1 Unix utilities and shell builtins

1.1 File system

■ **cat** concatenates and prints files:
`A` shows all nonprinting characters,
`b` numbers nonempty output lines,
`n` numbers all output lines,
`s` suppresses repeated empty output lines.
■ **tac** does the same in reverse.
★ **rev** reverses lines characterwise.
■ **nl** numbers lines of files:
`s` adds „string" after line number,
`w` uses „number" columns for line numbers.

■ **chgrp** changes group ownership.
■ **chmod** changes permissions of a file:
`ugoa` permissions of the owner, group, other/all users,
`+-=` adds, removes or sets selected file mode bits,
`rwx` selects file mode bits: read/write/execute (4/2/1).
■ **chown** changes owner of a file.
★ **umask** sets file mode creation mask.
■ **touch** changes file timestamps:
`a` only the access time,
`m` only the modification time,
`t` uses custom stamp instead of current time,
`c` does not create files.

■ **shasum** prints or checks SHA message digests:
`a` algorithm: 1, 224, 256, 384, 512, 512224 or 512256,
`b` reads in binary mode,
`c` checks SHA sums read from the „files".
■ See also **cksum** (CRC checksums) and **md5sum**.
■ **wc** prints newline, word and byte counts (`lwc`):
`m` prints the character counts,
`L` prints the maximum display width.

■ **dd** converts and copies a file:
`if=` reads from a file instead of standard input,

`of=` writes to a file insteaed of standard output,
`bs=` up to „bytes" bytes at a time,
`count=` copies only „n" input blocks.

■ **cp** copies files and directories:
`b` makes a backup of each existing destination file,
`f` removes an existing destination file if needed,
`i` prompts before overwrite,
`n` does not overwrite existing files,
`L` always follows symlinks in „source",
`P` never follows symlinks in „source",
`r` copies directories recursively,
`s` makes symbolic links instead,
`l` hard links files instead,
`t` copies all „source" arguments into „directory",
`T` treats „destination" as a normal file,
`u` copies only newer source files,
`v` explains what is being done.

■ **mv** moves (renames) files:
`b` makes a backup of each existing destination file,
`i` prompts before overwriting,
`f` does not prompt before overwriting,
`n` does not overwrite existing destination files.
`t` moves all „source" arguments into „directory",
`T` treats „destination" as a normal file,
`u` moves only newer source files,
`v` explains what is being done.

■ **rm** removes files or directories:
`f` never prompts,
`i` always prompts,
`r` removes directories and their contents.

■ See also **rmdir** (directories removal) and **shred**.

■ **mkdir** makes directories (`mkdir p`: with parents as needed, no error if existing).

■ **df** reports file system disk space usage:
`h` prints size in powers of 1024,

`i` list inode information instead of block usage,
`t` limits listing to file systems of given type,
`x` limits listing to file systems not of given type,
`T` prints file systems types.

■ **du** estimates file space usage:
`a` writes counts for all files, not just directories,
`c` produces a grand total,
`d` the depth at which summing should occur,
`h` prints sizes in human readable format,
`s` diplays only a total,
`X` excludes files that match pattern.

★ **file** determines file type.

★ **fsck** checks and repairs a Linux filesystem:
`a` automatically repairs (without any question!),
`t` specifies the type(s) of filesystem to be checked,
`A` tries to check all filesystems in one run,
`M` skips mounted filesystems,
`R` skips the root filesystem.

■ **ln** makes hard links between files (not directories; only in the same file system):
`s` makes symbolic links instead.

■ **ls** lists directory contents:
`a` does not ignore entries starting with dot,
`F` appends indicator to entries,
`h` prints human readable sizes,
`i` prints the index number of each file,
`l` prints permissions, number of hard links, owner, group, size, last-modified date as well,
`r` reverses order while sorting,
`R` lists subdirectories recursively,
`S` sorts by file size (largest first),
`t` sorts by modification time (newest first),
★ **tree** lists tree-like contents of directories.

★ **mount** mounts a filesystem.

■ **pwd** prints name of current directory.

★ **tar** stores and extracts files from a disk archive:
`c` creates a new archive,
`x` extracts files,
`t` lists the contents of an archive,
`v` verbosely lists files processed,
`j` bzip2 compression,
`z` uses zip/gzip (gz compression),
`f` uses archive file or device (???),
`k` does not replace existing files when extracting.

■ **tee** duplicates pipe content:
`a` appends to the given files, does not overwrite,
`i` ignores interrupts.

★ Missing: **cmp**, **fuser**, **pax**, **type**.

1.2 Processes

■ **chroot** changes the root directory for the current running process and their children.

