

# Session de formation 2018



- 12 Mars**      Guide de survie à Linux : les commandes de base pour débuter sur un serveur linux
- 13 Mars**      Linux avancé : manipuler et filtrer des fichiers sans connaissance de programmation
- 15 Mars**      Initiation à l'utilisation du cluster bioinformatique itrop
- 22 Mars**      Initiation à git
- 23 Mars**      Initiation aux gestionnaires de workflow South Green:  
Galaxy ou TOGGLE
- 26 Mars**      Initiation aux analyses de données transcriptomiques





IRD

Institut de Recherche  
pour le Développement

# South Green

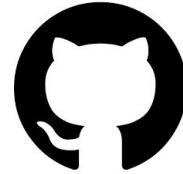
bioinformatics platform



plateau i-trop



[www.southgreen.fr](http://www.southgreen.fr)



<https://github.com/SouthGreenPlatform>



*The South Green portal: a comprehensive resource for tropical and Mediterranean crop genomics*, Current Plant Biology, 2016

# Session de formation 2018



- Toutes nos formations :  
<https://southgreenplatform.github.io/trainings/>
- Topo & TP : [Linux For Jedi](#)
- Environnement de travail : [Logiciels à installer](#)



# Linux Avancé

[www.southgreen.fr](http://www.southgreen.fr)

<https://southgreenplatform.github.io/trainings>



## The objectif!

Optimiser vos analyses bioinformatiques sur un cluster en utilisant la puissance de Linux



## Applications

- Manipuler des fichiers de sortie : ***head, tail, sort, cut, wc, grep***
- Rediriger la sortie d'une commande dans un fichier : ***>, >>***
- Enchaîner plusieurs commandes : ***|***
- Réaliser la même action sur plusieurs fichiers
- Modifier le contenu d'un fichier : ***sed, awk***

# Rappel Commandes de Base





# Practice

1

Go to [Practice 1](#) on our github

# Les entrées / sorties



**pour sauvegarder la sortie d'une  
commande dans un fichier**

```
cut -d: -f1 /etc/passwd
```

On peut rediriger la sortie d'une commande

dans un fichier avec le caractère >

au lieu d'afficher sur le terminal

```
cut -d: -f1 /etc/passwd > userName.txt
```

Redirection	Action
Command > file	<ul style="list-style-type: none"><li>• si le fichier n'existe pas : il sera créé</li><li>• si le fichier existe : efface le contenu</li></ul>
Command >> file	<ul style="list-style-type: none"><li>• si le fichier n'existe pas : il sera créé</li><li>• si le fichier existe : écrit à la fin du fichier</li></ul>



# Practice

2

Go to [Practice 2](#) on our github

```
cut -d: -f1 /etc/passwd
```

On peut rediriger la sortie d'une commande

en entrée d'une autre commande

avec le caractère |

= *workflow*

```
cmd1 | cmd2 | cmd3
```

- Pipe permet d'enchaîner l'exécution de plusieurs programmes/commandes
- Redirection sans utilisation de fichier intermédiaire

# Pipe pour rediriger une commande

```
cut -d: -f1 /etc/passwd
```

Root

troot

iroot

ctroot

//

# Pipe pour rediriger une commande

```
cut -d: -f1 /etc/passwd
```

Root

troot

iroot

ctroot

//

```
cut -d: -f1 /etc/passwd | sort
```

abate

adm

adroot

ais

#albar

alvaro-wis

anthony

apache

# Pipe pour rediriger une commande

```
cut -d: -f1 /etc/passwd
```

Root

troot

iroot

ctroot

//

```
cut -d: -f1 /etc/passwd | sort
```

abate

adm

adroot

ais

#albar

alvaro-wis

anthony

apache

```
cut -d: -f1 /etc/passwd | sort | head
```



# Practice

3

Go to [Practice 3](#) on our github



# Des commandes pour rechercher dans des fichiers

**grep**

# Commande grep

**grep**

*pour rechercher un motif dans une ligne*

```
grep [options] "motif" [file1, ... ]
```

# Commande grep

**grep**

*pour rechercher un motif dans une ligne*

*grep [options] "motif" [file1, ... ]*

Option	Description
-c	Compte le nombre de lignes dans lesquelles le motif a été trouvé
-n	Affiche le numéro de ligne et la ligne dans laquelle le motif a été trouvé
-i	insensible à la casse (majuscule/minuscule) du motif
-v	Affiche seulement les lignes sans le motif
-l	Affiche uniquement les noms de fichiers dans lesquels le motif a été trouvé



