Formation Linux pour les utilisateurs du CTIG

Module 2 : Linux commandes

Création de la formation
à partir des supports

Pascal Croiseau (GABI)
Olivier Filangi (PEGASE)
Sylvie Nugier (CTIG)
François Laperruque (SAGA)

Linux

- Système d'exploitation :
- Multi taches
- Multi utilisateurs

- GNU/Linux est un Unix sur des architectures type x86 (32 et 64 bits).

- Composé :
- d'un noyau,
- d'un système de gestion de fichiers hiérarchisé,
- des interpréteurs de commandes,
- des commandes,
- d'utilitaires
- de services de communication
- d'un environnement graphique
- d'applications pour les utilisateurs.

Linux

- Logiciels libres
  - La liberté d'exécuter le programme, pour tous les usages
  - La liberté d'étudier le fonctionnement du programme, et de l'adapter à vos besoins : l'accès au code source
  - La liberté de redistribuer des copies, donc d'aider votre voisin
  - La liberté d'améliorer le programme et de publier vos améliorations, pour en faire profiter toute la communauté
- Distributions packagées : **RedHat**, SuSE, Debian,Mandriva, Ubuntu, Fedora

Plan

- Le système de fichier
- Commandes de manipulation de fichiers
- Transfert des fichiers entre machines distantes
- Compression/Décompression/Archive
- Extraction d'information des fichiers
- Les processus sous Linux
- Les scripts SHELL

Le système de fichier

CTIG – CATI IPBI
Formation Linux pour les utilisateurs du CTIG

Les types de fichiers

- Répertoire, fichier, lien (raccourci)
- **Se déplacer dans l'arborescence : cd**
- **Lister les éléments d'un répertoire : ls**
  - ls [-options] [nom_répertoire]
  - -a : affiche les fichiers cachés
  - -l : utilise le format long
  - -h : ajouter une lettre indiquant l'unité de taille
  - -t : trie
  - -r : inverse l'ordre
- Détails : **man** ls

Exemples :
ls -ltr
ls -als
ls -ls *
ls -lh

lien

- **crée un lien symbolique : ln**
- *ln -s nom_fichier nom_lien_symbolique*

snugier@dga12:/formation/exemples# echo TOTO > ficTest

snugier@dga12:/formation/exemples# ls -ls

0 -rw-r--r-- 1 snugier formation 5 23 avril 16:21 ficTest

snugier@dga12:/formation/exemples# cd ..

snugier@dga12:/formation# ln -s exemples/ficTest ficTestSymb2

snugier@dga12:/formation# ls -ls ficTestSymb2

0 lrwxrwxrwx 1 snugier formation 16 23 avril 16:23 ficTestSymb2 -> exemples/ficTest

- Les liens sont utiles si vous souhaitez qu'un fichier apparaisse dans plusieurs répertoires, ou sous un nom différent. Si le fichier est encombrant une copie par cp entraînerait un gâchis alors qu'un lien permettra de réduire l'utilisation d'espace disque.

Droits sous Linux

Type – Droits propriétaire – Droits groupe – Droits autres

Fichier
r : lecture
w : écriture
X : exécution

Répertoire
r : lecture du contenu
w : ajout ou suppression d'un fichier (quelque soit les droits du fichier)
X : autorise le passage (accès au contenu)

```
[ofilangi@dga11 exemple]$ ls -l
total 72
-rw-r--r-- 1 ofilangi ugen 52121 20 déc.  11:29 main.f90
-rw------- 1 ofilangi ugen    65 20 déc.  11:28 myprivatefile
-rw-rw---- 1 ofilangi ugen  9051 20 déc.  11:29 notice.html
```

