Biogenouest

ifb
INSTITUT FRANÇAIS DE BIOINFORMATIQUE

EMBRC FRANCE
CENTRE NATIONAL DE RESSOURCES BIOLOGIQUES MARINES

OCEANOMICS

IDEALG
seaweed for the future

Abims 4

01/06/16

Système Unix Avancé

Formation 2016

Le Corguillé – 1.12

UPMC SORBONNE UNIVERSITÉS

cnrs

RAPPELS

Rappel | Kit de survie

ls	: lister le contenu d'un dossier
cd	: changer de répertoire
pwd	: où suis-je ?
cat	: afficher le contenu d'un fichier texte
less	: consulter le contenu d'un fichier texte long
head / tail	: afficher le début ou la fin d'un fichier texte
gedit	: éditer un fichier texte
file	: connaitre le type de fichier (texte, binaire, pdf ...)

Rappel | Kit de survie

mkdir : créer un dossier

rmdir : effacer un dossier vide

rm : effacer un fichier

rm -r : effacer un dossier et ses fichiers

cp : copier/coller

mv : couper/coller

Rappel | Kit de survie

cmd & : lancer une commande en mode background

cmd : mettre une commande en mode background

[ctrl]+[z]

bg

INTRODUCTION

Introduction

- Le but ici sera d'acquérir les commandes avancées pour la manipulation de fichiers

- Certains analyses bio informatiques/statistiques peuvent :
  - nécessiter des centaines de fichiers d'entrée
  - nécessiter des fichiers d'entrée volumineux : Ko, Mo, Go, To
  - produire des centaines de fichiers de sortie
  - produire des fichiers de sortie volumineux : Ko, Mo, Go, To

Introduction

- Certains fichiers trop volumineux ne peuvent souvent pas être ouverts dans un éditeur de texte classique.

  (pas assez de RAM disponible)

  - Donc certaines opérations simples se compliquent :
    - compter le nombre de fois où apparaît un caractère
    - substituer toutes les occurrences d'un mot par un autre
    - trier un tableau / extraire certaines colonnes d'un tableau
    - …

Introduction

- Autre problème :

  Manuellement,

  On peut effectuer 1 opération sur 1 fichier

  Mais on ne peut pas effectuer 1000 opérations sur 1 fichier

  ...

  Et 1000 x 1000 = ...

Récupérer les données sur Internet

- Récupération du jeu de données sur Internet

```
$ wget http://application.sb-roscoff.fr/download/fr2424/
abims/lecorguille/cours/linux.tgz
```

- Décompression de l'archive compressée

```
$ tar -zxvf linux2.tgz
```

- Ouvrir les pdf des cours : evince

```
$ evince *.pdf
```

Connection

- Connection
  - sur un noeud du cluster via l'ordonnanceur SGE

```
$ qlogin
```

LES OUTILS

Les outils | dos2unix

- Nécessité de conversion entre OS :
  - Unix termine chaque ligne par \n
  - MS-DOS par \r et \n

Ex d'erreur :

```
$ ./phyml-mpi-multi.sh
    -bash: ./phyml-mpi-multi.sh: /bin/sh^M: bad interpreter:
    No such file or directory
```

dos2unix

```
$ dos2unix phyml-mpi-multi.sh
```

* \n : saut de ligne [LF]
* \r : retour chariot [CR]

Les outils | Description

- Obtenir les statistiques d'un fichier
  - Nombre de lignes : wc -l

```
$ wc -l adress.csv
      6 adress.csv
```

  - Nombre de mots : wc -w

```
$ wc -w adress.csv
     76 adress.csv
```

  - Nombre de caractères : wc -m

```
$ wc -m adress.csv
     488 adress.csv
```

Les outils | Recherche

- Recherche les lignes contenant l'expression : grep

```
$ grep ">" insulin.fas
      >gi|163659904|ref|NM_000618.3| Homo sapiens insulin-like g
      >gi|163659900|ref|NM_001111284.1| Homo sapiens insulin-lik
$ grep -v ">" insulin.fas
      ATGTCGTACTAGTCGCATCGTAGTCGTAGCACTGATCTATCGTAGCTGCGTACGTATC
      …
```

- Comptage de lignes contenant un mot dans un fichier

```
$ grep ">" -c insulin.fas
      5
```

Les outils | Tableaux - cut

- Récupérer des colonnes dans un fichier : cut

```
$ cut -f 1 acteur.tab          # récupération de la colonne 1
      Chuck
      Sylvester
      Steven

$ cut -f "2,3" acteur.tab      # récupération des colonnes 2,3
      Norris       70
      Stallone     64
      Seagal       59
```

```
Chuck                Norris      70
Sylvester            Stallone    64
Steven               Seagal      59
```

L'âge des acteurs

Les outils | Tableaux - sort

- Trier un fichier tabulé : sort

```
$ sort -k 1,1 pop_ville.tab          # classement par la colonne 1
      Paris    4193031
      Roscoff  3705
      Tokyo    13010279

$ sort -k 2,2 pop_ville.tab          # classement par la colonne 2
      Tokyo    13010279
      Roscoff  3705
      Paris    4193031

$ sort -k 2,2 -n pop_ville.tab       # classement numérique
      Roscoff  3705
      Paris    4193031
      Tokyo    13010279
```

```
Roscoff     3705
Paris       4193031
Tokyo       13010279
```

Population des villes

Les outils | Tableaux - join

- Fusion de tableaux : join

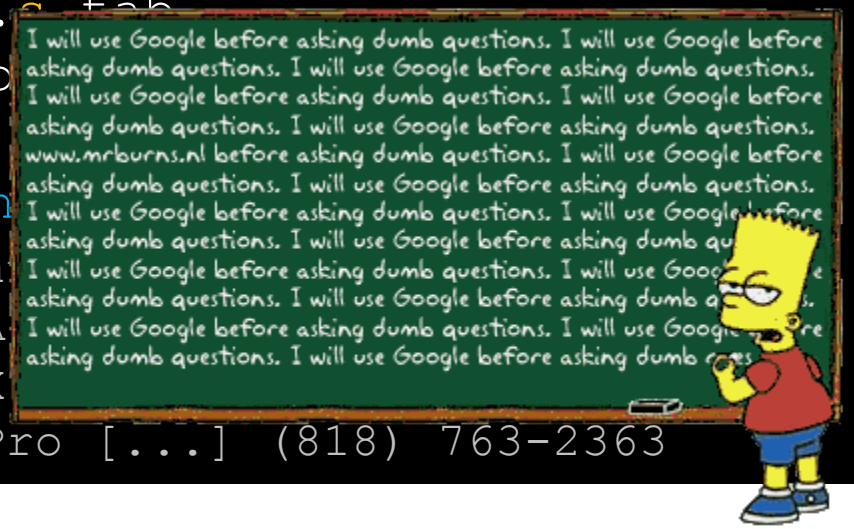
```
$ join -help
      […]
      Important: FILE1 and FILE2 must be sorted on the join fields.

$ sort -k 1,1 address.tab > address.s.tab
$ sort -k 1,1 phone_number.tab > phone_number.s.tab

$ join -i -1 1 -2 1 address.s.tab phone_number.s.tab
      Canet Guillaume Artmedia 20, Avenue [...] +33(0)1-43-17-33-00
      Li Jet The One Foundation Room 3A T [...] +86 (0)10-65568141
      Norris Chuck Box 872 Navasota, TX 7 [...] (424) 208-7321
      Stallone Sylvester Rogue Marble Pro [...] (818) 763-2363
```

```
Norris          Chuck       Box 872 Navasota, TX 77868      Li          +86 (0)10-65568141
Li              Jet         The One Foundation Room 3A      Canet       +33(0)1-43-17-33-00
Schwarzenegger  Arnold      Governor's Office S             Stallone    (818) 763-2363
Reno            Jean        c/o Gaumont 30, Avenue Char     Norris      (424) 208-7321
Stallone        Sylvester   Rogue Marbl
Canet           Guillaume   Artmedia 20, Avenue
```

