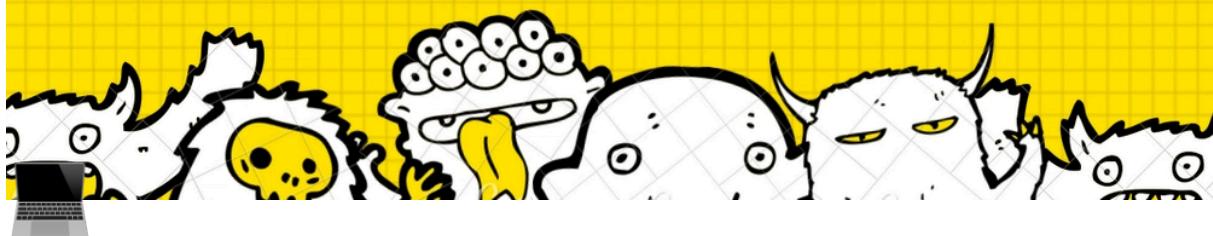


# LET'S HAVE FUN WITH LINUX COMMAND LINE



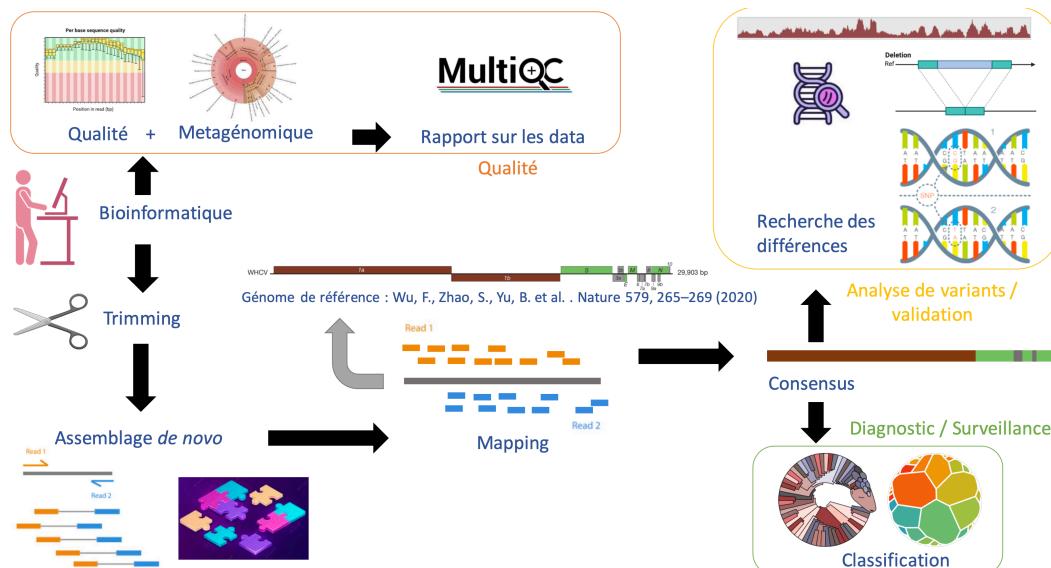
## Linux for Bioinformatics

### Agenda

#### ▼ Session 1: Introduction aux ressources bio-informatiques et aux formats de données génomiques

##### ▼ LOGIQUE DES PIPELINES BIO-INFORMATIQUES

Un pipeline? C'est quoi?



**Suite d'étapes organisées** permettant de transformer des données biologiques brutes issues d'un séquençage en résultats interprétables

ex: comme des variants, des profils d'expression, ou des annotations fonctionnelles

Chaque étape du pipeline s'appuie sur des formats de fichiers spécifiques, qui servent d'entrée et de sortie pour les outils utilisés.

La bonne compréhension de ces formats est essentielle pour suivre, modifier ou concevoir un pipeline de manière rigoureuse et reproductible.

## ▼ LES FORMATS DE FICHIERS EN BIO-INFORMATIQUE

À mesure que les données biologiques sont devenues numériques, de nombreux formats ont émergé pour répondre à des besoins spécifiques : du simple fichier texte aux formats complexes intégrant annotations et métadonnées. Nous verrons plus bas les principaux formats utilisés en bio-informatique, leurs caractéristiques et leurs usages.

### 1. Formats de séquence

#### FASTA

Le format **FastA** est le format le plus simple pour représenter une séquence, et il est accepté par presque tous les programmes d'analyse de séquences.

Il contient uniquement un **nom de séquence**, une **description** (métadonnées, informations du séquenceur, annotations, etc.) et la **séquence elle-même** – il peut s'agir d'acides nucléiques ou d'acides aminés, tant que le format est respecté.

Chaque séquence est composée **d'au moins deux lignes** :

1. La première est **l'en-tête** de la séquence, qui commence toujours par un >
  - Tout ce qui se trouve entre le symbole > et le **premier espace** est considéré comme **l'identifiant** de la séquence (ex. accession GenBank)
  - Tout ce qui suit cet espace est considéré comme **la description** (cela peut inclure des métadonnées, un numéro de série de la machine, l'orientation de la lecture, etc.)
2. Ensuite vient **la séquence** elle-même.
  - À noter : la séquence peut s'étendre sur **plusieurs lignes** en fonction de sa longueur.

```
>Chr1 CHROMOSOME dumped from ADB: Jun/20/09 14:53; last updated: 2009-02-02
CCCTAACCTAAACCTAACCTAACCTCTGAATCCTAAATCCCTAAATCTTAAATCCTACATCCAT
GAATCCCTAAATCCTAACCTAACCGAACCGGTTCTGGTTGAAAATCATGGTATATAATGATAATT
ATCGTTTATGTAAATTGCTTATGTGTGAGATTAAAAATCATTTGAGGTCAATACAATCTTATTCT
TGTGGTTCTTCCTCACTTAGCTATGGATGGTTTATCTTCAAGGTTATTGATACAAGCTTGCTACGATCTA
CATTGGGAATGTGAGTCTTATGTAACCTTACGGGTTGGTTATCTCAAGGTTATTGATACAAGCTTGCTACGATCTA
TGTTGGACATTATTGTCACTTACTCCTTGTGAAATGTTGTTCTATCAATTATCTTGTGGAAAATTATT
TAGTTGAGGATGAAGTCTTCCTCGTGTGTTACGCTTGTCTATCTCATCTCAATGATATGGATGTTCTT
CATTATTCTGAAGTCTCTGCTTGTGATTTCCTAGCCAAAGGATTGGTGGTTGAAGACACATCATATCAA
AAAAGCTATCGCCCTCGACGCTCTTCTGGTACACATTTGGCACTCAAAAAGTATTTTAGATGT
TTGTTTGTCTTGAAGTAGTTCTTGCCTTGGTACGATTTGGTGGATGTTCAAGACTTCTC
GGTACTGCAAAGTCTTCGCCGCTGATTAATTATCCATTTCCTTGTGCTAGATATTAGTAATCTGAAGTCAACT
ATATACAACCTATAATTAAAATATGATGACACACAGTTACACATAAAATCTGAAATCAACTCATATACCC
GTTATCCCCACAATCATGCTTCAAAAGCAAAGTATGTCAACAATTGGTTAAATTAGAAGTTTCCAC
TTATGACTTAAGAACACTGTGAGCAGAAAGTGGCAACACCCCCCACCTCCCCCCCCCCCCAATTGAGA
AGTCATAATTATAATTAAATAAGTTATGGTTAGAGTTTACTCTTTTATTTCTTTCTTT
```

