Finding Things (Core)

Linux Commands Course · Section 4

IDSchool

Finding Commands — which, whereis, type

which

Shows the full path of a command as found in $PATH.

```
which ls
```

Output example:

```
/bin/ls
```

If nothing prints, the command isn’t in your PATH.

---

whereis

Locates executables, source code, and man pages for a command.

```
whereis ls
```

Example output:

```
ls: /bin/ls /usr/share/man/man1/ls.1.gz
```

Useful when you want both program and documentation locations.

---

type

Displays how a command name will be interpreted by the shell.

```
type echo
```

Possible results:

- **builtin** – internal to the shell
- **alias** – shortcut defined by the user
- **file** – external program in the PATH

`type` is the most complete tool for identifying command nature.

Searching Files — find

find scans directories recursively and matches patterns or conditions.

Basic syntax:

```
find [path] [tests] [actions]
```

Example — find files by name:

```
find . -name "notes.txt"
```

The dot (.) means “start from current directory”.

Search by Type, Size, and Time

By file type:

```
find /etc -type d
```

→ shows only directories.

By size:

```
find /var/log -size +10M
```

→ finds files larger than 10 MB.

By modification time (in days):

```
find /home -mtime -2
```

→ modified in the last 2 days.

Combining Conditions

You can combine filters with logical operators.

Example — find .log files modified recently:

```
find /var/log -type f -name "*.log" -mtime -1
```

You can also negate tests:

```
find /etc -type f ! -name "*.conf"
```

→ every file that is *not* a .conf file.

Running Actions — -exec

Execute a command on each found file.

Example — list detailed info:

```
find . -type f -name "*.sh" -exec ls -lh {} \;
```

Each `{}` represents the current file; `\;` ends the `-exec` clause.

Or remove safely (after verifying!):

```
find ~/Downloads -type f -name "*.tmp" -exec rm -i {} \;
```

Avoid Unwanted Paths — -prune

Exclude directories from search with -prune.

Example — skip .git folders:

```
find . -path "./.git" -prune -o -type f -name "*.py" -print
```

How it works:

- -prune skips matched directories.
- The -o means “OR” — only the right side runs when left fails.

Using find with xargs

xargs efficiently passes found files to another command.

Example — count lines in all .c files:

```
find . -name "*.c" | xargs wc -l
```

Faster than repeated -exec calls.

For safety with spaces in filenames, use -print0 + xargs -0:

```
find . -name "*.txt" -print0 | xargs -0 rm -i
```

Locate — database-based search

locate searches a prebuilt database of filenames — much faster than find.

```
locate passwd
```

The database is usually updated daily.

If results seem outdated, refresh manually:

```
sudo updatedb
```

locate searches **by name only**, not by content or modification time.

Comparing find vs locate

Feature	find	locate
Searches live filesystem	✅	❌ (uses index)
Needs database update	❌	✅
Can filter by time/size/type	✅	❌
Speed	Slower	Instant
Accuracy	Always current	May be outdated

Use locate for quick lookups, and find for precise, real-time results.

Recap

- **Command locations:** which, whereis, type
- **File system search:** find (name, size, time, exec, prune)
- **Indexed search:** locate, updatedb
- Combine with xargs for high performance.

These are your search toolkit for any Linux environment.