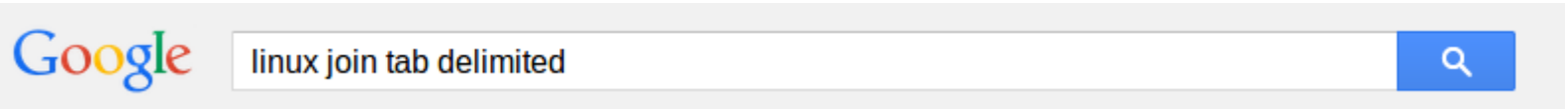
Les outils | Tableaux - join

- Fusion de tableaux : join

```
$ join -help
      […]
      Important: FILE1 and FILE2 must be sorted on the join fields.

$ sort -k 1,1 address.tab > address.s.tab
$ sort -k 1,1 phone_number.tab > pho

$ join -i -1 1 -2 1 address.s.tab ph
      Canet Guillaume Artmedia 20, Aven
      Li Jet The One Foundation Room 3A
      Norris Chuck Box 872 Navasota, TX
      Stallone Sylvester Rogue Marble Pro [...] (818) 763-2363
```



- Pour les fichiers tabulés



Les outils | Liste - uniq

- Récupérer un liste de terme : uniq

```
$ sort condition1.go > condition1.s.go
$ uniq condition1.s.go
      GO:0000166      nucleotide binding
      GO:0003824      catalytic activity
      GO:0005488      binding
```

- Compter les occurrences

```
$ uniq -c condition1.s.go
      2 GO:0000166      nucleotide binding
      1 GO:0003824      catalytic activity
      7 GO:0005488      binding
```

```
GO:0016301      kinase activity
GO:0005975      carbohydrate metabolic process
GO:0006091      generation of precursor metabolites and energy
GO:0009056      catabolic process
GO:0016787      hydrolase activity
GO:0005488      binding
GO:0005488      binding
GO:0016301      kinase activity
GO:0005975      carbohydrate metabolic process
GO:0006091      generation of precursor metabolites and energy
```

GO term exprimés dans la condition 1

Les outils | sed

- sed : subtitution

```
Chuck       Norris      70
Sylvester   Stallone    64
Steven      Seagal      59
```

```
$ sed "s/\t/;/g" acteur.tab
      Chuck;Norris;72
      Sylvester;Stallone;66
      Steven;Seagal;61:
```

  - s : mode substitution
  - en rouge : motif recherché
  - en vert : motif de remplacement
  - g : global pour ne pas s'arrêter à la première occurrence

Les outils | Rechercher / Remplacer

- sed …

afficher les lignes 10 à 12

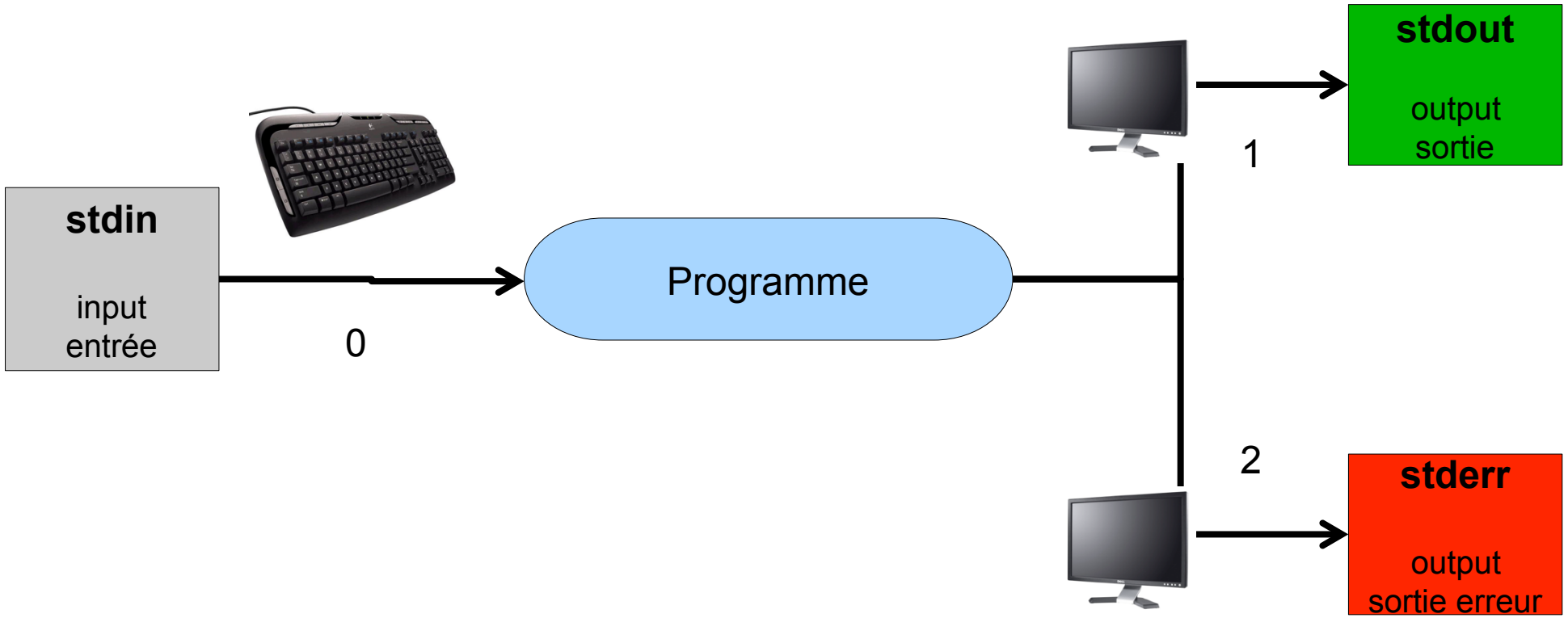
```
$ sed -n '10,12p' fichier
```

supprimer une ligne

```
$ sed "10d" fichier
```

REDIRECTION – LES FLUX

Redirection | Les flux



Redirection | Les flux

- Rediriger un flux consiste à écrire dans un fichier le résultat d'une commande plutôt que sur l'écran

  - Certains programmes proposent dans leurs options d'indiquer le fichier de sortie

```
$ blastn -query insulin.fas -db nt
      BLASTN 2.2.23+
      Reference: Stephen F. Altschul, Thomas L. Madden, Alejandro A.
      ...

$ blastn -query insulin.fas -db nt -out insulin_vs_nt.blast
```