Extensions courantes : .fasta, .fa, .fna, .faa, .frn

#### Comment sont ils générés?

Les anciens séquenceurs NGS produisent des données sous ce format (e.g Sanger sequencing)

La plupart des BD stocke les séquences sous ce format directement téléchargeable

FastA peut aussi être généré à partir d'un fichier FastQ

#### FASTQ

File containing sequencing reads or nucleotides sequences and quality score for each base position.

The result of base-calling and demultiplexing of raw data from a sequencing instrument. Il stocke les "reads" issus du séquençage ou séquences de nucléotides avec les scores de qualité associés à chaque position.

#### Structure :

1. `@` + identifiant de read (souvent lié à la machine Illumina).
  2. Séquence
  3. Ligne `+` (optionnellement répétition de l'identifiant)
  4. Scores de qualité pour chaque base (Phred, encodés en ASCII)
- **Extensions :** `.fastq`, `.fq`

```
@HWI-ST911:111:C0N4WACXX:5:1101:2249:2216 1:N:0:TTAGGC CGATC:@@@FF
NATGGCACCATAAAAAGAATGTTTATGGTGTGAGAAGGACAAGCTGAAGAAGAAATTAGTCCTGCACTGATGTTGCAAATGCAAAGAAA
+
#2A2<CCFHIIIIIIIIIIIGCHIIIGIFFHIIDGHIGHIIIIICHGIIIGGCCECEGICFHCECDEFFFFDEEEEEDDDDCDDCDDBBC
@HWI-ST911:111:C0N4WACXX:5:1101:2509:2197 1:N:0:TTAGGC CGATC:1+4=B
NATGAGATAAACATCAATTGCTTTAATGAAGTACAGTCTTGAAATAATGAGTTGAACACTCTCTGCAACTTTGGAAACTTAAAGTTGTAATG
+
#4A2<AADHIIIIIIIIHHIIIIIIIFGII@GIIFIIGIIIEIDHEIIIHIIIIIIIIICHIIIHHEEDFFFFEEEEEADDFC
@HWI-ST911:111:C0N4WACXX:5:1101:3746:2179 1:N:0:TTAGGC CCATC:+11+A
NATGTCATCCATCTTCTATCTAAAAAGAATCAAAAAGGGATAGTACAGAGGGAAAGTTCAATCCAGAGGACGATGAAACACTGATTGATGG
+
```

HW-ST911	the unique instrument name
111	the run id
C0N4WACXX	the flowcell id
5	flowcell lane
1101	tile number within the flowcell lane
2249	'x'-coordinate of the cluster within the tile
2216	'y'-coordinate of the cluster within the tile
1	the member of a pair, 1 or 2 (paired-end or mate-pair reads only)
Y	Y if the read is filtered, N otherwise
18	0 when none of the control bits are on
TTAGGC, CGATC	index sequence

Quels sont les outils qui utilisent ce format?

- Aligners (Bowtie, Tophat2)
- Assemblers (Velvet, Spades)
- QC tools (Trimmomatic, FastQC)

#### Quality scores

Les **scores de qualité** (ou *quality scores*) permettent d'évaluer la **fiabilité de chaque base** dans une lecture (*read*) de séquençage.

Ils représentent la **probabilité qu'une base ait été mal identifiée** (erreur d'appel de base, ou *base call error*).

La majorité des séquenceurs utilisent aujourd'hui le codage **Phred-33** (un encodage standard où les scores sont transformés en caractères ASCII pour compacter les données). Chaque score Phred correspond à une **probabilité d'erreur** selon la formule :

$$Q = -10 \times \log_{10}(p)$$

où  $p$  est la probabilité que la base soit incorrecte. Par exemple, un score de 20 signifie une erreur sur 100 bases, un score de 30 une erreur sur 1 000 bases.

Ces scores sont stockés dans les fichiers **FastQ** et sont utilisés pour **filtrer les bases** ou les **lectures complètes** de mauvaise qualité, souvent selon une **valeur seuil** (par exemple, la moyenne des scores sur un read).

Ils sont essentiels dans de nombreuses étapes de l'analyse NGS (Next Generation Sequencing) : **contrôle qualité** (*FastQC*), **alignement** (ex. *BWA*, *Bowtie2*), **assemblage de génomes**, ou **détection de variants** (*variant calling*, comme les SNP – *Single Nucleotide Polymorphisms*)

## 2. Alignement

### SAM (Sequence Alignment/Map)

C'est le format le plus simple et lisible par un humain parmi les trois.

Il est généré par pratiquement tous les algorithmes d'alignement existants et sert à stocker des alignements entre reads et génome de référence en **texte**

Peut être lu par des visualiseurs pour un affichage graphique des alignements.

Il se compose:

- d'une **en-tête**,
- d'une **ligne pour chaque lecture (read)** de votre jeu de données, et
- de **11 champs séparés par des tabulations** décrivant cette lecture



- **Structure** : header ( @ ) + lignes d'alignement (nom du read, position, CIGAR, ...)

