GNU Parallel

Parallel shell jobs

CONCEPTS

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

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

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

REPLACEMENT STRINGS

`{}`
The argument itself

`{.}`
Argument without extension

`{/}`
Basename (no directory)

`{/.}`
Basename without extension

`{#}`
Job sequence number

RUNNING JOBS

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

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

SAFETY AND PREVIEW

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

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

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