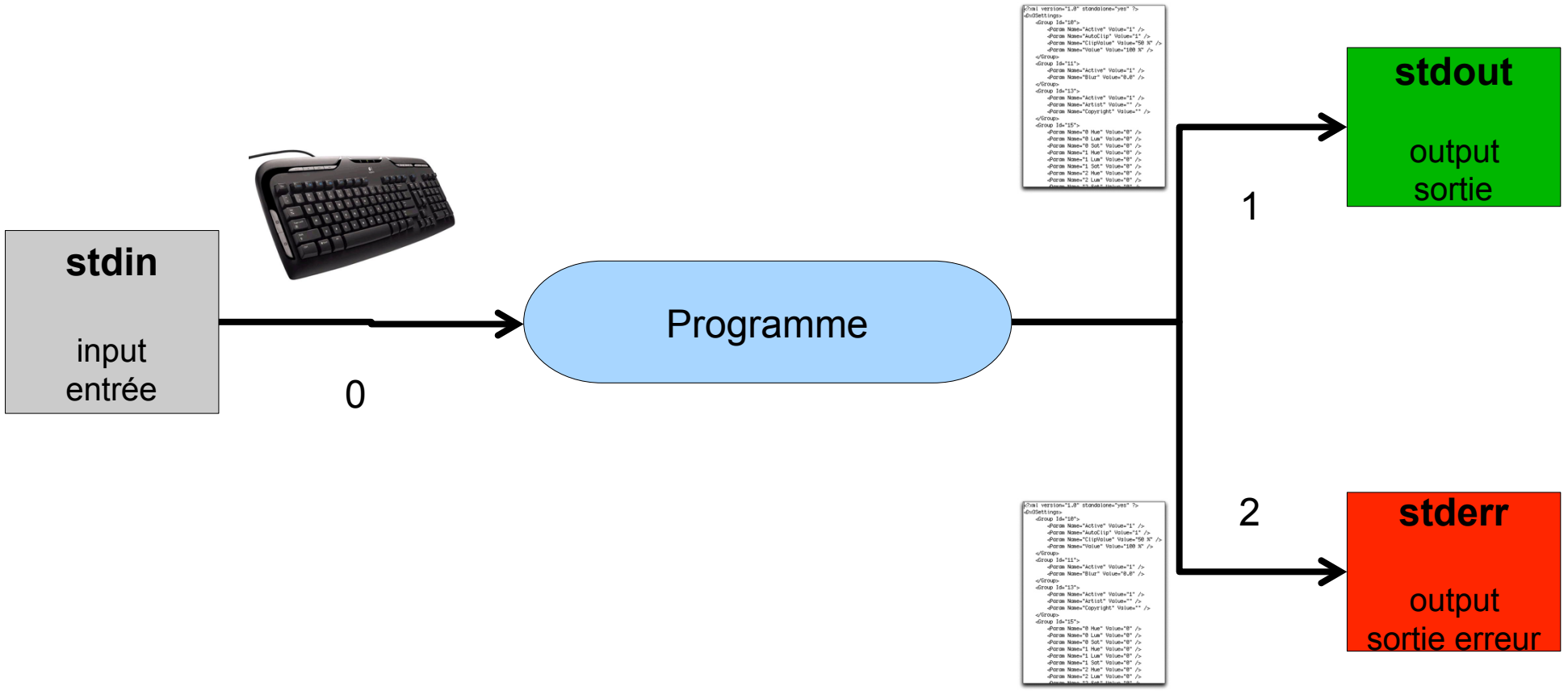
  - Mais pour d'autres, ça n'est pas la cas

```
$ ls
$ cut
$ sort
```

Redirection | Sorties

- Les sorties sont par défaut affichées sur l'écran

  => donc le but est de rediriger ce flux d'information dans des fichiers



Redirection | Sorties

- Sortie écran

```
$ infoseq nr.fsa -only -name -length -noheading
      Display basic information about sequences
      NP_268346.1    81
      XP_642131.1    169
      …
```

- Redirection vers des fichiers de stdout1

```
$ infoseq nr.fsa -only -name -length -noheading > nr.infoseq
      Display basic information about sequences
      Warning: Sequence 'fasta::nr.fsa:XP_635368.1' has zero length,
      Warning: Sequence 'fasta::nr.fsa:XP_720761.1' has zero length,
      …
```

- Redirection vers des fichiers de stderr2

```
$ infoseq nr.fsa -only -name -length -noheading 2> nr.infoseq.err
      NP_268346.1    81
      XP_642131.1    169
$ wc -l nr.infoseq.err
```

Redirection | Sorties

- Redirection de tous les flux

```
$ infoseq nr.fsa -only -name -length -noheading 2> nr.infoseq.err
1> nr.infoseq
      # 2 fichiers

$ infoseq nr.fsa -only -name -length -noheading >& nr.infoseq
      # 1 fichier

$ infoseq nr.fsa -only -name -length -noheading > nr.infoseq 2>&1
      # 1 fichier
```

Redirection | Sorties

- Exemples

```
$ cat insulin.fas
      >gi|163659904|ref|NM_000618.3| Homo sapiens insulin-like g
      TTTTGTAGATAAATGTGAGGATTTTCTCTAAATCCCTCTTCTGTTTGCTAAATCTCAC
      ATTCAGAGCAGATAGAGCCTGCGCAATGGAATAAAGTCCTCAAAATTGAAATGTGACA

$ cat *.fas
      >gi|163659904|ref|NM_000618.3| Homo sapiens insulin-like g
      TTTTGTAGATAAATGTGAGGATTTTCTCTAAATCCCTCTTCTGTTTGCTAAATCTCAC
      ATTCAGAGCAGATAGAGCCTGCGCAATGGAATAAAGTCCTCAAAATTGAAATGTGACA
      [...]

$ cat *.fas > allgenes.fas

$ cat aquaporin.fas >> allgenes.fas
                              # en mode append : c'est à dire
                              # qu'il n'écrase pas le fichier
                              # mais met au bout de celui-ci
```

Redirection | Sorties

- Récupérer des colonnes dans un fichier : cut

  Attention aux redirections qui vous feront perdre vos données

```
$ cut -f "1,2" tableau.tab > tableau.tab
$ cat tableau.tab
$
```

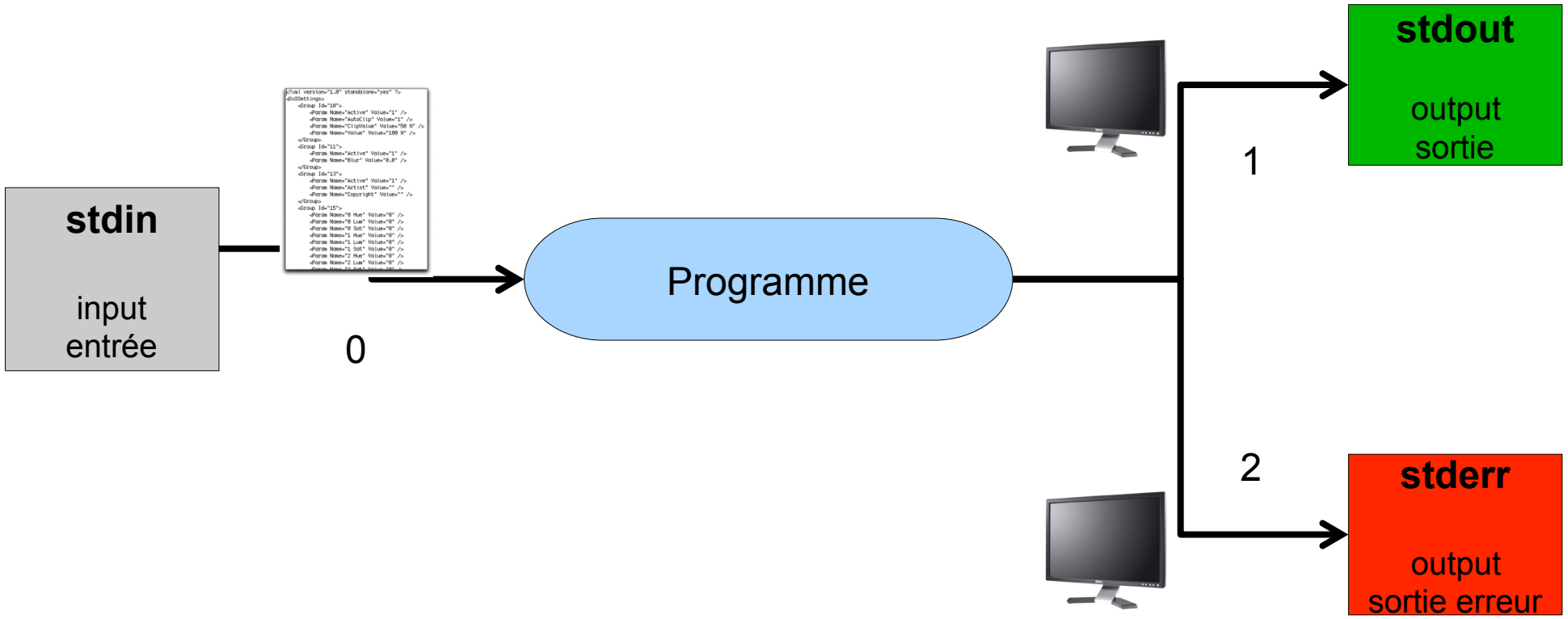
```
Chuck       Norris      70
Sylvester   Stallone    64
Steven      Seagal      59
```

