GNU Parallel for Bioinformatics.

Pierre Lindenbaum
@yokofakun
`http://plindenbaum.blogspot.com`

October 15, 2013

Abstract

This document follows Ole Tange's **parallel** tutorial `http://www.gnu.org/software/parallel/parallel_tutorial.html`.
The sources of this document are available at `https://github.com/lindenb/courses/tree/master/about.parallel`

Contents

1 Input Source

1.1 A single input source

1.1.1 Input can be read from the command line.

Example: determine the file type of a list of **bam** .

```
$ parallel file ::: samtools-0.1.18/examples/*.bam
```

output:

```
samtools-0.1.18/examples/ex1a.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1b.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1f.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1f-rmduppe.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1f-rmdupse.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1_sorted.bam: gzip compressed data, extra field
samtools-0.1.18/examples/toy.bam: gzip compressed data, extra field
```

1.1.2 The input source can be a file

Example: determine the file type of a list of **bam** .

```
$ find samtools-0.1.18/examples/ -name "*.bam" -type f > listbams.txt
$ parallel -a listbams.txt file
```

output:

```
samtools-0.1.18/examples/ex1a.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1b.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1f.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1f-rmduppe.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1f-rmdupse.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1_sorted.bam: gzip compressed data, extra field
samtools-0.1.18/examples/toy.bam: gzip compressed data, extra field
```

1.1.3 STDIN (standard input) can be the input source

Example: determine the file type of a list of **bam** .

```
$ find samtools-0.1.18/examples/ -name "*.bam" -type f | parallel file
```

output:

```
samtools-0.1.18/examples/ex1a.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1b.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1f.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1f-rmduppe.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1f-rmdupse.bam: gzip compressed data, extra field
samtools-0.1.18/examples/ex1_sorted.bam: gzip compressed data, extra field
samtools-0.1.18/examples/toy.bam: gzip compressed data, extra field
```

Example: indexing sorted sorted **bam** files with **samtools** . :

```
$ find dir1 -name "*.bam" | grep sorted |\
        parallel -a - 'samtools index '
```

or , without '-a -'

```
$ find dir1 -name "*.bam" | grep sorted |\
        parallel 'samtools index '
```

Ole Tange: "The `'-a -'` construct is unnatural to me. It makes sense when you have multiple `'-a'` but if it is the only one, leave it out. That will also make it easier for people who are used to xargs."

1.2 Multiple input source

```
\example{Print the combinations of two lists of nucleotides }
$  parallel echo ::: A T G C ::: a t g c
```

output:

```
A a
A t
A g
A c
T a
T t
T g
T c
G a
G t
G g
G c
C a
C t
C g
C c
```

1.2.1 If one of the input sources is too short, its values will wrap

Example: Print the combinations of two lists of nucleotides. The second list is shorter .

```
$  parallel echo ::: A T G C N ::: a t
```

output:

```
A a
A t
T a
T t
G a
G t
C a
C t
N a
N t
```

1.2.2 The input sources can be files

Example: Print the combinations of two files of nucleotides. .

```
$ echo -e "A\nT\nG\nC" > ATGC.txt
$ echo -e "a\nt\ng\nc" > atgc.txt
$  parallel  -a ATGC.txt -a  atgc.txt echo
```

output:

```
A a
A t
A g
A c
T a
T t
T g
T c
G a
G t
G g
G c
C a
C t
C g
C c
```

1.2.3 STDIN can be one of the input sources using '-'

Example: Print the combinations of one file of nucleotides and stdin .

```
$ echo -e "a\nt\ng\nc" > atgc.txt
$ echo -e "A\nT\nG\nC" |\
       parallel  -a - -a  atgc.txt echo
```

output:

```
A a
A t
A g
A c
T a
T t
T g
T c
G a
G t
G g
G c
C a
C t
C g
C c
```

1.2.4 Instead of -a files can be given after '::::'

Example: Print the combinations of two files of nucleotides. .

```
$ echo -e "A\nT\nG\nC" > ATGC.txt
$ echo -e "a\nt\ng\nc" > atgc.txt
parallel  echo ::::  ATGC.txt  ::::  atgc.txt
```

output:

```
A a
A t
A g
A c
T a
T t
T g
T c
G a
G g
```

```
G t
G c
C a
C t
C g
C c
```

1.2.5 ':::' and '::::' can be mixed:

I created a random list of genes using:

```
for L in 01 02 03 04 05 06 07 08 09 ; do curl -s "http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/kgXref.
    txt.gz" | gunzip -c | cut -d ' ' -f 4 | grep _ | uniq | head -n 10 | shuf | head -n 5 | sort > list_genes_${L
    }.txt ; done
```

Example: search two genes in a file containing the filenames of lists of genes. .

```
$ ls list_genes_*.txt > list_of_files.txt
$ parallel grep -Hn ::: B7ZGX9 I7FC33 :::: list_of_files.txt
```

output:

```
list_genes_01.txt:1:B7ZGX9_HUMAN
list_genes_02.txt:1:B7ZGX9_HUMAN
list_genes_04.txt:1:B7ZGX9_HUMAN
list_genes_07.txt:1:B7ZGX9_HUMAN
list_genes_08.txt:1:B7ZGX9_HUMAN
list_genes_09.txt:1:B7ZGX9_HUMAN
list_genes_02.txt:3:I7FC33_HUMAN
list_genes_06.txt:3:I7FC33_HUMAN
list_genes_07.txt:4:I7FC33_HUMAN
list_genes_08.txt:3:I7FC33_HUMAN
```

1.3 Matching arguments from all input sources

1.3.1 With `--xapply` you can get one argument from each input source

Example: Print the pairs of bases from two sets of nucleotides. . with '`--xapply`'

```
parallel --xapply echo ::: A T G C ::: a t g c
```

output:

```
A a
T t
G g
C c
```

without '`--xapply`': **Example:** Print the pairs from two sets of nucleotides. .

```
$ parallel echo ::: A T G C ::: a t g c
```

output:

```
A a
A t
A g
A c
T a
T t
T g
T c
G a
G t
G g
G c
C a
C t
C g
C c
```

1.3.2 If one of the input sources is too short, its values will wrap

Example: Print the pairs of two sets of nucleotides. .

```
$ parallel --xapply echo ::: A T ::: g c n
```

output:

```
A g
T c
A n
```

1.4 Changing the argument separator

GNU **parallel** can use other separators than ':::' or '::::'. This is typically useful if ':::' or '::::' is used in the command to run. **Example:** Print the combinations of 3 sets of swissprot accessions .

```
$parallel --arg-sep yoyo echo yoyo B7ZGX9 I7FC33 yoyo EIF4G1 PABPC1 yoyo B7ZGX9_HUMAN C9J4L2_HUMAN
```

output:

```
B7ZGX9 EIF4G1 B7ZGX9_HUMAN
B7ZGX9 EIF4G1 C9J4L2_HUMAN
B7ZGX9 PABPC1 B7ZGX9_HUMAN
B7ZGX9 PABPC1 C9J4L2_HUMAN
I7FC33 EIF4G1 B7ZGX9_HUMAN
I7FC33 EIF4G1 C9J4L2_HUMAN
I7FC33 PABPC1 B7ZGX9_HUMAN
I7FC33 PABPC1 C9J4L2_HUMAN
```

1.4.1 Changing the argument file separator

Example: Search two accessions in a set of files containing some swissprot accessions .

```
$ ls list_genes_0* |\
        parallel --arg-file-sep schtroumph 'grep -nH' ::: B7ZGX9 I7FC33 schtroumph -
```

output:

```
list_genes_01.txt:1:B7ZGX9_HUMAN
list_genes_02.txt:1:B7ZGX9_HUMAN
list_genes_04.txt:1:B7ZGX9_HUMAN
list_genes_07.txt:1:B7ZGX9_HUMAN
list_genes_08.txt:1:B7ZGX9_HUMAN
list_genes_09.txt:1:B7ZGX9_HUMAN
list_genes_02.txt:3:I7FC33_HUMAN
list_genes_06.txt:3:I7FC33_HUMAN
list_genes_07.txt:4:I7FC33_HUMAN
list_genes_08.txt:3:I7FC33_HUMAN
```

1.4.2 Changing the argument delimiter

Example: Search two accessions in a set of files containing some swissprot accessions .

```
$ $ echo -n "7ZGX9,I7FC33" | \
        parallel -d , 'grep -nH' :::: - ::: list_genes_0*
```

output:

```
list_genes_01.txt:1:B7ZGX9_HUMAN
list_genes_02.txt:1:B7ZGX9_HUMAN
list_genes_04.txt:1:B7ZGX9_HUMAN
list_genes_07.txt:1:B7ZGX9_HUMAN
list_genes_08.txt:1:B7ZGX9_HUMAN
list_genes_09.txt:1:B7ZGX9_HUMAN
list_genes_02.txt:3:I7FC33_HUMAN
list_genes_06.txt:3:I7FC33_HUMAN
list_genes_07.txt:4:I7FC33_HUMAN
list_genes_08.txt:3:I7FC33_HUMAN
```

1.4.3 NULL can be given as '\0'

Example: Search two accessions in a set of files containing some swissprot accessions .

```
$ echo -n -e "7ZGX9\0I7FC33" | parallel -d '\0' 'grep -nH' :::: - ::: list_genes_0*
```

output:

```
list_genes_01.txt:1:B7ZGX9_HUMAN
list_genes_02.txt:1:B7ZGX9_HUMAN
list_genes_04.txt:1:B7ZGX9_HUMAN
list_genes_07.txt:1:B7ZGX9_HUMAN
list_genes_08.txt:1:B7ZGX9_HUMAN
list_genes_09.txt:1:B7ZGX9_HUMAN
list_genes_02.txt:3:I7FC33_HUMAN
list_genes_06.txt:3:I7FC33_HUMAN
list_genes_07.txt:4:I7FC33_HUMAN
list_genes_08.txt:3:I7FC33_HUMAN
```

1.4.4 A shorthand for '-d \0' is '-0'

Example: Search two accessions in a set of files containing some swissprot accessions . This will often be used to read files from 'find ... -print0'.

```
$ find ./ -name "list_genes_0*.txt" -print0 |\
        parallel -0 'grep -nH' ::: B7ZGX9 I7FC33 :::: -
```

output:

```
./list_genes_01.txt:1:B7ZGX9_HUMAN
./list_genes_04.txt:1:B7ZGX9_HUMAN
./list_genes_02.txt:1:B7ZGX9_HUMAN
./list_genes_09.txt:1:B7ZGX9_HUMAN
./list_genes_07.txt:1:B7ZGX9_HUMAN
./list_genes_08.txt:1:B7ZGX9_HUMAN
./list_genes_06.txt:3:I7FC33_HUMAN
./list_genes_02.txt:3:I7FC33_HUMAN
./list_genes_07.txt:4:I7FC33_HUMAN
./list_genes_08.txt:3:I7FC33_HUMAN
```

