Core utilities

This article deals with so-called *core* utilities on a GNU/Linux system, such as *less*, *ls*, and *grep*. The scope of this article includes, but is not limited to, those utilities included with the GNU `coreutils (https://www.archlinux.org/packages/?name=coreutils)` package. What follows are various tips and tricks and other helpful information related to these utilities.

Contents

Related articles

Bash

Zsh

General recommendations

GNU Project

sudo

cron

man page

Securely wipe disk#shred

File permissions and attributes

Color output in console

Basic commands

The following table lists basic shell commands every Linux user should be familiar with. See the below sections and *Related articles* for details.

Command	Description	Manual page	Example
man	Show manual page for a command	man(7) (https://jlk.fjfi.cvut.cz/arch/manpages/man/man.7)	man ed
cd	Change directory (shell built-in command)	cd(1p) (https://jlk.fjfi.cvut.cz/arch/manpages/man/cd.1p)	cd /etc/pacman.d
mkdir	Create a directory	mkdir(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/mkdir.1)	mkdir ~/newfolder
rmdir	Remove empty directory	rmdir(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/rmdir.1)	rmdir ~/emptyfolder
rm	Remove a file	rm(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/rm.1)	rm ~/file.txt
rm -r	Remove directory and contents		rm -r ~/.cache
ls	List files	ls(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ls.1)	ls *.mkv
ls -a	List hidden files		ls -a /home/archie
ls -al	List hidden files and file properties		
mv	Move a file	mv(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/mv.1)	mv ~/compressed.zip ~/archive/compressed2.zip
cp	Copy a file	cp(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/cp.1)	cp ~/.bashrc ~/.bashrc.bak
chmod +x	Make a file **executable**	chmod(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/chmod.1)	chmod +x ~/.local/bin/myscript.sh
cat	Show file contents	cat(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/cat.1)	cat /etc/hostname
strings	Show printable characters in binary files	strings(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/strings.1)	strings /usr/bin/free
find	Search for a file	find(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/find.1)	find ~ -name myfile
mount	Mount a partition	mount(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/mount.8)	mount /dev/sdc1 /media/usb
df -h	Show remaining space on all partitions	df(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/df.1)	
ps -A	Show all running processes	ps(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ps.1)	
killall	Kill all running instances of a process	killall(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/killall.1)	
ss -at	Display a list of open TCP sockets	ss(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ss.8)	

cat

cat is a standard Unix utility that concatenates and lists files.

- Because *cat* is not built into the shell, on many occasions you may find it more convenient to use a **redirection**, for example in scripts, or if you care a lot about performance. In fact `< file` does the same as `cat file`.
- *cat* is able to work with multiple lines:

```
$ cat << EOF >> path/file
first line
...
last line
EOF
```

Alternatively, using `printf`:

```
$ printf '%s\n' 'first line' ... 'last line'
```

- If you need to list file lines in reverse order, there is a utility called **tac** (*cat* reversed).

dd

dd is a utility for Unix and Unix-like operating systems whose primary purpose is to convert and copy a file.

Similarly to *cp*, by default *dd* makes a bit-to-bit copy of the file, but with lower-level I/O flow control features.

Tip: By default, *dd* outputs nothing until the task has finished. To monitor the progress of the operation, add the `status=progress` option to the command.

For more information see **dd(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/dd.1)** or the **full documentation (https://www.gnu.org/software/coreutils/dd)**.

grep

grep (from **ed**'s *g/re/p*, *global/regular expression/print*) is a command line text search utility originally written for Unix. The *grep* command searches files or standard input for lines matching a given regular expression, and prints these lines to the program's standard output.

- Remember that *grep* handles files, so a construct like `cat file | grep pattern` is replaceable with `grep pattern file`
- There are *grep* alternatives optimized for VCS source code, such as **the_silver_searcher (https://www.archlinux.org/packages/?name=the_silver_searcher)** and **ack (https://www.archlinux.org/packages/?name=ack)**.