L'âge des acteurs

Redirection | Entrée

- L'entrée est par défaut l'entrée clavier

  => donc le but est de rediriger un fichier vers le programme



Redirection | Entrée

- Entrée clavier

```
$ phyml
      . Enter the sequence file name >
```

- Entrée par fichier en argument

```
$ wc -l annuaire.csv
     311 annuaire.csv
$ tar -zxvf archive.tgz
```

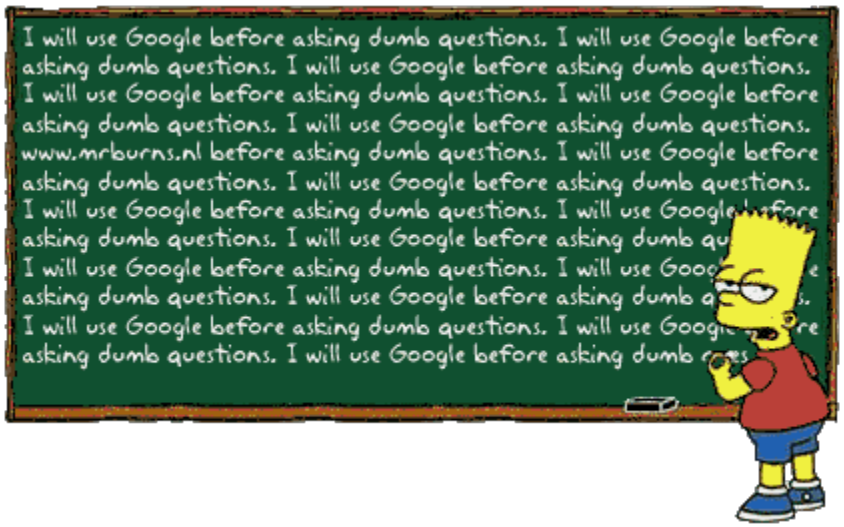
- Entrée par fichier par redirection

```
$ wc -l < annuaire.csv
     311
$ tar -zxv < archive.tgz
```

Redirection | Entrée

- Fusion de tableaux : join

```
$ join -i -1 1 -2 1 <(sort -k 1,1 address.tab)
<(sort -k 1,1 phone_number.tab)
      Canet Guillaume Artmedia 20, Avenue [...] +33(0)1-43-17-33-00
      Li Jet The One Foundation Room 3A T [...] +86 (0)10-65568141
      Norris Chuck Box 872 Navasota, TX 7 [...] (424) 208-7321
      Stallone Sylvester Rogue Marble Pro [...] (818) 763-2363
```



LANCEMENT SEQUENCIEL

Redirection | Lancement séquentiel

- Lancer plusieurs commandes en 1 fois : ;

```
$ muscle -in seq.fas -phyiout seq.phy ; phyml -i seq.phy -d aa
      seq 5 seqs, max length 7321, avg  length 7140
      00:00:00    10 MB(-2%)  Iter   1  100.00%  K-mer dist pass 1
      00:00:00    10 MB(-2%)  Iter   1  100.00%  K-mer dist pass 2
      …
      . Building BioNJ tree...
      . Maximizing likelihood (using NNI moves)...
```

- Avec condition de réussite : &&

```
$ muscle -in seq2.fas -phyiout seq2.phy ; phyml -i seq2.phy -d aa
      *** ERROR ***  Invalid file format, expected '>' to start FASTA label
      . The file 'seq2.phy' does not exist.

$ muscle -in seq2.fas -phyiout seq2.phy && phyml -i seq2.phy -d aa
      *** ERROR ***  Invalid file format, expected '>' to start FASTA label
```

- ATTENTION : un seul "&" implique des lancements en parallèle

```
$ muscle -in seq2.fas -phyiout seq2.phy & phyml -i seq2.phy -d aa
      . The file 'seq2.phy' does not exist.

      seq 5 seqs, max length 7321, avg  length 7140
```

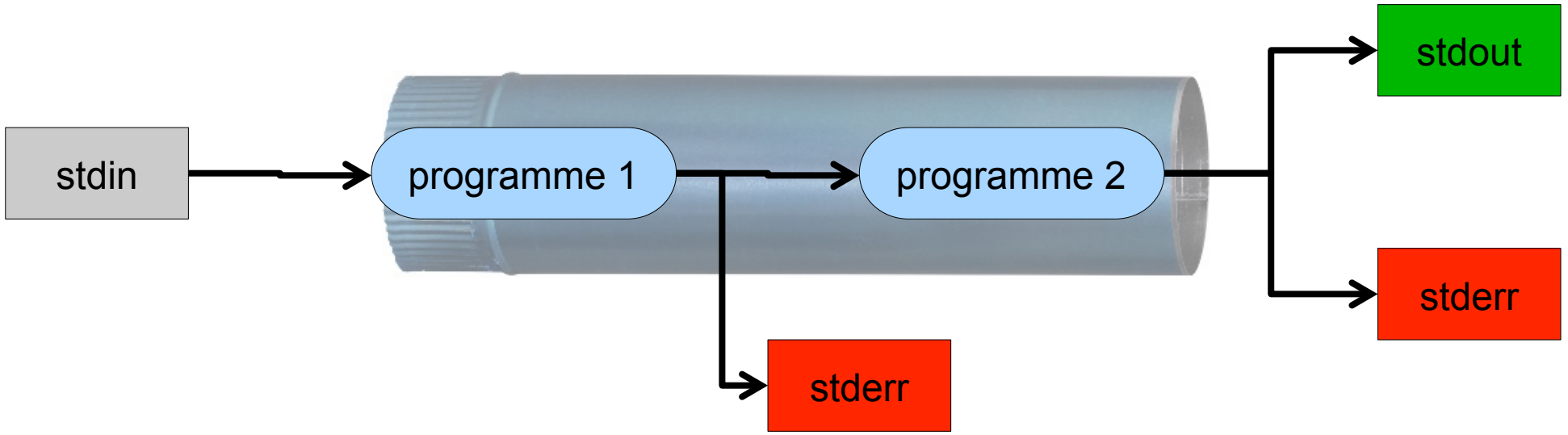
REDIRECTION – PIPE

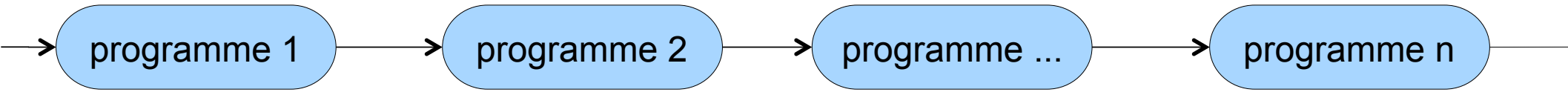
Redirection | Pipe

- Tuyau ou pipeline : pipe : |

Redirection | Pipe

- Tuyau ou pipeline : pipe : |





Redirection | Pipe

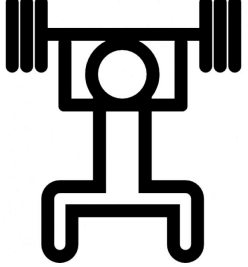
- Exemple 1

```
$ ls | wc -l          # une manière de compter le nombre de fichiers
      35
$ ls > /tmp/fichier.txt ; wc -l /tmp/fichier.txt; rm /tmp/fichier.txt
                      # version longue
```

- Exemple 2

```
$ ps -edf                              # rapide pour la consultation
$ ps -edf | tail                       # affiche les dernieres lignes

$ ps -edf | grep "lecorguille"         # filtrage en direct
```

Exercices



acteur.tab

```
Chuck       Norris      70
Sylvester   Stallone    64
Steven      Seagal      59
```

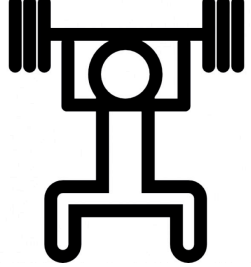
acteur.csv

```
Chuck;Norris;70
Sylvester;Stallone;64
Steven;Seagal;59
```

