

```
killall pppd
```

End the all processes with name pppd by sending signal SIGKILL.

```
killall -9 pppd
```


2.3.15 klog

Execution with `sudo` is required.

This command can be used to print the kernel logs.

2.3.16 logger

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

Synopsis:

```
logger [ option ] [ 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 socket as specified with socket instead of builtin syslog routines
-d	Use a datagram instead of a stream connection to this socket

Table 49: logger options



Examples:

Send the message System rebooted to the syslogd 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!!!"
```

2.3.17 openssl

The openssl program is a command line tool for using the various cryptography functions of OpenSSL's crypto library from the shell. 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:

 For detail description this command, visit Linux manual pages.

Examples:

 Generate a new key for the SSH server.

```
openssl genrsa -out /etc/certs/ssh\_rsa\_key 512
```

Generate a new 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
```

2.3.18 passwd

This program can be used to change user passwords. A user with the *User* role can change only their own password. A user with the *Admin* 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 50: passwd options



Examples:

Change the password for the logged-in user john.

```
$ passwd
```

Changing password for john.

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

Administrator is changing the password for the john user. This operation is not permitted to a user with the *User* role.

```
# passwd john
```

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

2.3.19 pidof

This program lists the PIDs of all processes with names that match the names on the command line.

Synopsis:

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

Options:

Option	Description
-s	Display only a single PID.

Table 51: pidof options

2.3.20 ps

This program can be used to view information related with the processes on a system.

Synopsis:

```
ps [options]
```

2.3.21 restore

Execution with `sudo` is required.

This program can be used to restore configuration of the router from a file containing router's configuration. The backup of router's configuration can be generated into a file by *backup* command, see Chapter 2.3.1.

Synopsis:

```
restore <filename>
```



Examples:

Restore configuration from file */tmp/my.cfg*.

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

2.3.22 rlog

Execution with `sudo` is required.

This command can be used to print the emergency logs.

2.3.23 slog

Execution with `sudo` is required.

This command can be used to print system log (file `/var/log/message`).

Synopsis:

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

Options:

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

Table 52: slog options



Examples:

Continuous listing the system log. Listing stops when reaching the maximum number of lines of log.

```
slog -f
```

2.3.24 service

Execution with `sudo` is required.

This program can be used to start, stop or restart specified service.

Synopsis:

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



Examples:

Start service cron.

```
service cron start
```

Restart service ppp.

```
service ppp restart
```

2.3.25 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 <code>sh -c</code>
-s SH	Shell to use instead of default shell

Table 53: 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
```

2.3.26 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
-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 users cached credentials
-K	Remove the users 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 the specified user
-V	Display the version of <code>sudo</code>
-e	Edit files as the specified user

Table 54: sudo options



Examples:

Run a command as the superuser:

```
sudo command
```

List the commands available to the user:

```
sudo -l
```

Run a command as a specified user (e.g., admin):

```
sudo -u admin command
```

Edit a file as another user with elevated permissions:

```
sudo -e /path/to/file
```

Run a command in non-interactive mode, without prompting for a password:

```
sudo -n command
```


2.3.27 sysex

Execution with `sudo` is required.

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

Synopsis:

```
sysex {merge|unmerge|refresh}
```

Options:

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

Table 55: sysex options

2.3.28 times

This command is a shell builtin that provides information on the accumulated user and system times for the current shell and all of its child processes. It is an essential tool for evaluating the performance of scripts, helping users and administrators gauge the resource usage and efficiency of their commands or scripts executed within the shell.

2.3.29 top

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 <i>top</i> to other programs or to a file. In this mode, <i>top</i> 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 56: touch options

Keys:

Key	Description
n/m/p/t	Sort by pid/mem/cpu/time
r	Reverse sort
q, ^C	Exit

Table 57: touch keys



Examples:

Run *top* program in batch mode and exit after 5 iterations.

```
top -b -n5
```

Run *top* program and update the output every 10 seconds. Exit by *q* key.

```
top -d10
```

2.3.30 umask

This command is a built-in shell command used to set the default permission or file mode creation mask. The setting of the umask determines the permissions that are not set on newly created files and directories. This command is crucial for controlling the default file permissions and ensuring security by restricting the default set of permissions when new files or directories are created.

Synopsis:

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

Description:

`umask` is used without any arguments to display the current umask value in a shell session. The umask can be set by providing an octal or symbolic value representing the set of permissions to mask out (i.e., permissions that won't be set on new files and directories). For example, a umask of 022 prevents new files from being created with write permissions for group and others.

Options:

- `-S`: Display the current umask in a symbolic format, which can be more understandable than the default octal representation.
- `-p`: Display the output in a format that can be reused as input, facilitating the replication of umask settings in scripts or session initializations.



Examples:

Display the current umask value:

```
umask
```

Set a new umask value using octal notation:

```
umask 077
```

This command configures the shell so that new files and directories are created with permissions allowing only the owner to read, write, and execute them, while preventing group and others from any access.

Display the current umask in a symbolic format:

```
umask -S
```

Setting the umask is an essential part of system administration and security, as it helps ensure that files and directories are not inadvertently created with overly permissive or restrictive permissions.