- To include file line numbers in the output, use the `-n` option.

Note: Some commands send their output to **stderr(3) (https://jlk.fjfi.cvut.cz/arch/manpages/man/stderr.3)**, and grep has no apparent effect. In this case, redirect *stderr* to *stdout* with `command 2>&1 | grep args` or (for Bash 4) `command |& grep args`. See also **I/O Redirection (http://www.tldp.org/LDP/abs/html/io-redirection.html)**.

For color support, see **Color output in console#grep**.

find

find is part of the **findutils (https://www.archlinux.org/packages/?name=findutils)** package, which belongs to the **base (https://www.archlinux.org/groups/x86_64/base/)** package group.

Tip: **fd (https://github.com/sharkdp/fd)** is a modern and user friendly alternative to `find`, that tries to improve performance, and offer more friendly defaults. For example `fd PATTERN` instead of `find -iname '*PATTERN*'`. It features colorized output (similar to `ls`), smart-case search by default, hidden files ignoring, and more. **fd (https://www.archlinux.org/packages/?name=fd)**

One would probably expect a *find* command to take as argument a file name and search the filesystem for files matching that name. For a program that does exactly that see **#locate** below.

Instead, find takes a set of directories and matches each file under them against a set of expressions. This design allows for some very powerful "one-liners" that would not be possible using the "intuitive" design described above. See **UsingFind (http://mywiki.wooledge.org/UsingFind)** for usage details.

iconv

iconv converts the encoding of characters from one codeset to another.

The following command will convert the file `foo` from ISO-8859-15 to UTF-8, saving it to `foo.utf`:

```
$ iconv -f ISO-8859-15 -t UTF-8 foo > foo.utf
```

See **iconv(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/iconv.1)** for more details.

Convert a file in place

Tip: You can use **recode (https://www.archlinux.org/packages/?name=recode)** instead of iconv if you do not want to touch the mtime.

Unlike **sed**, *iconv* does not provide an option to convert a file in place. However, `sponge` from the **moreutils (https://www.archlinux.org/packages/?name=moreutils)** package can help:

```
$ iconv -f WINDOWS-1251 -t UTF-8 foobar.txt | sponge foobar.txt
```

See **sponge(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/sponge.1)** for details.

ip

ip allows you to show information about network devices, IP addresses, routing tables, and other objects in the Linux **IP** software stack. By appending various commands, you can also manipulate or configure most of these objects.

Note: The *ip* utility is provided by the **iproute2 (https://www.archlinux.org/packages/?name=iproute2)** package, which is included in the **base (https://www.archlinux.org/groups/x86_64/base/)** group.

Object	Purpose	Manual page
ip addr	protocol address management	ip-address(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-address.8)
ip addrlabel	protocol address label management	ip-addrlabel(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-addrlabel.8)
ip l2tp	tunnel Ethernet over IP (L2TPv3)	ip-l2tp(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-l2tp.8)
ip link	network device configuration	ip-link(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-link.8)
ip maddr	multicast addresses management	ip-maddress(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-maddress.8)
ip monitor	watch for netlink messages	ip-monitor(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-monitor.8)
ip mroute	multicast routing cache management	ip-mroute(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-mroute.8)
ip mrule	rule in multicast routing policy db	
ip neigh	neighbour/ARP tables management	ip-neighbour(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-neighbour.8)
ip netns	process network namespace management	ip-netns(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-netns.8)
ip ntable	neighbour table configuration	ip-ntable(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-ntable.8)
ip route	routing table management	ip-route(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-route.8)
ip rule	routing policy database management	ip-rule(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-rule.8)
ip tcp_metrics	management for TCP Metrics	ip-tcp_metrics(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-tcp_metrics.8)
ip tunnel	tunnel configuration	ip-tunnel(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-tunnel.8)
ip tuntap	manage TUN/TAP devices	
ip xfrm	manage IPsec policies	ip-xfrm(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ip-xfrm.8)

The `help` command is available for all objects. For example, typing `ip addr help` will show you the command syntax available for the address object. For advanced usage see the **iproute2 documentation (http://www.policyrouting.org/iproute2.doc.html)**.