- Exercice 1 : acteur.csv
  - Récupérer le nom des acteurs

- Exercice 2 : acteur.csv
  - Classer les acteurs par âge décroissant

Exercices | Solutions



- Exercice 1

Corrections

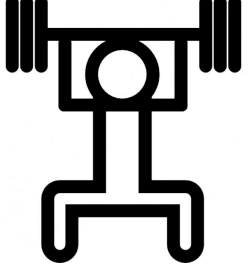
```
$ cut -d ";" -f 2 acteur.csv

                                # -d : séparateur de champs : ;
                                # -f : champs souhaités : 2

$ cut --delimiter=";" --fields=2 acteur.csv

                                # version plus jolie
```

- Exercice 2

Corrections

```
$ sort -t ";" -k 3,3 acteur.csv

                                # -t : séparateur de champs : ;
                                # -k : colonne visée : 3

$ sort --field-separator=";" --key=3,3 acteur.csv
```

Exercices

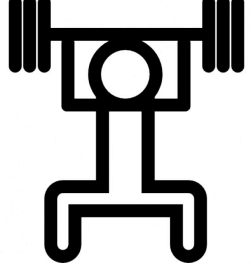


- Exercice 3 : annuaire.csv
  - Trier par le nom d'équipe (colonne 6)
  - Récupérer les colonnes nom (1), prénom (2), unité (5) et équipe (6)
  - Ne garder que les personnes de l'unité umr7144 (5)
  - Et mettre le tout dans un fichier nommé annuaire_umr7144.csv

  - **EN UNE LIGNE DE COMMANDE**

NOM	PRENOM	EMAIL	TELEPHONE	UNITE	EQUIPE
Le Corguille	Gildas	gildas.lecorguille{AT}sb-roscoff.fr	02 98 29 23 81	fr2424	service informatique et bioinformatique
Corre	Erwan	erwan.corre{AT}sb-roscoff.fr	02 98 29 23 81	fr2424	service informatique et bioinformatique

Exercices | Solutions

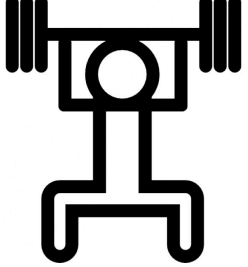


- Exercice 3

Corrections

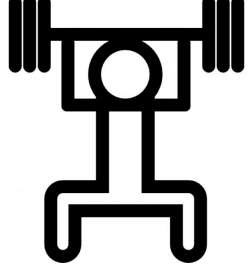
```
$ grep "umr7144" annuaire.csv | cut -f "1,2,5,6" -d ";" |
sort -k 4,4 -t ";" > annuaire_umr7144.csv

$ sort -k 6,6 -t ";" | cut -f "1,2,5,6" -d ";" |
grep "umr7144" annuaire.csv > annuaire_umr7144.csv

$ cut -f "1,2,5,6" -d ";" annuaire.csv | grep "umr7144" |
sort -k 4,4 -t ";" > annuaire_umr7144.csv

$ grep "umr7144" annuaire.csv | sort -key=6,6 -field-separator=";"
| cut --fields="1,2,5,6" --delimiter=";" > annuaire_umr7144.csv

$ . ..
```

Exercices



- Exercice 4 : condition2.go
  - Récupérer le numéro du GO le plus abondant dans la condition 2
  - **EN UNE LIGNE DE COMMANDE**

Exercices | Solutions



- Exercice 4

Corrections

```
$ sort condition1.go | uniq -c | sort -k 1,1 -n | tail -n 1 |
cut -f 1 | cut -f 2 -d ":"
```

```
$ sort condition1.go | uniq -c | sort -k 1,1 -rn | head -n 1 |
cut -f 1 | cut -f 2 -d ":"
```

EXPRESSION RÉGULIÈRE

Expression régulière | Définition

« Les expressions régulières (ou regexp) sont des chaînes de caractères que l'on appelle parfois motif et qui décrivent un ensemble de chaînes de caractères possibles selon une syntaxe précise. »

- Exemples en langage naturel :
  - chaine_de_caractere@chaine_de_caractere.chaine_de_charactere
  - 2xchiffre 2xchiffre 2xchiffre 2xchiffre 2xchiffre

Expression régulière | Les classes

[0-9]	Chiffre décimal
[a-z]	Lettre minuscule
[A-Z]	Lettre majuscule
[a-zA-Z]	Caractère alphabétique
[0-9a-zA-Z]	Caractère alphanumérique
[\t]	Espace blanc ou tabulation
.	N'importe quel caractère : joker
[^ATGC]	Tout sauf les lettres A,T,C et G

- Exemples en construction
  - [0-9a-zA-Z]@[0-9a-zA-Z].[a-zA-Z]
  - [0-9][0-9] [0-9][0-9] [0-9][0-9] [0-9][0-9] [0-9][0-9]

Expression régulière | Les répétitions

En grep / python / perl ...		En sed
?	répété 0 ou 1 fois	\ ?
+	répété de 1 à n fois	\+
*	répété de 0 à n fois	*
{2}	répété 2 fois	\{2\}
{2,5}	répété de 2 à 5 fois	\{2,5\}
{2,}	répété de 2 à n fois	\{2,\}

- Exemples en construction
  - [0-9a-zA-Z-_.]+@[0-9a-zA-Z-]+.[a-zA-Z]{2,3}
  - [0-9]{2} [0-9]{2} [0-9]{2} [0-9]{2} [0-9]{2}

Expression régulière | Caractères spéciaux

- ^ indique un début de ligne
- $ indique une fin de ligne
- \t tabulation
- \n saut de ligne sous linux et mac

Expression régulière | Caractères spéciaux

- \ caractere special car participe aux caracteres speciaux
  caractere d'echappement pour les caracteres speciaux

Syntaxe	Caractère texte
(	\(
[	\[
+	\+
|	|
$	\$

\t	t
\n	n

Expression régulière | Caractères spéciaux

- grep : rechercher
  - Par défaut :
    - + → \+
  - Recommandation :
    - l'option -P

-P, --perl-regexp

MOTIF est une expression régulière en Perl

- sed : rechercher / remplacer
  - Par défaut :
    - – + → \+
    - – { } → \{ \}
  - Recommandation :
    - l'option -r

-r, --regexp-extended

utiliser la syntaxe des expressions régulières étendues dans le script.

Exemples

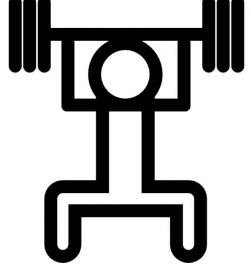
- Exemples complets
  - [0-9a-zA-Z\._\-]+@[0-9a-zA-Z]+\.[a-zA-Z]{2,3}

- En vrai

```
$ egrep --color "[0-9]{2} [0-9]{2} [0-9]{2} [0-9]{2} [0-9]{2}"
annuaire.csv
      Gaillard;Fanny;fanny.gaillard{AT}sb-roscoff.fr;02 98 29 23 89;fr2424;serv
      Houbin;Celine;celine.houbin{AT}sb-roscoff.fr;02 98 29 25 31;fr2424;servic
      Simon;Nathalie;nathalie.simon{AT}sb-roscoff.fr;02 98 29 25 34;umr7144;Div
```

  - egrep accepte des expressions régulières plus complexe que grep

Exercices



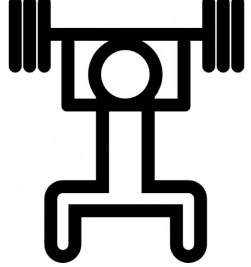
- Exercice 1

```
$ grep "1" --color patelles_roscoff.csv
        35,6    18,6    0
        36,7    13,1    0
        38,7    15,1    0
        37,4    16,4    0
        48,8    19,8    0
        43,9    17,1    1
```

  - Trouver toutes les patelles à la coque percée

- Exercice 2
  - Trouver toutes les personnes dont le nom est Thomas dans annuaire.csv
- Exercice 3
  - Trouver toutes les personnes dont le prénom est Thomas dans annuaire.csv