This section aims to provide a clear understanding of how the `umask` command functions, its syntax, options, and usage examples to guide users in managing default permissions effectively.

2.3.31 umupdate

Execution with `sudo` is required.

This program can be used for adding or deleting of a router app from the command line.

Synopsis:

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

Options:

Option	Description
-a	Add new or update installed router app. Enter path to the installation file.
-d	Delete an installed router app with the specified name. List of installed router apps can be obtained by <code>service module list</code> command.

Table 58: umupdate options

2.3.32 whoami

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

Synopsis:

```
whoami
```



Examples:

Print the current user's name.

```
whoami
```

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

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.
-s	Manually create an ARP address mapping entry for host hostname with hardware address set to hw_addr.
-d	Remove any entry for the specified host.
-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.
-D	Use the interface ifa's hardware address.
-f	Similar to the -s option, only this time the address info is taken from file filename set up. The name of the data file is very often <i>/etc/ethers</i> , but this is not official. If no filename is specified <i>/etc/ethers</i> is used as default. The format of the file is simple; it only contains ASCII text lines with a hardware address and a hostname separated by whitespace. Additionally the pub, temp and netmask flags can be used.

Table 59: arp options

With no flags, the program displays the current ARP entry for hostname. The host may be specified by name or by number, using Internet dot notation. For detail description of this command, visit Linux manual pages.



Examples:

View arp table without translating IP addresses to domain names

```
arp -n
```

2.4.2 brctl

This command has read-only rights.

This command can be used to set up, maintain, and inspect the Ethernet bridge configuration in the Linux kernel.

An Ethernet bridge is a device commonly used to connect different networks of Ethernets together, so that these Ethernets will appear as one Ethernet to the participants.

Each of the Ethernets being connected corresponds to one physical interface in the bridge. These individual Ethernets are bundled into one bigger ('logical') Ethernet, this bigger Ethernet corresponds to the bridge network interface.

Synopsis:

```
brctl [<commands>]
```

Options:

Command	Parameters	Description
addbr	<bridge>	Add bridge
delbr	<bridge>	Delete bridge
addif	<bridge> <device>	Add interface to bridge
delif	<bridge> <device>	Delete interface from bridge
setageing	<bridge> <time>	Set ageing time
setbridgepri	<bridge> <prio>	Set bridge priority
setfd	<bridge> <time>	Set bridge forward delay
sethello	<bridge> <time>	Set hello time
setmaxage	<bridge> <time>	Set max message age
setpathcost	<bridge> <port> <cost>	Set path cost
setportprio	<bridge> <port> <prio>	Set port priority
show		Show list of bridges
showmacs	<bridge>	Show list of mac address
showstp	<bridge>	Show bridge stp info
stp	<bridge> {on off}	Turn stp on/off

Table 60: brctl commands



Examples:

Create bridge between eth0 and eth1.

```
brctl addbr br0
```

```
brctl addif br0 eth0
```

```
brctl addif br0 eth1
```

2.4.3 curl

Curl (transfer a URL) is a tool to transfer data from or to a server, using one of the supported protocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, TELNET and TFTP). It is an alternative to `wget` (see Chapter 2.4.29).

Synopsis:

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

Options:



Type `curl --help` for options to show in the command line or visit online manual page at [curl.1 the man page](#).

2.4.4 email

The program can be used for sending email.



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 61: email options



Examples:

Send system logs to the address john.doe@email.com.

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

2.4.5 ethtool

This command has read-only rights.

This command can be used to display or change Ethernet card settings.

Synopsis:

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

Options:

For detail description this command, visit Linux manual pages.



Examples:

View the status of the interface eth0.

```
ethtool eth0
```

Switch interface eth0 to mode 10 Mbit/s, half duplex.

```
ethtool -s eth0 speed 10 duplex half autoneg off
```

Turn on autonegacion on the interface eth0.

```
ethtool -s eth0 autoneg on
```


2.4.6 ifconfig

This command has read-only rights.

This command can be used to configure a network interface.

Synopsis:

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

Options:

Option	Description
broadcast <addr.>	If the address argument is given, set the protocol broadcast address for this interface.
pointtopoint <ad.>	This keyword enables the point-to-point mode of an interface, meaning that it is a direct link between two machines with nobody else listening on it.
netmask <address>	Set the IP network mask for this interface.
dstaddr <address>	Set the remote IP address for a point-to-point link (such as PPP).
metric <NN>	This parameter sets the interface metric.
mtu <NN>	This parameter sets the Maximum Transfer Unit of an interface.
trailers	This flag used to cause a non-standard encapsulation of inet packets on certain link levels.
arp	Enable or disable the use of the ARP protocol on this interface.
allmulti	Enable or disable all-multicast mode. If selected, all multicast packets on the network will be received by the interface.
multicast	Set the multicast flag on the interface. This should not normally be needed as the drivers set the flag correctly them-selves.
promisc	Enable or disable the promiscuous mode of the interface. If selected, all packets on the network will be received by the interface.
txqueuelen <NN>	Set the length of the transmit queue of the device.
up down	This flag causes the interface to be activated. This flag causes the driver for this interface to be shut down.

