**Unix Commands**

**Getting help in Unix**

* man – view manual pages for Unix commands

Unix Shell Commands

* clear – clear screen
* history – show history of previous commands

Time and Date commands

* date – show current date and time
* sleep – wait for a given number of seconds
* uptime – find out how long the system has been up

Unix users commands

These commands allow you to get basic information about Unix users in your environment.

* whoami – show your username
* id – print user identity
* groups – show which groups user belongs to
* passwd – change user password
* who – find out who is logged into the system
* last – show history of logins into the system

Unix file operations

Navigating filesystem and managing files and access permissions:

* ls – list files and directories
* cp – copy files (work in progress)
* rm – remove files and directories (work in progress)
* mv – rename or move files and directories to another location
* chmod – change file/directory access permissions
* chown – change file/directory ownership

Text file operations in Unix

Most of important configuration in Unix is in clear text files, these commands will let you quickly inspect files or view logs:

* cat – concatenate files and show contents to the standard output
* more – basic pagination when viewing text files or parsing Unix commands output
* less – an improved pagination tool for viewing text files (better than more command)
* head – show the first 10 lines of text file (you can specify any number of lines)
* tail – show the last 10 lines of text file (any number can be specified)
* grep – search for patterns in text files

Unix directory management commands

Navigating filesystems and managing directories:

* cd – change directory
* pwd – confirm current directory
* ln – make links and symlinks to files and directories
* mkdir – make new directory
* rmdir – remove directories in Unix

Unix system status commands

Most useful commands for reviewing hostname configuration and vital stats:

* hostname – show or set server hostname
* w – display system load, who’s logged in and what they are doing
* uname – print Unix system information

Reboot

* shutdown – graceful shutdown and reboot of your system
* halt – ungraceful (without stopping OS services) shutdown
* reboot – ungraceful reboot (without stopping OS services)

Networking commands in Unix

Most useful commands for inspecting network setup and exploring network connections and ports:

* ifconfig – show and set IP addresses (found almost everywhere)
* ip – show and set IP addresses (in recent Linux versions)
* ping – check if remote host is reachable via ICMP ping
* netstat – show network stats and routing information
* nslookup – query domain names and IP addresses
* 5 ways to use nslookup

Process management

Listing processes and confirming their status, and stopping processes if needed:

* ps – list processes
* top – show tasks and system status
* htop
* kill – kill a process (stop application running)

Remote access commands

ssh is really the only way to go, but it’s important to know telnet as well:

* telnet – clear-text (insecure) remote access protocol
* ssh – Secure SHell – encrypted remote access client
  + check out the SSH reference!

File transfers commands

Always useful to know how to copy files between servers or just download some package from the web:

* ftp – clear-text (insecure!) File Transfer Protocol client
* sftp – secure (encrypted) version of FTP
* scp – secure (encrypted) version of cp command
* wget – download files from remote servers, HTTP/HTTPS and FTP

Unix file operations

* basename – get filename from the full path (remove directory names)
* [ln](https://www.unixtutorial.org/commands/ln) – make links and symlinks to files and directories
* [find](https://www.unixtutorial.org/commands/find) – finding files and directories in Unix
* [rsync](https://www.unixtutorial.org/commands/rsync) - synchronise local and remote directories and files

Unix system status commands

* dmesg – show latest kernel buffer messages
* last – show history of user logins and reboot/shutdown commands
* w – show who is logged on and what they are doing
* who -r – confirm current run-level of your Unix/Linux OS
* uname – print Unix system information: hostname, kernel version, etc
* lsb\_release – find Linux Standard Base (LSB) information

Privileged Access

* su – switch user (commonly used to become root)
* sudo – run commands with elevated (usually root-like) privileges
  + be sure to check out sudo reference
* visudo – edit the /etc/sudoers file safely

Advanced process management in Unix

* ps -aef – show full listing of processes on your system
* ptree – show process tree in Solaris (can be done with ps in Linux)
* kill – kill a process (or send a specific signal to it)
* nice – start a process with certain priority (CPU resourcing)
* renice – update a CPU priority of an existing process
* pmap – print process map – detailed low-level memory usage
* pfiles – shows list of files/network ports open by process in Solaris

Text Manipulation commands

* awk – text processing and data extraction in text files
* grep – find and extract lines from text
* egrep – extended grep – advanced pattern matching in text files
* sed – stream editor (search/replace values in a string)
* tr – translate characters in a string using provided rules