The **Network configuration** article shows how the *ip* command is used in practice for various common tasks.

Note: You might be familiar with the **ifconfig** command, which was used in older versions of Linux for interface configuration. It is now deprecated in Arch Linux; you should use *ip* instead.

locate

Install the **mlocate (https://www.archlinux.org/packages/?name=mlocate)** package. The package contains an `updatedb.timer` unit, which invokes a database update each day. The timer is enabled right after installation, **start** it manually if you want to use it before reboot. You can also manually run *updatedb* as root at any time. By default, paths such as `/media` and `/mnt` are ignored, so *locate* may not discover files on external devices. See **updatedb(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/updatedb.8)** for details.

The *locate* command is a common Unix tool for quickly finding files by name. It offers speed improvements over the **find** tool by searching a pre-constructed database file, rather than the filesystem directly. The downside of this approach is that changes made since the construction of the database file cannot be detected by *locate*.

Before *locate* can be used, the database will need to be created. To do this, execute `updatedb` as root.

See also **How locate works and rewrite it in one minute (http://jvns.ca/blog/2015/03/05/how-the-locate-command-works-and-lets-rewrite-it-in-one-minute/)**.

less

less is a terminal pager program used to view the contents of a text file one screen at a time. Whilst similar to other pagers such as **more** and **pg**, *less* offers a more advanced interface and complete **feature-set (http://www.greenwoodsoftware.com/less/faq.html)**.

See **List of applications#Terminal pagers** for alternatives.

Vim as alternative pager

Vim includes a script to view the content of text files, compressed files, binaries and directories. Add the following line to your shell configuration file to use it as a pager:

~/.bashrc

```
alias less='/usr/share/vim/vim80/macros/less.sh'
```

There is also an alternative to the *less.sh* macro, which may work as the `PAGER` environment variable. Install **`vimpager (https://www.archlinux.org/packages/?name=vimpager)`** and add the following to your shell configuration file:

~/.bashrc

```
export PAGER='vimpager'
alias less=$PAGER
```

Now programs that use the `PAGER` environment variable, like **git**, will use *vim* as pager.

ls

ls lists directory contents.

See `info ls` or **the online manual (https://www.gnu.org/software/coreutils/manual/html_node/ls-invocation.html#ls-invocation)** for more information.

exa (https://the.exa.website) is a modern, and more user friendly alternative to `ls` and `tree`, that has more features, such as displaying **Git** modifications along with filenames, colouring differently each columnn in `--long` mode, or displaying `--long` mode metadata along with a `tree` view. **`exa (https://www.archlinux.org/packages/?name=exa)`**

Long format

The `-l` option displays some metadata, for example:

```
$ ls -l /path/to/directory
```

```
total 128
drwxr-xr-x 2 archie users  4096 Jul  5 21:03 Desktop
drwxr-xr-x 6 archie users  4096 Jul  5 17:37 Documents
drwxr-xr-x 2 archie users  4096 Jul  5 13:45 Downloads
-rw-rw-r-- 1 archie users  5120 Jun 27 08:28 customers.ods
-rw-r--r-- 1 archie users  3339 Jun 27 08:28 todo
-rwxr-xr-x 1 archie users  2048 Jul  6 12:56 myscript.sh
```

The `total` value represents the total disk allocation for the files in the directory, by default in number of blocks.