2 Building the command line

2.0.5 No command means arguments are commands

If no command is given after **parallel** the arguments themselves are treated as commands: **Example:** Get the file type and list a directory and print the workind directory .

```
parallel ::: 'ls -la toy*' 'file toy.bam' pwd
```

output:

```
toy.bam: gzip compressed data, extra field
/path/to/samtools-0.1.18/examples
-rw-rw-r-- 1 lindenb lindenb 478 Mar 27  2013 toy.bam
-rw-rw-r-- 1 lindenb lindenb 176 Mar 27  2013 toy.bam.bai
-rw-rw-r-- 1 lindenb lindenb 254 Mar 27  2013 toy.dict
-rw-r--r-- 1 lindenb lindenb  98 Mar 27  2013 toy.fa
-rw-rw-r-- 1 lindenb lindenb  32 Mar 27  2013 toy.fa.fai
-rw-r--r-- 1 lindenb lindenb 786 Apr 22  2011 toy.sam
-rw-rw-r-- 1 lindenb lindenb 176 Jul 19 09:28 toy_sorted.bai
-rw-rw-r-- 1 lindenb lindenb 478 Aug  6 15:11 toy_sorted.bam
-rw-rw-r-- 1 lindenb lindenb 176 Oct  2 10:24 toy_sorted.bam.bai
```

The command can be a script, a binary or a Bash function if the function is exported using 'export -f'. **Example:** Index a list of sorted BAMs with samtools .

```
$ index_bam_with_samtools() {
echo "Indexing_$1" && samtools index $1
}
$ export -f index_bam_with_samtools
$ find ./ -name "*_sorted.bam" |\
   parallel -a - index_bam_with_samtools
```

output:

```
Indexing ./ex1b_sorted_sorted.bam
Indexing ./ex1b_sorted.bam
Indexing ./sorted_sorted_sorted.bam
Indexing ./ex1f-rmduppe_sorted.bam
Indexing ./ex1f-rmdupse_sorted.bam
Indexing ./toy_sorted.bam
Indexing ./ex1_sorted_sorted.bam
Indexing ./ex1a_sorted.bam
Indexing ./ex1_sorted.bam
Indexing ./ex1f-rmduppe_sorted_sorted.bam
Indexing ./ex1f-rmdupse_sorted_sorted.bam
Indexing ./ex1f_sorted_sorted.bam
Indexing ./ex1a_sorted_sorted.bam
Indexing ./sorted_sorted.bam
Indexing ./ex1f_sorted.bam
```

3 Replacement strings

3.1 The 5 replacement strings

parallel has several replacement strings. If no replacement strings are used the default is to append '{}': **Example:** get the headers from a list of fasta files .

```
$ parallel  grep -Hn ">"  ::: toy.fa ex1.fa
toy.fa:1:>ref
toy.fa:3:>ref2
ex1.fa:1:>seq1
ex1.fa:29:>seq2
```

output:

```
toy.fa:1:>ref
toy.fa:3:>ref2
ex1.fa:1:>seq1
ex1.fa:29:>seq2
```

The default replacement string is '{}': **Example:** get the headers from a list of fasta files .

```
$ parallel  grep -Hn ">"  {}  ::: toy.fa ex1.fa
```

output:

```
toy.fa:1:>ref
toy.fa:3:>ref2
ex1.fa:1:>seq1
ex1.fa:29:>seq2
```

The replacement string '{}' can be changed with '-I': **Example:** get the fasta headers from a list of fasta files .

```
$ parallel -I FILE_NAME grep -Hn ">"  FILE_NAME ::: toy.fa ex1.fa
```

output:

```
toy.fa:1:>ref
toy.fa:3:>ref2
ex1.fa:1:>seq1
ex1.fa:29:>seq2
```

The replacement string '{.}' removes the extension: **Example:** Sort a list of BAMs: .

```
$ find ./ -name "*.bam" |\
  parallel -a - 'samtools sort' {} {.}_sorted &&\
  find ./ -name "*_sorted.bam"
```

output:

```
./examples/ex1b_sorted.bam
./examples/ex1f-rmduppe_sorted.bam
./examples/ex1f-rmdupse_sorted.bam
./examples/toy_sorted.bam
./examples/ex1a_sorted.bam
./examples/ex1_sorted.bam
./examples/ex1f_sorted.bam
```

The replacement string '{.}' can be changed with '--extensionreplace': **Example:** Sort a list of BAMs .

```
$ find ./ -name "*.bam" |\
  parallel --extensionreplace BARBAPAPA -a - 'samtools sort' {} BARBAPAPA_sorted &&\
  find ./ -name "*_sorted.bam"
```

output:

```
./examples/ex1b_sorted.bam
./examples/ex1f-rmduppe_sorted.bam
./examples/ex1f-rmdupse_sorted.bam
./examples/toy_sorted.bam
./examples/ex1a_sorted.bam
./examples/ex1_sorted.bam
./examples/ex1f_sorted.bam
```

The replacement string '{/}': removes the path: **Example:** List the basenames of a list of FASTA files. .

```
$ find ~/dir1 ~/dir2 ~/dir3  -name "*.fasta" |\
  parallel -a - echo {/}
```

output:

```
seq1.fasta
seq2.fasta
alnfile.fasta
test_project.fasta
hs_owlmonkey.fasta
genomic-seq.fasta
testaln.fasta
test.fasta
```

Example: copy all fasta files into the current working directory .

```
$ ls -l *.fasta

ls: cannot access *.fasta: No such file or directory

$ find ~/tmp/ /home/lindenb/daily/  -name "*.fasta" |\
   parallel -a - cp  {} {/}  && \
   ls -l *.fasta
```

output:

```
-rw------- 1 lindenb lindenb    294 Oct  3 10:22 alnfile.fasta
-rw------- 1 lindenb lindenb 171524 Oct  3 10:22 genomic-seq.fasta
-rw------- 1 lindenb lindenb    416 Oct  3 10:22 hs_owlmonkey.fasta
-rw------- 1 lindenb lindenb   7194 Oct  3 10:22 seq1.fasta
-rw------- 1 lindenb lindenb  25756 Oct  3 10:22 seq2.fasta
-rw------- 1 lindenb lindenb   4620 Oct  3 10:22 testaln.fasta
-rw------- 1 lindenb lindenb    804 Oct  3 10:22 test.fasta
-rw------- 1 lindenb lindenb   3475 Oct  3 10:22 test_project.fasta
```

The replacement string '{/}' can be replaced with '--basenamereplace'.
Example: copy all fasta files into the current working directory .

```
$ find ~/tmp/ /home/lindenb/daily/  -name "*.fasta" |\
   parallel --basenamereplace BASE_FILE_NAME -a - cp {}  BASE_FILE_NAME && \
   ls -l *.fasta
```

output:

```
-rw------- 1 lindenb lindenb    294 Oct  3 11:34 alnfile.fasta
-rw------- 1 lindenb lindenb 171524 Oct  3 11:34 genomic-seq.fasta
-rw------- 1 lindenb lindenb    416 Oct  3 11:34 hs_owlmonkey.fasta
-rw------- 1 lindenb lindenb   7194 Oct  3 11:34 seq1.fasta
-rw------- 1 lindenb lindenb  25756 Oct  3 11:34 seq2.fasta
-rw------- 1 lindenb lindenb   4620 Oct  3 11:34 testaln.fasta
-rw------- 1 lindenb lindenb    804 Oct  3 11:34 test.fasta
-rw------- 1 lindenb lindenb   3475 Oct  3 11:34 test_project.fasta
```

The replacement string {/.} removes the path and the extension.
Example: Sorting the BAM in the current working directory .

```
$ ls -l sorted_*.bam

ls: cannot access sorted_*.bam: No such file or directory

$ find ./ -name "*.bam" |\
  parallel -a - '../samtools sort' {} sorted_{/.} && \
  ls -l sorted_*.bam
```

output:

```
-rw-rw-r-- 1 lindenb lindenb 126888 Oct  3 10:31 sorted_ex1a.bam
-rw-rw-r-- 1 lindenb lindenb 126583 Oct  3 10:31 sorted_ex1.bam
-rw-rw-r-- 1 lindenb lindenb 126878 Oct  3 10:31 sorted_ex1b.bam
-rw-rw-r-- 1 lindenb lindenb 208594 Oct  3 10:31 sorted_ex1f.bam
-rw-rw-r-- 1 lindenb lindenb 180639 Oct  3 10:31 sorted_ex1f-rmduppe.bam
-rw-rw-r-- 1 lindenb lindenb 132225 Oct  3 10:31 sorted_ex1f-rmdupse.bam
-rw-rw-r-- 1 lindenb lindenb    478 Oct  3 10:31 sorted_toy.bam
```

The replacement string '{//}' keeps only the path.
Example: print the path of the fasta files .

```
find dir -name "*.fa" |\
 parallel echo {//} |\
 sort | uniq
```

output:

```
dir/dir1
dir/dir2
dir/dir3
```

The replacement string '{//}' can be changed with '--dirnamereplace'.
Example: print the path of the fasta files .

```
find dir -name "*.fa" |\
 parallel --dirnamereplace BARBALALA echo BARBALALA |\
 sort | uniq
```

output:

```
dir/dir1
dir/dir2
dir/dir3
```

The replacement string {#} gives the job number.
Example: Print the basename of some FASTA files and the job number .

```
$ find ~/tmp -name "*.fasta" |\
  parallel -a - echo {#} {/}
```

output:

```
1 seq1.fasta
2 seq2.fasta
4 test_project.fasta
3 alnfile.fasta
5 hs_owlmonkey.fasta
6 genomic-seq.fasta
7 testaln.fasta
8 test.fasta
```

The replacement string {#} can be changed with `--seqreplace`.

Example: Print the basename of some FASTA files and the job number .

```
$ find ~/tmp -name "*.fasta" |\
  parallel -a - --seqreplace JOBNUM echo JOBNUM {/}
```

output:

```
1 seq1.fasta
2 seq2.fasta
3 alnfile.fasta
4 test_project.fasta
5 hs_owlmonkey.fasta
6 genomic-seq.fasta
7 testaln.fasta
8 test.fasta
```

3.2 Positional replacement strings

With multiple input sources the argument from the individual input sources can be access with {number}: The positional replacement strings can also be modified using `'/'` or `'//'` or `'/.'` or `'.'`:

Example: Aligning with 'BWA aln' two pairs of fastqs on two indexed references .

```
$ parallel  bwa aln -f {1//}/{2/.}_{1/.}.sai {2} {1} \
 ::: examples/01_F.fastq.gz examples/01_R.fastq.gz examples/02_F.fastq.gz  examples/02_R.fastq.gz \
 ::: examples/toy.fa examples/ex1.fa
```

will generate

```
bwa aln -f examples/toy_01_F.fastq.sai examples/toy.fa examples/01_F.fastq.gz
bwa aln -f examples/ex1_01_F.fastq.sai examples/ex1.fa examples/01_F.fastq.gz
bwa aln -f examples/toy_01_R.fastq.sai examples/toy.fa examples/01_R.fastq.gz
bwa aln -f examples/ex1_01_R.fastq.sai examples/ex1.fa examples/01_R.fastq.gz
bwa aln -f examples/toy_02_F.fastq.sai examples/toy.fa examples/02_F.fastq.gz
bwa aln -f examples/ex1_02_F.fastq.sai examples/ex1.fa examples/02_F.fastq.gz
bwa aln -f examples/toy_02_R.fastq.sai examples/toy.fa examples/02_R.fastq.gz
bwa aln -f examples/ex1_02_R.fastq.sai examples/ex1.fa examples/02_R.fastq.gz
```

output:

```
$ ls -lah examples/*.sai

-rw-rw-r-- 1 lindenb lindenb  27K Oct  3 12:38 examples/ex1_01_F.fastq.sai
-rw-rw-r-- 1 lindenb lindenb  27K Oct  3 12:38 examples/ex1_01_R.fastq.sai
-rw-rw-r-- 1 lindenb lindenb  27K Oct  3 12:38 examples/ex1_02_F.fastq.sai
-rw-rw-r-- 1 lindenb lindenb  27K Oct  3 12:38 examples/ex1_02_R.fastq.sai
-rw-rw-r-- 1 lindenb lindenb 4.0K Oct  3 12:38 examples/toy_01_F.fastq.sai
-rw-rw-r-- 1 lindenb lindenb 4.0K Oct  3 12:38 examples/toy_01_R.fastq.sai
-rw-rw-r-- 1 lindenb lindenb 4.0K Oct  3 12:38 examples/toy_02_F.fastq.sai
-rw-rw-r-- 1 lindenb lindenb 4.0K Oct  3 12:38 examples/toy_02_R.fastq.sai
```