Table 62: ifconfig options



Examples:

View the status of all interfaces.

```
ifconfig
```

Activation of loopback with IP address 127.0.0.1/8.

```
ifconfig lo up
```

Activation of virtual interface eth0:0 with IP address 192.168.2.1/24.

```
ifconfig eth0:0 192.168.2.1 netmask 255.255.255.0 up
```

2.4.7 ip

This command has read-only rights.

This command can be used to configure a network interface or show the current configuration. Type `ip --help` for help in the terminal.



The SPECTRE v3 routers support more ip options and commands (options: `-d[etails]` , `-t[imestamp]` , `-b[atch]` , `<filename>` , `-rc[vbuf]`) ; objects: `addrlabel` , `ntable` , `tuntap` , `mrule` , `netns` , `l2tp` , `tcp_metrics` , `token`). For information how to use, type `ip <object> help` , for detailed description of all options, visit Linux manual pages or look up them online.

Synopsis:

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

Options:

Option	Description
<code>-V[ersion]</code>	Print the version of the ip utility and exit
<code>-s[tatistics]</code>	Output more information. If the option appears twice or more, the amount of information increases.
<code>-r[esolve]</code>	use the system's name resolver to print DNS names instead of host addresses
<code>-f[amily] <family></code>	Specifies the protocol family to use. The protocol family identifier can be one of inet, inet6, bridge , ipx, dnet or link.
<code>-o[neline]</code>	output each record on a single line, replacing line feeds with the '\ ' character

Table 63: ip options

Object	Description
<code>link</code>	network device
<code>addr</code>	protocol (IP or IPv6) address on a device
<code>route</code>	routing table entry
<code>rule</code>	rule in routing policy database
<code>neigh</code>	manage ARP or NDISC cache entries
<code>tunnel</code>	tunnel over IP
<code>maddr</code>	multicast address
<code>mroute</code>	multicast routing cache entry
<code>monitor</code>	watch for netlink messages
<code>xfrm</code>	manage IPsec policies

Table 64: ip objects



Examples:

View the status of all interfaces.

```
ip link show
```

View the route table.

```
ip route list
```

Add routing networks 192.168.3.0/24 through interface eth0.

```
ip route add 192.168.3.0/24 dev eth0
```

Add routing IP address 192.168.3.1 through gateway 192.168.1.2.

```
ip route add 192.168.3.1 via 192.168.1.2
```

Add default gateway 192.168.1.2.

```
ip route add default via 192.168.1.2
```

2.4.8 ipcalc

This utility is designed to calculate and display various network settings based on an IP address and optionally a netmask or prefix length. It is a valuable tool for network administrators and anyone needing to quickly derive network configuration details.

Usage:

```
ipcalc [-bnmphps] ADDRESS[/PREFIX] [NETMASK]
```

Description:

`ipcalc` takes an IP address, and optionally a slash notation prefix or a netmask, and calculates the resulting broadcast, network, netmask, and IP prefix. It simplifies network configuration and planning tasks.

Options:

- `-b`: Displays the broadcast address for the given IP network.
- `-n`: Calculates and displays the network address.
- `-m`: Shows the default netmask for the provided IP address.
- `-p`: Displays the prefix length for the given IP address/netmask.
- `-h`: Resolves and shows the hostname associated with the IP address.
- `-s`: Suppresses error messages, making the output cleaner in scripts or batch operations.



Examples:

Calculate the network settings for an IP address with a specific prefix:

```
ipcalc -bnmp 192.168.1.100/24
```

Determine the default netmask for an IP address:

```
ipcalc -m 192.168.1.100
```

For further details on using *ipcalc* refer to the BusyBox documentation or use the built-in help with `ipcalc --help`

2.4.9 iw

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 65: 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.
scan	Trigger a scan and dump the results.
station dump	Show information about all stations.
interface add	Add a new virtual interface.
phy	Show information about physical devices.

Table 66: iw commands



Examples:

```
iw dev wlan0 scan
```

This command initiates a scan for wireless networks on the device `wlan0`. It is useful for finding available wireless networks and their details like SSID, signal strength, and security protocols.

```
iw dev wlan0 link
```

Displays the status of the current wireless link on `wlan0`, including the SSID, signal quality, and other connection details. This is helpful for diagnosing connection issues or verifying the connected network.

```
iw dev wlan0 station dump
```

Provides detailed information about all stations (devices) currently connected to `wlan0`. This includes data rates, signal strength, and other relevant statistics, useful for network analysis and troubleshooting.

```
iw phy phy0 info
```

Shows detailed information about the physical device `phy0`, including supported wireless standards, frequencies, and capabilities. This is valuable for understanding the hardware's wireless capabilities.

```
iw dev wlan0 interface add wlan1 type monitor
```

Adds a new virtual interface `wlan1` of type `monitor` to the device `wlan0`. Monitor mode allows the capture of packets without being associated with a network, useful for network analysis and wireless testing.

