TP2

Objectif

Tester la présence d'un read particulier dans un ensemble de reads

Données

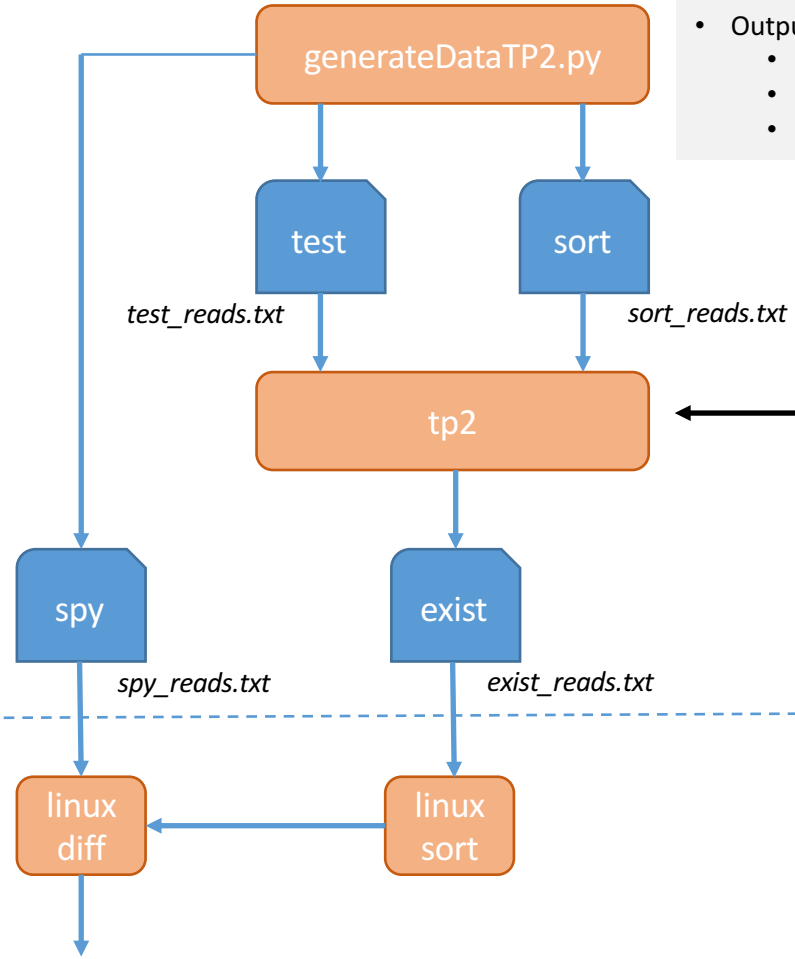
Résultat du TP1 : sort_read.txt

- Fichier de reads triés
- Taille des reads = 100 bp

TP 2

python **generateDataTP2.py** <sort_size> <test_size> <percentage>

- parameters
  - sort_size = number of reads in sort_reads.txt
  - test_size = number of reads in test_reads.txt
  - Percentage = percentage of reads in test_reads.txt that are in sort_reads.txt
- Output
  - Sort_reads.txt = list of sorted reads
  - Test_reads.txt = list of test reads
  - Spy_reads.txt = list of reads that are present both in test/sort files



Tp2 program output reads of test_reads.txt that are present in sort_reads.txt

Results are stored in exist_reads.txt

Protocol to check the correctness of tp2

NB: reads of spy_reads.txt are already sorted

TP2 : déroulement

1. Ecrire le programme tp2.c
   - en entrée :
     - sort_reads.txt (résultat TP1)
     - test_reads.txt : jeu de reads test. même format que sort_reads.txt
   - en sortie : exist_reads.txt : même format que sort_read.txt
   - **Objectif : minimiser l'empreinte mémoire**
2. Tester le programme sur des petits jeux de données
3. Mesurer les performances du programme avec les paramètres suivants
   - sort_reads.txt
     - généré avec génome de taille 10 Mbp et 30 Mbp. (couverture = 30)
   - test_read.txt
     - Nombre total de reads : 10^3, 10^5
     - Pourcentage de reads présents : 10%, 30%, 70%

12 situations

TP2

A rendre avant le 10/12/2019

1. Le code source avec (beaucoup de) commentaires
2. Explications sur la stratégie employée
3. Rapport sur le comportement du programme en fonction des données
   - Empreinte mémoire
   - Temps d'exécution
     - Temps de construction de la structure de données
     - Temps d'interrogation

Envoyez à lavenier@irisa.fr :

- Code source (en C)
- Le rapport (format pdf)