3.3 Input from columns

The columns in a file can be bound to positional replacement strings using `'--colsep'`. Here the columns are separated with TAB:

Example: use ‘paste‘ to get two columns containing two FASTQs forward and reverse and align with ‘bwa mem‘ .

```
$ find examples/ -name "*.fastq.gz" |\
  sort |\
  paste -- - - |\
  parallel  --colsep '\t' bwa mem examples/ex1.fa {1} {2} ">" {1//}/{1/.}_{2/.}.sam
```

will generate

```
bwa mem examples/ex1.fa examples/01_F.fastq.gz examples/01_R.fastq.gz > examples/01_F.fastq_01_R.fastq.sam
bwa mem examples/ex1.fa examples/02_F.fastq.gz examples/02_R.fastq.gz > examples/02_F.fastq_02_R.fastq.sam
bwa mem examples/ex1.fa examples/03_F.fastq.gz examples/03_R.fastq.gz > examples/03_F.fastq_03_R.fastq.sam
bwa mem examples/ex1.fa examples/04_F.fastq.gz examples/04_R.fastq.gz > examples/04_F.fastq_04_R.fastq.sam
bwa mem examples/ex1.fa examples/05_F.fastq.gz examples/05_R.fastq.gz > examples/05_F.fastq_05_R.fastq.sam
```

output:

```
$ ls -la examples/*.sam

-rw-rw-r-- 1 lindenb lindenb 447025 Oct  3 13:03 examples/01_F.fastq_01_R.fastq.sam
-rw-rw-r-- 1 lindenb lindenb 447025 Oct  3 13:03 examples/02_F.fastq_02_R.fastq.sam
-rw-rw-r-- 1 lindenb lindenb 447025 Oct  3 13:03 examples/03_F.fastq_03_R.fastq.sam
-rw-rw-r-- 1 lindenb lindenb 447025 Oct  3 13:03 examples/04_F.fastq_04_R.fastq.sam
-rw-rw-r-- 1 lindenb lindenb 447025 Oct  3 13:03 examples/05_F.fastq_05_R.fastq.sam
```

3.4 Header defined replacement strings

With **'--header'** GNU **parallel** will use the first value of the input source as the name of the replacement string. Only the non-modified version '{}' is supported.
Example: global alignment of oligonucleotides with primer3/ntdpal .

```
$ parallel --header : primer3-2.3.5/src/ntdpal {FORWARD} {REVERSE} g \
    ::: REVERSE ATCTGACTCGTGC ACTGATCGATCGATCG \
    ::: FORWARD ATAGTAATAT ACTATA GAAATTC
```

output:

```
|ATAGTAATAT|   |ATCTGACTCGTGC| g score=2.00 len=4 |6,0|7,1|8,2|9,3|
|ACTATA|   |ATCTGACTCGTGC| g score=1.00 len=6 |0,5|1,6|2,7|3,9|4,10|5,11|
|GAAATTC|   |ATCTGACTCGTGC| g score=2.00 len=3 |3,0|5,1|6,2|
|ATAGTAATAT|   |ACTGATCGATCGATCG| g score=2.00 len=8 |0,4|1,5|2,6|3,7|6,8|7,9|8,12|9,13|
|ACTATA|   |ACTGATCGATCGATCG| g score=3.00 len=6 |0,0|1,1|2,2|3,4|4,5|5,8|
|GAAATTC|   |ACTGATCGATCGATCG| g score=1.00 len=6 |0,7|1,8|2,11|3,12|5,13|6,14|
```

3.5 More than one argument

With **'--xargs'** will GNU **parallel** fit as many arguments as possible on a single line:

```
$ seq 1 100000 | parallel --xargs echo  | wc -l
5
```

The 100000 arguments fitted on 5 lines.

The maximal length of a single line can be set with **'-s'**. With a maximal line length of 10000 chars 595 commands will be run:

```
$ seq 1 100000 | parallel --xargs -s  1000 echo  | wc -l
595
```

3.6 Quoting

Command lines that contain special characters may need to be protected from the shell.
Example: linearize some FASTA files with awk .

```
 find dir1 -name "*.fa" |\
 xargs awk '/^>/ {printf("\n%s\t",$0);next;} { printf("%s",$0);} END { printf("\n");} '
```

output

```
>seq1    CACTAGTGGCTCATTGTAAATGTGTGGTTTAACTCGTCCATGGCCCAGCATTAGGGAGCTGTGGACCCTGCAGCCTGGCTGTGGGGGCCGCAGT
>seq2    TTCAAATGAACTTCTGTAATTGAAAAATTCATTTAAGAAATTACAAAATATAGTTGAAAGCTCTAACAATAGACTAAACCAAGCAGAAGAAAGA
>ref     AGCATGTTAGATAAGATAGCTGTGCTAGTAGGCAGTCAGCGCCAT
>ref2    aggttttataaaacaattaagtctacagagcaactacgcg
(...)
```

This won't work:

```
Command lines that contain special characters may need to be protected from the shell.
\begin{lstlisting}
 find dir1 -name "*.fa" |\
 parallel awk '/^>/ {printf("\n%s\t",$0);next;} { printf("%s",$0);} END { printf("\n");} '
```

To quote the command use **'-q'**:

Example: linearize some FASTA files with awk .

```
 find dir1 -name "*.fa" |\
 parallel -q awk '/^>/ {printf("\n%s\t",$0);next;} { printf("%s",$0);} END { printf("\n");} '

>seq1    CACTAGTGGCTCATTGTAAATGTGTGGTTTAACTCGTCCATGGCCCAGCATTAGGGAGCTGTGGACCCTGCAGCCTGGCTGTGGGGGCCGCAGT
>seq2    TTCAAATGAACTTCTGTAATTGAAAAATTCATTTAAGAAATTACAAAATATAGTTGAAAGCTCTAACAATAGACTAAACCAAGCAGAAGAAAGA

>ref     AGCATGTTAGATAAGATAGCTGTGCTAGTAGGCAGTCAGCGCCAT
>ref2    aggttttataaaacaattaagtctacagagcaactacgcg
```

Or you can quote the critical part using **\'**.

Example: linearize some FASTA files with awk .

```
$ find dir -name "*.fa" |\
 parallel awk \' '/^>/ {printf("\n%s\t",$0);next;} { printf("%s",$0);} END { printf("\n");} ' \'

>seq1    CACTAGTGGCTCATTGTAAATGTGTGGTTTAACTCGTCCATGGCCCAGCATTAGGGAGCTGTGGACCCTGCAGCCTGGCTGTGGGGGCCGCAGT
>seq2    TTCAAATGAACTTCTGTAATTGAAAAATTCATTTAAGAAATTACAAAATATAGTTGAAAGCTCTAACAATAGACTAAACCAAGCAGAAGAAAGA

>ref     AGCATGTTAGATAAGATAGCTGTGCTAGTAGGCAGTCAGCGCCAT
>ref2    aggttttataaaacaattaagtctacagagcaactacgcg
```

3.7 Trimming space

Space can be trimmed on the arguments using **'--trim'**

Example: Print A T G C, trim left spaces .

```
$ parallel --trim l echo [{}] :::  " A " " T " " C " " G "
```

output:

```
[A ]
[T ]
[C ]
[G ]
```

Example: Print A T G C, trim right spaces .

```
$ parallel --trim r echo [{}] :::  " A " " T " " C " " G "
```

output:

```
[ A]
[ T]
[ C]
[ G]
```

Example: Print A T G C, trim left and right spaces .

```
$ parallel --trim lr echo [{}] :::  " A " " T " " C " " G "
```

output:

```
[A]
[T]
[C]
[G]
```

4 Controling the output

The output can prefixed with the argument:

```
$ parallel --tag echo prefix-{} ::: A T G C
```

output:

```
A       prefix-A
T       prefix-T
G       prefix-G
C       prefix-C
```

To prefix it with another string use –tagstring:

```
parallel --tagstring suffix-{} echo  ::: A T G C
```

output:

```
suffix-A        A
suffix-T        T
suffix-G        G
suffix-C        C
```

4.1 To see what commands will be run without running them

Example: Align a set of FASTQs with bwa aln .

```
$ find ./ -name "*.fastq.gz" |\
        parallel --dryrun  bwa aln -f {/.}.sai toy.fa {}
```

output:

```
bwa aln -f 02_R_.fastq.sai toy.fa ./02_R_.fastq.gz
bwa aln -f 03_R_.fastq.sai toy.fa ./03_R_.fastq.gz
bwa aln -f 01_R_.fastq.sai toy.fa ./01_R_.fastq.gz
bwa aln -f 05_F_.fastq.sai toy.fa ./05_F_.fastq.gz
bwa aln -f 05_R_.fastq.sai toy.fa ./05_R_.fastq.gz
bwa aln -f 04_R_.fastq.sai toy.fa ./04_R_.fastq.gz
bwa aln -f 03_F_.fastq.sai toy.fa ./03_F_.fastq.gz
bwa aln -f 04_F_.fastq.sai toy.fa ./04_F_.fastq.gz
bwa aln -f 01_F_.fastq.sai toy.fa ./01_F_.fastq.gz
bwa aln -f 02_F_.fastq.sai toy.fa ./02_F_.fastq.gz
```

Hack from Stephen Turner @genetics_blog

```
In bash, ^foo^bar repeats the latest command, replacing the first instance of 'foo' with 'bar'.
ith GNU parallel to actually run last dry run commands: \$ ^--dry-run^
```