2.4.10 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 67: nc options



Example:

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

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

2.4.11 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 68: 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 .1.3.6.1.4.1.8072.2.3.2.2 6
```

Uses SNMP version 1 to send an INFORM message with specified enterprise OID and trap parameters.

2.4.12 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 69: 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.

2.4.13 netstat

This program can be used to display the networking information.

Synopsis:

```
netstat [<options>]
```

Options:

Option	Description
-l	display listening server sockets
-a	display all sockets (default: connected)
-e	display other/more information
-n	don't resolve names
-r	display routing table
-t	tcp sockets
-u	udp sockets
-w	raw sockets
-x	unix sockets

Table 70: netstat options

2.4.14 ntpdate

Execution with `sudo` is required.

This program can be used to set the system time from 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 71: ntpdate options



Examples:

Set the system time according to the NTP server `time.windows.com`.

```
ntpdate time.windows.com
```

2.4.15 ping

This program can be used to send ICMP echo request to network host.

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 received.
-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 72: ping options



Examples:

Send one ICMP packet Echo Request with size 500 B on IP address 10.0.0.1.

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

2.4.16 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:

- **-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.



Examples:

Ping a host only a certain number of times:

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

Ping a host with a specific interval and packet size:

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

The `ping6` command is essential for network administrators and users who need to verify IPv6 connectivity or diagnose IPv6 network problems.

2.4.17 route

This command has read-only rights.

This program can be used to show and manipulate the IP routing table.

Synopsis:

```
route [ -n ] [ -e ] [ -A ] [ add | del | delete ]
```

Options:

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

Table 73: route options



For detail description this command, visit [Linux manual pages](#).



Examples:

View the routing table without translating IP addresses to domain names.

```
route -n
```

Add routing networks 192.168.3.0/24 through eth0.

```
route add -net 192.168.3.0/24 dev eth0
```

Add routing IP addresses 192.168.3.1 through 192.168.1.2 gateway.

```
route add -host 192.168.3.1 gw 192.168.1.2
```

Add default gateway 192.168.1.2

```
route add default gw 192.168.1.2
```

2.4.18 scp

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

Synopsis:

```
scp [-12346BCpqr] [-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 74: scp options

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



Examples:

Copy the file `/etc/version` from remote host `remotehost.edu` to the local host, into subdirectory `myFolder` in user's home directory.

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

Copy the file `/etc/version` from the local host to remote host `remotehost.edu`, into user's home directory.

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

Copy the directory `/home/user` from the local host to a remote host's `/tmp/bar` directory.

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

2.4.19 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:

- -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.

IPv4 options:

- -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.

IPv6 options:

- -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).

**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`.

2.4.20 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
<code>-h, -help</code>	Display the help.
<code>-H</code>	Display configuration file directives understood.
<code>-v 1 2c 3</code>	Specifies SNMP version to use.
<code>-V, -version</code>	Display package version number.
<code>-Cf</code>	If <code>-Cf</code> is not specified, some applications (<i>snmpdelta</i> , <i>snmpget</i> , <i>snmpgetnext</i> and <i>snmpstatus</i>) 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.
<i>other</i>	<i>For other options see the command help.</i>

Table 75: snmpget options

**Examples:**

Retrieve the variable `system.sysDescr.0` from the host *zeus* using the community string *public*.

```
snmpget -c public zeus system.sysDescr.0
```


2.4.21 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.
<i>other</i>	<i>For other options see the command help.</i>

Table 76: 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 77: Type options



Examples:

set the variables sysContact.0 and ipForwarding.0:

```
system.sysContact.0 = STRING: "dpz@noc.rutgers.edu"
```

```
ip.ipForwarding.0 = INTEGER: not-forwarding(2)
```

```
snmpset -c private -v 1 test-hub system.sysContact.0 s dpz@noc.rutgers.edu
```

```
ip.ipforwarding.0 = 2
```

2.4.22 snmptrap



See similar programs `snmpinform` (Chapter 2.4.11) and `snmptrap` (Chapter 2.4.12).

This program can be used to send a 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 78: snmptrap options



Examples:

Send TRAP with info about the status of a digital input BIN0 to the IP address 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 TRAP "warm start" to the IP address 192.168.1.2.

```
snmptrap -g 1 192.168.1.2
```

2.4.23 ssh

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

Synopsis:

```
ssh [-4AaCfGgKkMMNnqsTtVvXxYy] [-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 79: ssh options



Examples:

```
ssh -l username remote_host
```

This command is used for a basic login to a remote server. Replace `username` with your actual username and `remote\_host` with the address of the remote server. It initiates an SSH session where you are prompted for the password of the specified user account on the remote server.

```
ssh -p 2222 username@remote_host
```

If the SSH server is listening on a non-standard port (other than the default port 22), the `-p` option allows you to specify the port to connect to.

```
ssh -i /path/to/private_key username@remote_host
```

In this scenario, the `-i` option allows you to specify a private key to use for authentication, instead of the default key. This is particularly useful when multiple keys are used for different servers or accounts.

```
ssh username@remote_host 'command'
```

Here, you can execute a single command on the remote server without entering into a full login shell. Replace `command` with the desired command.

For example, `ssh user@server 'ls -l /var/www'` lists the contents of the specified directory on the remote server.