# Practice

4

Go to [Practice 4](#) on our github

5

Go to [Practice 5](#) on our github



# Des commandes pour rechercher et modifier des fichiers

**sed**

## sed : rechercher et modifier une ligne

Selection de lignes dans un fichier vérifiant une expression régulière  
ET applicant une modification ou un traitement

```
sed "s/motif recherché/nouveau motif/" file
```

substitution

motif recherché

nouveau motif

fichier à parser

## Sed : Quelques exemples

Example	Description
sed "s/linux/LINUX/" file	Change la 1ère occurrence de “linux” par “LINUX”
sed "s/linux/LINUX/3" file	Change la 3ème occurrence de “linux” par “LINUX”
sed "s/linux/LINUX/g" file	Change toutes occurrences de “linux” par “LINUX”

## Sed : Quelques exemples

### Example

### Description

```
sed "s/>/>MOT-/g" seq.fasta
```

Substitution de > par >**MOT-** au niveau de chaque identifiant de séquences

```
sed "s/\|/-/g" contigs.fasta
```

Substitution de | par - (le | est “despecialisé” par le \)



# Practice

6

Go to [Practice 6](#) on our github

## sed : rechercher et modifier une ligne

Selection de lignes dans un fichier vérifiant une expression régulière  
ET applicant une modification ou un traitement

```
sed "s/[0-9][0-9]* /nouveau motif/" file
```

substitution

motif recherché

nouveau motif

fichier à parser

Recherche une chaîne de caractères  
commençant par un chiffre suivi par 0 ou plusieurs nombres

=> Chaîne de caractère enregistrée dans la variable \1



# Practice

6

*Exercice bonus*



# Des commandes pour rechercher et modifier des fichiers

**awk**

## awk: Langage pour manipuler un fichier ligne par ligne

- Nom des auteurs : “Aho, Weinberger, and Kernighan”

## awk: Langage pour manipuler un fichier ligne par ligne

- Nom des auteurs : “Aho, Weinberger, and Kernighan”
- Un langage de programmation qui permet facilement de manipuler des fichiers tabulés (blast, sam, vcf) et d'extraire une partie des données
- Un langage utilisé pour rechercher des motifs et pour effectuer des opérations, des actions associées.

## awk: Langage pour manipuler un fichier ligne par ligne

- Un langage de programmation qui permet facilement de manipuler des fichiers tabulés (blast, sam, vcf) et d'extraire une partie des données.
- Un langage utilisé pour rechercher des motifs et pour effectuer des opérations, des actions associées.

### Principales caractéristiques d'awk

- Pour awk, le fichier en entrée est tabulé
- Comme tout langage de programmation, awk a des variables et peut appliquer des conditions
- awk peut faire des opérations sur les nombres et les chaînes de caractères
- Awk peut générer et afficher des données/rapports suite à des manipulations

## awk: Langage pour manipuler un fichier ligne par ligne

Syntax : *awk [-F] 'program' file*

Option	Description
-F	Donne la nature des séparateurs de champs

## awk: Langage pour manipuler un fichier ligne par ligne

Syntax : `awk [-F] 'program' file`

Option	Description
<code>-F</code>	Donne la nature des séparateurs de champs

Variables prédéfinies utilisées par awk

Variable	Description
<code>\$0</code>	ligne entière
<code>NR</code>	Numéro de la ligne lue
<code>NF</code>	Nombre de champs dans la ligne

awk voit le fichier en entrée  
comme des enregistrements et des champs

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	r��aron@videotron.ca

File: contact.txt

# Rechercher un motif & modifier un fichier

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	r��aron@videotron.ca

File: contact.txt

```
awk '{print $0}' contact.txt
```

```
Helene 56 edu hcyr@sun.com
jean 32 ri jeanc@inexpress.net
julie 22 adm juliem@sympatico.ca
michel 24 inf michel@uqo.ca
richard 25 inf r  aron@videotron.ca
```

Affiche chaque  
ligne

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	r̄caron@videotron.ca

File: contact.txt

```
$awk '{print NR, $1, $2}' contact.txt
```

```

1 Helene 56
2 jean 32
3 julie 22
4 michel 24
5 richard 25