Exercices | Solutions



- Exercice 1

Corrections

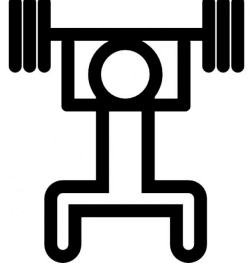
```
$ grep "1$" --color patelles_roscoff.csv          # fini pas un 1
      43,9    17,1    1
      42,8    15,8    1
      47,4    22,6    1
      21      7,1     1
```

- Exercice 2

Corrections

```
$ grep "^Thomas" --color annuaire.csv             # commence par Thomas
```

Exercices | Solutions



- Exercice 3

Corrections

```
$ grep ";Thomas" --color annuaire.csv

# il faut au moins 1 caractère avant Thomas

$ grep "^[a-zA-Z]\+;Thomas" --color annuaire.csv

$ grep -P "^[a-zA-Z]+;Thomas" --color annuaire.csv
```

Les outils | sed (le retour)

- sed : subtitution

```
Chuck       Norris      70
Sylvester   Stallone    64
Steven      Seagal      59
```

```
$ sed   's/\(.*\)\t\(.*\)\t\(.*\)/\2;\1;\3/g' acteur.tab

      Norris;Chuck;72
      Stallone;Sylvester;66
      Seagal;Steven;61
```

```
$ sed -r 's/(.*)\t(.*)\t(.*)/\2;\1;\3/g' acteur.tab

      Norris;Chuck;72
      Stallone;Sylvester;66
      Seagal;Steven;61
```

  - \(\) : groupe (sauce sed) sinon ()
  - \1 : référence arrière au premier groupe (sauce sed)

Les outils | sed (le retour)

- sed : subtitution

```
$ grep ">" nr.fsa | head -n 3
>gi|15674171|ref|NP_268346.1| 30S ribosomal protein S18 [Lactococcus lactis subsp. lactis Il
>gi|66816243|ref|XP_642131.1| hypothetical protein DDB_G0277827 [Dictyostelium discoideum AX
>gi|66818355|ref|XP_642837.1| hypothetical protein DDB_G0276911 [Dictyostelium discoideum AX

$ grep ">" nr.fsa |
sed 's/^>gi|.*|.*|\([A-Z]\{2\}_[0-9]*\.[0-9]*\)|.*\[\(.*\)\].*$/\1\t\2/'

NP_268346.1     Lactococcus lactis subsp. lactis Il1403
XP_642131.1     Dictyostelium discoideum AX4
XP_642837.1     Dictyostelium discoideum AX4

$ grep ">" nr.fsa |
sed 's/^>gi|.*|.*|\([A-Z]\{2\}_[0-9]*\.[0-9]*\)|.*\[\(.*\)\].*$/\1\t\2/'
```

  - \(\) : groupe (sauce sed)
  - \1 : référence arrière au premier groupe (sauce sed)
  - \{ \} : répétition (sauce sed)

http://www.grymoire.com/Unix/Sed.html

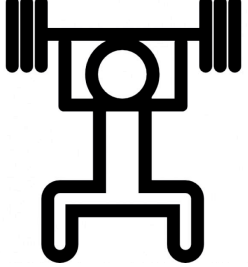
Les outils | sed (le retour)

- sed -r : subtitution

```
$ grep ">" nr.fsa | head -n 3
>gi|15674171|ref|NP_268346.1| 30S ribosomal protein S18 [Lactococcus lactis subsp. lactis Il
>gi|66816243|ref|XP_642131.1| hypothetical protein DDB_G0277827 [Dictyostelium discoideum AX
>gi|66818355|ref|XP_642837.1| hypothetical protein DDB_G0276911 [Dictyostelium discoideum AX

$ grep ">" nr.fsa |
sed 's/^>gi\|.*\|.*\|([A-Z]{2}_[0-9]*\.[0-9]*)\|.*\[(.*)\].*$/\1\t\2/'

NP_268346.1     Lactococcus lactis subsp. lactis Il1403
XP_642131.1     Dictyostelium discoideum AX4
XP_642837.1     Dictyostelium discoideum AX4

$ grep ">" nr.fsa |
sed -r 's/^>gi\|.*\|.*\|([A-Z]{2}_[0-9]*\.[0-9]*)\|.*\[(.*)\].*$/\1\t\2/'
```

  - ( ) : groupe (sauce sed -r)
  - \1 : référence arrière au premier groupe (sauce sed)
  - { } : répétition (sauce sed -r)

Exercices

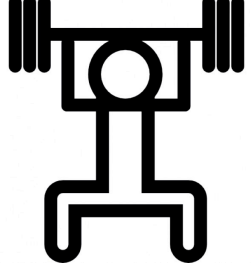


- Exercice 1 v2 : condition2.go
  - Récupérer le numéro du GO le plus abondant dans la condition 2
  - **EN UNE LIGNE DE COMMANDE**

Pour rappel

```
$ sort condition1.go | uniq -c | sort -k 1,1 -n | tail -n 1 |
cut -f 1 | cut -f 2 -d ":"
```

Exercices



- Exercice 1 : condition2.go

Pour rappel

```
$ sort condition1.go | uniq -c | sort -k 1,1 -n | tail -n 1 |
cut -f 1 | cut -f 2 -d ":"
```

Corrections

```
$ sort condition1.go | uniq -c | sort -k 1,1 -n | tail -n 1 |
sed -r "s/^.*GO:([0-9]{7}).*$/\1/"
```

AWK

Les outils | awk

- Outil et langage de programmation
- Traitement de fichier
  - Réorganisation
  - Filtrage
  - Calcul
  - Opérations complexes

- Utilisable sur un fichier ou en pipe
- Utilisation sur une seule ligne de commande ou via un fichier

Les outils | awk - Syntaxe

- Usage

```
awk ' ' fichier                 # le programme : ' '
awk '{ }' fichier               # un bloc : { }
```

```
$ cut -f "2,3" acteur.tab       # récupération des colonnes 2,3
      Norris 70
      Stallone    64
      Seagal 59
```

```
$ awk '{ print $2,$3 }' acteur.tab
      Norris 70
      Stallone    64
      Seagal 59
```

Les outils | awk - Syntaxe

- Notion de champs et imprimer

```
print $0            # imprime la ligne entière
print $1,$3         # imprime les champs 1 et 3 séparés par un espace

print "caracters;"$1";"$3";"$5+$7    # alternance chaine de caractère
```

```
$ awk '{ print $2,$3 }' acteur.tab
      Norris 70
      Stallone    64
      Seagal 59
```

```
$ awk '{ print $2"\t"$3" ans" }' acteur.tab
      Norris      70 ans
      Stallone    64 ans
      Seagal      59 ans
```

Les outils | awk - Syntaxe

- Motifs

```
{action}                # exécute action pour toutes les lignes
/regex/ {action}        # exécute action pour les lignes répondant à regex
prédicat {action}       # exécute action pour les lignes répondant au prédicat
```

- Les opérateurs

```
==, !=                  # égalité, non égalité (caractères et chiffres)
~, !~                   # match, non match (regex)
<, >, <=, >=            # comparaisons numériques
+, -, /, *, %           # opérations numériques
&&, ||                  # opérateurs logiques 'et' 'ou'
```

Les outils | awk - Syntaxe

- Motifs

- Les opérateurs

```
$ awk '$3 >= 60 { print $2,$3 }' acteur.tab
      Norris 70
      Stallone    64

$ awk '$2 ~ /^S.*/ { print $2,$3 }' acteur.tab
      Stallone    64
      Seagal 59

$ awk '$2 ~ /^S.*/ && $3 >= 60 { print $2,$3 }' acteur.tab
      Stallone    64
```