Below, each file and subdirectory is represented by a line divided into 7 metadata fields, in the following order:

- type and permissions:
  - the first character is the entry type, see `info ls -n "What information is listed"` for an explanation of all the possible types; for example:
    - `-` denotes a normal file;
    - `d` denotes a directory, i.e. a folder containing other files or folders;
    - `p` denotes a named pipe (aka FIFO);
    - `l` denotes a symbolic link;
  - the remaining characters are the entry's **permissions**;
- number of **hard links** for the entity; files will have at least 1, i.e. the showed reference itself; folders will have at least 2: the showed reference, the self-referencing `.` entry, and then a `..` entry in each of its subfolders;
- owner **user** name;
- **group** name;
- size;
- last modification timestamp;
- entity name.

File names containing spaces enclosed in quotes

By default, file and directory names that contain spaces are displayed surrounded by single quotes. To change this behavior use the `-N` or `--quoting-style=literal` options. Alternatively, set the `QUOTING_STYLE` **environment variable** to `literal`. **[1] (https://unix.stackexchange.com/questions/258679/why-is-ls-suddenly-surrounding-items-with-spaces-in-single-quotes)**

lsblk

`lsblk(8)` (https://jlk.fjfi.cvut.cz/arch/manpages/man/lsblk.8) will show all available **block devices** along with their partitioning schemes, for example:

```
$ lsblk -f
```

```
NAME   FSTYPE   LABEL       UUID                                 MOUNTPOINT
sda
├─sda1 vfat                C4DA-2C4D                            /boot
├─sda2 swap                5b1564b2-2e2c-452c-bcfa-d1f572ae99f2 [SWAP]
└─sda3 ext4                56adc99b-a61e-46af-aab7-a6d07e504652 /
```

The beginning of the device name specifies the type of block device. Most modern storage devices (e.g. hard disks, **SSDs** and USB flash drives) are recognised as SCSI disks (`sd`). The type is followed by a lower-case letter starting from `a` for the first device (`sda`), `b` for the second device (`sdb`), and so on. *Existing* partitions on each device will be listed with a

number starting from `1` for the first partition (`sda1`), `2` for the second (`sda2`), and so on. In the example above, only one device is available (`sda`), and that device has three partitions (`sda1` to `sda3`), each with a different **file system**.

Other common block device types include for example `mmcblk` for memory cards and `nvme` for **NVMe** devices. Unknown types can be searched in the **kernel documentation (https://www.kernel.org/doc/Documentation/devices.txt)**[**dead link** 2017-11-11].

mkdir

mkdir makes directories.

To create a directory and its whole hierarchy, the `-p` switch is used, otherwise an error is printed. As users are supposed to know what they want, `-p` switch may be used as a default:

```
alias mkdir='mkdir -p -v'
```

The `-v` switch make it verbose.

Changing mode of a just created directory using *chmod* is not necessary as the `-m` option lets you define the access permissions.

Tip: If you just want a temporary directory, a better alternative may be **mktemp**: `mktemp -p` .

mv

mv moves and renames files and directories.

To limit potential damage caused by the command, use an alias:

```
alias mv='timeout 8 mv -iv'
```

This alias suspends *mv* after eight seconds, asks for confirmation before overwriting any existing files, lists the operations in progress and does not store itself in the shell history file if the shell is configured to ignore space starting commands.

od

The **od** (*o*ctal *d*ump) command is useful for visualizing data that is not in a human-readable format, like the executable code of a program, or the contents of an unformatted device. See the **manual (https://www.gnu.org/software/coreutils/manual/html_node/od-invocation.html#od-invocation)** for more information.

pv

You can use **pv (https://www.archlinux.org/packages/?name=pv)** (*pipe viewer*) to monitor the progress of data through a pipeline, for example:

```
# dd if=/source/filestream | pv -monitor_options -s size_of_file | dd of=/destination/filestream
```

In most cases `pv` functions as a drop-in replacement for `cat`.

rm

rm removes files or directories.

