Gnu Parallel 조사

- 6개의 Major한 Area존재함.
- 챕터 4~ 9 + 10

Chap4. Input sources

- 가능한 input 파일로는 files, command line, stdin 총3가지가 있음.
- `-a`옵션으로 파일을 입력할 수 있음.
  - `parallel -a abc-file -a def-file echo`
  - `abc-file`과 `def-file`의 각 값들을 짝지어서 출력해줌.
- `:::+` 리스트 항목 조합
- `::::+` 파일 항목 조합
- `-arg-sep` 옵션을 통해 `:::` 의 형태를 바꿀 수 있다.
- `-arg-file-sep`옵션을 통해 `::::`의 형태를 바꿀 수 있다.
- `-d`옵션을 통해 파싱에 사용할 구분자를 지정할 수 있다.
- `-E`옵션을 통해 특정 입력을 만났을 때 멈출 수 있다.
- `--no-run-if-empty`옵션을 통해 빈 라인은 무시할 수 있다.

Chap5. Build the command line

- 만약 parallel이 주어지고 이후에 아무 것도 주어지지 않은채로 argument가 주어진다면 해당 명령어에 대한 인자로 자기 자신을 사용한다는 뜻이다.
- export 기능을 이용해 사용자 정의 스크립트를 사용할 수 있다.
  - `export -f func_name`을 통해 함수 사용을 알린다.
  - `parallel func_name ::: a b c`
- 만약 env_parallel을 사용한다면 alias도 사용가능하다.

Replace Strings

- `{}` → value 그대로 출력 -`mydir/mysub/myfile.mytxt`
  - `-I` 옵션을 통해 변경 가능
- `{.}` → `mydir/mysub/myfile`
  - `--extensionreplace` 옵션을 통해 형태 변경 가능
- `{/}` → `myfile.mytxt`
  - `--basenamereplace` 옵션을 통해 형태 변경 가능

- `{//}` → `mydir/subdir`
  - `--dirnamereplace` 옵션을 통해 형태 변경 가능
- `{/.}` → `myfile`
  - `--basenameextensionreplace` | `--bner` 옵션을 통해 형태 변경 가능
- `{#}` → job의 순서 숫자
  - `--seqreplace` 옵션을 통해 형태 변경 가능
- `{%}` → job 슬롯 숫자
  - `--slotreplace` 옵션을 통해 형태 변경 가능
- Perl expression을 통해 이미 존재하는 문자열을변경할수도 있음.
  - Perl Expression에 대해 알아보고 더 자세히 알아볼 것임.
- `{n}` → n번째 argument값을 가져온다는 것임. rust랑 비슷한 것 같음.
- `--colsep` 옵션을 이용하여 텍스트파일을 파싱하여 인풋으로서 사용할 수 있음.
- `--header` 첫번째 인자를 헤더로서 사용하는 방법이다.
  - `parallel --header : echo f1={f1} ::: f1 A B`

```
f1=A
f1=B
```

- `--plus` 옵션을 이용해 더 다양한 string replacement를 사용할 수 있음.
  - `{##}` total number of jobs
  - `{:-string}` 만약 argurment가 빈다면 string을 입력함.
  - `{:n}` n번째 문자부터 끝까지 출력함.
  - `{:n1:n2}` n1번째 문자부터 n2번째 문자까지 출력함.
  - `{#string}` string으로 시작하는 argument는 삭제함.
  - `{%string}` string으로 끝나는 argument는 삭제함.
  - `{/str1/str2}` str1은 str2로 바꿈
  - `{^str}` 만약 str(한 문자)로 시작하면 해당 단어를 대문자들로 바꿈.
  - `{^^str}` 만약 str(한 문자)가 포함되어 있으면 해당 단어를 대문자들로 바꿈
  - `{,str}` 만약 str(한 문자)로 시작하면 해당 단어를 소문자들로 바꿈.

    - `{,,str}` 만약 str(한 문자)가 포함되어 있으면 해당 단어를 소문자들로 바꿈.

Insert more than one arg

- 많은 argument를 넣는 경우 `--xargs` 옵션을 통해 한번에 입력해 줄 수 있다.
- 보통 최대한 많은 아규먼트를 합쳐서 넣어줌.
- `-S` 옵션을 이용하면 최댓값을 지정할 수 있어서 다음과 같이 적용할 수 있다.
- `cat num30000 | parallel --xargs -S 30000 'echo {} | wc -w'`
- `-m` 옵션과 `--jobs` 옵션을 통해 병행으로 작업을 할 수 있음.
    - 하지만 첫 작업은 균등하게 작업하지 않고 두 번째 작업부터 균등하게 작업하기 시작함.
- arguemnt에 제한을 두기위해서는 `-Nn`을 사용해야 한다. `n`개 만큼 반복하게 된다.
- 특정 문맥을 반복하기 위해서는 `-X`옵션을 사용한다. 이는 `-m`처럼 동작한다.
- `-N0`의 경우 하나의 arguement를 읽지만 사용하지는 않는다.

Quote the command line

- 커맨드 라인 인용하기
- perl 사용함.

Trim space from arguments

- python의 split 같은 것임.
- `--trim` 옵션을 추가하고 이후 `l` | `r`로 사용이 가능하다.
- `parallel --trim l echo pre-{}-post ::: ' A '` → pre-A -post
- `parallel --trim lr echo pre-{}-post ::: ' A '` → pre-A-post

Chap6. Control the output

Tag output

- 접두사로 argument를 기술하고 이후에 출력을 씀
  `parallel --tag echo foo-{} ::: A B C`

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

- `--tagstring {}` 형태로 사용할 수도 있음.
  `parallel --tagstring {}-bar echo foo-{} ::: A B C`

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

See what is being run

- 어떤 명령어가 실행되는지 확인하기 위해서는 `--dryrun` 을 사용하면 된다.
  `parallel --dryrun echo {} ::: A B C`

```
echo A
echo B
echo C
```

- `--verbose` 옵션을 사용하면 어떤 명령어를 실행했는지와 그 결과가 나옴.
  `parallel --verbose echo {} ::: A B C`

```
echo A
A
echo B
B
echo C
C
```