4.2 To print the command before running them use –verbose

Example: Show how to align a set of FASTQs with bwa aln .

```
$ find ./ -name "*.fastq.gz" |\
        parallel --verbose  bwa aln -f {/.}.sai toy.fa {}
```

output:

```
bwa aln -f 02_R_.fastq.sai toy.fa ./02_R_.fastq.gz
bwa aln -f 03_R_.fastq.sai toy.fa ./03_R_.fastq.gz
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
[bwa_aln] 190bp reads: max_diff = 8
```

```
[bwa_aln] 225bp reads: max_diff = 9
[bwa_aln_core] calculate SA coordinate... 0.01 sec
[bwa_aln_core] write to the disk... 0.00 sec
[bwa_aln_core] 1000 sequences have been processed.
[main] Version: 0.7.4-r385
[main] CMD: bwa aln -f 02_R_.fastq.sai toy.fa ./02_R_.fastq.gz
[main] Real time: 0.017 sec; CPU: 0.020 sec
bwa aln -f 01_R_.fastq.sai toy.fa ./01_R_.fastq.gz
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
[bwa_aln] 190bp reads: max_diff = 8
[bwa_aln] 225bp reads: max_diff = 9
[bwa_aln_core] calculate SA coordinate... 0.00 sec
[bwa_aln_core] write to the disk... 0.00 sec
[bwa_aln_core] 1000 sequences have been processed.
[main] Version: 0.7.4-r385
[main] CMD: bwa aln -f 03_R_.fastq.sai toy.fa ./03_R_.fastq.gz
[main] Real time: 0.016 sec; CPU: 0.012 sec
bwa aln -f 05_F_.fastq.sai toy.fa ./05_F_.fastq.gz
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
[bwa_aln] 190bp reads: max_diff = 8
[bwa_aln] 225bp reads: max_diff = 9
[bwa_aln_core] calculate SA coordinate... 0.01 sec
[bwa_aln_core] write to the disk... 0.00 sec
[bwa_aln_core] 1000 sequences have been processed.
[main] Version: 0.7.4-r385
[main] CMD: bwa aln -f 01_R_.fastq.sai toy.fa ./01_R_.fastq.gz
[main] Real time: 0.012 sec; CPU: 0.012 sec
bwa aln -f 05_R_.fastq.sai toy.fa ./05_R_.fastq.gz
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
[bwa_aln] 190bp reads: max_diff = 8
[bwa_aln] 225bp reads: max_diff = 9
[bwa_aln_core] calculate SA coordinate... 0.00 sec
[bwa_aln_core] write to the disk... 0.00 sec
[bwa_aln_core] 1000 sequences have been processed.
[main] Version: 0.7.4-r385
[main] CMD: bwa aln -f 05_F_.fastq.sai toy.fa ./05_F_.fastq.gz
[main] Real time: 0.013 sec; CPU: 0.012 sec
bwa aln -f 04_R_.fastq.sai toy.fa ./04_R_.fastq.gz
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
[bwa_aln] 190bp reads: max_diff = 8
[bwa_aln] 225bp reads: max_diff = 9
[bwa_aln_core] calculate SA coordinate... 0.01 sec
[bwa_aln_core] write to the disk... 0.00 sec
[bwa_aln_core] 1000 sequences have been processed.
[main] Version: 0.7.4-r385
[main] CMD: bwa aln -f 05_R_.fastq.sai toy.fa ./05_R_.fastq.gz
[main] Real time: 0.012 sec; CPU: 0.012 sec
bwa aln -f 03_F_.fastq.sai toy.fa ./03_F_.fastq.gz
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
[bwa_aln] 190bp reads: max_diff = 8
[bwa_aln] 225bp reads: max_diff = 9
[bwa_aln_core] calculate SA coordinate... 0.01 sec
[bwa_aln_core] write to the disk... 0.00 sec
[bwa_aln_core] 1000 sequences have been processed.
[main] Version: 0.7.4-r385
[main] CMD: bwa aln -f 04_R_.fastq.sai toy.fa ./04_R_.fastq.gz
[main] Real time: 0.012 sec; CPU: 0.012 sec
bwa aln -f 04_F_.fastq.sai toy.fa ./04_F_.fastq.gz
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
[bwa_aln] 190bp reads: max_diff = 8
[bwa_aln] 225bp reads: max_diff = 9
```

```
[bwa_aln_core] calculate SA coordinate... 0.00 sec
[bwa_aln_core] write to the disk... 0.00 sec
[bwa_aln_core] 1000 sequences have been processed.
[main] Version: 0.7.4-r385
[main] CMD: bwa aln -f 03_F_.fastq.sai toy.fa ./03_F_.fastq.gz
[main] Real time: 0.013 sec; CPU: 0.012 sec
bwa aln -f 02_F_.fastq.sai toy.fa ./02_F_.fastq.gz
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
[bwa_aln] 190bp reads: max_diff = 8
[bwa_aln] 225bp reads: max_diff = 9
[bwa_aln_core] calculate SA coordinate... 0.00 sec
[bwa_aln_core] write to the disk... 0.00 sec
[bwa_aln_core] 1000 sequences have been processed.
[main] Version: 0.7.4-r385
[main] CMD: bwa aln -f 04_F_.fastq.sai toy.fa ./04_F_.fastq.gz
[main] Real time: 0.012 sec; CPU: 0.012 sec
bwa aln -f 01_F_.fastq.sai toy.fa ./01_F_.fastq.gz
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
[bwa_aln] 190bp reads: max_diff = 8
[bwa_aln] 225bp reads: max_diff = 9
[bwa_aln_core] calculate SA coordinate... 0.00 sec
[bwa_aln_core] write to the disk... 0.00 sec
[bwa_aln_core] 1000 sequences have been processed.
[main] Version: 0.7.4-r385
[main] CMD: bwa aln -f 02_F_.fastq.sai toy.fa ./02_F_.fastq.gz
[main] Real time: 0.016 sec; CPU: 0.012 sec
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
[bwa_aln] 190bp reads: max_diff = 8
[bwa_aln] 225bp reads: max_diff = 9
[bwa_aln_core] calculate SA coordinate... 0.01 sec
[bwa_aln_core] write to the disk... 0.00 sec
[bwa_aln_core] 1000 sequences have been processed.
[main] Version: 0.7.4-r385
[main] CMD: bwa aln -f 01_F_.fastq.sai toy.fa ./01_F_.fastq.gz
[main] Real time: 0.012 sec; CPU: 0.012 sec
```

4.3 GNU parallel will postpone the output until the command completes

Example: Align the FASTQs forward with bwa aln, wait a few seconds and then align the FASTQs reverse .

```
$ parallel -j 3 --verbose 'bwa aln -f 0{}_F.fastq.sai toy.fa 0{}_F_.fastq.gz ; sleep {} ; bwa aln -f 0{}_R.fastq.
    sai toy.fa 0{}_R_.fastq.gz ' ::: 1 2 3 4 5
```

output:

```
$ parallel -j 3 --verbose  'bwa aln -f 0{}_F.fastq.sai toy.fa 0{}_F_.fastq.gz ; sleep {} ; bwa aln -f 0{}_R.fastq.
    sai toy.fa 0{}_R_.fastq.gz ' ::: 1 2 3 4 5

bwa aln -f 01_F.fastq.sai toy.fa 01_F_.fastq.gz ; sleep 1 ; bwa aln -f 01_R.fastq.sai toy.fa 01_R_.fastq.gz
bwa aln -f 02_F.fastq.sai toy.fa 02_F_.fastq.gz ; sleep 2 ; bwa aln -f 02_R.fastq.sai toy.fa 02_R_.fastq.gz
bwa aln -f 03_F.fastq.sai toy.fa 03_F_.fastq.gz ; sleep 3 ; bwa aln -f 03_R.fastq.sai toy.fa 03_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 01_F.fastq.sai toy.fa 01_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 01_R.fastq.sai toy.fa 01_R_.fastq.gz
(...)
bwa aln -f 04_F.fastq.sai toy.fa 04_F_.fastq.gz ; sleep 4 ; bwa aln -f 04_R.fastq.sai toy.fa 04_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 02_F.fastq.sai toy.fa 02_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 02_R.fastq.sai toy.fa 02_R_.fastq.gz
(...)
bwa aln -f 05_F.fastq.sai toy.fa 05_F_.fastq.gz ; sleep 5 ; bwa aln -f 05_R.fastq.sai toy.fa 05_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 03_F.fastq.sai toy.fa 03_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 03_R.fastq.sai toy.fa 03_R_.fastq.gz
```

```
(...)
[main] CMD: bwa aln -f 04_F.fastq.sai toy.fa 04_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 04_R.fastq.sai toy.fa 04_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 05_F.fastq.sai toy.fa 05_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 05_R.fastq.sai toy.fa 05_R_.fastq.gz
```

4.4 To get the output immediately use '--ungroup'

Example: Align the FASTQs forward with bwa aln, wait a few seconds and then align the FASTQs reverse .

```
$ parallel -j 3 --verbose --ungroup 'bwa aln -f 0{}_F.fastq.sai toy.fa 0{}_F_.fastq.gz ; sleep {} ; bwa aln -f 0{}
    _R.fastq.sai toy.fa 0{}_R_.fastq.gz ' ::: 1 2 3 4 5
```

output:

```
bwa aln -f 01_F.fastq.sai toy.fa 01_F_.fastq.gz ; sleep 1 ; bwa aln -f 01_R.fastq.sai toy.fa 01_R_.fastq.gz
bwa aln -f 02_F.fastq.sai toy.fa 02_F_.fastq.gz ; sleep 2 ; bwa aln -f 02_R.fastq.sai toy.fa 02_R_.fastq.gz
(...)
bwa aln -f 03_F.fastq.sai toy.fa 03_F_.fastq.gz ; sleep 3 ; bwa aln -f 03_R.fastq.sai toy.fa 03_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 01_F.fastq.sai toy.fa 01_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 02_F.fastq.sai toy.fa 02_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 03_F.fastq.sai toy.fa 03_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 01_R.fastq.sai toy.fa 01_R_.fastq.gz
(...)
bwa aln -f 04_F.fastq.sai toy.fa 04_F_.fastq.gz ; sleep 4 ; bwa aln -f 04_R.fastq.sai toy.fa 04_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 04_F.fastq.sai toy.fa 04_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 02_R.fastq.sai toy.fa 02_R_.fastq.gz
(...)
bwa aln -f 05_F.fastq.sai toy.fa 05_F_.fastq.gz ; sleep 5 ; bwa aln -f 05_R.fastq.sai toy.fa 05_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 05_F.fastq.sai toy.fa 05_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 03_R.fastq.sai toy.fa 03_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 04_R.fastq.sai toy.fa 04_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 05_R.fastq.sai toy.fa 05_R_.fastq.gz
(...)
```

'-ungroup' is fast, but can cause half a line from one job to be mixed with half a line of another job. That has happend in the second line, where the line '4-middle' is mixed with '2-start'. To avoid this use '--linebuffer' (which, however, is much slower).