```

Affiche

le numéro de la ligne lue  
 Puis le 1<sup>er</sup> champs  
 puis le 2<sup>ème</sup> champs du  
 fichier tabulé

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	r��aron@videotron.ca

```
$awk '{print $1,$2};  
END { print NR << lignes lues en tout >>; }' contact.txt
```

Helene 56

Jean 32

Julie 22

Michel 24

Richard 25

5 lignes lues en tout

Instruction ex  ut  e une fois le fichier lu dans son int  gralit  

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	rcharon@videotron.ca

```
$awk ' {print $1,$3; somme+=$2}
END { print « Somme des ages égale à », somme; }'
contact.txt
```

Helene edu  
jean ri  
julie adm  
michel inf  
richard inf

Somme des ages égale à 159

On ajoute l'âge (\$2) à la variable somme à chaque ligne lue

Puis on affiche la somme calculée à la fin de la lecture du fichier

Helene	56	edu	hcyr@sun.com
jean	32	ri	jeanc@inexpress.net
julie	22	adm	juliem@sympatico.ca
michel	24	inf	michel@uqo.ca
richard	25	inf	r��aron@videotron.ca

File: contact.txt

```
$awk '{somme+=$2}
END { print << Age moyen = >>, somme/NR; }' contact.txt
```

Age moyen = 31,8

On ajoute l' ge (\$2)   la variable **somme**   chaque ligne lue

Puis on affiche la moyenne une fois le fichier lu

awk: Langage pour manipuler un fichier ligne par ligne

avec une liste d'instructions et de **conditions aussi**

Condition {Instr-1; Instr-2; ...; Instr-n}

```
awk '$2>24 && $2<50{ print « Age de », $1, « compris entre 24 et 50, égal à », $2; }' contact.txt
```

L age d Helene est supérieur à 24 et égal à 56  
L age d jean est supérieur à 24 et égal à 32  
L age d richard est supérieur à 24 et égal à 25

Avec 2 conditions

## awk: Langage pour manipuler un fichier ligne par ligne

```
awk '$3 == "inf" {print $0}' contact.txt
```

```
michel 24 inf michel@uqo.ca
richard 25 inf rcaron@videotron.ca
```

```
$awk '/j/ {print $0}' contact.txt
```

```
jean 32 ri jeanc@inexpress.net
julie 22 adm juliem@sympatico.ca
```

## awk: Langage pour manipuler un fichier ligne par ligne

```
awk ' {print $1,$2-10}' contact.txt
```

```
Helene 46
Jean 12
Julie 12
Michel 14
Richard 15
```

```
awk '$2 > 30 && $3 == "ri" {print $0}' contact.txt
```

```
jean 32 ri jeanc@inexpress.net
```

Ces commandes peuvent être utilisées avec en entrée la sortie standard ou un fichier tabulé (comme .gff, fichier blast m8 , .vcf)



# Practice

7

*Go to [Practice 7](#) on our github*



# Exécuter une même commande sur plusieurs fichiers

**for (bash)**

# Exécuter une boucle

*for...*



- Pour parcourir une liste
- Exécuter les mêmes instructions sur chaque élément de la liste

```
for file in *;  
do  
    instruction1;  
    instruction2;  
done;
```



# Practice

8

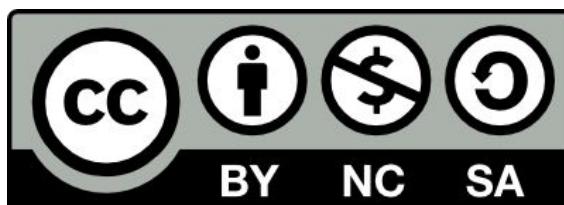
*Go to [Practice 8](#) on our github*

## Formateurs itrop / South Green

- Christine Tranchant-Dubreuil
- Sébastien Ravel
- Alexis Dereeper
- Ndomassi Tando
- François Sabot
- Gautier Sarah
- Bruno Granouillac



# Merci pour votre attention !



Le matériel pédagogique utilisé pour ces enseignements est mis à disposition selon les termes de la licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage dans les Mêmes Conditions (BY-NC-SA) 4.0 International:

<http://creativecommons.org/licenses/by-nc-sa/4.0/>