```
ssh -L local_port:remote_host:remote_port username@ssh_server
```

This example sets up SSH port forwarding. It forwards connections to a local port to a specified port on a remote machine. It's commonly used to securely access a service on the remote server that isn't exposed to the public internet.

2.4.24 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:

Common options include:

- `-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.



Examples:

Show all current qdiscs:

```
tc -s qdisc show
```

Limit the outbound traffic rate on an interface:

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

For a comprehensive guide on setting up Quality of Service (QoS) with `tc`, refer to our Quality of Service (QoS) Application Note available at <https://icr.advantech.com/download/application-notes#quality-service-qos>.

This section aims to introduce the `tc` command, highlighting its significance in network traffic management and Quality of Service (QoS) configuration. Through practical examples, users are guided on how to employ `tc` for optimizing network performance, with additional resources available for deeper exploration of QoS setups.

2.4.25 tcpdump

This program can be used to dump traffic on a network.

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 detail description this command, visit Linux manual pages.

Examples:

View 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
```

2.4.26 traceroute

This program can be used to track the route to a network 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 <pausesecs>]  
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 80: traceroute options

2.4.27 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:

- `-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.



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
```

`traceroute6` offers valuable insights into the network path and performance characteristics between the source and an IPv6 destination, aiding in network troubleshooting and analysis.

2.4.28 vconfig

This command has read-only rights.

This program can be used to create and remove virtual ethernet devices.

Synopsis:

```
vconfig command [OPTIONS]
```

Options:

Command [OPTIONS]	Description
add IFACE VLAN_ID	Creates a vlan-device on IFACE. The resulting vlan-device will be called according to the nameing convention set.
rem VLAN_NAME	Removes the named VLAN_NAME.
set_flag IFACE 0 1 VLAN_QOS	When 1, ethernet header reorders are turned on. Dumping the device will appear as a common ethernet device without vlans. When 0(default) however, ethernet headers are not reordered, which results in vlan tagged packets when dumping the device. Usually the default gives no problems, but some packet filtering programs might have problems with it.
set_egress_map VLAN_- NAME SKB_PRIO VLAN_QOS	This flags that outbound packets with a particular skb-priority should be tagged with the particular vlan priority vlan-qos. The default vlan priority is 0.
set_ingress_map VLAN_- NAME SKB_PRIO VLAN_QOS	RThis flags that inbound packets with the particular vlan priority vlan-qos should be queued with a particular skb-priority. The default skb-priority is 0.
set_name_type NAME_- TYPE	Sets the way vlan-device names are created. Use vconfig without arguments to see the different formats.

Table 81: vconfig commands and options



Vlan ID 4091 and 4092 are reserved for the system



Examples:

Create VLAN ID 1 on eth0 Ethernet interface.

```
vconfig add eth0 1
```

2.4.29 wget

This program can be used to retrieve files via HTTP or FTP.

Synopsis:

```
wget [-c] [-q] [-O <document file>] [--header 'header: value'] [-Y on/off] [-P <DIR>] <url>
```

Options:

Option	Description
--spider	Only check URL existence: \$? is 0 if exists
-c	Continue retrieval of aborted transfers
-q	Quiet mode – do not print
-P DIR	Save to DIR (default .)
-S	Show server response
-T SEC	Network read timeout is SEC seconds
-O FILE	Save to filename ('-' for stdout)
-o FILE	Log messages to FILE
-U STR	Use STR for User-Agent header
-Y on/off	Use proxy ('on' or 'off')

Table 82: wget options



Examples:

Download a file my.cfg from HTTP server with IP address 10.0.0.1.

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

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 proffile.

Synopsis:

```
awk [-v] [-F] [-f] ... [<program-text>] [<file> ...]
```

Options:

Option	Description
-v	Assign the value <i>val</i> to the variable <i>var</i> , before execution of the program begins. Such variable values are available to the BEGIN block of an AWK program.
-F	Use for the input field separator (the value of the FS predefined variable).
-f	Read the AWK program source from the file program-file, instead of from the first command line argument. Multiple -f (or --file) options may be used.

Table 83: `awk` options



Examples:

Show IP address of Gateway

```
route -n | awk '/^-0\.0\.0\.0/ { print $2 }'
```

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.

2.5.3 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.

2.5.4 echo

This command prints the strings to standard output.

Synopsis:

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

Options:

Option	Description
-n	Do not output the trailing newline
-e <level>	Enable interpretation of backslash escapes
-E <size>	Disable interpretation of backslash escapes (default)

Table 84: 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
```

2.5.5 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.

2.5.6 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).

2.5.7 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.

2.5.8 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. This command is integral for configuring the shell environment dynamically.

Synopsis:

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

Description:

Without arguments, `export` displays a list of all names that are exported in the shell session. When accompanied by `NAME` or `NAME='VALUE'`, the command sets the environment variable `NAME` to `VALUE`, or exports the `NAME` if `VALUE` is omitted. This enables configurations like setting the `PATH`, defining the home directory, and configuring terminal settings to be inherited by child processes.

Options: This command typically does not include options for its operation but may be used in conjunction with shell-specific flags to modify its behavior or output formatting.



Examples:

Set the `HOME` environment variable:

```
export HOME='/root'
```

Export the `PATH` environment variable, appending a new directory:

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

Display exported names:

```
export
```

These examples illustrate setting environment variables that control the shell and subprocesses' behavior, like the home directory, executable search path, and other crucial runtime configurations.