Les outils | awk - Syntaxe

- Les blocs BEGIN et END

```
BEGIN {action}        # exécute action une fois au début

{action}              # exécute action à chaque ligne

END {action}          # exécute action une fois à la fin
```

```
$ awk 'BEGIN {print "Nom;Prénom;Age"}{print $2";"$1";"$3}' acteur.tab
Nom;Prénom;Age
Norris;Chuck;72
Stallone;Sylvester;66
Seagal;Steven;61
```

Les outils | awk - Syntaxe

- Manipulation de chaîne de caractère (string)

```
length(s)                                   # longueur de s
substr(s,o,l)                               # coupe s à partir de o sur une longueur l
tolower(s)                                  # minuscule
toupper(s)                                  # majuscule
split(s,t,d)                                # découpe s en un t avec d
```

ex : split($2,mavariable,"|") ; print(mavariable[2])

```
gsub(r,t,s)                                 # remplace tous les r par t dans s
sub(regexp, replacement, target)            # rechercher / remplacer (équivalent seq)
```

ex : sub(";", "\t", $2)

- Arithmétique

```
int(x)          # partie entière de x
log(x)          # logarithme de x
sqrt(x)         # racine carrée de x
```

Les outils | awk - Utilisation

- Traiter

```
$ awk '{total = total + $3; count = count + 1 } END {print total / count}' acteur.tab
        66.3333



# version courte
# += incrémente la variable moyenne avec $3
# NR est une variable fournis correspondant au nombre de ligne parcourue

$ awk '{total += $3} END {print total / NR}' acteur.tab
        66.3333
```

```
Chuck       Norris      70
Sylvester   Stallone    64
Steven      Seagal      59
```

Les outils | awk - Syntaxe

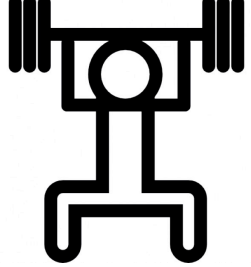
- Syntaxe suite
  - préciser le délimiteur

```
$ awk -F ';' 'BEGIN {action} motif {action} END {action}' fichier
```

  - mode script

```
$ awk -f script.awk fichier
```

Exercices



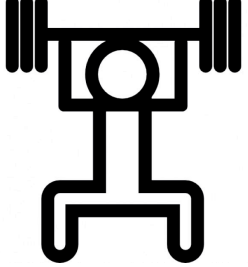
- Exercice 1 – v2

```
$ grep "1" --color patelles_roscoff.csv
        35,6    18,6    0
        36,7    13,1    0
        38,7    15,1    0
        37,4    16,4    0
        48,8    19,8    0
        43,9    17,1    1
```

  - Trouver toutes les patelles à la coque percée

- Exercice 2 – v2
  - Trouver toutes les personnes dont le nom est Thomas dans annuaire.csv
- Exercice 3 – v2
  - Trouver toutes les personnes dont le prénom est Thomas dans annuaire.csv

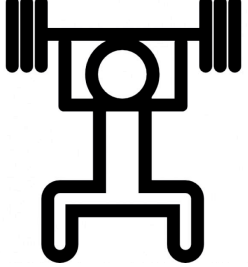
Exercices



- Input :
  - spur_transcriptome.fna
  - spur_transcriptome-orfs.gff3
- Output :
  - récupérer les orfs qui représentent au moins 50% de la longueur de la séquence

- Les outils supplémentaires :
  - infoseq : pour récupérer des informations basiques sur une séquence

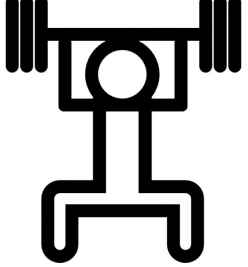
Exercices



- La démarche :
  - récupérer les longueurs des transcrits
  - fusionner les informations issues de la recherche d'ORF avec le fichier contenant les longueurs des séquences
  - calculer la part en % qu'occupe les ORF sur les transcrits
  - filter les orfs qui représentent au moins 50% de la longueur des transcrits

- [bonus] :
  - calculer les tailles des 5'UTR, ORF et 3'UTR
  - ajouter des entêtes de colonnes au fichier généré

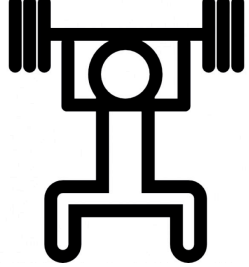
Exercices



Corrections

```
# rechercher les ORFs a partir de transcrits

transcripts_to_best_scoring_ORFs.pl -t ../input/spur_transcriptome.fna
```

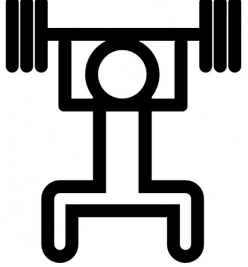
Exercices



Corrections

```
# recuperer les longueurs des transcrits

infoseq -noheading -only -name -length -sequence ../input/spur_transcriptome.fna |
awk '{print $1"\t"$2}' |
sort  -k"1,1" > spur_transcriptome.infoseq
```

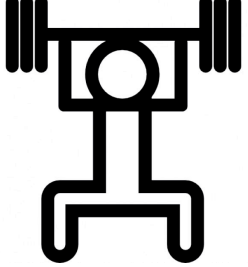
Exercices



Corrections

```
# fusionner les informations issues de la recherche d'orfs avec le fichier contenant
# les longueurs des sequences

grep "mRNA" spur_transcriptome-orfs.gff3 |
sed -r "s/gi\|[0-9]+\|gb\|(.*)\|/\1/" |
sort -k"1,1" > spur_transcriptome-orfs.sorted.gff3
```

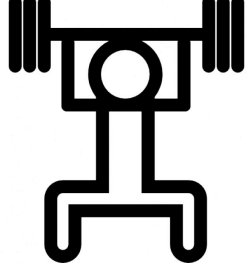
Exercices



Corrections

```
# fusionner les informations issues de la recherche d'orfs avec le fichier contenant
# les longueurs des sequences

grep "mRNA" spur_transcriptome-orfs.gff3 |
sed -r "s/gi\|[0-9]+\|gb\|(.*)\|/\1/" |
sort -k"1,1" > spur_transcriptome-orfs.gff3


join -1 1 -2 1 spur_transcriptome-orfs.sorted.gff3
spur_transcriptome.infoseq > spur_transcriptome-orfs.sorted.merge.tab
```

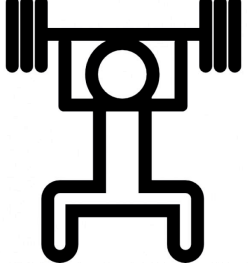
Exercices



Corrections

```
# calculer la part en % qu'occupe les ORF sur les transcrits
# filter les orfs qui representent au moins 50% de la longueur des transcrits

awk '
        {coverage = (($5-$4)/$10)*100}
        coverage >= 50 {print $0"\t"coverage}
    '
spur_transcriptome-orfs.sorted.merge.tab >
spur_transcriptome-orfs.sorted.merge.filtered50.tab
```

Exercices



Corrections

```
#[bonus]
# calculer les tailles des 5'UTR, ORF et 3'UTR

awk '
                {orf=$5-$4}
                $7 == "+" {utr5 = $4-1; utr3 = $10-$5}
                $7 == "-" {utr5 = $10-$5; utr3 = $4-1}
                {print $0"\t"utr5"\t"orf"\t"utr3}
        '
spur_transcriptome-orfs.sorted.merge.tab >
spur_transcriptome-orfs.sorted.merge.utr.tab
```

BATCH

Batch – Besoin

- Traiter par lot un grand nombre de fichiers

“batch processing”