- **Usage** : inspection manuelle, conversion vers BAM ou CRAM

### Sequence Alignment/Map format

Similaire à FASTQ mais contient des informations supplémentaires

```
@HISEQ2000-02:420:C2E47ACXX:7:2214:18015:39495 99 [chr1_17644] 37 37M = 17919 314
CACTCGAGCCTGGGTGACAGAGCAGATTCCCTCTAAAGATTTAATAAATTTAAATAAATTTAAAGTTTG
EAD@:@?@A@:@?>?@?@?A?@?>@?ACCAA@A@:@AAAAA?AAA?AAA?AAA?ACCCBBBBAABA@
RG:Z:UM0098:1 XT:A:R NM:i:0 SM:i:0 AM:i:0 X0:i:4 XI:i:0 XM:i:0 X0:i:0 XG:i:0 MD:Z:37
Field          QNAME      @HISEQ2000-02:420:C2E47ACXX:7:2214:18015:39495
FLAG           99
RNAME          chr1
POS            17644
MAPQ          37
CIGAR          37M
NM:i:0
MD:Z:37
MN:i:0/PNEXT
MPOL/PNEXT
ISIZE/TLEN
SEQ             CACTCGAGCCTGGGTGACAGAGCAGATTCCCTCTAAAGATTTAATAAATTTAAATAAATTTAAAGTTTG...
QUAL           EAD@:@?@A@:@?>?@?@?A?@?>@?ACCAA@A@:@AAAAA?AAA?AAA?AAA?ACCCBBBBAABA@...
TAGS           RG:Z:UM0098:1 XT:A:R NM:i:0 SM:i:0 AM:i:0 X0:i:4 XI:i:0 XM:i:0 X0:i:0 XG:i:0 MD:Z:37
```

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	[^@]{1,254}	Query/mate NAME
2	FLAG	Int	[0,2^15-1]	binary FLAG
3	RNAME	String	\w{1,-1}\w*	Reference sequence NAME
4	POS	Int	[0,2^31-1]	1-based leftmost mapping POSITION
5	MAPQ	Int	[0,2^8-1]	MAPping Quality
6	CIGAR	String	\w{1,((0-9)+[MDNSHPX]*)*}	CIGAR string
7	RNEXT	String	\w{1,((0-9)+[MDNSHPX]*)*}	Ref. name of the mate/next read
8	PNEXT	Int	[0,2^31-1]	Position of the mate/next read
9	TLEN	Int	[-2^15+1,2^15-1]	observed Template LENgth
10	SEQ	String	\w{1,(A-Za-z\*\w)*}	segment SEQuence
11	QUAL	String	[!-]+	ASCII of Phred-scaled base QUALity+33

<http://samtools.github.io/hts-specs/SAMv1.pdf>

Que stocke la séquence CIGAR?

Le champ **CIGAR** est une représentation compacte qui **décrit la forme exacte de l'alignement** entre le read et la séquence de référence, en détaillant les correspondances, insertions, suppressions, et autres modifications.

Op	Description
M	Match (alignment column containing two letters). This could contain two different letters (mismatch) or two identical letters. USEARCH generates CIGAR strings containing Ms rather than X's and =s (see below).
D	Deletion (gap in the target sequence).
I	Insertion (gap in the query sequence).
S	Segment of the query sequence that does not appear in the alignment. This is used with soft clipping, where the full-length query sequence is given (field 10 in the SAM record). In this case, S operations specify segments at the start and/or end of the query that do not appear in a local alignment.
H	Segment of the query sequence that does not appear in the alignment. This is used with hard clipping, where only the aligned segment of the query sequences is given (field 10 in the SAM record). In this case, H operations specify segments at the start and/or end of the query that do not appear in the SAM record.
=	Alignment column containing two identical letters. USEARCH can read CIGAR strings using this operation, but does not generate them.
X	Alignment column containing a mismatch, i.e. two different letters. USEARCH can read CIGAR strings using this operation, but does not generate them.

Code	Signification	Description technique
M	Correspondance (match ou mismatch)	Base du read alignée sur la référence (exacte ou non)
I	Insertion dans le read	Base(s) présente(s) dans le read mais absente(s) dans la référence
D	Délétion dans le read	Base(s) présente(s) dans la référence mais absente(s) dans le read
N	Saut dans la référence	Région ignorée dans la référence (ex. intron en RNA-seq)
S	Soft clipping	Bases non alignées, visibles dans la séquence du read
H	Hard clipping	Bases non alignées, supprimées de la séquence du read
=	Match exact	Base du read identique à celle de la référence
X	Mismatch	Base du read différente de celle de la référence

## BAM

- **But :** version **binaire et compressée** du SAM.
- **Avantages :**
  - Taille réduite, indexable.
  - Lecture rapide avec Samtools, IGV...

## CRAM

- **But :** compression encore plus efficace que BAM, en utilisant le génome de référence

Extension : .cram

Il s'agit d'un format relativement récent, très similaire au format BAM puisqu'il conserve également les mêmes informations que le format SAM et est compressé, mais il est plus intelligent dans la manière dont il stocke les données.

### 3. Variants

## VCF (Variant Call Format)

Le **format VCF (Variant Calling Format)** est un fichier texte à colonnes séparées par des tabulations, utilisé pour décrire les **variants nucléotidiques simples (SNVs)** ainsi que les **insertions, délétions** et autres variations de séquence.

- **But** : décrire les variants (SNP, indels, ...).
  - **Structure** :
    - En-tête ( ## ) avec métadonnées.
    - Colonnes obligatoires : CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO, FORMAT + échantillon

	Champ	Nom	Description
Header			
(a) VCF example			<pre>##fileformat=VCFv4.1 ##fileDate=20110413 ##reference=file:///refs/human_NCBI36.fasta ##contig=<id=x,length=155270568,md5=7e02e2508297b7764e31db80c254b0d,species="homo ##alt='&lt;ID=DEL,Description="Deletion"&gt;' ##filter='&lt;ID=QD,Number=1,Type=String,Description="Quality"&gt;' ##format='&lt;ID=DP,Number=1,Type=Integer,Description="Read' ##info='&lt;ID=DEL,Number=1,Type=Integer,Description="End' #chrom="" &lt;del&gt;="" .="" 0="" 0:20:36<="" 1="" 100="" 1:12:-="" 1:13="" 1:16="" 2="" 2:20="" 2:29="" 3="" aat="" allele"&gt;="" c="" depth"&gt;="" filter="" format="" ggt="" gt:dp="" gt:gq:dp="" h2:aat="" id="" info="" of="" pass="" pos="" position="" pre="" qual="" ref="" sample1="" sample2="" sapiens"&gt;="" svtype="DEL;END=299" t,ct="" the="" variant"&gt;=""> </id=x,length=155270568,md5=7e02e2508297b7764e31db80c254b0d,species="homo></pre>
Body			
(b) SNP	(c) Insertion	(d) Deletion	(e) Replacement
Alignment	VCF representation	VCF representation	VCF representation
A ACGT	100 110 120 290 300 ACGTACGTACTAGTAGCTAGTCGT[...]	12345 1234 AC-GT 1 AC-GT ACGT A-TT	100 T ->DEL> GTAC
(f) Large structural variant			
Alignment	VCF representation	VCF representation	VCF representation
A ACGT	100 110 120 290 300 ACGTACGTACTAGTAGCTAGTCGT[...]	12345 1234 AC-GT 1 AC-GT ACGT A-TT	100 T ->DEL> GTAC
(g) Aligning ambiguity	Possible representation	Possible representation	Recommended VCF representation
1234567890	POS REF ALT	POS REF ALT	POS REF ALT
TTCCCTCTA	1 TTCCCTCT CTTACCTA	1 T C 4 C A 7 CCT T	1 T C 4 C A 5 CCT C
CTTACCT-A	~ ~ ~		
Header			
1	CHROM		Nom du chromosome sur lequel se trouve le variant (ex. chr1, chrX, MT)
2	POS		Position exacte du variant sur le chromosome
3	ID		Identifiant du variant (souvent issu de dbSNP ou d'une base annotée ; peut être . si inconnu)
4	REF		Base(s) de référence à cette position
5	ALT		Base(s) alternative(s) observée(s) dans l'échantillon
6	QUAL		Score de qualité Phred pour la base alternative (indique la confiance dans la détection)
7	FILTER		Résultat du filtrage qualité (ex. PASS si le variant est retenu, sinon nom du filtre échoué)
8	INFO		Informations supplémentaires décrivant le variant ou sa position