2.5.9 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 ...]
```

Description:

When called without arguments, `hash` displays the contents of the hash table, including command names and their associated full paths. Specifying one or more names as arguments adds those commands to the hash table, assuming they can be found in the directories listed in the \$PATH environment variable. This command is particularly useful in scripts and sessions where certain commands are executed repeatedly, reducing overall command lookup times.

Options:

- `-r`: Resets the hash table, clearing all remembered locations.
- `-l`: Displays output in a format that is reusable as shell input.
- `-p pathname name`: Defines a pathname for a command name, bypassing \$PATH search and hash table lookup.
- `-d name`: Removes the specified name from the hash table.
- `-t name`: Displays the remembered location of the specified command, without altering the hash table.



Examples:

Display the hash table:

```
hash
```

Add a command to the hash table:

```
hash myscript
```

Reset the hash table:

```
hash -r
```

Specify a pathname for a command:

```
hash -p /usr/local/bin/myscript myscript
```

Remove a command from the hash table:

```
hash -d myscript
```

This section aims to elucidate the purpose and usage of the `hash` command, demonstrating how it manages command locations to enhance shell efficiency. Through examples, readers can learn how to view, modify, and reset the hash table according to their needs.

2.5.10 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>
```

Description:

Accepting a numerical value as input, the `inc` command calculates and displays the value increased by one. This command simplifies operations that involve numerical incrementation, streamlining the process of generating sequences or adjusting values dynamically in scripts.

Options: The `inc` command operates directly with the numerical value provided as an argument without the need for additional options. Its simplicity ensures ease of use and integration into larger command sequences or scripts.



Examples:

Increment a given value:

```
inc 5
```

This example takes the number 5 as input and outputs 6, demonstrating the basic functionality of the `inc` command to increment a number.

Given its proprietary nature, users should refer to specific documentation or support resources for advanced usage, compatibility information, and integration guidance.

2.5.11 let

This command in Linux is a built-in shell command primarily used for evaluating arithmetic expressions. It provides a straightforward method for performing numerical calculations, supporting a wide range of operators similar to those found in traditional programming languages. This command allows for direct manipulation and evaluation of shell variables and expressions within scripts.

Synopsis:

```
let "expression"
```

Description:

Let evaluates each expression argument as an arithmetic expression. Variable names in expressions refer directly to shell variables without needing a dollar sign prefix. Arithmetic expansions performed by *let* allow assignment, various arithmetic operations, conditional expressions, and even bitwise operations, making it versatile for scripting applications where numerical computation is required.

Options: The *let* command does not have specific options. However, expressions need to be quoted to prevent the shell from interpreting special characters.



Examples:

Increment a variable:

```
count=5
let "count += 1"
echo $count # Outputs: 6
```

Perform arithmetic with variables:

```
a=10
b=20
let "c = a * b"
echo $c # Outputs: 200
```

Use conditional expressions:

```
let "d = (a > b) ? a : b"
echo $d # Outputs: 20
```

These examples illustrate the command's capability to handle arithmetic operations, providing a powerful tool for numerical calculations within shell scripts. The versatility of **let** supports complex expressions, making it a valuable resource for script authors seeking to include mathematical logic and operations in their work.

2.5.12 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.

2.5.13 nohup

`nohup` is short for No Hang-up. Nohup is a supplemental command that tells the system not to stop another command once it has started. That means itll keep running until its done, even if the user that started it logs out.

Synopsis:

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

Options:

Option	Description
program	Program to be run immune to hang-ups with output to a non-tty.
arguments	Arguments for the program.

Table 85: nohup options

2.5.14 printf

This command formats and prints ARGUMENT(s) according to FORMAT, where FORMAT controls the output exactly as in C printf. For detailed information please see documentation for C programming language.

Synopsis:

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

Format options:

Option	Description
d or i	Signed decimal integer
u	Unsigned decimal integer
o	Unsigned octal
x	Unsigned hexadecimal integer
X	Unsigned hexadecimal integer (uppercase)
f	Decimal floating point, lowercase
e	Scientific notation (mantissa/exponent), lowercase
E	Scientific notation (mantissa/exponent), uppercase
g	Use the shortest representation: %e or %f
G	Use the shortest representation: %E or %F
a	Hexadecimal floating point, lowercase
A	Hexadecimal floating point, uppercase
c	Character
s	String of characters
p	Pointer address
n	Nothing printed
%	A % followed by another % character will write a single % to the stream.

Table 86: printf format specifiers



Examples:

Print number 10 in unsigned hexadecimal integer (uppercase) format.

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

```
Output: A
```

Example of printing system variables.

```
printf "User '%s' in directory '%s'.\n" "$USER" "$PWD"
```

```
User 'root' in directory '/home/httpd'.
```

2.5.15 read

This command in shell scripting is utilized to read a line of input from standard input (stdin) or a file descriptor. It is frequently employed in scripts to capture user input or to process text files or streams line by line.

2.5.16 readonly

This command in shell scripting is used to mark variables and functions as immutable. Once a variable or a function is set to read-only, its value or function body cannot be changed or unset. Attempting to modify a read-only variable or function will result in an error.

2.5.17 return

This command is utilized within shell functions to terminate the function execution and optionally return an exit status to the calling environment. This command is similar in behavior to the *exit* command but is specifically designed for use within functions.

