14/05/2014

Formation scripting

Cycle de formation ABiMS 2014

Alexandre CORMIER
Gwendoline ANDRES
Camille VACQUIE

IFB
INSTITUT FRANÇAIS DE BIOINFORMATIQUE

Pourquoi le scripting ?

- Automatisation de tâches répétitives
- Traitement de gros volumes de données
- Minimiser les risques d'erreurs

Les langages de scripting

- Shell : sh, ksh, bash, les C shells

- Langages interprétés de plus haut niveau
  - Perl
  - Python
  - Ruby

Pourquoi le BASH ?

- Langage natif: unix/linux et MacOS
- Simple et assez complet
- Rapide
- Moins d'apprentissage que pour Perl ou Python

Scripting Shell

- Regrouper et enchaîner un ensemble de commandes

Scripting Shell

- Regrouper et enchaîner un ensemble de commandes
- Avantages
  - Automatisation et planification
  - Gain en temps
  - Portable et simplicité
  - Fonctionne sur tous les Unix-Linux

Scripting Shell

- Regrouper et enchaîner un ensemble de commandes
- Avantages
  - Automatisation et planification
  - Gain en temps
  - Portable et simplicité
  - Fonctionne sur tous les Unix-Linux
- Inconvénients
  - Syntaxe
  - Messages d'erreurs
  - Relative lenteur par rapport à du C

Que mettre dans un script ?

- La première ligne obligatoire: chemin de l'interpréteur

  #!/bin/bash

- Des commentaires

  # Exemple de commentaire

- Des commandes et des variables
- Des fonctions
- Debugger un script

  #!/bin/bash –x

  set –x … set +x

  echo

Variables

- Locales : affectation locale au shell

```
MYSEQ=hsfau.fasta
NOM="Bacillus Subtilis"
```

  - Attention : pas d'espace autour du =
  - Bonne habitude = mettre des quotes ‘’

- Globales : elles sont propagées aux sous-processus

```
export MYSEQ=hsfau.fasta
```

- Référence au contenu d'une variable : $

```
echo $MYSEQ
echo ${MYSEQ}
```

Variables systèmes

- $HOME, $PATH, $PWD, $USER, $RANDOM…
- Variables SHELL

  $# : nombre de paramètres

  $0 : Le nom du programme en cours d'execution

  $1, $2…: les arguments passés en paramètres

  $? : code de retour

  $$: PID du shell

Variables systèmes

$# -> nombre de paramètres

Guillemets ou quotes !

- Avant évaluation d'un texte, le shell substitue les valeurs des variables.

Guillemets ou quotes !

- Avant évaluation d'un texte, le shell substitue les valeurs des variables.

- Doubles quotes ou guillemets:
  - Remplacement par le contenu

```
MYSEQ="seq1.fasta"
echo $MYSEQ       => seq1.fasta
```

Guillemets ou quotes !

- Avant évaluation d'un texte, le shell substitue les valeurs des variables.
- Doubles quotes ou guillemets:
  - Remplacement par le contenu

```
MYSEQ="seq1.fasta"
FILE="$MYSEQ"
echo $FILE        => seq1.fasta
```

- Simples quotes:
  - Pas d'évaluation et donc de remplacement

```
FILE='$MYSEQ'
echo $FILE        => $MYSEQ
```

Évaluation des commandes

- Stocker le résultat d'une commande dans une variable
- Backquote (Alt Gr + 7) ou () :

```
DATE=`date`
DATE=$(date)
```

Découpage de chaînes

- BASH dépendant
- Exemple :

```
CHAINE="FORMATIONSCRIPTING2014"
echo ${CHAINE:9:9}      => SCRIPTING
```

- Éliminer la fin d'une chaîne : %

```
SEQ="file.fasta"
echo ${SEQ%.*}   => file
```

Le branchement conditionnel

- Si condition VRAI => Action1
- Sinon => Rien ou Action2
- Entourer les variables de guillemets
- Espaces importants

```
if [ "$MYSEQ" = "hsfau.fasta" ]
then
    echo $MYSEQ
else
    echo "Not ok"
fi
```

Les tests possibles

- Sur fichier : -r, -w, -d, -f …
- Sur contenu de chaîne : -z …
- Sur valeurs numériques : entiers…
- Négation : !
- Et : -a
- Ou : -o
- …

Les opérateurs booléen

- -o ||: ou

```
[ $rep = oui –o $rep = yes ]
[ $rep = oui ] || [ $rep = yes ]
```

- -a &&: et

```
[ $rep = oui –a $confirmation = oui ]
[ $rep = oui ] && [ $confirmation = oui ]
```

- ! : négation

```
[ $USER != root ]
```

Les tests sur fichiers

- -d : répertoire existe
- -f : fichier existe
- -r : fichier existe et lisible
- -s : fichier existe et taille > 0
- -w : fichier existe et modifiable
- -x : fichier existe et exécutable
- f1 –nt f2 : vrai si f1 plus récent

Les tests sur chaîne

- -z c1 : vrai si taille nulle
- -n c1 : vrai si taille non nulle
- c1 == c2 : vrai si chaines sont égales
- c1 != c2 : vrai si chaines sont différentes