- **Usage :**

Le format VCF est largement utilisé dans les analyses génomiques pour **décrire les variants génétiques** tels que les SNPs et les indels.

Il est produit en sortie par des outils d'appel de variants comme **GATK** ou **Samtools**, ainsi que par certains logiciels de haplotypage.

Ce format sert aussi d'entrée pour des logiciels d'annotation fonctionnelle des variants tels que **SNPeff**, ou pour des outils de filtrage et d'analyse comme **VCFtools**.

Par ailleurs, de nombreuses bases de données publiques de variants, comme **dbSNP**, exigent ou distribuent leurs données au format VCF, ce qui en fait un standard incontournable dans le domaine.

## 4. Annotations génomiques

### GFF / GTF

Le **format General Feature Format (GFF)** est un fichier texte tabulé qui permet de décrire toutes sortes de **caractéristiques annotées sur une séquence d'acide nucléique ou de protéine**.

Il peut contenir des informations sur des régions codantes (CDS), des microARN, des domaines de liaison, des cadres de lecture ouverts (ORFs), et bien plus. Cependant, plusieurs versions différentes de ce format ont été développées, ce qui a causé des problèmes d'incompatibilité. La version la plus récente, appelée **GFF3**, a été conçue pour résoudre ces problèmes et est désormais la norme acceptée.

Le fichier GFF3 contient 9 champs obligatoires, dont certains peuvent être laissés vides ou avec une valeur par défaut.

- **But :** annoter les éléments génomiques (gènes, exons, CDS, UTR...).
- **Structure :**
  - En-tête (##).
  - 9 colonnes : séquence, source, type, début, fin, score, brin, phase, attributs/groupes
- **GTF :** variante de GFF, format strict pour gènes/transcrits (inclut `gene_id`, `transcript_id`).

```

0 ##gff-version 3.2.1
1 ##sequence-region ctg123 1 1497228
2 ctg123 . gene      1000 9000 . + . ID=gene00001;Name=EDEN
3 ctg123 . TF_binding_site 1000 1012 . + . ID=tfbs00001;Parent=gene00001
4 ctg123 . mRNA     1050 9000 . + . ID=mRNA00001;Parent=gene00001;Name=EDEN.1
5 ctg123 . mRNA     1050 9000 . + . ID=mRNA00002;Parent=gene00001;Name=EDEN.2
6 ctg123 . mRNA     1300 9000 . + . ID=mRNA00003;Parent=gene00001;Name=EDEN.3
7 ctg123 . exon    1300 1500 . + . ID=exon00001;Parent=mRNA00003
8 ctg123 . exon    1050 1500 . + . ID=exon00002;Parent=mRNA00001,mRNA00002
9 ctg123 . exon    3000 3902 . + . ID=exon00003;Parent=mRNA00001,mRNA00003
10 ctg123 . exon   5000 5500 . + . ID=exon00004;Parent=mRNA00001,mRNA00002,mRNA00003
11 ctg123 . exon   7000 9000 . + . ID=exon00005;Parent=mRNA00001,mRNA00002,mRNA00003
12 ctg123 . CDS    1201 1500 . + 0 ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
13 ctg123 . CDS    3000 3902 . + 0 ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
14 ctg123 . CDS    5000 5500 . + 0 ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
15 ctg123 . CDS    7000 7600 . + 0 ID=cds00001;Parent=mRNA00001;Name=edenprotein.1
16 ctg123 . CDS   1201 1500 . + 0 ID=cds00002;Parent=mRNA00002;Name=edenprotein.2
17 ctg123 . CDS   5000 5500 . + 0 ID=cds00002;Parent=mRNA00002;Name=edenprotein.2
18 ctg123 . CDS   7000 7600 . + 0 ID=cds00002;Parent=mRNA00002;Name=edenprotein.2
19 ctg123 . CDS   3301 3902 . + 0 ID=cds00003;Parent=mRNA00003;Name=edenprotein.3
20 ctg123 . CDS   5000 5500 . + 1 ID=cds00003;Parent=mRNA00003;Name=edenprotein.3
21 ctg123 . CDS   7000 7600 . + 1 ID=cds00003;Parent=mRNA00003;Name=edenprotein.3
22 ctg123 . CDS   3391 3902 . + 0 ID=cds00004;Parent=mRNA00003;Name=edenprotein.4
23 ctg123 . CDS   5000 5500 . + 1 ID=cds00004;Parent=mRNA00003;Name=edenprotein.4
24 ctg123 . CDS   7000 7600 . + 1 ID=cds00004;Parent=mRNA00003;Name=edenprotein.4

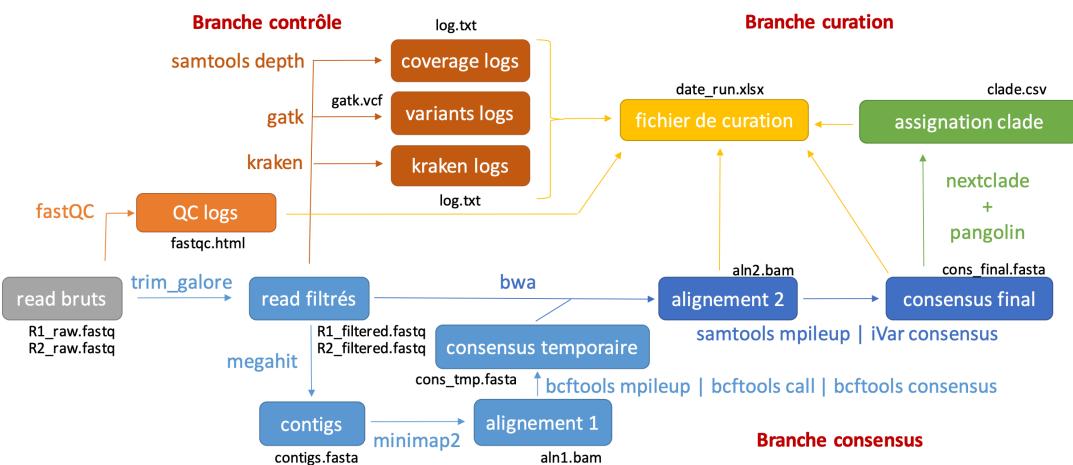
```