★ **at** schedules commands to be executed once, at a particular time in the future: it accepts times of the form `HH:MM`, `midnight`, `noon` or `teatime`; `MMDD[CC]YY`, `MM/DD/[CC]YY`, `DD.MM.[CC]YY` or `[CC]YY-MM-DD` (the specification of a date must follow the specification of the time of day). You can also give times like `now + 3 hours`.

★ **bg** resumes suspended jobs in the background.
★ **fg** resumes suspended jobs in the foreground.
★ **jobs** lists the active jobs.
★ **command &** runs command in the background.

★ **cron**: a daemon executing scheduled commands.
★ **crontab** maintain individual users' crontab files.

★ **kill** sends a `TERM` signal to a process.
★ **killall** kills processes by name.

★ **ps** reports a snapshot of the current processes:
`e` selects all processes,
`f` does full-format listing,
`C` selects processes by command name,
`p` selects processes by PID,
`u` selects processes by EUID or name.
★ **pstree** displays a tree of processes.

★ **nice** changes process priority.

★ **pgrep**, **pkill** looks up or signals processes based on name and other attributes.

★ **time** runs programs and summarizes system resource usage.

★ **top** displays linux processes.

1.3 User environment

★ **clear** clears the terminal screen.
★ **env** runs a program in a modified environment.
★ **exit** terminates the calling process.
★ **finger** looks up user information.
★ **history** displays the history list.
★ **mesg** displays messages from other users.

★ **passwd** changes user password:
`d` deletes an account's password (makes it empty),
`e` expires an account's password,
`n` sets minimum days to change password,
`w` sets warning days before password expire,
`x` sets the maximum number of days a password remains valid.

★ **su** changes user ID or becomes superuser.
★ **sudo** executes a command as another user.

★ **uname** prints system information:
`a` all information, in the following order:
`s` the kernel name,
`n` the network node hostname,
`r` the kernel release,

`v` the kernel version,
`m` the machine hardware name,
`p` the processor type,
`i` the hardware platform,
`o` the operating system.

★ **uptime**: how long has the system been running?

★ **wall** writes a message to all users,
★ **write** sends a message to another user.

★ **who** shows who is logged on,
★ **w** does the same and shows what they are doing,
★ **whoami** prints effective userid.

1.4 Text processing

★ **awk** is a pattern scanning / processing language, a pseudo-C interpretor. Sample code:

```
BEGIN {print "- Start -"}
/word/ {print NR ")" $1, $2}
END {print "- End -"}
```

Examples of conditions:

(a) `/word[0+9]+/`: regular expressions
(b) `!/word[0+9]+/`: regexes inverted
(c) ∼ and `!`∼: matches / does not match.
(d) `length($0) > 18`.

Important variables:

(a) FS: field separator (tab),
(b) OFS: output field separator,
(c) RS: record separator (new line),
(d) NR: number of the current record,
(e) NF: number of fields in the current record.

★ **grep** prints lines matching a pattern:
`c` prints a count of matching lines instead,
`e` uses a „regexp” pattern,
`f` obtains patterns from a file,

`i` ignores case disctinctions,
`v` inverts the sense of matching,
`w` selects only lines containing matches that form whole words,
`n` prints line numbers as well,
`A` prints „num” lines of trailing content,
`B` prints „num” lines of leading content,
`C` prints „num” lines of both contents,
`R` ???,
★ **sed**: a stream editor filtering/transforming text.

■ **comm** compares two sorted files line by line.
■ **shuf** generates random permutations:
`e` treats each „arg” as an input line,
`i` treats each number .. through .. as an input line,
`n` outputs at most „count” lines,
`r` output lines can be repeated (with `-n`).
■ **sort** sorts lines of text files:
`c` checks for sorted input,
`f` folds lower case to upper case characters,
`g` compares general numerical values,
`h` compares human readable numbers,
`k` sorts via a key,
`n` compares string numerical values,
`r` reverses the results,
`s` stabilizes the sort.
■ **tsort** performs topological sort.
■ **uniq** omits repeated lines:
`c` prefixes lines by the number of occurences,
`d` only prints duplicate lines, one for each group,
`f` avoids comparing first fields,
`i` ignores differences in case,
`s` avoids comparing first characters,
`w` compares no more than n characters.

■ **cut** prints selected parts of lines:
`--complement` complements the selection,
`c` selects only these characters,

`d` uses „delim” instead of Tab for field delimeter,
`f` selects only these fields,
`s` does not print lines not containing delimeters.
■ **join** joins lines of two files on a common field.
■ **paste** merges lines of files.
`d` reuses characters from „list” instead of tabs,
`s` pastes one file at a time, not in parallel.
■ **tr** translates or deletes characters:
`c` uses the complement of „set1”,
`d` deletes characters, does not translate,
`s` replaces each sequence of a repeated character that is listed in the last specified „set” with a single occurrence of that character.

