·····Slurm·····

·····Slurm··ID·····

Slurm job array ·····
····· TSV····· BAM ·····

·····

samples.tsv ····· ID · BAM ····· TAB ····· glob ·····
····· ID·····

```
from pathlib import Path
import csv, re

with open('samples.tsv', newline='') as source:
    rows = list(csv.reader(source, delimiter='\t'))
if not rows or any(len(row) != 2 for row in rows):
    raise SystemExit('·····')
ids = [row[0] for row in rows]
if len(set(ids)) != len(ids):
    raise SystemExit('··ID··')
checked = []
for sample_id, filename in rows:
    if not re.fullmatch(r'[A-Za-z0-9][A-Za-z0-9_.-]*', sample_id):
        raise SystemExit(f'····ID: {sample_id!r}')
    if any(char in filename for char in '\t\r\n'):
        raise SystemExit('BAM·····TAB···')
    path = Path(filename)
    if not path.is_absolute() or not path.is_file() or path.stat().st_size == 0:
        raise SystemExit(f'·····: {filename!r}')
    checked.append((sample_id, str(path)))
root = Path('array-run-001')
if root.exists() or root.is_symlink():
    raise SystemExit('·····')
(root / 'logs').mkdir(parents=True)
(root / 'results').mkdir()
with (root / 'plan.tsv').open('x', newline='') as out:
    csv.writer(out, delimiter='\t', lineterminator='\n').writerows(checked)
print(f'prepared {len(checked)} samples in {root}')
```

·· | ···· | ···

···· | ·1····· | ·Slurm·····0·····

···· | %N····· | ·····

·· | %A_%a····· | ·····

·· | ··ID·JobID·· | ·····

·····

··· array.sbatch····· BAM ····· ID ··· 1 ·····
·····

```
#!/usr/bin/env bash
set -euo pipefail
root="${1:?·····}"
[[ "$root" = /* ]] || exit 2
: "${SLURM_ARRAY_TASK_ID:?···Slurm····}"
: "${SLURM_ARRAY_TASK_COUNT:?·····}"
[[ "$SLURM_ARRAY_TASK_ID" =~ ^[1-9][0-9]*$ && "$SLURM_ARRAY_TASK_COUNT" =~ ^[1-9][0-9]*$ ]] || exit 2
(( SLURM_ARRAY_TASK_ID <= SLURM_ARRAY_TASK_COUNT )) || exit 2
cd "$root"
sha256sum -c plan.sha256
row=$(sed -n "${SLURM_ARRAY_TASK_ID}p" plan.tsv)
IFS=$'\t' read -r id bam extra <<< "$row"
[[ "$id" =~ ^[A-Za-z0-9][A-Za-z0-9_.-]*$ && -z "$extra" ]] || exit 2
test -s "$bam"
d="$root/results/$id/$SLURM_JOB_ID"
mkdir -p "$root/results/$id"; mkdir "$d"
printf 'sample=%s\narray_task=%s\njob=%s\ninput=%s\n' \
  "$id" "$SLURM_ARRAY_TASK_ID" "$SLURM_JOB_ID" "$bam" > "$d/metadata.txt"
samtools --version > "$d/version.txt"
samtools quickcheck -v "$bam"
/usr/bin/time -v -o "$d/time.txt" \
  samtools flagstat -O json "$bam" > "$d/flagstat.json" 2> "$d/run.log"
python - "$d/flagstat.json" <<'PY'
import json, sys
with open(sys.argv[1]) as source:
    result = json.load(source)
for key in ('QC-passed reads', 'QC-failed reads'):
    if not isinstance(result.get(key), dict):
        raise SystemExit(f'missing or invalid flagstat section: {key}')
PY
printf '%s\n' BASIC_BAM_QC_JSON_OK > "$d/status.txt"
```

····· BAM ····· flagstat JSON ·····
·····

……

LIMIT·MEM · WALL …… 1 · CPU…… Slurm CP
U ……Slurm ……

```
set -euo pipefail
: "${LIMIT:?……}" "${MEM:?……}"
: "${WALL:?……00:10:00}"
[[ "$LIMIT" =~ ^[1-9][0-9]*$ ]] || exit 2
python prepare_array.py
root="$PWD/array-run-001"
n=$(wc -l < "$root/plan.tsv")
test "$n" -ge 1
(cd "$root" && sha256sum plan.tsv > plan.sha256)
sbatch --version > "$root/slurm-version.txt"
sbatch --array="1-${n}%${LIMIT}" --ntasks=1 --cpus-per-task=1 \
  --mem="$MEM" --time="$WALL" --output="$root/logs/%A_%a.out" \
  --error="$root/logs/%A_%a.err" "$PWD/array.sbatch" "$root"
```

…… 1 …… LIMIT ……
…… MaxArraySize……

……

…… MEM · WALL ……reads ……
…… LIMIT ……
……

…… CPU……
…… ID ……

……

·· sbatch …… JobID…… squeue …… sacct … task ……JobID ····
…… SLURM_ARRAY_TASK_ID ……. metadata ……. ID ……
……

……
…… BAM ……
··

····

Slurm··Job Array…… https://slurm.schedmd.com/job_array.html

Slurm··sbatch…… https://slurm.schedmd.com/sbatch.html

samtools flagstat……JSON·· https://www.htslib.org/doc/samtools-flagstat.html