Example: Align the FASTQs forward with bwa aln, wait a few seconds and then align the FASTQs reverse .

```
$ parallel -j 3 --verbose --linebuffer 'bwa aln -f 0{}_F.fastq.sai toy.fa 0{}_F_.fastq.gz ; sleep {} ; bwa aln -f
    0{}_R.fastq.sai toy.fa 0{}_R_.fastq.gz ' ::: 1 2 3 4 5
```

output:

```
bwa aln -f 01_F.fastq.sai toy.fa 01_F_.fastq.gz ; sleep 1 ; bwa aln -f 01_R.fastq.sai toy.fa 01_R_.fastq.gz
bwa aln -f 02_F.fastq.sai toy.fa 02_F_.fastq.gz ; sleep 2 ; bwa aln -f 02_R.fastq.sai toy.fa 02_R_.fastq.gz
bwa aln -f 03_F.fastq.sai toy.fa 03_F_.fastq.gz ; sleep 3 ; bwa aln -f 03_R.fastq.sai toy.fa 03_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 01_F.fastq.sai toy.fa 01_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 03_F.fastq.sai toy.fa 03_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 02_F.fastq.sai toy.fa 02_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 01_R.fastq.sai toy.fa 01_R_.fastq.gz
(...)
bwa aln -f 04_F.fastq.sai toy.fa 04_F_.fastq.gz ; sleep 4 ; bwa aln -f 04_R.fastq.sai toy.fa 04_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 04_F.fastq.sai toy.fa 04_F_.fastq.gz
```

```
(...)
[main] CMD: bwa aln -f 02_R.fastq.sai toy.fa 02_R_.fastq.gz
(...)
bwa aln -f 05_F.fastq.sai toy.fa 05_F_.fastq.gz ; sleep 5 ; bwa aln -f 05_R.fastq.sai toy.fa 05_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 05_F.fastq.sai toy.fa 05_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 03_R.fastq.sai toy.fa 03_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 04_R.fastq.sai toy.fa 04_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 05_R.fastq.sai toy.fa 05_R_.fastq.gz
(...)
```

To force the output in the same order as the arguments use '`--keep-order`'/'`-k`'. **Example:** Align the FASTQs forward with bwa aln, wait a few seconds and then align the FASTQs reverse .

```
$ parallel -j 2 --verbose --keep-order 'bwa aln -f 0{}_F.fastq.sai toy.fa 0{}_F_.fastq.gz ; sleep {} ; bwa aln -f
    0{}_R.fastq.sai toy.fa 0{}_R_.fastq.gz ' ::: 1 2 3 4 5
```

output:

```
bwa aln -f 01_F.fastq.sai toy.fa 01_F_.fastq.gz ; sleep 1 ; bwa aln -f 01_R.fastq.sai toy.fa 01_R_.fastq.gz
bwa aln -f 02_F.fastq.sai toy.fa 02_F_.fastq.gz ; sleep 2 ; bwa aln -f 02_R.fastq.sai toy.fa 02_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 01_F.fastq.sai toy.fa 01_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 01_R.fastq.sai toy.fa 01_R_.fastq.gz
(...)
bwa aln -f 03_F.fastq.sai toy.fa 03_F_.fastq.gz ; sleep 3 ; bwa aln -f 03_R.fastq.sai toy.fa 03_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 02_F.fastq.sai toy.fa 02_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 02_R.fastq.sai toy.fa 02_R_.fastq.gz
(...)
bwa aln -f 04_F.fastq.sai toy.fa 04_F_.fastq.gz ; sleep 4 ; bwa aln -f 04_R.fastq.sai toy.fa 04_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 03_F.fastq.sai toy.fa 03_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 03_R.fastq.sai toy.fa 03_R_.fastq.gz
(...)
bwa aln -f 05_F.fastq.sai toy.fa 05_F_.fastq.gz ; sleep 5 ; bwa aln -f 05_R.fastq.sai toy.fa 05_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 04_F.fastq.sai toy.fa 04_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 04_R.fastq.sai toy.fa 04_R_.fastq.gz
(...)
[main] CMD: bwa aln -f 05_F.fastq.sai toy.fa 05_F_.fastq.gz
(...)
[main] CMD: bwa aln -f 05_R.fastq.sai toy.fa 05_R_.fastq.gz
```

Another example with '`-k`'.
Example: get the flanking sequence +/- 100bp from a VCF file using samtools faidx . .

```
grep -vE "^#" ~/data.vcf |\
awk '{printf("%s:%d-%d\n",$1,$2 -100, $2+100);}' |\
parallel -k samtools faidx  hg18.fa
```

output:

```
>chr1:762173-762373
GTATAGTCTCCTCGTCATGTCTGCCGCTTCTTCCTGAGTCAGGGAATATCTCTTAGGCCA
TATCTATTATAGTCGTGGTCTGACTTATATTTGTGGTCAAtttttttttcttaatttttc
gtagagacggggtctcactatgttgcccaggctggtctcaaactctaagtgatcctcctg
cctcagcctcccaaactgctg
>chr1:860908-861108
AGGTCATCCCCACGCTCACACACAGAGCTAGGCACTCCCTGTGCCCAGGCTGGGCTCCAG
CCTCGCAGCTGCCCACGGGGTCAGCTTTTCCCGGTCTCGTTCTGCAGCCAGGACGGCAAC
CTTCCCACCCTCATATCCAGCGTCCACCGCAGCCGCCACCTCGTTATGCCCGAGCATCAG
AGCCGCTGTGAATTCCAGAGA
>chr1:861530-861730
AGGGCTCCTGGACGGAGGGGGTCCCCGGTCCCGCCTCCTAGGGCTCCTGGACGGAAGGGG
TCCCCGGTCCCGCCTCCTAGGGCTCCTGGACGGAAGGGGTCCCCGGTCCCGCCTCCTAGG
GCTCCTGGACGGAGGGGGTCCCCGGTCGGTCCCGCCTTCTAGGGCTCCGGGAAGGATGGG
GTTCTCGGGAGGGAAGGGATC
>chr1:866219-866419
GGCCCTGTGGCCGCCCATCCTCCTGCCCCGTGCCCGAGACCAGCCCAGGGGCCGAGCACG
GCCGAGTGGTGTGGTCAGTTCCCCACCTCAGTGTTCTACGCCAGGACGCGGGCTGGGGAG
GATGAGGGCGCATAGCCGGGGGGATCACTGCTGTTGTCCCCCACCCAGATCTCCTGAGGG
TCCGGCAGGAGGTGGCGGCTG
>chr1:866411-866611
TGGCGGCTGCAGCTCTGAGGGGCCCCAGTGGCCTGGAAGCCCACCTGCCCTCCTCCACGG
```

```
CAGGTCAGCGTCGGAAGCAGGGCCTGGCTCAGCACCGGGAGGGCGCCGCCCCAGCTGCCG
CCCCGTCCTTCTCGGAGAGGTACTGGGGTGGCTGCCGTTCTCTGCTTGTTTCTGGGGTGC
CGCCCGCACCCCCGCGCTCTC
>chr1:870803-871003
TCAGGTCAAAGAGGTCTTTAAATTGCTTCCTGTCCTCATCCTTCCTGTCAGCCATCTTCC
TTCGTTTGATCTCAGGGAAGTTCAGGTCTTCCAGCTGGAAGGCCAAAGAACCAGGGGCTC
AGGTGAGAGAGGGCAGGGGCTGGCGGCCACAGCAGGGCCAGGCATCGCCAGACCCACCAC
CAGGGCCCCATGTGGCCAATT
>chr1:871234-871434
GGAAACGCCTGGTTCTGGCCAGTTCTCCAACACCTACCCCCTCTCCAAGTCGAATCATCC
GGGCACGGCCCTGGCCGCCTGGCACTGTTTCCAAACCCTCGCCCTGGTCTCAAGTCATAG
TGCGCTAGATCTGAAACCCAGGAAGTCACAACACACCCCCAGGTCCCCTCGCCGAGCCGC
ACCCGCTCTTTGCCACTGATC
```

4.4.1 Saving output into files

GNU **parallel** can save the output of each job into files.
Example: Align two sets of oligos using primer3/ntdpal, save the result in a structured output .

```
$ parallel --files --verbose primer3-2.3.5/src/ntdpal {1} {2} g \
        ::: AATCGTACGTACG ATAGCATCGA \
        ::: AATCGTACGTCG  ATAGCATCGAG
```

output:

```
/home/lindenb/package/primer3-2.3.5/src/ntdpal AATCGTACGTACG AATCGTACGTCG g
/home/lindenb/package/primer3-2.3.5/src/ntdpal AATCGTACGTACG ATAGCATCGAG g
/tmp/eLAmSSq0DY.par
/home/lindenb/package/primer3-2.3.5/src/ntdpal ATAGCATCGA AATCGTACGTCG g
/tmp/Yv7YvDPzzw.par
/home/lindenb/package/primer3-2.3.5/src/ntdpal ATAGCATCGA ATAGCATCGAG g
/tmp/mea6UUq1Li.par
/tmp/VLbG2gkq2c.par

$ head /tmp/eLAmSSq0DY.par /tmp/Yv7YvDPzzw.par /tmp/mea6UUq1Li.par /tmp/VLbG2gkq2c.par
==> /tmp/eLAmSSq0DY.par <==
|AATCGTACGTACG|   |AATCGTACGTCG| g score=11.00 len=12 |0,0|1,1|2,2|3,3|4,4|5,5|6,6|7,7|8,8|9,9|11,10|12,11|

==> /tmp/Yv7YvDPzzw.par <==
|AATCGTACGTACG|   |ATAGCATCGAG| g score=3.00 len=10 |1,0|2,1|3,2|4,3|5,4|6,5|7,7|8,8|10,9|12,10|

==> /tmp/mea6UUq1Li.par <==
|ATAGCATCGA|   |AATCGTACGTCG| g score=3.00 len=6 |4,0|5,1|6,2|7,3|8,4|9,6|

==> /tmp/VLbG2gkq2c.par <==
|ATAGCATCGA|   |ATAGCATCGAG| g score=10.00 len=10 |0,0|1,1|2,2|3,3|4,4|5,5|6,6|7,7|8,8|9,9|
```

By default GNU **parallel** will cache the output in files in '/tmp'. This can be changed by setting `'$TMPDIR'` or `'--tmpdir'`.
Example: Align two sets of oligos using primer3/ntdpal, save the result in a structured output .

```
$ parallel --files --verbose --tmpdir . primer3-2.3.5/src/ntdpal {1} {2} g \
        ::: AATCGTACGTACG ATAGCATCGA \
        ::: AATCGTACGTCG  ATAGCATCGAG
```

output:

```
/home/lindenb/package/primer3-2.3.5/src/ntdpal AATCGTACGTACG AATCGTACGTCG g
/home/lindenb/package/primer3-2.3.5/src/ntdpal AATCGTACGTACG ATAGCATCGAG g
./0gCXxOB_fm.par
/home/lindenb/package/primer3-2.3.5/src/ntdpal ATAGCATCGA AATCGTACGTCG g
./SD67tLyZTs.par
/home/lindenb/package/primer3-2.3.5/src/ntdpal ATAGCATCGA ATAGCATCGAG g
./RR54nlC0yu.par
./K1c6EBsxZq.par
```

The output files can be saved in a structured way using `'--results'`.
Example: Align a set of five FASTQs, save stdin and stdout in a structured output under the ALN directory .

```
parallel --result ALN  bwa aln -f 0{}_R.fastq.sai toy.fa 0{}_R_.fastq.gz  ::: 1 2 3 4 5
```

output:

```
$ find ALN
ALN
ALN/1
ALN/1/2
ALN/1/2/stdout
ALN/1/2/stderr
ALN/1/1
ALN/1/1/stdout
ALN/1/1/stderr
ALN/1/5
ALN/1/5/stdout
ALN/1/5/stderr
ALN/1/4
ALN/1/4/stdout
ALN/1/4/stderr
ALN/1/3
ALN/1/3/stdout
ALN/1/3/stderr
```