2.5.18 shlock

This command is a proprietary utility designed to lock a specified file during the execution of a script. This mechanism ensures that the file is not accessed or modified by other processes until the script completes its execution. If the file is already locked, *shlock* will wait until the lock is released before proceeding.

Synopsis:

```
shlock <filename>
```

Description:

Upon invocation, `shlock` attempts to create a lock on the specified file, preventing other instances or processes from modifying the file concurrently. This command is particularly useful in scripts that perform critical operations on files, requiring exclusive access to prevent data corruption or loss. If a lock cannot be immediately acquired due to an existing lock on the file, `shlock` enters a wait state until the lock becomes available.

Options: This command operates straightforwardly with the primary argument being the path to the file to be locked. There are no additional options, reflecting its focused purpose.



Examples:

Lock a file for exclusive script access:

```
shlock /path/to/myfile
```

This example demonstrates the basic usage of the `shlock` command to lock a file named `myfile`. The script that executed this command will have exclusive access to `myfile` until the script terminates, at which point the lock is released.

Given its proprietary nature, users are advised to consult specific documentation or support resources for detailed usage instructions, compatibility information, and advanced features.

2.5.19 set

This command is a shell built-in that sets or unsets shell options and positional parameters. It can modify the operational behavior of the shell, set positional parameters to the script, and enable or disable shell features.

2.5.20 shift

This command in shell scripting is used to shift the positional parameters to the left by a specified number of positions, which effectively decreases the number of positional parameters. This is particularly useful in scripts that process an arbitrary number of arguments in a loop.

2.5.21 sleep

This program can be used to delay for a specified amount of time.

Synopsis:

```
sleep <time>
```



Examples:

Sleep for 30 second.

```
sleep 30
```

2.5.22 source

This command in Unix-like operating systems is a shell built-in command used to read and execute commands from a specified file in the current shell environment. This command is commonly used to apply configuration changes, set environment variables, or define functions without starting a new shell session. The `source` command ensures that any variables or functions declared in the file are available in the current shell session.

Synopsis:

```
source filename [arguments]
```

Description:

The `source` command reads and executes commands from the given filename argument in the current shell context. If the specified file contains export statements, function definitions, or variable assignments, they will be applied to the running shell session, affecting its behavior. This command is particularly useful for scripts and configuration files that need to modify the environment of the current shell. Arguments can be passed to the sourced script, which then access them as positional parameters.

Options: The `source` command typically does not have options. Its behavior is straightforward: it executes commands from a file in the current shell.



Examples:

Source a shell script to set environment variables:

```
source /path/to/envvars.sh
```

Execute a script with arguments:

```
source /path/to/script.sh arg1 arg2
```

This documentation aims to clarify the purpose and usage of the `source` command, demonstrating how it is used to execute commands from a file in the current shell session, thereby altering the shell's environment or behavior based on the contents of the file.

2.5.23 test

This command in Linux is a fundamental tool utilized for evaluating conditional expressions. It allows scripts and users to check file types, compare values, and perform logical operations, serving as the backbone for conditional statements in shell scripting. This command facilitates decision-making processes in scripts by testing expressions and returning an exit status based on the evaluation result.

Synopsis:

```
test EXPRESSION
```

```
[ EXPRESSION ]
```

Description:

The *test* command evaluates the `EXPRESSION` provided as an argument. If the `EXPRESSION` evaluates to true, *test* returns an exit status of 0 (success). If the `EXPRESSION` evaluates to false, it returns a non-zero exit status (failure). This behavior integrates seamlessly with the *if* statements in shell scripts, enabling conditional execution of commands.

Commonly Used Tests:

- `-f FILE`: Returns true if `FILE` exists and is a regular file.
- `-d DIRECTORY`: Returns true if `DIRECTORY` exists and is a directory.
- `-z STRING`: Returns true if `STRING` is empty.
- `-n STRING`: Returns true if `STRING` is not empty.
- `STRING1 = STRING2`: Returns true if the strings are equal.
- `STRING1 != STRING2`: Returns true if the strings are not equal.
- `-eq, -ne, -lt, -le, -gt, -ge`: Numerical comparisons between integers.



Examples:

Check if a file exists:

```
test -f /path/to/file && echo "File exists." || echo "File does not exist."
```

Compare two strings:

```
test "string1" = "string2" && echo "Equal" || echo "Not equal"
```

Check if a variable is set:

```
test -n "$VARIABLE" && echo "Variable is set" || echo "Variable is not set"
```

The `test` command's versatility and simplicity make it an indispensable part of shell scripting, providing the means to perform checks and comparisons that guide the flow of execution.

2.5.24 trap

This command in Unix-like operating systems is a shell builtin that allows scripts to execute a command or a set of commands when receiving specified signals. This functionality is crucial for ensuring that scripts can handle unexpected events gracefully, perform cleanup operations, or take specific actions when interrupted.

Synopsis:

```
trap [COMMANDS] [SIGNALS]
```

Description:

`trap` enables the specification of commands (`COMMANDS`) to be executed automatically in response to various signals (`SIGNALS`). Signals are operating system messages that communicate to processes about events like termination requests, keyboard interrupts, or other conditions that require attention. By default, certain signals cause a script to terminate, but with *trap*, scripts can respond in user-defined ways, allowing for more robust and reliable behavior.