Droits – Nb liens – Propriétaire

Groupe – Taille – Date – Nom

Droits d'accès

- **Modification des autorisations : chmod**
  - chmod [options] [*droits*] [nom_fichier|repertoire]
  - u : user, g : groupe, o : other, a : all
  - (=) appliquer, (+) ajouter, (-) enlever
  - r : lecture, w : écriture, x : exécution
  - **-R** : changement récursif de tous les objets de l'arborescence

chmod g+w repertoire1	Autorise les utilisateurs de votre groupe à ajouter et à supprimer des fichiers dans repertoire1.
chmod u=rwx cmd1 cmd2	Donne les droits de lecture, écriture et exécution au fichier cmd1 et cmd2 pour le propriétaire seulement.
chmod –R g+w fic*	Donne les droits d'écriture aux utilisateurs du groupe pour tous les fichiers et répertoires commençant par fic dans toute l'arborescence.
chmod go-x rep1	Supprime le droit d'accès au groupe ainsi qu'à tous les utilisateurs

Droits d'accès

- **Changement du groupe : chgrp**
  - chgrp [options] [groupe] [nom_fichier|repertoire]
  - -R : changement récursif de tous les objets de l'arborescence
  - Ex : chgrp -R nvgroupe repertoire1

Commandes de manipulation de fichiers

CTIG – CATI IPBI
Formation Linux pour les utilisateurs du CTIG

Manipulation de fichier

- **Création / suppression de fichier / répertoire**
  - **mkdir / rmdir [nom_rep]** : crée / supprime un répertoire
    - *mkdir nouveau_repertoire*
    - *mkdir -p rep1/repA/nvrep (Crée les répertoires parents s'ils manquent).*
  - **touch / rm [nom_fichier]** : créé fichier vide / supprime un fichier
    - *rm -rf* : attention danger !
    - rm -i : pour avoir une confirmation

Manipulation de fichier

- **La copie de fichiers**
  - **cp** nom_fic_src nom_fic_dest :
    - => copie de fichiers
    - *cp fichier1 fichier2*
  - **cp -r** nom_rep_src nom_rep_dest
    - => copie de répertoires
    - *cp -r rep1 rep2*
  - **cp -p** nom_rep_src nom_rep_dest
    - *lors de la copie les attributs du fichier seront préservés (propriétaire, groupe, **date**...).*

Manipulation de fichier

- **Déplacer / renommer un fichier**
  - **mv** nom_fic nom_rep
  - Déplace :
    - *mv nom_fichier répertoire_accueil*
  - Renomme :
    - *mv ancien_nom_fichier nouveau_nom_fichier*
  - Déplace et renomme :
    - *mv ancien_nom_fichier répertoire_accueil/nouveau_nom_fichier*

Recherche de fichier

- **find** nom_rep [-option] [paramètre]
  - **find /home/jdupont -name "*.Z" :** les objets se terminant par « .Z »
  - **find . -type d** : uniquement des dossiers
  - **find . -type f** : uniquement des fichiers
  - **Ne pas faire : find / -size +1000k** : si taille > 1Mo

    Parcours de tous les espaces disques attachés au serveur (consommateur en CPU) !

Des caractères spéciaux ...

- **Les caractères jockers**
  - **?** équivaut à un seul caractère
    - *ls bov?.seq*
  - ***** équivaut à 0, 1 ou plusieurs caractères
    - *ls *.seq*
    - *rm bacterie**
  - **[ ]** équivaut à un caractère parmi une sélection
    - *ls [123]**
    - *ls f[a-c]**

Des caractères spéciaux ...

- **Entrée / sortie standard**
  - La plupart des commandes utilisent les entrées / sorties standards : **entrée standard = clavier, sortie standard = terminal**
  - Il est possible de rediriger les E/S avec les opérateurs « **<** », « **>** », « **|** », « **>>** »
  - **Remarques** :
    - « 2 > » pour la redirection de la sortie des erreurs (stderr) ;
    - « & > » pour rediriger en même temps la sortie standard et la sortie erreur.

Des caractères spéciaux ...

- **Redirection**
  - **commande > nom_fichier_sortie** : redirige la sortie standard vers un nouveau fichier
    - *date > fic_result.csv*
  - **commande1 | commande2 :** redirige la sortie standard vers un autre prog.
    - *ls -ltr | more*
  - **commande >> nom_fichier_sortie :** redirige la sortie standard vers un fichier existant et ajoute en fin de fichier
    - *ls -ltr >> fic_result.csv*

Des caractères spéciaux ...