Example: Align two sets of oligos using primer3/ntdpal, save the result in a structured output, under the NTDPAL directory .

```
 parallel --result NTDPAL  primer3-2.3.5/src/ntdpal {1} {2} g \
        ::: AATCGTACGTACG ATAGCATCGA \
        ::: AATCGTACGTCG  ATAGCATCGAG
```

output:

```
$ find NTDPAL/
NTDPAL/
NTDPAL/1
NTDPAL/1/AATCGTACGTACG
NTDPAL/1/AATCGTACGTACG/2
NTDPAL/1/AATCGTACGTACG/2/ATAGCATCGAG
NTDPAL/1/AATCGTACGTACG/2/ATAGCATCGAG/stdout
NTDPAL/1/AATCGTACGTACG/2/ATAGCATCGAG/stderr
NTDPAL/1/AATCGTACGTACG/2/AATCGTACGTCG
NTDPAL/1/AATCGTACGTACG/2/AATCGTACGTCG/stdout
NTDPAL/1/AATCGTACGTACG/2/AATCGTACGTCG/stderr
NTDPAL/1/ATAGCATCGA
NTDPAL/1/ATAGCATCGA/2
NTDPAL/1/ATAGCATCGA/2/ATAGCATCGAG
NTDPAL/1/ATAGCATCGA/2/ATAGCATCGAG/stdout
NTDPAL/1/ATAGCATCGA/2/ATAGCATCGAG/stderr
NTDPAL/1/ATAGCATCGA/2/AATCGTACGTCG
NTDPAL/1/ATAGCATCGA/2/AATCGTACGTCG/stdout
NTDPAL/1/ATAGCATCGA/2/AATCGTACGTCG/stderr

$ cat NTDPAL/1/AATCGTACGTACG/2/ATAGCATCGAG/stdout
|AATCGTACGTACG|   |ATAGCATCGAG| g score=3.00 len=10 |1,0|2,1|3,2|4,3|5,4|6,5|7,7|8,8|10,9|12,10|
```

This is useful if you are running multiple variables.

Example: Align two sets of oligos using primer3/ntdpal, save the result in a structured output, under the NTDPAL directory, use the data headers for the directories names .

```
$ parallel --header : --result NTDPAL  primer3-2.3.5/src/ntdpal {1} {2} g \
        ::: primer-foward AATCGTACGTACG ATAGCATCGA \
        ::: primer-reverse AATCGTACGTCG  ATAGCATCGAG
```

Generated files:

```
$ find NTDPAL/

NTDPAL/
NTDPAL/primer-foward
NTDPAL/primer-foward/AATCGTACGTACG
NTDPAL/primer-foward/AATCGTACGTACG/primer-reverse
NTDPAL/primer-foward/AATCGTACGTACG/primer-reverse/ATAGCATCGAG
NTDPAL/primer-foward/AATCGTACGTACG/primer-reverse/ATAGCATCGAG/stdout
NTDPAL/primer-foward/AATCGTACGTACG/primer-reverse/ATAGCATCGAG/stderr
NTDPAL/primer-foward/AATCGTACGTACG/primer-reverse/AATCGTACGTCG
NTDPAL/primer-foward/AATCGTACGTACG/primer-reverse/AATCGTACGTCG/stdout
NTDPAL/primer-foward/AATCGTACGTACG/primer-reverse/AATCGTACGTCG/stderr
NTDPAL/primer-foward/ATAGCATCGA
NTDPAL/primer-foward/ATAGCATCGA/primer-reverse
NTDPAL/primer-foward/ATAGCATCGA/primer-reverse/ATAGCATCGAG
NTDPAL/primer-foward/ATAGCATCGA/primer-reverse/ATAGCATCGAG/stdout
NTDPAL/primer-foward/ATAGCATCGA/primer-reverse/ATAGCATCGAG/stderr
NTDPAL/primer-foward/ATAGCATCGA/primer-reverse/AATCGTACGTCG
NTDPAL/primer-foward/ATAGCATCGA/primer-reverse/AATCGTACGTCG/stdout
NTDPAL/primer-foward/ATAGCATCGA/primer-reverse/AATCGTACGTCG/stderr
```

The directories are named after the variables and their values.

5 Control the execution.

5.1 Number of simultaneous jobs.

The number of concurrent jobs is given with `'--jobs'`/`'-j'`. By default `'--jobs'` is the same as the number of CPU cores. `'--jobs 0'` will run as many jobs in parallel as possible.
Example: sort a set of BAMS, using two parallel jobs .

```
$ ls *.bam | parallel --verbose -j 2 samtools sort {} sorted_{.}
```

output:

```
samtools sort ex1a.bam sorted_ex1a
samtools sort ex1.bam sorted_ex1
samtools sort ex1b.bam sorted_ex1b
samtools sort ex1f.bam sorted_ex1f
samtools sort ex1f-rmduppe.bam sorted_ex1f-rmduppe
samtools sort ex1f-rmdupse.bam sorted_ex1f-rmdupse
samtools sort toy.bam sorted_toy
```

6 Interactiveness.

Example: sort a set of BAMS, prompt user for confirmation .

```
$ ls *.bam | parallel --verbose --interactive  samtools sort {} sorted_{.}
```

output:

```
samtools sort ex1a.bam sorted_ex1a ?...y
samtools sort ex1.bam sorted_ex1 ?...n
samtools sort ex1b.bam sorted_ex1b ?...n
samtools sort ex1f.bam sorted_ex1f ?...n
samtools sort ex1f-rmduppe.bam sorted_ex1f-rmduppe ?...y
samtools sort ex1f-rmdupse.bam sorted_ex1f-rmdupse ?...y
samtools sort toy.bam sorted_toy ?...y

$ ls sorted*.bam
sorted_ex1a.bam
sorted_ex1f-rmduppe.bam
sorted_ex1f-rmdupse.bam
sorted_toy.bam
```

6.1 Timing

Some jobs do heavy I/O when they start. To avoid a thundering herd GNU **parallel** can delay starting new jobs. `'--delay X'` will make sure there is at least X seconds between each start.
Example: sort a set of BAMS, using two parallel jobs, wait 0.2 seconds between each jobs .

```
ls *.bam | parallel -j 2 --verbose --delay 2.0 samtools sort {} sorted_{.}
```

output:

```
samtools sort ex1a.bam sorted_ex1a
samtools sort ex1.bam sorted_ex1
samtools sort ex1b.bam sorted_ex1b
samtools sort ex1f.bam sorted_ex1f
samtools sort ex1f-rmduppe.bam sorted_ex1f-rmduppe
samtools sort ex1f-rmdupse.bam sorted_ex1f-rmdupse
samtools sort toy.bam sorted_toy
```

If jobs taking more than a certain amount of time are known to fail, they can be stopped with `'--timeout'`.
Example: index a set of sorted BAMS, cancel is the job takes more than 5 seconds .

```
ls ex1f-rmdupse_sorted.bam ~/tmp/BIGBAM/sorted_.bam ex1a_sorted.bam |\
        parallel --timeout 5.0 --verbose samtools index {}
samtools index ex1a_sorted.bam
samtools index ex1f-rmdupse_sorted.bam
samtools index /home/lindenb/tmp/BIGBAM/sorted_.bam #no bam.bai
```

Based on the runtime of completed jobs GNU **parallel** can estimate the total runtime with `'--eta'`.
Example: index a set of sorted BAMS, print an estimation of the total runtime .

```
$ ls ex1f-rmdupse_sorted.bam BIGBAM/sorted_.bam ex1a_sorted.bam |\
        parallel --eta --verbose samtools index {}
```

output:

```
samtools index ex1a_sorted.bam
samtools index ex1f-rmdupse_sorted.bam


Computers / CPU cores / Max jobs to run
1:local / 2 / 2

Computer:jobs running/jobs completed/%of started jobs/Average seconds to complete
local:2/0/100%/0.0s samtools index BIGBAM/sorted_.bam
ETA: 28s 0left 19.00avg  local:0/3/100%/19.0s
```

6.2 Progress

GNU **parallel** can give progress information with `'--progress'`.
Example: index a set of sorted BAMS, print the progress .

```
$ ls *_sorted.bam BIGBAM/sorted_.bam  |\
   parallel --progress  samtools index {}
```

output:

```
Computers / CPU cores / Max jobs to run
1:local / 2 / 2

Computer:jobs running/jobs completed/%of started jobs/Average seconds to complete
local:0/8/100%/7.2s
```

A logfile of the jobs completed so far can be generated with `'--joblog'`.
Example: index a set of sorted BAMS, log the jobs into the file 'log.txt' .

```
$ ls *_sorted.bam BIGBAM/sorted_.bam  |\
   parallel --joblog log.txt  samtools index {}

$ cat log.txt
Seq     Host    Starttime       Runtime Send    Receive Exitval Signal  Command
1       :       1381155497.299  0.002   0       0       0       0       samtools index ex1a_sorted.bam
2       :       1381155497.301  0.004   0       0       0       0       samtools index ex1b_sorted.bam
3       :       1381155497.306  0.006   0       0       0       0       samtools index ex1f-rmduppe_sorted.bam
4       :       1381155497.313  0.006   0       0       0       0       samtools index ex1f-rmdupse_sorted.bam
5       :       1381155497.319  0.009   0       0       0       0       samtools index ex1f_sorted.bam
6       :       1381155497.326  0.006   0       0       0       0       samtools index ex1_sorted.bam
8       :       1381155497.334  0.004   0       0       0       0       samtools index toy_sorted.bam
7       :       1381155497.332  59.246  0       0       0       0       samtools index BIGBAM/sorted_.bam
```

The log contains the job sequence, which host the job was run on, the start time and run time, how much data was transferred if the job was run on a remote host, the exit value, the signal that killed the job, and finally the command being run.