To limit potential damage caused by the command, use an alias:

```
alias rm='timeout 3 rm -Iv --one-file-system'
```

This alias suspends *rm* after three seconds, asks confirmation to delete three or more files, lists the operations in progress, does not involve more than one file systems and does not store itself in the shell history file if the shell is configured to ignore space starting commands. Substitute `-I` with `-i` if you prefer to confirm even for one file.

Zsh users may want to put `noglob` before `timeout` to avoid implicit expansions.

To remove directories believed to be empty, use *rmdir* as it fails if there are files inside the target.

sed

sed is stream editor for filtering and transforming text.

Here is a handy **list (http://sed.sourceforge.net/sed1line.txt)** of *sed* one-liners examples.

Tip: More powerful alternatives are **AWK** and the **Perl** language.

seq

seq prints a sequence of numbers. Shell built-in alternatives are available, so it is good practice to use them as explained on **Wikipedia**.

ss

ss is a utility to investigate network ports and is part of the **`iproute2 (https://www.archlinux.org/packages/?name=iproute2)`** package in the **`base (https://www.archlinux.org/groups/x86_64/base/)`** group. It has a similar functionality to the **deprecated (https://www.archlinux.org/news/deprecation-of-net-tools/)** netstat utility.

Common usage includes:

Display all TCP Sockets with service names:

```
$ ss -at
```

Display all TCP Sockets with port numbers:

```
$ ss -atn
```

Display all UDP Sockets:

```
$ ss -au
```

For more information see **ss(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/ss.8)** or `ss.html` from the **iproute2 (https://www.archlinux.org/packages/?name=iproute2)** package.

tar

As an early Unix archiving format, .tar files—known as "tarballs"—are widely used for packaging in Unix-like operating systems. Both **pacman** and **AUR** packages are compressed tarballs, and Arch uses **GNU's** *tar* program by default.

For .tar archives, *tar* by default will extract the file according to its extension:

```
$ tar xvf file.EXTENSION
```

Forcing a given format:

File Type	Extraction Command
`file.tar`	`tar xvf file.tar`
`file.tgz`	`tar xvzf file.tgz`
`file.tar.gz`	`tar xvzf file.tar.gz`
`file.tar.bz`	`bzip -cd file.bz | tar xvf -`
`file.tar.bz2`	`tar xvjf file.tar.bz2` `bzip2 -cd file.bz2 | tar xvf -`
`file.tar.xz`	`tar xvJf file.tar.xz` `xz -cd file.xz | tar xvf -`

The construction of some of these *tar* arguments may be considered legacy, but they are still useful when performing specific operations. See **tar(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/tar.1)** for details.

which

which shows the full path of shell commands. In the following example the full path of `ssh` is used as an argument for `journalctl`:

```
# journalctl $(which sshd)
```

wipefs

wipefs can list or erase **file system**, **RAID** or **partition-table** signatures (magic strings) from the specified device. It does not erase the file systems themselves nor any other data from the device.

See **`wipefs(8) (https://jlk.fjfi.cvut.cz/arch/manpages/man/wipefs.8)`** for more information.

For example, to erase all signatures from the device `/dev/sdb` and create a signature backup `~/wipefs-sdb-offset.bak` file for each signature:

```
# wipefs --all --backup /dev/sdb
```

See also

- **A sampling of coreutils (http://www.reddit.com/r/commandline/comments/19garq/a_sampling_of_coreutils_120/) , part 2 (http://www.reddit.com/r/commandline/comments/19ge6v/a_sampling_of_coreutils_2040/) , part 3 (http://www.reddit.com/r/commandline/comments/19j1w3/a_sampling_of_coreutils_4060/)** - Overview of commands in coreutils
- **GNU Coreutils online documentation (https://www.gnu.org/software/coreutils/manual/coreutils.html)**

- **Learn the DD command (https://www.linuxquestions.org/questions/linux-newbie-8/learn-the-dd-command-362506/)**

Retrieved from "https://wiki.archlinux.org/index.php?title=Core_utilities&oldid=510582#lsblk"