- **Et encore ...**
  - **.** le répertoire courant (où l'on se trouve)
    - *ls -lsd .*
  - **..** le répertoire supérieur (le père)
    - *cd ../..*
  - **~** le répertoire de base (home directory)
    - *cp ~/fic1* **.**
  - **&** lancement d'une commande en arrière plan. La ligne de commande n'est pas bloquée.
    - *xterm &*

Les pagineurs

- **Afficher le contenu d'un fichier**
  - **cat [-options] nom_fichier** : lit / concatène un fichier
    - *cat nom_fic1 nom_fic2 > nom_fic_3*
  - **more** nom_fichier : visualise page par page
    - balayage par la touche « espace »
    - Permet de faire des recherches (touche « / »)
    - sortir avec le « q »
  - **less** nom_fichier : visualise avant + arrière
    - balayage par les flèches « haut » et « bas »
    - recherche caractère par la touche « / »
    - sortir avec le « q »

Les éditeurs de texte

- **Modifier le contenu d'un fichier**
  - **vi** : standard mais peu ergonomique
  - **vim** : vi (amélioré : syntaxique ...)
  - **nano** : simple d'utilisation
  - **nedit** : en mode graphique, assez intuitif
  - **gedit** : standard, en mode graphique, assez intuitif
  - **emacs** : dispose de fonctions évoluées