Same command with `'--timeout'`.
Example: index a set of sorted BAMS with timeout=5 sec, log the jobs into the file 'log.txt', BIGBAM/-sorted_.bam fails .

```
$ ls *_sorted.bam BIGBAM/sorted_.bam  |\
    parallel --timeout 5.0 --joblog log.txt  samtools index {}

$ cat log.txt
Seq      Host     Starttime        Runtime Send     Receive Exitval Signal   Command
1        :        1381155638.224   0.002    0        0       0       0        samtools index ex1a_sorted.bam
2        :        1381155638.225   0.008    0        0       0       0        samtools index ex1b_sorted.bam
3        :        1381155638.233   0.006    0        0       0       0        samtools index ex1f-rmduppe_sorted.bam
4        :        1381155638.235   0.009    0        0       0       0        samtools index ex1f-rmdupse_sorted.bam
5        :        1381155638.243   0.007    0        0       0       0        samtools index ex1f_sorted.bam
6        :        1381155638.248   0.008    0        0       0       0        samtools index ex1_sorted.bam
8        :        1381155638.258   0.003    0        0       0       0        samtools index toy_sorted.bam
7        :        1381155638.255   6.188    0        0       -1      15       samtools index BIGBAM/sorted_.bam
```

With a joblog GNU **parallel** can be stopped and later pickup where it left off. It it important that the input of the completed jobs is unchanged.

```
$ parallel --joblog log.txt  bwa aln -f 0{}_R.fastq.sai toy.fa 0{}_R_.fastq.gz  ::: 1 2
$ cat log.txt
Seq      Host     Starttime        Runtime Send     Receive Exitval Signal   Command
1        :        1381156165.018   0.012    0        0       0       0        bwa aln -f 01_R.fastq.sai toy.fa 01_R_.
     fastq.gz
2        :        1381156165.028   0.016    0        0       0       0        bwa aln -f 02_R.fastq.sai toy.fa 02_R_.
     fastq.gz
```

With a joblog GNU **parallel** can be stopped and later pickup where it left off. It it important that the input of the completed jobs is unchanged. **Example:** Resume the previous command: create a BAM for the remaining files DOESNTEXIST 4 5, 'Starttime' doesn't change for 1 and 2 .

```
$  parallel --resume --joblog log.txt  \
 bwa aln -f 0{}_R.fastq.sai toy.fa 0{}_R_.fastq.gz  ::: 1 2 3 DOESNTEXIST 4 5
$ cat log.txt

Seq      Host     Starttime        Runtime Send     Receive Exitval Signal   Command
1        :        1381156165.018   0.012    0        0       0       0        bwa aln -f 01_R.fastq.sai toy.fa 01_R_.
     fastq.gz
2        :        1381156165.028   0.016    0        0       0       0        bwa aln -f 02_R.fastq.sai toy.fa 02_R_.
     fastq.gz
3        :        1381156215.782   0.014    0        0       0       0        bwa aln -f 03_R.fastq.sai toy.fa 03_R_.
     fastq.gz
5        :        1381156215.800   0.026    0        0       0       0        bwa aln -f 04_R.fastq.sai toy.fa 04_R_.
     fastq.gz
6        :        1381156215.830   0.014    0        0       0       0        bwa aln -f 05_R.fastq.sai toy.fa 05_R_.
     fastq.gz
4        :        1381156215.791   0.139    0        0       0       6        bwa aln -f 0DOESNTEXIST_R.fastq.sai toy.fa
     0DOESNTEXIST_R_.fastq.gz
```

With `'--resume-failed GNU parallel will re-run the jobs that failed'` **Example:** Align some FASTQs with 'bwa aln', but a FASTQ doesn't exist .

```
$ parallel --resume-failed --verbose --joblog log.txt  bwa aln -f 0{}_R.fastq.sai toy.fa 0{}_R_.fastq.gz  ::: 1 2 3
      DOESNTEXIST 4 5
bwa aln -f 0DOESNTEXIST_R.fastq.sai toy.fa 0DOESNTEXIST_R_.fastq.gz
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
[bwa_aln] 190bp reads: max_diff = 8
[bwa_aln] 225bp reads: max_diff = 9
[bwa_seq_open] fail to open file '0DOESNTEXIST_R_.fastq.gz'. Abort!
```

now create the file 0DOESNTEXIST_R_.fastq.gz

```
$ cp 01_R_.fastq.gz 0DOESNTEXIST_R_.fastq.gz
```

and re-run the command. Only one command is run.

```
$ parallel --resume-failed --verbose --joblog log.txt  bwa aln -f 0{}_R.fastq.sai toy.fa 0{}_R_.fastq.gz  ::: 1 2 3
      DOESNTEXIST 4 5
bwa aln -f 0DOESNTEXIST_R.fastq.sai toy.fa 0DOESNTEXIST_R_.fastq.gz
[bwa_aln] 17bp reads: max_diff = 2
[bwa_aln] 38bp reads: max_diff = 3
[bwa_aln] 64bp reads: max_diff = 4
[bwa_aln] 93bp reads: max_diff = 5
[bwa_aln] 124bp reads: max_diff = 6
[bwa_aln] 157bp reads: max_diff = 7
```

```
[bwa_aln] 190bp reads: max_diff = 8
[bwa_aln] 225bp reads: max_diff = 9
[bwa_aln_core] calculate SA coordinate... 0.01 sec
[bwa_aln_core] write to the disk... 0.00 sec
[bwa_aln_core] 1000 sequences have been processed.
[main] Version: 0.7.4-r385
[main] CMD: bwa aln -f 0DOESNTEXIST_R.fastq.sai toy.fa 0DOESNTEXIST_R_.fastq.gz
[main] Real time: 0.014 sec; CPU: 0.012 sec
```

6.3 Termination

todo

6.4 Limiting the ressources

todo

7 Remote execution

7.1 Sshlogin

(on remote side, add parallel to the PATH if needed in .bashrc)

```
PATH=${PATH}:/commun/data/packages/parallel/bin
```

The most basic sshlogin is `'-S host'`.
Example: print four bases on the remote server .

```
$ parallel -S user@host echo ::: A T G C
A
T
C
G
```

The special sshlogin `':'` is the local machine.
Example: print four bases on the remote server .

```
$ parallel -S : echo ::: A T G C
A
T
G
C
```

If ssh is not in `'$PATH'` it can be prepended to `'$SERVER1'`.
Example: print four bases on the remote server using "/usr/bin/ssh" .

```
$ parallel -S '/usr/bin/ssh 'user@host echo ::: A T G C
A
T
C
G
```

Several servers can be given using multiple `'-S'`.
Example: print four bases using two remote servers .

```
$ parallel -S user@host1 -S user@host2 echo ::: A T G C
A
T
C
G
```

Or they can be separated by ','.
Example: print four bases using two remote servers .

```
$ parallel -S user@host1,user@host2 echo ::: A T G C
A
T
C
G
```

The can also be read from a file (replace user@ with the user on $SERVER2).
Example: print four bases using two remote servers .

```
$ echo "user@host1" > nodefile
$ echo "4//usr/bin/ssh/_user@host2" >> nodefile
$ parallel --sshloginfile nodefile echo ::: A T G C
A
T
G
C
```

The special `--sshloginfile '..'` reads from `/.parallel/sshloginfile`.

7.2 Transferring files

GNU **parallel** can transfer the files to be processed to the remote host. It does that using rsync.
Example: copy the BAMs on the remote server .

```
$ parallel -S user@host --transfer file ::: *.bam
```

output

```
parallel: Warning: ssh to user@host only allows for 10 simultaneous logins.
You may raise this by changing /etc/ssh/sshd_config:MaxStartup on user@host.
Using only 9 connections to avoid race conditions.
ex1f-rmdupse.bam: gzip compressed data, extra field
ex1b.bam: gzip compressed data, extra field
ex1f-rmduppe.bam: gzip compressed data, extra field
ex1f.bam: gzip compressed data, extra field
ex1.bam: gzip compressed data, extra field
ex1a.bam: gzip compressed data, extra field
toy.bam: gzip compressed data, extra field
```

If the files is processed into another file, the resulting file can be transferred back.
Example: copy the BAMs on the remote server, sort them with samtools, fetch the sorted bam .

```
$ parallel -S user@host --transfer  --return {/.}_s.bam samtools sort {} {/.}_s ::: ex1f.bam  ex1.bam toy.bam
```

output

```
$ ls -la
-rw-rw-r-- 1 lindenb lindenb 207931 Oct  8 11:28 ex1f_s.bam
-rw-rw-r-- 1 lindenb lindenb 126522 Oct  8 11:28 ex1_s.bam
-rw-rw-r-- 1 lindenb lindenb    502 Oct  8 11:28 toy_s.bam
```

To remove the input and output file on the remote server use `'--cleanup'`.
Example: copy the BAMs on the remote server, sort them with samtools, fetch the sorted bam, cleanup on server side .

```
$ parallel -S user@host --transfer  --cleanup  --return {/.}_s.bam samtools sort {} {/.}_s ::: ex1f.bam  ex1.bam
    toy.bam
```

There is a short hand for `'--transfer'` `'--return'` `'--cleanup'` called `'--trc'`.
Example: copy the BAMs on the remote server, sort them with samtools, fetch the sorted bam, cleanup on server side .

```
$ parallel -S user@host --trc {/.}_s.bam samtools sort {} {/.}_s ::: ex1f.bam  ex1.bam toy.bam
```

Some jobs need a common database for all jobs. GNU **parallel** can transfer that using `'--basefile'` which will transfer the file before the first job.
Example: transfert the file ex1.fa on a remote server, grep three oligos .

```
$ parallel -S user@host --basefile ex1.fa grep -inH {} ex1.fa  :::  GCCTGGCT CCAGCT ATCACC
```

output

```
ex1.fa:3:GTGGACCCTGCAGCCTGGCTGTGGGGGCCGCAGTGGCTGAGGGGTGCAGAGCCGAGTCAC
ex1.fa:9:CTTCTTCCAAAGATGAAACGCGTAACTGCGCTCTCATTCACTCCAGCTCCCTGTCACCCA
ex1.fa:11:AGCCCAGCTCCAGATTGCTTGTGGTCTGACAGGCTGCAACTGTGAGCCATCACAATGAAC
ex1.fa:14:CATCCCTGTCTTACTTCCAGCTCCCCAGAGGGAAAGCTTTCAACGCTTCTAGCCATTTCT
ex1.fa:23:TTGGGCTGTAATGATGCCCCTTGGCCATCACCCAGTCCCTGCCCCATCTCTTGTAATCTC
```

7.3 Working dir

The default working dir on the remote machines is the login dir. This can be changed with `'--workdir'` mydir.

Files transferred using `'--transfer'` and `'--return'` will be relative to mydir on remote computers, and the command will be executed in the dir mydir.

Example: copy the BAMs on the remote server, print the working directory, sort the BAMs with samtools, fetch the sorted bam, cleanup on server side. Use the login directory .

```
$ parallel --workdir . -S user@host --trc {/.}_s.bam  pwd "&&" samtools sort {} {/.}_s  :::  ex1f.bam  ex1.bam toy.
  bam
/home/user/package/samtools-0.1.18/examples
/home/user/package/samtools-0.1.18/examples
/home/user/package/samtools-0.1.18/examples
```

The special mydir value `'...'` will create working dirs under `' /.parallel/tmp/'` on the remote computers. If `'--cleanup'` is given these dirs will be removed.

Example: copy the BAMs on the remote server, print the working directory, sort the BAMs with samtools, fetch the sorted bam, cleanup on server side. Use the ' /.parallel/tmp/' directory .

```
$ parallel --workdir ... -S user@host --trc {/.}_s.bam  pwd "&&" samtools sort {} {/.}_s  :::  ex1f.bam  ex1.bam toy.
  bam
/home/lindenb/.parallel/tmp/hardyweinberg-10672-2
/home/lindenb/.parallel/tmp/hardyweinberg-10672-3
/home/lindenb/.parallel/tmp/hardyweinberg-10672-1
```

Example: copy the BAMs on the remote server, print the working directory, sort the BAMs with samtools, fetch the sorted bam, cleanup on server side. Use the ' /tmp/' directory .

```
$ parallel --workdir /home/user/tmp -S user@host --trc {/.}_s.bam  pwd "&&" samtools sort {} {/.}_s  :::  ex1f.bam
    ex1.bam toy.bam
/home/user/tmp
/home/user/tmp
/home/user/tmp
```

7.4 Avoid overloading sshd

If many jobs are started on the same server, sshd can be overloaded. GNU **parallel** can insert a delay between each job run on the same server.

