

GNU parallel

Gnu.org의 공식 문서와 튜토리얼, GNU Parallel 2018 참고

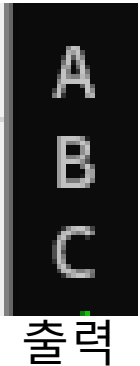
Input을 파일, 커맨드, stdin으로 받을 수 있다.

-하나의 input

Command line input

```
$ parallel echo ::: A B C
```

A, B, C 문자를 입력으로 echo를 병렬로 실행시키는 예제

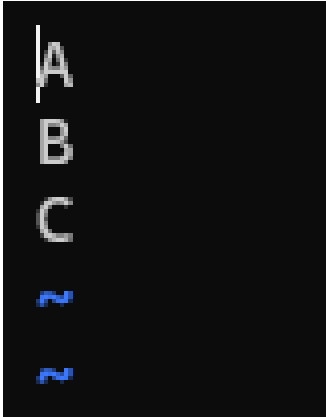


출력

File input (-a 파일이름)

```
parallel -a abc-file echo
```

Abc-file에서 데이터를 가져와 input으로 echo 병렬 실행



Abc-file 내용

Stdin input

```
cat abc-file | parallel echo
```

여러 개의 input

여러 개의 input source가 들어오면 GNU는 가능한 모든 조합을 생성

```
parallel echo ::: A B C ::: D E F
```

```
parallel -a abc-file -a def-file echo
```



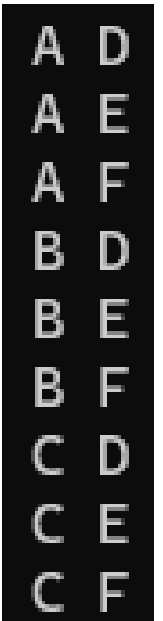
D
E
F
~
~

Def-file 내용

Stdin

```
cat abc-file | parallel -a - -a def-file echo
```

'-' 을 하나의 input으로 해서 동작 할 수 있다.



A D
A E
A F
B D
B E
B F
C D
C E
C F

실행 내용

-a 대신 ::: 사용

-a 옵션 대신 ::: 을 사용할 수 있다.

```
parallel echo :::: abc-file def-file
```

::: 과 ::: 를 함께 사용 가능

A	D
A	E
A	F
B	D
B	E
B	F
C	D
C	E
C	F

--link

가능한 모든 조합을 생성하는 것이 아니라 하나 씩 쌍을 이루게 하는것

```
parallel --link echo ::: A B C ::: D E F
```

A	D
B	E
C	F

Link 시 데이터의 개수가 다른 경우

적은 개수의 데이터를 반복시킨다.

```
parallel --link echo ::: A B C D E ::: F G
```

```
A F  
B G  
C F  
D G  
E F
```

F와 G가 반복되는 것을 볼 수 있다.

:::+ 와 :::+

앞쪽의 input source와 link 시키는 표현

::+, :::+

```
parallel echo :::: abc-file :::+ G H I :::: def-file
```

A G D
A G E
A G F
B H D
B H E
B H F
C I D
C I E
C I F

Abc-file 과 G H I 가 link 하고 그 link되 데이터와 def-file의 모든 가능한 경우를 생성

```
parallel echo :::: abc-file ::: G H I ::::+ def-file
```

def-file 과 G H I 가 link 하고 그 link되 데이터와 abc-file의 모든 가능한 경우를 생성

데이터의 개수가 다른 경우

개수가 많은 쪽의 데이터는 무시된다.

```
parallel echo ::: A B C D E :::+ F G
```

A F
B G

--arg-sep, --arg-file-sep

⋮, ⋮⋮ 대신에 원하는 구분자를 사용할 수 있다.

--arg-sep 원하는구분자: ⋮ 대신 원하는 구분자 사용

--arg-file-sep 원하는구분자: ⋮⋮ 대신 원하는 구분자를 설정

```
parallel --arg-sep ,, --arg-file-sep // echo ,, A B C // def-file
```

⋮ 대신 ,, 을

⋮⋮ 대신 // 을 사용하는 예제

--arg-sep ;; , --arg-sep " 등 다양하게 사용 가능

A	D
A	E
A	F
B	D
B	E
B	F
C	D
C	E
C	F

record delimiter (-d)

Gnu에서는 input source의 데이터를 \n 으로 구분한다.
-d 옵션을 이용하여 이 구분 문자를 추가할 수 있다.

```
parallel -d _ echo :::: abc_-file
```

```
A  
B  
C
```

```
parallel echo :::: abc_-file
```

```
A_B_C_
```

Abc_-file 내용

```
A_B_C_  
~  
~
```


Stop (-E)

특정 값을 만나면 멈추는 명령

```
parallel -E E echo ::: A B C D E F G
```

A

-E E 로 E를 만나면 그만

B

C

결과에서 A B C D 만 나오고 E 이후의 데이터는 사용하지 않는다.

D

Empty line skipping (--no-run-if-empty)

Empty인 값은 skip하는 명령

```
(echo 1; echo; echo 2) | parallel --no-run-if-empty echo
```

1

2

--no-run-fi-empty 없이 실행

1

2

공백 까지 출력된 것을 알 수 있다.

Command가 없는 경우

Arguments 가 command

```
parallel ::: ls 'echo foo' pwd
```

```
abc-file  
abc0-file  
abc_-file  
def-file  
fixedlen  
num1000000  
num128  
num30000  
num8  
num_%header  
tsv-file.tsv  
foo  
/home/dudcks/gnu_study
```

명령 ls, echo foo, pwd 가 각각 실행된 것을 알 수 있다.