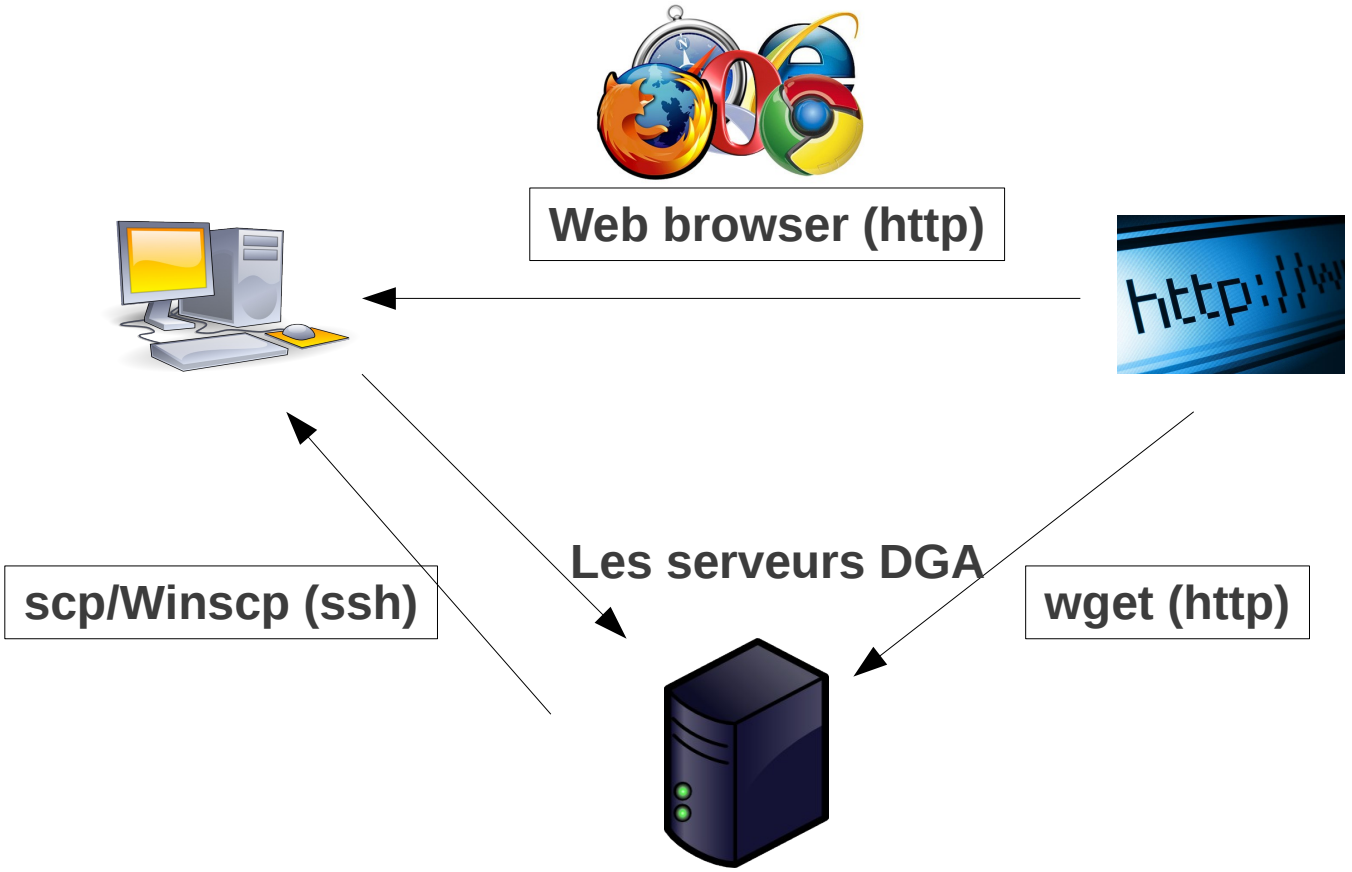
Contrôle de l'espace disque

- **df [-option] [nom_partition]** :
  - Affiche l'espace disque des partitions
    - *df -h :* format « humain »
- **du [-option] [nom_répertoire]** :
  - Affiche l'espace disque par répertoire
    - *du -sh* : espace global utilisé

Transfert des fichiers entre machines distantes

CTIG – CATI IPBI
Formation Linux pour les utilisateurs du CTIG

Télécharger / Transférer



Télécharger / Transférer

- **Directement depuis internet**
  - Téléchargement de fichiers depuis Internet :
    - Copier l'url du fichier à télécharger
    - **wget** http://url.a.telecharger/nom_fichier
    - La commande wget enregistre dans le répertoire courant le fichier

Télécharger / Transférer

- **Transférer**
  - Utilisation de **scp** (secure copy)
    - **scp [user@host1:]file1 [user@host2:]file2 :** copie de fichier à travers le réseau
    - *scp fichier_source user@dga11.jouy.inra.fr:destination* (copie depuis le poste local vers le serveur "dga11")
  - *WinSCP, Filezilla* : copie via interface graphique

Exercices

- Télécharger les fichiers jeux de données : pedigree_qtlmas_2009 et phenotype_qtlmas_2009 - https://forge-dga.jouy.inra.fr/projects/sform-ctig/files
  - Sur votre machine puis transferer ces fichiers sur DGA12 en utilisant scp, winswp (ou filezilla)
  - Directement sur la machine DGA12 en utilisant wget

Compression/Décompression/Archive

CTIG – CATI IPBI
Formation Linux pour les utilisateurs du CTIG

Décompresser / Compresser

- **Plusieurs formats**
  - **gzip** : compresse un fichier en .gz
    - *gzip fichier_à_compresser* =>création d'un fichier.gz
  - **gunzip ou gzip -d** : décompresse un fichier .gz
    - *gunzip fichier.gz*
- Autres formats : bz2, zip, rar, Z

Archiver / Désarchiver

- **Tar**
  - **tar -cvf** : archive une arborescence
    - tar -cvf formation.tar /home/formation => création d'un fichier.tar
  - **tar -xvf** : redéploie une arborescence
    - tar -xvf formation.tar /tmp
- Astuce: combinaison du tar + gzip (.tgz)
  - **tar -cvzf** : archive + compression
  - **tar -xvzf** : désarchive + décompression

pigz et pbzip2

Compression/Décompression en parallèle

- **Pigz**
  - utilise les librairies zlib et pthread
  - capable de mobiliser tous les CPUs disponibles
  - utilise les mêmes arguments que gzip et peut donc le remplacer directement
  - un fichier compressé avec pigz sera décompressé avec gzip et inversement
- **Pbzip2**
  - compatible avec bzip2
- Les durées d'exécution sont quasiment inversement proportionnelles au nombre de processeurs disponibles