Common Signals:

- `SIGINT`: Interrupt signal, typically sent by pressing Ctrl+C.
- `SIGTERM`: Termination signal, requesting the program to end gracefully.
- `SIGHUP`: Hangup signal, often sent when a terminal window is closed.
- `SIGQUIT`: Quit signal, similar to `SIGINT`, but also generates a core dump.
- `EXIT`: Not a signal, but can be used with *trap* to specify commands to run when the script exits.



Examples:

Trap an interrupt signal and execute a cleanup function:

```
trap cleanup_function SIGINT
```

Remove temporary files upon script exit:

```
trap 'rm -f /tmp/mytempfile' EXIT
```

Reset trap to default behavior for `SIGINT`:

```
trap - SIGINT
```

The `trap` command's ability to catch and respond to signals makes it an invaluable tool in creating more interactive and user-friendly shell scripts. It enhances script reliability and control over the script's execution flow, especially in long-running processes or scripts that manage critical tasks.

This documentation segment aims to elucidate the functionality and utility of the `trap` command in shell scripting, offering a comprehensive overview of its syntax, signal handling capabilities, and practical examples to guide users in implementing effective signal management and cleanup procedures in their scripts.

2.5.25 type

This command is a shell builtin that displays the kind of command the shell will execute, given a command name as an argument. This includes identifying whether the command is a shell builtin, an alias, a function, a keyword, or an external file. The `type` command is an essential tool for debugging and scripting, as it clarifies command resolution and helps script authors understand how their commands will be interpreted by the shell.

2.5.26 unset

This command in shell scripting is used to remove variables or functions from the shell environment. By unsetting a variable or function, you effectively delete it, making its value or definition no longer accessible in the current session. This command is particularly useful in scripting to ensure that the environment is clean or to prevent accidental reuse of variables and functions with stale data.

2.5.27 wait

This command in Unix-like operating systems is a shell builtin that suspends the execution of the shell script until processes identified by their Process ID (PID) have terminated or until a specified job number has completed. It is primarily used in scripts to halt script progress until background processes or jobs have finished executing, making it essential for scripts that rely on the completion of parallel processes.

2.5.28 xargs

This program executes the command on every item given by standard input.

Synopsis:

```
xargs [<commands>] [<options>] [<args> ...]
```

Options:

Option	Description
-r	Do not run command for empty readed lines
-t	Print the command line on stderr before executing it

Table 87: xargs options



Examples:

Find files named core in or below the directory /tmp and delete them. Note that this will work incorrectly if there are any filenames containing newlines or spaces.

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


3. Related Documents

You can obtain product-related documents on the **Engineering Portal** at icr.advantech.com.

To access your router's documents or firmware, go to the [Router Models](#) page, locate the required model, and select the appropriate tab below.

Documents that are common to all models and describe specific functionality areas are available on the [Application Notes](#) page.

The **Router Apps** installation packages and manuals are available on the [Router Apps](#) page.

If you are interested in further options for extending router functionality, either through scripts or custom Router Apps, please see the information available on the [Development](#) page.

Appendix: Command Index

A

arp	60
awk	90

B

backup	40
basename	15
brctl	61
break	91

C

cat	15
cd	16
chdir	16
chmod	40
chown	41
clog	41
cmp	16
continue	91
cp	17
curl	62
cut	18

D

date	41
dd	19
decode	19
defaults	42
df	42
dirname	20
dmesg	43

E

echo	91
email	62
ethtool	63
eval	92
exec	92
exit	92

export	92
--------------	----

F

faillock	44
find	20
free	45
fwupdate	45

G

grep	21
gsminfo	4
gunzip	22
gzip	22

H

hash	93
head	23
hwclock	5

I

id	46
ifconfig	64
inc	94
io	6
ip	65
ipcalc	67
iw	68

J

jq	24
----------	----

K

kill	47
killall	47
klog	48

L

led	7
less	25
let	95
ln	26
local	96
logger	48
lpm	8
ls	27

M

mac	9
mkdir	28
mv	28

N

nc	69
net-snmpinform	70
net-snmptrap	71
netstat	72
nohup	96
ntpdate	72

O

openssl	49
---------------	----

P

passwd	50
pidof	51
ping	73
ping6	74
printf	97
ps	51
pwd	29

R

read	98
readlink	29
readonly	98
realpath	30

reboot	9
report	10
restore	51
return	98
rlog	52
rm	30
rmdir	31
route	75

S

scp	76
sed	31
service	52
set	99
shift	99
shlock	98
shred	32
sipcalc	78
sleep	99
slog	52
sms	10
snmpget	79
snmpset	80
snmptrap	81
source	100
split	33
ssh	82
status	11
stty	12
su	53
sudo	54
sync	33
sysexr	56

T

tail	34
tar	35
tc	84
tcpdump	85
test	101
times	56
top	57
touch	36
tpm2	13
traceroute	86
traceroute6	87
trap	102

type 102

U

umask 58

umupdate 59

unset 103

V

vconfig 88

vi 37

W

wait 103

wc 37

wget 89

whoami 59

X

xargs 103

xxd 38

Z

zcat 39