## 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 (default) and grep for the string foobar
  ```
  cat bigfile | parallel --pipe grep foobar
  ```

### Input sources

- ```echo ::: cmd line input source```
- `cat input_from_stdin | parallel echo`
- ```echo ::: multiple input sources ::: with values```
- `parallel -a input_from_file echo`
- ```echo :::: input_from_file```
- ```echo :::: input_from_file ::: and command line```

### Replacement string

<table>
<thead>
<tr>
<th>Value if input is mydir/mysubdir/myfile.myext</th>
<th>Replacement string</th>
</tr>
</thead>
<tbody>
<tr>
<td>mydir/mysubdir/myfile.myext</td>
<td><code>{}</code></td>
</tr>
<tr>
<td>mydir/mysubdir/myfile</td>
<td><code>{.}</code></td>
</tr>
<tr>
<td>myfile.myext, mydir/mysubdir, myfile</td>
<td><code>{/}, {///}, {./}</code></td>
</tr>
<tr>
<td>The sequence number of the job</td>
<td><code>#{}</code></td>
</tr>
<tr>
<td>The job slot number</td>
<td><code>{#}</code></td>
</tr>
<tr>
<td>Value from the second input source</td>
<td><code>{2}</code></td>
</tr>
<tr>
<td>Combination of <code>{2}</code> and <code>{.} {/} {///} {./}</code></td>
<td><code>{{2}} {2/} {2///} {2/.}</code></td>
</tr>
<tr>
<td>Change $ with perl expression</td>
<td><code>{= perl expression =}</code></td>
</tr>
</tbody>
</table>

### 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 - run "hostname; echo foo/bar" on server1 and server2

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

### Pipe mode

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

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

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


(CC-By-SA) 2022-11-03 Ole Tange