Extraction d'information des fichiers

CTIG – CATI IPBI
Formation Linux pour les utilisateurs du CTIG

Les commandes utilitaires

- sort [-options] nom_fichier : trie un fichier
  - *sort -n -k 1* : trie numérique 1ère col.
- wc [-options] nom_fichier : compte les mots
  - *wc -c nom_fichier* : compte les caractères
  - *wc -w nom_fichier* : compte les mots
  - *wc -l nom_fichier* : compte les lignes

Extraction de données

- **Les filtres (1)**
  - **head [-nombre] nom_fichier** : lit le début d'un fichier
    - *head -100 nom_fic* (100 premières)
  - **tail [-f] [+/-nombre] nom_fichier** : lit la fin de fichier
    - *tail **-f** /var/log/message* (actualise)
    - *tail -n 100 nom_fichier* (100 dernières)
    - *tail -n +6 nom_fichier* (à partir de la 6)

Extraction de données

- **Les filtres (2)**
  - **cut [-options] nom_fichier** : découpe les champs (verticalement)
    - *cut -c 1* (récupère les premier car.)
    - *cut -f 2,3* (récupère les champs 2 et 3)
    - *cut -d "|"* (séparé par un délimiteur)
  - **split [-options] nom_fichier** (csplit) : découpe les champs (horizontalement)
    - *split -l 500 fichier.txt fichier_split*. (découpe toutes les 500 lignes, les fichiers s'appelleront _split.*)

Extraction de données

- **Recherche textuelle**
  - **grep [-options] 'motif' nom_fichier[s]**
    - Outil de recherche textuelle dans les fichiers
    - Utilisation possible des jokers
      - *grep SEQRES fichier_pdb* (recherche simple)
      - *grep -i* (ne distingue pas la casse)
      - *grep -c* (compte le nombre de ligne)
      - *grep -n* (affiche le numéro de ligne)
      - *grep -v* (toutes les lignes sauf)

Exercices

- Trier le fichier de généalogie sur la colonne des mères
- Donner l'indice de la plus grande mère
- Donner le nombre de progéniture du père 301

Extraction de données

- **Comparaison de fichiers**
  - **diff [-options] nom_fic1 nom_fic2** : compare deux fichiers ligne par ligne
    - diff fic_1 fic_2
  - **comm [-options] nom_fic1 nom_fic2** : compare des fichiers triés
    - *1ère colonne : lignes uniquement fic1*
    - *2ème colonne : lignes uniquement fic2*
    - *3ème colonne : lignes communes fic1 + fic2*

Extraction de données

- **Redirection**
  - **commande > nom_fichier_sortie** : redirige la sortie standard vers un nouveau fichier
    - *grep -i Human uniprot.fasta > fic_result*
  - **commande1 | commande2 :** redirige la sortie standard vers un autre prog.
    - *grep -i Human uniprot.fasta | wc -l*
  - **commande >> nom_fichier_sortie :** redirige la sortie standard vers un fichier existant et ajoute en fin de fichier
    - *grep -i bovin uniprot.fasta >> fic_result*

