GNU Parallel Parallel shell jobs

START HERE

`parallel <cmd> {} ::: <a> <b>`
Run per argument

`parallel --joblog <file> ...`
Log every job

`ls | parallel <cmd> {}`
Args from stdin

`parallel {.} {/} {#}`
Name placeholders

`parallel --dry-run <cmd> {} ::: <a>`
Preview commands

`parallel -j 4 ...`
Limit to 4 jobs

`parallel -k ...`
Keep input order

`parallel --bar ...`
Show progress bar

Input ::: / :::: / stdin → **split** → **Arguments** one per job → **fill {}** → **Job slots** -j N at a time → **group** → **Output** per job, in order with -k

CONCEPTS

Think xargs with sane defaults: one job per argument, one job per CPU thread, and output of each job never interleaved.

Several ::: groups give every combination; use :::+ to pair them up instead.

INPUTS

`parallel <cmd> ::: <a> <b> <c>`
Arguments on the command line

`<producer> | parallel <cmd>`
One argument per stdin line

`parallel <cmd> :::: <file>`
One argument per line of a file

`parallel -0 <cmd>`
NUL-separated stdin (find -print0)

`parallel -d <delim> <cmd>`
Custom input delimiter

INPUTS (cont.)

`parallel <cmd> ::: <a> <b> ::: <x> <y>`
All combinations of two sources

`parallel <cmd> ::: <a> <b> :::+ <x> <y>`
Pair sources (a-x, b-y)

REPLACEMENT STRINGS

`{}`
The argument itself

`{.}`
Argument without extension

`{/}`
Basename (no directory)

`{/.}`
Basename without extension

`{#}`
Job sequence number

`{%}`
Job slot number, 1 to -j

`{1} {2} {-1}`
Argument from source n; negative counts from end

REPLACEMENT STRINGS (cont.)

With no replacement string in the command, {} is appended. Replacement strings are shell-quoted for you.

RUNNING JOBS

`parallel -j <N> ...`
Run N jobs at once

`parallel -n <N> ...`
Up to N arguments per command

`parallel -X ...`
Pack many args into each command line

SAFETY AND PREVIEW

`parallel --dry-run <cmd> ::: <a> <b>`
Print commands, run nothing

`parallel --halt now,fail=1 ...`
Stop all jobs at first failure

SAFETY AND PREVIEW (cont.)

`parallel -q <cmd> '<arg with $>'`
Quote the command (simple commands only)

Pipes, redirects and && inside the command need single quotes around the whole command: parallel 'a {} | b > {.}.out' ::: ...

Shell functions need export -f <fn> in bash; aliases and zsh/fish functions need env_parallel.

OUTPUT AND PROGRESS

`parallel -k ...`
Print output in input order

`parallel --bar ...`
Progress bar on stderr

`parallel --tag ...`
Prefix each line with its argument

JOBLOG AND RESUME

`parallel --joblog <file> ...`
Record seq, time, exit code, command

`parallel --joblog <file> --resume ...`
Skip jobs already logged

`parallel --joblog <file> --resume-failed ...`
Rerun failed jobs, then the rest

SPLITTING STDIN

`<producer> | parallel --pipe <cmd>`
Give each job a block of stdin

`parallel --pipe -N <N> <cmd>`
Exactly N records per job

REMOTE HOSTS

`parallel -S <host1>,<host2> ...`
Run jobs on these ssh hosts

REMOTE HOSTS (cont.)

`parallel --transferfile {} -S <host> ...`
Copy the input file to the host first

RECIPES

Convert files, keep names

```
ls *.png | parallel convert {} {.}.jpg
# or safe for odd names:
find . -name '*.png' -print0 | parallel -0 convert {} {.}.jpg
```

Preview, run, log, resume

```
parallel --dry-run gzip {} ::: *.log
parallel -j 4 --bar --joblog jobs.log gzip {} ::: *.log
# interrupted? pick up where it stopped:
parallel -j 4 --joblog jobs.log --resume gzip {} ::: *.log
```

Parameter sweep

```
parallel --tag -k ./run.sh --lr {1} --bs {2} \
  ::: 0.1 0.01 ::: 32 64
```

Big file, many cores

```
cat big.txt | parallel --pipe --block 20M -k \
  grep -c '<pattern>' | paste -sd+ | bc
```