★ **diff** compares files line by line:
`y` outputs in two columns,
`i` ignores case differences,
`w` ignores all white space.

★ **fmt** is a simple optimal text formatter,
★ **fold** wraps each line to fit in specified width.

■ **head** outputs the first (last) part of files:
`c` the first „num” bytes,
`n` the first „num” lines,
■ **tail** the last „num” bytes:
`c` the last „num” bytes,
`n` the last „num” lines,
`f` outputs appended data as the file grows,
`s` sleeps for „n” seconds between iterations.
■ **split** splits a file into pieces:
`a` generates suffixes of length „n” (default 2),
`b` puts „size” bytes per output file,
`d` uses numeric (not alphabetic) suffixes,
`l` puts „number” lines/records per output file,
`n` generates „chunks” output files.
■ See also: **csplit**.

★ **less** is opposite of **more**.

★ **more** is a file perusal filter for crt viewing.

★ **xargs** builds and executes command lines from standard input.

★ **yes** outputs a string repeatedly until killed.

1.5 Shell builtins

★ **alias** allows a string to be substituted for a word.
★ **cd** changes the shell working directory:
- to the previous directory.
★ **echo*** displays a line of text:
e enables interpretation of backslash escapes,
n does not output the trailing newline.
★ **test** checks file types and compares values.
★ **unset** unsets a shell variable, removing it from memory and the shell's exported environment.
★ **wait** waits for process to change state.

1.6 Networking

★ **curl** transfers a URL.
★ **dig** is a DNS lookup utility (domain information groper).
x simplified reverse lookups.
★ **host** is a DNS lookup utility.
★ **ifconfig** configures a network interface.
★ **inetd** is a super-server daemon that provides Internet services.
★ **netcat**: arbitrary TCP and UDP connections and listens.
★ **netstat** prints network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.
★ **nslookup** queries Internet name servers interactively.
★ **ping** tests the reachability of a host on an IP network by sending ICMP ECHO_REQUE
c stops after sending „count" packets,
n numeric output only, avoids to lookup symbolic names for host addresses.
★ **rdate** sets the system's date from a remote host.
★ **rlogin** starts a terminal session on a remote host.
★ **route** shows and manipulates the IP routing table.
★ **ssh** is an OpenSSH SSH client (remote login program).
D (bind address)

`p` (port)

`X` (X11 forwarding)

★ **traceroute** is a computer network diagnostic tool for displaying the route (path) and measuring transit delays of

★ **wget** is a non-interactive network downloader.

`A`, `R` specifies lists of file suffixes or patterns (when wildcard characters appear) to accept or reject,

`b` goes to background immediately after startup,

`c` continues getting a partially-downloaded file,

`m` turns on options suitable for mirroring: infinite recursion and time-stamping,

`np` does not ever ascend to the parent directory when retrieving recursively,

`U` identifies as „agent-string” to the HTTP server.

`w` waits the specified number of seconds between the retrievals (see also `-random-wait`

1.7 Searching

★ **find** searches for files in a directory hierarchy.

★ **locate** finds files by names.

★ **whatis** displays one-line manual page description.

★ **whereis** locates the binary, source, and manual page files for a command.

1.8 Miscellaneous

★ **bc** is an arbitrary precision calculator language.

1. `echo 'obase=16;255' | bc` prints FF,
2. `echo 'ibase=2;obase=A;10' | bc` prints 2,
3. `scale=10` (after `bc -l`) sets working precision.

★ **dc** is a reverse-polish desk calculator. One of the oldest Unix utilities, predating even the invention of the C programming language.

★ **cal**, **ncal** displays a calendar.

`e` displays date of Easter,

`j` displays Julian days,

`m` displays the specified month,

`w` prints the numbers of the weeks,

`y` displays a calendar for the specified year,

`3` displays the previous, current and next month.

★ **date** prints or set the system date and time.

■ **seq** prints a sequence of numbers:

`w` equalizes width by padding with leading zeroes.

■ **sleep** delays for a specified amount of time.

★ **true**, **false** does nothing, (un)successfully.

2 Emacs shortcuts in Bash

1. `Ctrl-a` moves to the start of the line,
2. `Ctrl-e` moves to the end of the line,
3. `Ctrl-u` deletes to the beginning of the line.
4. `Ctrl-k` deletes to the end of the line.
5. `Ctrl-w` deletes to the start of the word.
6. `Ctrl-y` pastes text from the clipboard.
7. `Ctrl-l` clears the screen.
8. `Alt-r` undoes all changes to the line.
9. `Ctrl-r` searches incrementally up the history.