Champ	Nom	Description
1	Sequence ID	Identifiant de la séquence (ex. nom du chromosome, accession de génome)

2	Source	Origine de l'annotation (ex. nom du logiciel ou de la base de données, comme Genescane, GenBank)
3	Feature Type	Type de la caractéristique annotée (ex. exon, CDS, mRNA, domaine), basé sur l'ontologie des séquences
4	Feature Start	Position de début de la caractéristique sur la séquence (coordonnée 1-based)
5	Feature End	Position de fin de la caractéristique sur la séquence
6	Score	Score numérique (souvent une valeur statistique comme E-value ou P-value), ou . si non applicable
7	Strand	Brin de la séquence : + pour le brin positif, - pour le brin négatif, ou . si non applicable
8	Phase	Cadre de lecture pour les caractéristiques codantes : 0, 1 ou 2 indiquant combien de bases retirer avant le prochain codon
9	Attributes	Liste de paires clé-valeur séparées par des points-virgules, donnant des informations supplémentaires (ex. ID, Name, Parent)

## ▼ 5. Quelques formats complémentaires

- - **BED** : pour visualiser des plages génomiques (minimum : chrom, chromStart, chromEnd)
  - **CSV / JSON** : formats généraux pour métadonnées, tables, échanges avec autres outils
  - **PDB** : structures 3D de protéines (liaison avec PyMOL...)
  - **PED / MAP** : formats de généalogie et génotypage (PLINK)
  - **Tar.gz** : conteneur compressé pour regrouper logiciels ou gros jeux de données



## ▼ Session 2: Introduction Pratique à Linux pour la bio-informatique

### 😊 Au menu 🍽️

- Théorie générale sur Linux
- Installation
- Interpréteurs (shells)
- Commandes de base et structure d'une commande

## LET'S HAVE FUN WITH LINUX COMMAND LINE



- Navigation dans le système de fichiers
- Manipulations textuelles et analyse de données
- Téléchargement/transfert
- Compression et archivage

### ▼ Qu'est-ce qu'Unix/Linux ?

Unix est un **système d'exploitation** stable, puissant et multi-utilisateur, initialement conçu dans les années 1970, qui sert à contrôler un ordinateur et à gérer ses ressources (processeur, mémoire, disque, périphériques).

Il est à la base de nombreuses variantes modernes comme GNU/Linux, macOS ou BSD.

#### Caractéristiques fondamentales d'Unix

##### Il est **Multi-utilisateur**

Unix permet à plusieurs utilisateurs de travailler sur le même système simultanément, chacun avec un environnement isolé.

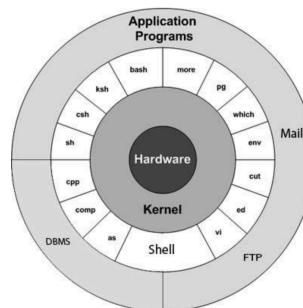
C'est une propriété essentielle pour les systèmes partagés (ex. clusters ou serveurs distants)

##### Il est **Multitâche**

Un même utilisateur peut exécuter plusieurs tâches ou programmes en parallèle. Le système répartit les ressources de manière efficace pour garantir que les processus tournent sans interférence.

##### Il a une **Architecture fondée sur un noyau (kernel)**

Le noyau est le cœur du système Unix.



#### **Noyau (Kernel)**

Le noyau est le cœur du système d'exploitation. Il interagit directement avec le matériel et prend en charge la majorité des tâches critiques telles que la gestion de la mémoire, l'ordonnancement des processus (gestion multitâche) et la gestion des fichiers.

#### **Shell (Interpréteur de commandes) !!!!**

Le shell est l'utilitaire qui interprète les requêtes de l'utilisateur. Lorsque vous tapez une commande dans le terminal, le shell l'interprète et exécute le programme correspondant. Tous les shells utilisent une syntaxe standard pour les commandes. Les shells les plus connus et largement disponibles dans les différentes variantes d'Unix sont le **C Shell**, le **Bourne Shell** et le **Korn Shell**.

▼ More?

Un **interpréteur de commandes**, ou **shell**, est un programme qui permet à l'utilisateur d'interagir avec le système d'exploitation en tapant des commandes dans un terminal. Il agit comme une interface entre l'utilisateur et le noyau (kernel) de Linux.

Lorsque vous saisissez une commande (comme `ls` ou `cd`), le shell analyse cette commande, vérifie sa syntaxe, et appelle le programme correspondant.

Il peut aussi chaîner plusieurs commandes, rediriger les sorties, gérer des variables, des boucles et des tests conditionnels.

Il existe plusieurs interpréteurs, chacun avec ses particularités :



- `sh` (Bourne Shell) : simple et très portable.
- `bash` (Bourne Again Shell) : le plus couramment utilisé, riche en fonctionnalités.
- `zsh` (Z Shell) : interactif et personnalisable, souvent apprécié pour le confort de l'utilisateur.
- `csh`, `tcsh`, `ksh` : autres shells avec des syntaxes ou usages particuliers.

Le shell par défaut dépend de la configuration du système et de l'utilisateur.

Il est possible de connaître son interpréteur avec la commande `echo $SHELL`, et de le changer temporairement ou définitivement.

### **Commandes et utilitaires**

Il existe de nombreuses commandes et utilitaires que vous pouvez utiliser dans vos activités quotidiennes.

Il existe plus de 250 commandes standard, sans compter les nombreuses autres disponibles via des logiciels tiers. Chaque commande dispose généralement de plusieurs options pour en adapter le comportement.

### **Fichiers et répertoires**

| dans Unix, *tout est fichier*

Toutes les données sous Unix sont organisées sous forme de fichiers.