Replacement Strings

```
parallel echo {} ::: A/B.C
```

1. {}: 확장자 제거 == --extensionreplace

```
A/B
```

2. {/}: path를 remove == --basenamereplace

```
B.C
```

3. {/}/: path만 keep == --dirnamereplcae

```
A
```

4. {/.}: path 와 확장자 제거 == --bner/ --basenameextensionreplace

```
B
```

5. {#} job number을 부여 == --seqreplce

```
parallel echo {#} ::: A B C
```

```
1  
2  
3
```

6. {%} job slot number 부여 == --slotreplace

```
parallel -j 2 echo {%} ::: A B C
```

```
1  
2  
1
```

-j number: 동시에 처리할 최대 작업 개수

-j 2: 최대 2개의 작업을 동시에 처리

Positional replacement strings

구분된 input sources 가 있으면 {number} 로 접근가능

```
parallel echo {1} AND {2} ::: A B ::: C D
```

```
A and C
```

```
A and D
```

```
B and C
```

```
B and D
```

string replacement와 함께 사용 가능

```
{3}
```

```
{3.}
```

```
{3//}
```

```
...
```

```
...
```

columns (--colsep)

tsv와 같이 column을 구분하기 위해서는 --colsep ' ' 을 사용하여 구분

f1	f2
A	B
C	D

tsv-file.tsv 내용

```
parallel --colsep '\t' echo 1={1} 2={2} ::: tsv-file.tsv
```

```
1=f1 2=f2  
1=A 2=B  
1=C 2=D
```

-d 'Wt ' 를 사용해 보았으나 제대로 구분하지 못하였음

header (--header :)

각 input의 첫 번째 값을 이름으로 해서 값을 구분

```
parallel --header : echo 1={f1} 2={f2} ::: f1 A B ::: f2 C D
```

f1 에 A B가 f2 에 C D가 있는 효과로 {f1}, {f2} 이런 식으로 원하는 데이터만 접근 가능하다.

{ } 만 사용 가능하며 다른 replacement({ } , {/} ...)은 사용할 수 없다.

-header 뒤 : 의 개수는 상관 없다.

```
parallel --header ::: echo 1={f1} 2={f2} ::: f1 A B ::: f2 C D
```

```
parallel --header : echo 1={f1} 2={f2} ::: f1 A B ::: f2 C D
```

More pre-defined replacement strings (--plus)

--plus 옵션으로 더 많은 replacement strings 가능

{..}: 최종 두 개의 확장자를 제외한 전체 경로

{...}: 최종 세 개의 확장자를 제외한 전체

--plus 옵션을 사용하면 {} 확장에 추가로 +가 붙은 새로운 패턴을 사용할 수 있습니다.

{+/.}: 디렉토리 경로 (dir/sub)

{+.}: 최종 한 개의 확장자

{+..}: 최종 두 개의 확장자

{+...}: 최종 세 개의 확장자

{##}: job의 총 개수

EX) dir/sub/file.ex1.ex2.ex3

```
parallel --plus echo {+}/{/} ::: dir/sub/file.ex1.ex2.ex3
```

{+}/{/}은 dir 부분 : dir/sub

{/}은 파일 이름: file.ex1.ex2.ex3

```
dir/sub/file.ex1.ex2.ex3
```

```
parallel --plus echo {.}.{+.} ::: dir/sub/file.ex1.ex2.ex3
```

{.}은 마지막 확장자 1개 제외한 나머지: dir/sub/file.ex1.ex2

{+.}은 최종 한 개의 확장자: ex3

```
dir/sub/file.ex1.ex2.ex3
```

Insert more than one argument (--xargs)

한 줄에 가능한 많은 인수를 넣어 명령어를 실행. 옵션을 사용하면 GNU Parallel이 입력에서 가능한 한 많은 인수를 수집, 명령어 한 줄로 최대한 많은 작업을 처리

```
cat num30000 | parallel --xargs 'echo {} | wc -w'
```

Num30000은 1부터 30000의 수가 적힌 파일
Wc -w 는 단어의 개수를 세는 작업

```
12439
10588
6973
```

총 30000개의 단어가 있다.
3개의 명령어로 나누어져 처리되었다.

-s 는 한 줄의 최대 길이를 제한하는 옵션

```
6218
5628
4997
4997
4997
3163
```

더 많은 명령어로 나누어져 실행된 것을
알 수 있다.

```
cat num30000 | parallel --xargs -s 30000 'echo {} | wc -w'
```

더 나은 병렬 처리

```
cat num30000 | parallel -j 4 -m 'echo {}' | wc -w
```

12439

10588

1744

1744

1744

1741

-m 은 균등하게 작업을 나누어 실행

세 번째 작업을 생성할 때에 마지막 인수를 읽게 되고 이를 4개의 작업으로 분리하여 병렬작업을 하게 된다.

12439

10588

6973

앞에서 -j와 -m을 사용하지 않은 것과 차이가 나는 모습

Trim space from arguments (--trim)

--trim을 사용하면 인수의 공백을 제거할 수 있다.

오른쪽 공백 제거 --trim r

```
parallel --trim r echo pre-{}-post ::: ' A '
```

```
pre- A-post
```

왼쪽 공백 제거 --trim l

```
parallel --trim l echo pre-{}-post ::: ' A '
```

```
pre-A -post
```

양쪽 공백 제거 --trim lr

```
parallel --trim lr echo pre-{}-post ::: ' A '
```

Tag output (--tag)

--tag 옵션을 사용하여 출력에 인수를 접두사로 추가

```
parallel --tag echo foo-{} ::: A B C
```

A	foo-A
B	foo-B
C	foo-C

--tag는 --tagstring {}의 축약형

```
parallel --tagstring {}-bar echo foo-{} ::: A B C
```

A-bar	foo-A
B-bar	foo-B
C-bar	foo-C

다른 문자열로 접두사를 추가할 수 있다.