Batch – jeu d'exemple

- Création du jeu de donnée du TP

```
$ split_multifasta.pl nr.fsa 50
```

Batch – Une solution quick and dirty

- Construction d'un fichier script

```
$ paste <(ls nr.fsa.*) <(ls nr.fsa.*) | head
   nr.fsa.1        nr.fsa.1
   nr.fsa.10       nr.fsa.10
   nr.fsa.100      nr.fsa.100
```

```
$ paste <(ls nr.fsa.*) <(ls nr.fsa.*) |
sed "s/^/infoseq -noheading -length -only /" | head
   infoseq -noheading -length -only nr.fsa.1    nr.fsa.1
   infoseq -noheading -length -only nr.fsa.10   nr.fsa.10
   infoseq -noheading -length -only nr.fsa.100  nr.fsa.100
```

```
$ paste <(ls nr.fsa.*) <(ls nr.fsa.*) |
sed "s/^/infoseq -noheading -length -only /" |
sed "s/\t/ > /" | head
   infoseq -noheading -length -only nr.fsa.1 > nr.fsa.1
   infoseq -noheading -length -only nr.fsa.10 > nr.fsa.10
   infoseq -noheading -length -only nr.fsa.100 > nr.fsa.100
```

Batch – Une solution quick and dirty

- Construction d'un fichier script

```
$ paste <(ls nr.fsa.*) <(ls nr.fsa.*) |
sed "s/^/infoseq -noheading -length -only /" |
sed "s/\t/ > /" | head
   infoseq -noheading -length -only nr.fsa.1 > nr.fsa.1
   infoseq -noheading -length -only nr.fsa.10 > nr.fsa.10
   infoseq -noheading -length -only nr.fsa.100 > nr.fsa.100
```

```
$ paste <(ls nr.fsa.*) <(ls nr.fsa.*) |
sed "s/^/infoseq -noheading -length -only /" |
sed "s/\t/ > /" |
sed "s/$/.length/" | head
   infoseq -noheading -length -only nr.fsa.1 > nr.fsa.1.length
   infoseq -noheading -length -only nr.fsa.10 > nr.fsa.10.length
   infoseq -noheading -length -only nr.fsa.100 > nr.fsa.100.length
```

Batch – Une solution quick and dirty

- Lancement

```
$ paste <(ls nr.fsa.*) <(ls nr.fsa.*) |
sed "s/^/infoseq -noheading -length -only /" |
sed "s/\t/ > /" |
sed "s/$/.length/" > infoseq.sh

$ bash infoseq.sh
```

– Ou

```
$ bash <(paste <(ls nr.fsa.*) <(ls nr.fsa.*) |
sed "s/^/infoseq -noheading -length -only /" |
sed "s/\t/ > /" |
sed "s/$/.length/" )
```

Batch – Inline loop - for in

- for in : Prototype

```
$ for file in $( ls nr.fsa.* ); do cmd $file; done
```

```
$ for file in $( ls nr.fsa.* ); do cmd $file; done
```

```
$ for file in $( ls nr.fsa.* ); do cmd $file; done
```

```
$ for file in $( ls nr.fsa.* ); do cmd $file; done
```

Batch – Inline loop - for in

- Exemples
  - Lancer un programme

```
$ for file in $( ls nr.fsa.* ); do infoseq -noheading -length -
only $file > $file.infoseq; done 2> /dev/null
```

Batch – Inline loop - for in

- Exemples
  - Formater le nom du fichier de sortie

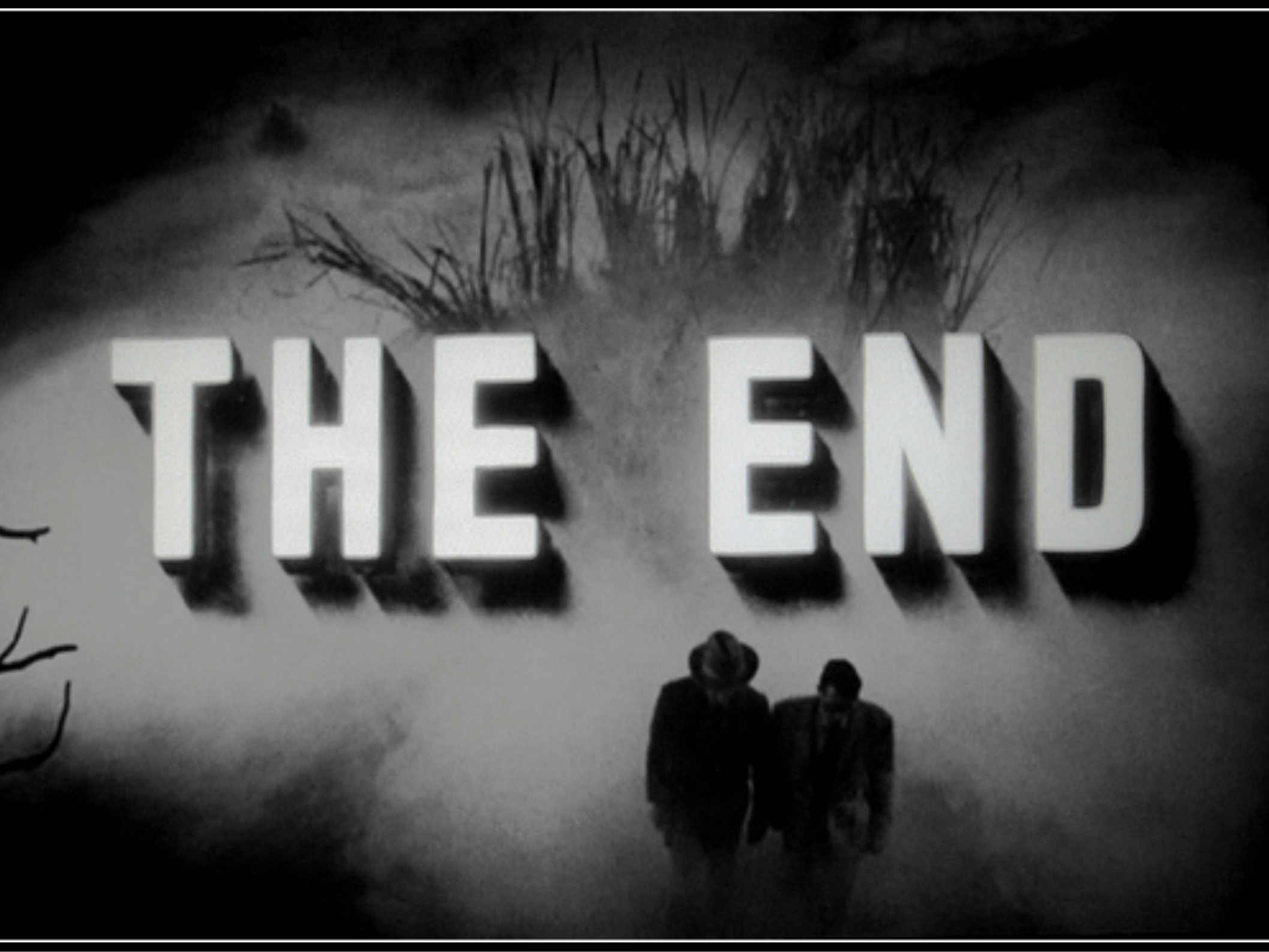
```
$ ls nr.fsa.1    nr.fsa.119  nr.fsa.139  nr.fsa.159  nr.fsa.179
nr.fsa.199  nr.fsa.218  nr.fsa.238  nr.fsa.258  nr.fsa.278
nr.fsa.298  nr.fsa.317  nr.fsa.337  nr.fsa.357  nr.fsa.44

$ for file in $( ls nr.fsa.* ); do mv $file $( echo $file | sed
-r "s/fsa.([0-9]+)/\1.fsa/" ); done 2> /dev/null

$ ls
nr.100.fsa  nr.120.fsa  nr.140.fsa  nr.160.fsa  nr.180.fsa  nr.
1.fsa    nr.21.fsa   nr.23.fsa   nr.25.fsa   nr.27.fsa
```

Batch – Inline loop - for in

- Exemples
  - Lancer une série de qsub

```
$ for file in $( ls nr.fsa.* ); do qsub -v INPUT=$file infoseq.qsub; done
```

```
#!/bin/bash
#$ -S /bin/bash
#$ -V
#$ -cwd
#$ -q short.q

infoseq -noheading -length -only $INPUT > $INPUT.infoseq
```


THE END