Les processus sous Linux

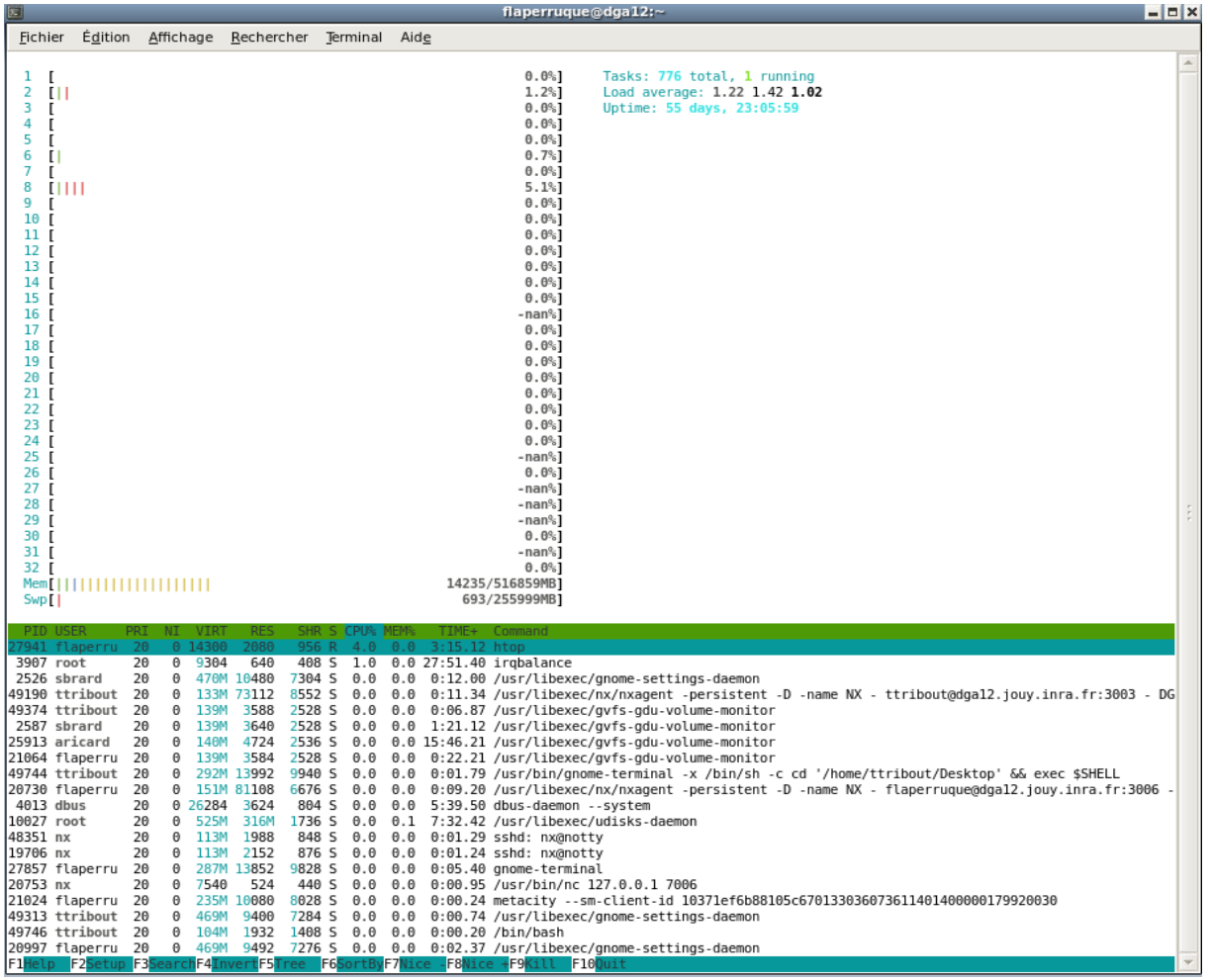
CTIG – CATI IPBI
Formation Linux pour les utilisateurs du CTIG

Affichage des processus en cours

- **Affichage des processus en cours d'exécution et dépendant de la fenêtre d'exécution**
  - ps
- **Affichage de tous les processus en cours d'exécution**
  - ps -e
- **Affichage de tous les processus avec rafraîchissement**
  - top

La commande htop

- **Présentation**
  - htop est un utilitaire similaire à la commande top de Linux, mais beaucoup plus pratique.
- **Fonctionnalités**
  - Permet d'afficher la charge de chaque CPU
  - permet d'utiliser la souris pour choisir un des process affichés, et d'entreprendre une action dessus (changement de priorité, tuer, ...).
  - permet également de faire des sélections de process par filtre, ...
  - le scrolling sur les lignes permet de voir la ligne de commande complète d'un processus



Interagir avec les processus

- A un processus donné peut être envoyé un signal. Ceci peut être vu comme une indication donnée au programme de manière asynchrone
- Un signal sera désigné soit par sa valeur numérique soit par son nom