Ces fichiers sont regroupés dans des répertoires (dossiers), eux-mêmes organisés selon une structure hiérarchique en forme d'arborescence, appelée **système de fichiers** (filesystem)

### **De Unix à Linux : continuité et évolutions**

**Linux** est un système d'exploitation de type Unix, libre et open-source, créé dans les années 1990 par Linus Torvalds. Il reprend les principes fondateurs d'Unix (structure modulaire, outils en ligne de commande, architecture multi-utilisateur/multitâche), mais avec une licence ouverte qui a favorisé son adoption massive dans le monde académique et scientifique.

Aujourd'hui, Linux n'est pas une seule entité mais une famille de systèmes : Ubuntu, Debian, CentOS, Fedora, etc. Toutes ces distributions partagent le même noyau et les mêmes principes Unix, avec des outils et interfaces adaptés à différents besoins.

## **Pourquoi Linux est-il si prisé en bio-info ?**

La bio-informatique repose sur le traitement de gros volumes de données et sur l'enchaînement automatique de nombreuses étapes d'analyse. Dans ce contexte, Linux présente plusieurs avantages majeurs :

- **Compatibilité avec la majorité des outils bioinformatiques** : la plupart des logiciels d'analyse (bwa, samtools, bowtie2, blast, etc.) sont conçus pour fonctionner en ligne de commande sur Linux.
- **Stabilité et robustesse** : Linux est reconnu pour son fonctionnement fiable, même sous forte charge (analyses lourdes, pipelines parallélisés).
- **Automatisation et scripting** : grâce à bash, awk, sed, etc., on peut automatiser l'analyse de milliers de fichiers, sans interface graphique.
- **Écosystème libre et communautaire** : de nombreux outils sont open-source, et partagés sur des plateformes comme GitHub, facilitant la collaboration et la reproductibilité.
- **Adapté au travail à distance** : Linux permet un accès sécurisé aux serveurs par SSH, ce qui est indispensable pour travailler sur des clusters ou serveurs de calcul.

## ▼ Installation + prise en main de l'environnement

### 1. Installation sur une machine virtuelle

Tester Linux sans affecter son système principal.

Exemple: VirtualBox pour installer une distribution (ex. Ubuntu) dans une machine virtuelle

### 2. Dual boot

Installer Linux aux côtés de Windows ou macOS sur une même machine.

Cette méthode offre de meilleures performances mais demande plus de précautions

### 3. Connexion à un serveur distant

Très fréquent en bioinfo via connexion SSH.

### 4. Utilisation dans le cloud ou via un conteneur :

des plateformes comme Google Colab, AWS, ou des environnements Docker peuvent offrir un accès temporaire à Linux pour exécuter des scripts

## ▼ Commandes de base et structure d'une commande

▼ Mais d'abord...

## Quelles sont les 5 commandes que vous utilisez le plus?

Answer here:



▼ Mentimeter

[menti.com](https://menti.com)

La structure d'une commande en ligne Linux est généralement la suivante :

commande [options] [arguments]

La **commande** est le nom du programme que vous souhaitez exécuter (les commandes que nous verrons plus bas sont des programmes en réalité, à la place nous pouvons aussi avoir un outil tel que fastp, trimomatic...)

Les **options** modifient le comportement de la commande, et les arguments spécifient les fichiers ou les données sur lesquels la commande doit agir.

Elles sont généralement précédées d'un tiret “-” ou de deux tirets pour les options plus longues “—”

Les **arguments** sont les données ou les fichiers sur lesquels la commande doit opérer ou les fichiers à traiter

### Exemples:

- `grep "mot" fichier.txt`
- `rm -f fichier.txt`

**50+ Essential Linux Commands: A Comprehensive Guide | DigitalOcean**  
Master essential Linux commands for file management, process control, networking, and system administration, from basic to advanced Linux commands  
<https://www.digitalocean.com/community/tutorials/linux-commands>



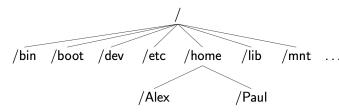
Golden rule !!

**Read your man**

## ▼ Navigation dans le système de fichiers

Naviguer dans le système de fichiers consiste à se déplacer entre les dossiers et à consulter les fichiers.

Sous Linux, le système est organisé en arborescence à partir de la racine `/`



Répertoire	Fonction principale
<code>/</code>	Racine du système de fichiers (point de départ de toute l'arborescence)
<code>/bin</code>	Programmes exécutables essentiels accessibles à tous les utilisateurs
<code>/boot</code>	Fichiers nécessaires au démarrage de Linux (ex: noyau, grub, initrd)
<code>/dev</code>	Fichiers représentant les périphériques (disques, imprimantes, ports, etc.)
<code>/etc</code>	Fichiers de configuration du système (réseau, utilisateurs, services...)
<code>/home</code>	Répertoires personnels des utilisateurs ordinaires
<code>/lib</code>	Bibliothèques partagées nécessaires au fonctionnement du système
<code>/mnt</code>	Point de montage pour des périphériques temporaires (clé USB, CD-ROM...)

/opt	Logiciels supplémentaires installés manuellement ou en dehors du gestionnaire
/proc	Répertoire virtuel contenant des infos système (mémoire, CPU, processus...)
/root	Répertoire personnel de l'utilisateur <b>root</b> (administrateur système)
/usr	Applications et fichiers partagés des utilisateurs (hiérarchie secondaire)
/var	Données variables : logs, mails, files d'attente, cache, etc.
/tmp	Fichiers temporaires, supprimés souvent au redémarrage

Note: Les fichiers cachés commencent par un .

#### Usual commands

```
pwd
cd # ~ : répertoire personnel de l'utilisateur , - : veut dire répertoire précédent
ls #(with options)
tree
```

### ▼ Gestion de fichiers

```
# Création
touch <fichier> — créer un fichier vide

mkdir <dossier> — créer un dossier

mkdir -p <chemin> - - créer une arborescence complète de dossiers

# Copie
cp <source> <destination> — copier un fichier

cp -r <source> <destination> — copier un dossier (récuratif)

# Déplacement / renommage
mv <source> <destination> — déplacer ou renommer un fichier ou dossier

# Suppression
rm <fichier> — supprimer un fichier

rm -r <dossier> — supprimer un dossier et son contenu

rm -f <fichier> — forcer la suppression (sans confirmation)

rm -rf <dossier> — suppression récursive forcée (dangereux)

# Affichage du contenu
cat <fichier> — afficher le contenu d'un fichier

less <fichier> — lecture page par page

head <fichier> — afficher les premières lignes
```

```

tail <fichier> — afficher les dernières lignes

file <fichier> — identifier le type d'un fichier

# Informations
ls -l — afficher les détails (taille, permissions, date)