Example: copy the BAMs on the remote server, sort the BAMs with samtools, fetch the sorted bam, cleanup on server side. Take five seconds between each call to sshd .

```
$ parallel -S user@host --sshdelay 5 --trc {/.}_s.bam  date "&&"samtools sort {} {/.}_s  :::  ex1f.bam  ex1.bam toy.
  bam
Tue Oct  8 12:26:45 CEST 2013
Tue Oct  8 12:26:50 CEST 2013
Tue Oct  8 12:26:55 CEST 2013
```

Sshd will be less overloaded if using `'--controlmaster'`, which will multiplex ssh connections.
Example: copy the BAMs on the remote server, sort the BAMs with samtools, fetch the sorted bam, cleanup on server side. Use `'--controlmaster'` .

```
$ parallel -S user@host --controlmaster  --trc {/.}_s.bam  samtools sort {} {/.}_s ::: ex1f.bam  ex1.bam toy.bam
```

7.5 Ignore hosts that are down

In clusters with many hosts a few of the are often down. GNU **parallel** can ignore those hosts. In this case the host nowhere.com is down.
Example: print combinations of bases on a remote set of servers even if one server is down .

```
$ parallel --filter-hosts  -S user@host,user@nowhere.com echo ::: A T G  ::: A T G
```

output

```
A A
A T
A G
T A
T T
T G
G A
G T
G G
```

7.6 Transfer environment variables and functions

Using `'--env'` GNU **parallel** can transfer an environment variable to the remote system.
Example: export the REFERENCE variable and extract some subsequences from the FASTA file .

```
$ export REFERENCE=/path/to/human_g1k_v37.fasta
$ parallel --env REFERENCE -S user@host -k "samtools faidx ${REFERENCE} {}" ::: "1:10000010-10000020" "MT:20-30" "
    3:1000050-1000080"
>1:10000010-10000020
CTACAATAAAT
>3:1000050-1000080
AAAAGCCCATCAAGGTTGTAAGAAGACTCCC
>MT:20-30
TATTAACCACT
```

This works for functions too.
Example: create and export a function to align two oligos with primer3/ntdpal, and then, align some combinations of oligos on a remote server .

```
$  align2primer() {
primer3-2.3.5/src/ntdpal $1 $2 g
}

$  export -f align2primer
$ parallel --env align2primer -S  user@host align2primer \
        ::: ACTGACGACTG ATCGATGACTAG  \
        ::: TGACGACTG TCGATGACT
```

output

```
|ACTGACGACTG|   |TGACGACTG| g score=9.00 len=9 |2,0|3,1|4,2|5,3|6,4|7,5|8,6|9,7|10,8|
|ACTGACGACTG|   |TCGATGACT| g score=5.00 len=8 |2,0|3,2|4,3|5,4|6,5|7,6|8,7|9,8|
|ATCGATGACTAG|   |TCGATGACT| g score=9.00 len=9 |1,0|2,1|3,2|4,3|5,4|6,5|7,6|8,7|9,8|
|ATCGATGACTAG|   |TGACGACTG| g score=5.00 len=9 |1,0|3,1|4,2|5,3|6,4|7,5|8,6|9,7|11,8|
```

GNU **parallel** can copy all defined variables and functions to the remote system. It just need to record which ones to ignore in `' /.parallel/ignored_vars'`. Do that by running this once:

```
$ parallel --record-env

$ cat ~/.parallel/ignored_vars
XAUTHORITY
XDG_CURRENT_DESKTOP
UBUNTU_MENUPROXY
LC_COLLATE
XDG_SEAT_PATH
MANDATORY_PATH
(...)
```

Now all new variables and functions defined will be copied when using `'--env _'`

8 –pipe

8.1 Chunk size

By default GNU **parallel** will start an instance of command_B, read a chunk of about 1 MB, and pass that to the instance. Then start another instance, read another chunk, and pass that to the second instance.
Example: count some chunks of SAM records .

```
$ samtools view  file.bam | parallel --pipe wc -l | head
6310
6347
6328
6337
6378
6302
6354
6352
6306
6334
```

ou can change the block size to 2 MB with `'--block'`.
Example: count some chunks of SAM records .

```
$ samtools view  file.bam | parallel--block 2M  --pipe wc -l | head
12657
12665
12680
12706
12640
12579
12600
12677
12643
12654
```

8.2 Records

Using `'-N400'` GNU **parallel** will read 400 records at a time.

```
$ gunzip -c  examples/0*.fastq.gz | parallel --pipe -N400 "paste␣-␣-␣-␣-␣|␣cut␣-f␣2␣|␣sort␣|␣uniq␣-dc␣"
      2 AATTGGGGAAAACCTCTTTAGTCTTGCTAGAGATTTAGACATCTAAATGAAAGAGGCTCAAAGAATGCCA
      2 GGAAATAAAGTCAAGTCTTTCCTGACAAGCAAATGCTAAGATAATTCATCATCACTAAACCAGTCCTATA
      2 GAAAAAAATTCTAAAATCAGCAAGAGAAAAGCATACAGTCATCTATAAAGGAAATCCCATCAGAATAACA
      2 ATGAACTAACTATATGCTGTTTACAAGAAACTCATTAATAAAGACATGAGTTCAGGTAAAGGGGTGGAAA
      2 AATTGGGGAAAACCTCTTTAGTCTTGCTAGAGATTTAGACATCTAAATGAAAGAGGCTCAAAGAATGCCA
      2 GGAAATAAAGTCAAGTCTTTCCTGACAAGCAAATGCTAAGATAATTCATCATCACTAAACCAGTCCTATA
      2 GAAAAAAATTCTAAAATCAGCAAGAGAAAAGCATACAGTCATCTATAAAGGAAATCCCATCAGAATAACA
      2 ATGAACTAACTATATGCTGTTTACAAGAAACTCATTAATAAAGACATGAGTTCAGGTAAAGGGGTGGAAA
      2 AATTGGGGAAAACCTCTTTAGTCTTGCTAGAGATTTAGACATCTAAATGAAAGAGGCTCAAAGAATGCCA
      2 GGAAATAAAGTCAAGTCTTTCCTGACAAGCAAATGCTAAGATAATTCATCATCACTAAACCAGTCCTATA
      2 GAAAAAAATTCTAAAATCAGCAAGAGAAAAGCATACAGTCATCTATAAAGGAAATCCCATCAGAATAACA
      2 ATGAACTAACTATATGCTGTTTACAAGAAACTCATTAATAAAGACATGAGTTCAGGTAAAGGGGTGGAAA
      2 AATTGGGGAAAACCTCTTTAGTCTTGCTAGAGATTTAGACATCTAAATGAAAGAGGCTCAAAGAATGCCA
      2 GGAAATAAAGTCAAGTCTTTCCTGACAAGCAAATGCTAAGATAATTCATCATCACTAAACCAGTCCTATA
      2 GAAAAAAATTCTAAAATCAGCAAGAGAAAAGCATACAGTCATCTATAAAGGAAATCCCATCAGAATAACA
      2 ATGAACTAACTATATGCTGTTTACAAGAAACTCATTAATAAAGACATGAGTTCAGGTAAAGGGGTGGAAA
      2 AATTGGGGAAAACCTCTTTAGTCTTGCTAGAGATTTAGACATCTAAATGAAAGAGGCTCAAAGAATGCCA
```

```
       2 GGAAATAAAGTCAAGTCTTTCCTGACAAGCAAATGCTAAGATAATTCATCATCACTAAACCAGTCCTATA
       2 GAAAAAAATTCTAAAATCAGCAAGAGAAAAGCATACAGTCATCTATAAAGGAAATCCCATCAGAATAACA
       2 ATGAACTAACTATATGCTGTTTACAAGAAACTCATTAATAAAGACATGAGTTCAGGTAAAGGGGTGGAAA
```

If a record is 75 lines -L can be used:

```
$ gunzip -c  examples/0*.fastq.gz | parallel --pipe -L4 "paste -- - - - - | cut -f 2 | sort | uniq -dc "
       4 AAAAAAAAAAAAAAAAAGAGAAAAGAAAAAAAAAAAAAACGAAATTACCAAATTGTTTTCCAAAGTGGAAA
       2 AAAAAAAAAAAAAAAAAGAGAAAAGAAAAAAAAAAAAAACGAAATTAGCAAATTGTTTTCCAAAGTGGCAAA
       2 AAAAAAAAAAAAAACGAAATTACCAAATTGTTTACCAAAGTGGAAAACAATTTATACTCCCACTAGCAATA
       2 AAAAAAAAAAAAAAGAGAAAAGAAAAAAAAAAAAAACGAAATTACCAAATTGTTTTCCAAAGTGGAAAACAA
       2 AAAAAAAAAAAAACGAAATTACCAAATTGTTTTCCAAAGTGGAAAACAATTTATACTCGCACTAGCAATAT
       2 AAAAAAAAAACGAAATTACCAAATTGTTTTCCAAAGTGGAAAACAATTTATACTCCCACTAGCAATATTT
       4 AAAAAAAAAAGAGAAAAGAAAAAAAAAAAAAACGAAATTACCAAATTGTTTTCCAAAGTGGAAAACAATTT
       2 AAAAAAAAACAAAAAGAGAAAATAAAAAAAAAAAAAACGAAATTACCAAATTGTTTTCCAAAGTGGAAAAC
       2 AAAAAAAAACGAAATTACCAAATTGTTTTCCAAAGTGGACAACAATTTATACTCCCACTAGCAATATTTG
       2 AAAAAAAAAGAGAAAAGAAAAAAAAAAAAAACGAAATTACCAAAATGTTTTCCAAAGTGGAAAACAATTTA
(...)
```

8.3 Header

If the input data has a header, the header can be repeated for each job by matching the header with '--header'. If headers start with '#'.
Example: split the ouput of samtools view but re-use the headers (starting with @) before piping and counting .

```
view -h path/big.bam  | parallel --header '(@.*\n)*' --pipe 'samtools view -f4 -Sc - ' 2> /dev/null
```

output:

```
10996
5496
5510
5611
5431
5549
5566
5431
5510
5392
```

8.4 Shebang

GNU Parallel is often called as:

```
  cat input_file | parallel command
```

With '--shebang' the input_file and parallel can be combined into the same script.
UNIX-scripts start with a shebang line like:

```
#!/bin/bash
```

GNU **parallel** can do that, too. With '--shebang' the arguments can be listed in the file. The parallel command is the first line of the script.
Example: index a set of sorted BAM files using samtools index and '--shebang' .

```
#!/usr/local/bin/parallel --shebang --verbose -r samtools index
examples/sorted_ex1a.bam
examples/sorted_ex1.bam
examples/sorted_ex1b.bam
examples/sorted_ex1f.bam
examples/sorted_ex1f-rmduppe.bam
examples/sorted_ex1f-rmdupse.bam
examples/sorted_toy.bam
```

Execute:

```
$ ./parallelindex

samtools index examples/sorted_ex1a.bam
samtools index examples/sorted_ex1.bam
samtools index examples/sorted_ex1b.bam
samtools index examples/sorted_ex1f.bam
samtools index examples/sorted_ex1f-rmduppe.bam
samtools index examples/sorted_ex1f-rmdupse.bam
samtools index examples/sorted_toy.ba
```

9 References

- O. Tange (2011): GNU Parallel - The Command-Line Power Tool, ;login: The USENIX Magazine, February 2011:42-47.
- Biostars.org : Tool: GNU Parallel - parallelize serial command line programs without changing them