

# GNU Parallel Cheat Sheet

GNU Parallel is a replacement for *xargs* and *for* loops. It can also split a file or a stream into blocks and pass those to commands running in parallel.

## Examples

**Compress all \*.html files in parallel – 2 jobs per CPU thread in parallel**

```
parallel --jobs 200% gzip ::: *.html
```

**Convert all \*.wav to \*.mp3 using lame – 1 job per CPU thread in parallel (default)**

```
parallel lame {} -o {}.mp3 ::: *.wav
```

**Chop bigfile into 1MB blocks and grep for the string foobar**

```
cat bigfile | parallel --pipe grep foobar
```

## Input sources

```
parallel echo ::: cmd line input source
```

```
cat input_from_stdin | parallel echo
```

```
parallel echo ::: multiple input sources ::: with values
```

```
parallel -a input_from_file echo
```

```
parallel echo :::: input_from_file
```

```
parallel echo :::: input_from_file ::: and command line
```

| Replacement string    | Value if input is mydir/mysubdir/myfile.myext   |
|-----------------------|-------------------------------------------------|
| {}                    | mydir/mysubdir/myfile.myext                     |
| {.}                   | mydir/mysubdir/myfile                           |
| {/}, {//}, {/.}       | myfile.myext, mydir/mysubdir, myfile            |
| {#}                   | <i>The sequence number of the job</i>           |
| {%}                   | <i>The job slot number</i>                      |
| {2}                   | <i>Value from the second input source</i>       |
| {2.} {2/} {2//} {2/.} | <i>Combination of {2} and {.} {/} {//} {/.}</i> |
| {= perl expression =} | <i>Change \$_ with perl expression</i>          |

## Control the output – keep the same order as the input, prepend with input value

```
parallel --keep-order --tag "sleep {}; echo {}" ::: 5 4 3 2 1
```

## Control the execution

**Run 2 jobs in parallel – command is a composed command**

```
parallel --jobs 2 "sleep {}; echo {}" ::: 5 4 3 2 1
```

**See what will be run**

```
parallel --dryrun echo {2} {1} ::: bird flower fish ::: Red Green Blue
```

## Remote execution

```
parallel -S server1 -S server2 "hostname; echo {}" ::: foo bar
```

## Pipe mode

```
cat bigfile | parallel --pipe wc -l
```

**Chop bigfile into one block per CPU thread and grep for foobar**

```
parallel -a bigfile --pipepart --block -1 grep foobar
```

## Read more – Your command line will love you for it

```
parallel --help; man parallel; man parallel_tutorial; www.pi.dk/1
```

GNU Parallel 2018 <https://doi.org/10.5281/zenodo.1146014>