```

## ▼ Commandes de gestion des droits d'accès

```

#Visualisation des permissions
ls -l

# Modification des permissions
chmod <mode> <fichier> — change les permissions

Exemples :
chmod 755 script.sh (mode numérique)

chmod u+x script.sh (mode symbolique : ajoute exécution pour le propriétaire)

# Modification du propriétaire
chown <nouveau_PROPRIÉTAIRE> <fichier> — change le propriétaire

chown <utilisateur>:<groupe> <fichier> — change propriétaire et groupe

Exemple : chown marieme:admin monscript.sh

# Modification du groupe
chgrp <groupe> <fichier> — change seulement le groupe associé au fichier

# Autres options utiles
chmod -R ... ou chown -R ... — application récursive à un dossier et son contenu

Exemple : chmod -R 755 mon_dossier/

```

Type	Symbole	Signification	S'applique à...
Lecture	r	Lire le contenu	Fichiers : voir le contenuDossiers : lister les fichiers ( ls )
Écriture	w	Modifier le contenu	Fichiers : modifier, supprimerDossiers : créer, supprimer fichiers
Exécution	x	Exécuter un fichier ou entrer dans un dossier	Fichiers : exécuter (scripts, binaires)Dossiers : y accéder ( cd )

Catégorie	Désignation	Exemple dans ls -l
Propriétaire	u (user)	-rwx----
Groupe	g (group)	-rwxr-x--
Autres	o (others)	-rwxr-xr-

rwx	Valeur numérique
-----	------------------

---	0
--x	1
-w-	2
-wx	3
r--	4
r-x	5
r-w-	6
rwx	7

### ▼ Exercice

- Quelle est la commande pour donner uniquement les droits `rwx-----` à un fichier ?
- Quelle différence y a-t-il entre les droits d'un fichier et ceux d'un dossier ?
- Que se passe-t-il si un fichier n'a pas le droit `x` mais est un script ?
- Quelle commande permet de voir précisément qui est le propriétaire d'un fichier ?

## ▼ Manipulations textuelles

Le traitement de fichiers texte est essentiel en bio-informatique

Des commandes comme `grep`, `cut`, `wc`, `sort`, `uniq` permettent de rechercher et manipuler des lignes et colonnes.

Les expressions régulières permettent des recherches complexes de motifs.

Des outils plus puissants comme `awk` (analyse ligne par ligne et colonnes) et `sed` (remplacement/filtrage) sont idéaux pour des traitements automatiques.

`rename` permet de renommer par lot

### Use case:

- Supprimer une colonne d'un CSV avec awk
- Renommer automatiquement des fichiers FASTQ avec rename

## ▼ Les redirections

Symbole	Fonction	Exemple
>	Redirection de la sortie standard vers un fichier	<code>commande &gt; fichier.txt</code>
<	Redirection de l'entrée standard depuis un fichier	<code>commande &lt; fichier.txt</code>
>>	Redirection de la sortie standard avec concaténation	<code>commande &gt;&gt; fichier.txt</code>
		Pipe : redirige la sortie standard vers l'entrée standard

## ▼ Téléchargement/transfert

De nombreux outils en bio-informatique nécessitent de télécharger ou partager des fichiers.

`wget` et `curl` permettent de récupérer des données sur Internet

`ssh` permet une connexion sécurisée à un serveur distant

`scp` et `sftp` permettent de transférer des fichiers

## ▼ Compression et archivage

La compression permet de réduire la taille des fichiers pour l'archivage ou le transfert.

`gzip` et `gunzip` compressent/décompressent un fichier

`tar` permet de regrouper plusieurs fichiers en une seule archive

`zip` et `unzip` sont plus courants sur d'autres systèmes (Windows)

La compression est utile pour sauvegarder des projets ou envoyer des données sur un serveur

## ▼ Pratiquons



### Instructions

Créer ce fichier `my_file.txt` contenant:

```
Name,Age,Country  
Yama,20,Senegal  
Alice,25,Nigeria  
Charles,35,Ghana
```



Extraire la seconde colonne dans un fichier `age.txt`

hint : commande à utiliser '`cut`'

Le fichier est un csv (**comma separated values**) donc? la virgule sert de delimiteur

pour la redirection vers un fichier utiliser '`>`'



**exple:** ls \*.fasta > mes.fasta.txt

```
cut -d , -f2 my_file.txt > age.txt
```

Puis supprimer cette seconde colonne du fichier

**Structure de dossiers** pour le projet:

Créer un dossier "analyses" qui contiendra les dossiers :

- data (contenant raw et clean)
- results

Allez dans `data/` et Téléchargez le fichier FASTA depuis NCBI: `wget -q -O my_L001.fasta.fa "https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=nuccore&id=NG_049243.1&rettype=fasta"`

Déplacez les fasta dans `data/raw`

afficher les 10 premières lignes de votre fichier fasta : `head`

Renommez le fichier afin de supprimer "L001\_" dans le nom

Comptez le nombre de séquences dans le fichier des fichiers FASTA (`grep '>' my.fasta.fa` )

## ▼ Session 3: Automation

Un script shell est un fichier texte contenant une suite de commandes qui seront exécutées automatiquement dans un interpréteur (souvent bash ou sh). Cela permet d'automatiser des tâches répétitives, comme le traitement de fichiers, la navigation dans des dossiers, ou l'exécution d'un pipeline bioinformatique.

### ▼ Mais d'abord, où est ce qu'il faut l'écrire?

#### Éditeurs de texte sous linux

Un **éditeur de texte** est un programme qui permet d'ouvrir, modifier et sauvegarder des fichiers texte.

En Linux, on les utilise souvent pour écrire des **scripts shell**, éditer des **fichiers de configuration**, programmer dans divers langages (Python, C, etc.)

Exemples:

nano ,vi ,vim ,emacs...

You can use cheat sheets 🧙

```
www.atmos.albany.edu  
https://www.atmos.albany.edu/daes/atmclasses/atm350/vi_cheat_sheet.pdf
```

Un script shell commence par une ligne appelée **shebang**, qui indique quel interpréteur utiliser.

▼ C'est quoi un interpréteur déjà?

Une interface entre l'utilisateur et le système d'exploitation

Par exemple

```
#!/bin/bash  
#!/bin/zsh  
#!/usr/bin/env bash
```

Comment savoir quel est notre interpréteur?

```
echo $SHELL
```

Ensuite, on écrit les commandes comme si on les tapait dans le terminal.