Unix filesystems commands

* fstyp – confirm a type of filesystem on the specified device
* df – shows filesystem usage for active (mounted) filesystems
* du – shows disk usage info for files and directories

Working with disks and filesystems

* mount – attach filesystem into the file tree
* umount – detach filesystem from the file tree
* dd – copy or securely erase disks and disk partitions
* fsck – check filesystem integrity
* growfs – grow filesystem
* tune2fs – adjust tunable filesystem parameters (for ext2/ext3)
* mkfs – make new filesystem

Networking

* iptables – manage firewall rules on a Linux server
* netstat – network statistics and network routing information
* traceroute – tracing ICMP routes to a remote host

systemd Commands

Recent Linux distros rely on systemd suite of commands for managing Linux OS and its services. This section lists systemd commands:

* systemctl – management of OS services
* journalctl – show latest (error) log messages on server or service level

Linux Commands List

Modern distributions of Linux do have their share of unique commands, and they’re mostly fairly recent. Some of these commands have been so popular than they found their way into other non-Linux distributions:

* apt and apt-get – package managers in Debian, Ubuntu and Linux Mint
* dpkg – Debian package manager (also available in Ubuntu)
* lsof– list open files (and unix sockets and network connections)
* lspci
* dmidecode– get description of system’s hardware components (and their model names and serial numbers)
* biosdecode
* systemctl
* journalctl
* lsmod
* insmod
* ip – routing, networking and network devices info
* sysctl
* lsb\_release – Linux Standard Base (LSB) release information
* cryptsetup – management of LUKS encrypted filesystems

unix filesystem operation commands

unix commands for effective shell scripting

unix user management

unix system monitoring

**To write the output of a command to a file, there are basically 10 commonly used ways.**

|| visible in terminal || visible in file || existing

Syntax || StdOut | StdErr || StdOut | StdErr || file

==========++==========+==========++==========+==========++===========

> || no | yes || yes | no || overwrite

>> || no | yes || yes | no || append

|| | || | ||

2> || yes | no || no | yes || overwrite

2>> || yes | no || no | yes || append

|| | || | ||

&> || no | no || yes | yes || overwrite

&>> || no | no || yes | yes || append

|| | || | ||

| tee || yes | yes || yes | no || overwrite

| tee -a || yes | yes || yes | no || append

|| | || | ||

n.e. (\*) || yes | yes || no | yes || overwrite

n.e. (\*) || yes | yes || no | yes || append

|| | || | ||

|& tee || yes | yes || yes | yes || overwrite

|& tee -a || yes | yes || yes | yes || append

List:

* command > output.txt

The standard output stream will be redirected to the file only, it will not be visible in the terminal. If the file already exists, it gets overwritten.

* command >> output.txt

The standard output stream will be redirected to the file only, it will not be visible in the terminal. If the file already exists, the new data will get appended to the end of the file.

* command 2> output.txt

The standard error stream will be redirected to the file only, it will not be visible in the terminal. If the file already exists, it gets overwritten.

* command 2>> output.txt

The standard error stream will be redirected to the file only, it will not be visible in the terminal. If the file already exists, the new data will get appended to the end of the file.

* command &> output.txt

Both the standard output and standard error stream will be redirected to the file only, nothing will be visible in the terminal. If the file already exists, it gets overwritten.

* command &>> output.txt

Both the standard output and standard error stream will be redirected to the file only, nothing will be visible in the terminal. If the file already exists, the new data will get appended to the end of the file..

* command | tee output.txt

The standard output stream will be copied to the file, it will still be visible in the terminal. If the file already exists, it gets overwritten.

* command | tee -a output.txt

The standard output stream will be copied to the file, it will still be visible in the terminal. If the file already exists, the new data will get appended to the end of the file.

* **(\*)**

Bash has no shorthand syntax that allows piping only StdErr to a second command, which would be needed here in combination with tee again to complete the table. If you really need something like that, please look at ["How to pipe stderr, and not stdout?" on Stack Overflow](https://stackoverflow.com/q/2342826/4464570) for some ways how this can be done e.g. by swapping streams or using process substitution.

* command |& tee output.txt

Both the standard output and standard error streams will be copied to the file while still being visible in the terminal. If the file already exists, it gets overwritten.

* command |& tee -a output.txt

Both the standard output and standard error streams will be copied to the file while still being visible in the terminal. If the file already exists, the new data will get appended to the end of the file.

**Detailed** <https://www.javatpoint.com/linux-awk-command>

awk