Les tests numériques

- x1 -eq x2 : vrai si égalité
- x1 -ne x2 : vrai si non égalité
- x1 -lt x2 : vrai si x1 inférieur
- x1 -le x2 : vrai si x1 inferieur ou égal
- x1 -gt x2 : vrai si x1 plus grand
- x1 -ge x2 : vrai si x1 plus grand ou égal

Passage de paramètres

- Arguments positionnels
  - $0, $1, $2

- Lecture avec ***read***

```
echo "Entrez le nom de la sequence"
read MYSEQ
```

Boucles

- Exécuter une suite d'instructions avec une variable parcourant une suite de valeurs

for *variable* **in** *expression*
do
    instructions
done

Boucles

```
REP=$(ls)

for FILE in $REP
do
    echo "Le nom du fichier est $FILE"
done
```

Bonnes Pratiques de Scripting

- Utiliser les variables
  - En début de script
  - Chemin de répertoire
- Commentaires
- Tester le code retour
- Envoyer un message en fin script

- Google

Excercices

1. Blast pas à pas

2. Parsing de fichiers d'annotation

3. Lancement automatique de jobs sur le cluster

1. Blast

1. **Boucle** : Ecrire un script *myblast.sh* qui lancera la commande ***blastall*** sur la banque *nr* pour chaque fichier de séquence. Les fichiers résultats du blast devront être nommé *protxx.seq.out*

blastall –p blastp –i input –d localisation-banque –o output

2. **Passage d'arguments** : Adapter le script afin de le rendre plus interactif. Passer le nom de la banque et le dossier d'input en argument à l'exécution

3. Afficher un message donnant le nom du script et le nom de la banque
   - **Test**: Si la banque est différente de nr ou uniprot, il faut afficher un message d'erreur et ne pas exécuter le blast
   - Même exercice mais le résoudre avec une instruction CASE
   - Ajouter l'envoi d'un email en fin de traitement.
   - En début de script tester l'existence du répertoire « resultat »

2. Qsub

Ecrire un script *multiHTSeq.sh* qui lancera un qsub pour la commande ***htseq-count*** sur un ensemble de fichiers de mapping (.bam).

- *Paramètre à passer en argument:*
  - *La file sur le cluster (long.q, short.q)*
  - *L'email de l'utilisateur*
  - *Le fichier gff*
- *Autres:*
  - *Vérifier que le nombre d'argument est correct, sinon afficher une aide*
  - *Le nom de chaque qsub: htseq-bam_file.sh | sans l'extension .bam*

samtools view -h bam_file | htseq-count -m intersection-nonempty -s no -t exon -i Parent - gff_file

Exemple de qsub: http://abims.sb-roscoff.fr/cluster

Ajouter vos script dans le $PATH

- Permet d'accéder de manière simple et rapide à un ensemble de scripts
- Modification du fichier .bashrc dans le dossier /home/user

```
SCRIPT="/projet/umr7139/ga/acormier/scripts/scripts/bin"
PATH=$PATH:$SCRIPT
```

```
$ ll /projet/umr7139/ga/acormier/scripts/scripts/bin
    total 0
    lrwxrwxrwx 1 acormier ga 15 avril  1 09:14 apsEsi -> ../utils/APS.sh
    lrwxrwxrwx 1 acormier ga 22 avril  1 11:35 apsSctg -> ../utils/sctgParser.py
    lrwxrwxrwx 1 acormier ga 24 avril  1 09:14 atomicHTSeq -> ../utils/atomic_HTSeq.py
    lrwxrwxrwx 1 acormier ga 28 avril 24 15:05 atomicHTSeqDev -> ../utils/atomic_HTSeq.dev.py
    lrwxrwxrwx 1 acormier ga 18 avril  1 09:15 gc4gff -> ../utils/GC4GFF.py
    lrwxrwxrwx 1 acormier ga 20 avril  1 09:17 splitBam -> ../utils/splitBam.sh
    lrwxrwxrwx 1 acormier ga 21 avril  1 09:15 teDensity -> ../utils/teDensity.py
```

Exemples


```
alias ..='cd ..'
alias ...='cd ../../'
alias ....='cd ../../../'
alias .....='cd ../../../../'
alias ......='cd ../../../../../'

alias space2tab="sed -i -r 's/^ *//;s/ {1,}/\t/g'"


extract () {
    if [ -f $1 ] ; then
      case $1 in
        *.tar.bz2)   tar xjf $1     ;;
        *.tar.gz)    tar xzf $1     ;;
        *.bz2)       bunzip2 $1     ;;
        *.rar)       unrar e $1     ;;
        *.gz)        gunzip $1      ;;
        *.tar)       tar xf $1      ;;
        *.tbz2)      tar xjf $1     ;;
        *.tgz)       tar xzf $1     ;;
        *.zip)       unzip $1       ;;
        *.Z)         uncompress $1  ;;
        *.7z)        7z x $1        ;;
        *)     echo "'$1' cannot be extracted via extract()" ;;
         esac
     else
         echo "'$1' is not a valid file"
     fi
}

alias cdfinal="cd /projet/umr7139/ga/acormier/finalresult/"
alias cdtmp="cd /scratch/umr8227/ga/acormier"
alias cdscript="cd /projet/umr7139/ga/acormier/scripts/scripts"
alias cdqsub="cd /projet/umr7139/ga/acormier/scripts/qsub"
```