On peut également utiliser des variables, des conditions (**if**), des boucles (**for**, **while**) et des fonctions dans l'algorithme de notre script

▼ Exemple simple de script `hello.sh` :

```
#!/bin/bash  
  
# good practice : je commente mon code  
  
name=$1 # stocke la variable passée en argument du script  
echo "Bonjour, $name!"
```

## How to make it executable?

```
chmod +x hello.sh  
.hello.sh xxxx
```

Que vous retourne le script?

Si je remplace \$name par \$USER qu'est ce que j'obtiens?

## Complexifier

### ▼ add condition

Syntaxe:

```
#!/bin/bash

if [ condition ]; then
    # commandes si la condition est vraie
elif [ autre_condition ]; then
    # commandes si l'autre condition est vraie
else
    # commandes si aucune condition n'est vraie
fi
```

Exemple:

```
#!/bin/bash

# good practice : je commente mon code
name=$1
if [ -z "$name" ]; then # si $name est vide
    echo "Usage : ./hello.sh [nom]"
else
    echo "Bonjour, $name !"
fi
```

▼ Exo: Script qui demande à l'utilisateur un nombre puis détermine si celui-ci est supérieur, inférieur ou égal à 10

```
#!/bin/bash

echo "Entrez un nombre :"
read nombre

if [ "$nombre" -gt 10 ]; then
    echo "Ce nombre est supérieur à 10"
elif [ "$nombre" -eq 10 ]; then
    echo "Ce nombre est égal à 10"
else
    echo "Ce nombre est inférieur à 10"
fi
```

hint: <https://github.com/RehanSaeed/Bash-Cheat-Sheet>

### ▼ loops

Syntax

```
for variable in valeur1 valeur2 valeur3; do
    # commandes
done
```

Exemples

```

for nom in A B C; do
    echo "Bonjour $nom"
done

for fichier in *.txt; do
    echo "Fichier trouvé : $fichier"
done

```

## ▼ case

### Syntaxe

```

case $variable in
pattern-1)
    commands;;
pattern-2)
    commands;;
pattern-3)
    commands;;
pattern-N)
    commands;;
*)
    commands;;
esac

```

### Exemple

```

#!/bin/bash

echo "Quelle couleur préférez-vous ?"
echo "1 - Bleu"
echo "2 - Rouge"
echo "3 - Jaune"
echo "4 - Vert"
echo "5 - Orange"

read color;
case $color in
    1) echo "Le bleu est une couleur primaire";;
    2) echo "Le rouge est une couleur primaire";;
    3) echo "Le jaune est une couleur primaire";;
    4) echo "Le vert est une couleur secondaire";;
    5) echo "Le orange est une couleur secondaire";;
    *) echo "Cette couleur n'est pas disponible";;
esac

```

## Reproducibility

### Principe

[https://opensilex-scripts.pages-forge.inrae.fr/opensilex-generic-scripts/best\\_practices/resources/conda\\_environment](https://opensilex-scripts.pages-forge.inrae.fr/opensilex-generic-scripts/best_practices/resources/conda_environment)

## Cheat Sheet

[https://docs.conda.io/projects/conda/en/4.6.0/\\_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf](https://docs.conda.io/projects/conda/en/4.6.0/_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf)

## Useful documentation

<https://github.com/RehanSaeed/Bash-Cheat-Sheet>

## Connexion à un serveur :

```
bash

# server copy command scp

scp -r standard_db playground@10.1.7.38:/home/playground/kraken_db/
kraken2-build --build --db /home/playground/kraken_db/standard_db

# Instructions

##### Connecting to remote server
# syntax : ssh <username>@<IP_adress>

ssh playground@10.1.7.38
password: GoodP@$wd2023

# !\ Vous vous trouvez dans /home/playground

##### create file with usernames

touch usernames # creer un fichier vide usernames
echo "mtop" >> usernames # l operateur '>>' ajoute "mtop" a la fin du fichier usernames
```

## Script de création des répertoires

```
#!/bin/zsh

while IFS= read -r username; do # lit chaque ligne du fichier
    echo "Creating $username" # affiche le nom de l'utilisateur
    mkdir -p users/$username # option -p fera que le dossier parent "users" sera crée s'il n'est pas présent
done < usernames # operateur < specifie que usernames est le fichier d'entree
```

\*\*Simulation\*\*

your user "account" : /home/playground/users/<the\_username\_you\_provided>

```
pour moi : /home/playground/users/mtop  
##### /!\ W A R N I N G ##### ##### ##### ##### #####
```

!! Note: ne vous déplacez pas dans l'arborescence des fichiers.  
Executez vos commandes à partir de votre répertoire personnel

Par contre, dans votre "compte" vous pouvez créer vos propres fichiers

Question?

un cd va vous ramener où?  
use pwd , et s'assurer que le chemin soit :

```
##### ##### ##### ##### ##### ##### ##### ##### #####
```

Une fois que vous êtes dans votre "compte"

```
##### working with environments  
#conda create --name workshop bioconda::kraken2  
#conda install bioconda::kraken2  
  
##### activer votre environnement  
conda activate workshop  
  
# tapez "kraken2" pour vous assurer que vous avez bien accès à l'outil
```

Vous devez maintenant copier vos fichiers de votre ordinateur vers le serveur

Pour cela, ouvrez un terminal sur votre machine locale

```
# Si vous êtes dans le répertoire contenant le fichier que vous voulez copier dans le serveur  
  
Syntax:  
scp <your_file> <username>@<server_IP_address>:</chemin/vers/le/répertoire/de/destination>  
  
Application:  
scp my.fasta fa playground@10.1.7.38:/home/playground/users/mtop  
  
# Par contre, si vous êtes un autre répertoire  
  
Syntax:  
scp </chemin/vers/your_file> <username>@<server_IP_address>:</chemin/vers/le/répertoire/de/destination>  
  
Application:  
scp /Users/marieme/Desktop/my.fasta fa playground@10.1.7.38:/home/playground/users/mtop  
  
# Le mot de passe du user: playground vous sera demandé à nouveau
```



Pour les petits fichiers scp peut faire l'affaire

Par contre pour les gros volumes de données la commande **rsync** est conseillée

Read the doc 😊

Maintenant vous pouvez lancer votre analyse avec kraken2.

Une base de données partagée a été téléchargée en amont

```
##### Useful paths
chemin vers la BD: /home/playground/kraken_db/standard_db
#####
kraken2 --threads 2 --db /home/playground/kraken_db/standard_db --output out.txt --report report.txt NC_0264

# copy output du serveur vers la machine : commande à taper dans la console de votre machine
scp playground@10.1.7.38:/home/playground/users/mtop/* .
```

**kraken2 --threads 2 --db /home/playground/kraken\_db/standard\_db --output out.txt --report report.txt <yourfile>.fasta**