Signal	Valeur numérique	Comportement par défaut	Description
SIGINT	2	Terminer le processus	Il s'agit d'une demande venant du clavier, le plus souvent à l'aide la combinaison de touches Ctrl-C.
SIGKILL	9	Terminer le processus	Ce signal permet d'arrêter tout programme car il ne peut être géré différemment que le comportement par défaut. L'arrêt du programme est brutal.
SIGUSR1	10	Terminer le processus	Ce signal n'a pas de signification particulière. Il peut être utilisé de manière différente par chaque programme.
SIGSEGV	11	Terminer le processus	Ce signal est envoyé à un programme lorsque qu'il tente d'accéder à un endroit invalide en mémoire.
SIGUSR2	12	Terminer le processus	Identique à SIGUSR1.
SIGTERM	15	Terminer le processus	Si le programme n'a pas prévu de gérer ce signal, l'arrêt sera aussi brutal que pour SIGKILL. Mais comme le comportement par défaut peut être changé, le programme a la possibilité de réaliser des opérations avant de se terminer.
SIGCHLD	17	Ignorer ce signal	Envoyé à un processus dont un fils est arrêté ou terminé.
SIGCONT	18	Reprendre le processus	Permet de faire se continuer un processus qui avait été arrêté par exemple à l'aide de SIGSTOP (voir ci-après).
SIGSTOP	19	Arrêter le processus	Ce signal demande au processus de suspendre son exécution. Comme SIGKILL, ce signal ne peut être géré différemment.

> kill -s SIGSTOP 256
> kill -SIGSTOP 256
> kill -19 256

Lancement en tache de fond

- **Objectifs**
  - Pouvoir fermer sa session sans interrompre les jobs en cours
  - Identifier ses jobs avec un ID
    - utile quand on lance un même programme avec différents jeux de paramètres
  - Pouvoir interrompre l'ensemble des taches liées à un script en 1 étape
    - l'ID représente le script dans son intégralité et pas la sous tache en cours de traitement

Système de contrôle des Jobs sous bash

- Exécution d'un job en arrière plan : &
- CTRL-Z : envoie d'un signal SIGSTOP et rend la main pour utiliser à nouveau le shell
- bg (background) : exécute en arrière plan la dernière commande exécutée (après un CTRL-Z)
- fg (foreground) : exécute la dernière commande (qui est en arrière plan) dans le shell courant
- Lister l'état des commandes en cours d'exécution : jobs

nohup

- Commande linux → man nohup
- nohup monscript > monscript.log &
  - Ne pas oublier le « & »
  - Le fichier log contient l'ensemble des informations en sortie d'écran
- L'ID du nohup n'apparait pas directement avec « top ».
- Pour le récupérer:
  - nohup monscript > monscript.log &
  - numero_process_nohup=$!
  - echo $numero_process_nohup
- Pour le tuer: kill -9 $numero_process_nohup

screen

- Commande linux → man screen
- Il s'agit d'une commande complexe. Seules les options de bases sont présentées ici.
- Une screen est un terminal virtuel. Une fois crée, elle existe tant que l'utilisateur ne l'a pas tuée.
  - Autrement dit, fermer une screen ne la tue pas
  - Le gros avantage est de pouvoir récupérer ses terminaux à tout endroit et à tout moment
  - Ces terminaux sont nommés par l'utilisateur et ont un ID
- En comparaison avec « nohup », les screen ne crée pas de fichier log. Les sorties d'écran apparaissent sur le terminal et l'utilisateur voit plus facilement la progression de ses scripts

screen

- Pour créer une screen: screen -S mascreen
- Pour afficher l'ID des screen existantes et leur statut: screen -r

  There are several suitable screens on:

  22405.mascreen1 (Detached)

  17664.mascreen2 (Detached)

  12178.mascreen3 (Detached)
- Pour fermer une screen:
  - screen -d 12178 ou screen -d 12 depuis un autre terminal
  - ctrl+A+D depuis la screen à fermer
- Pour tuer une screen
  - Exit ou ctrl+D depuis la screen à tuer
  - Kill -9 12178 depuis un autre terminal puis screen -wipe 12178

Les scripts SHELL

CTIG – CATI IPBI
Formation Linux pour les utilisateurs du CTIG

Les langages de scripting

- Langages interprétés
- Shell : sh, ksh, bash, C Shells
- Langages plus évolués
  - Perl, Python

Script

- **Enchaînement de commandes**
- **Interprétation par le shell vs compilation**
- **Avantages** :
  - Automatisation et planification des taches
  - Reproductible : Gain en temps
  - Portabilité : Système Unix
- **Inconvénients** :
  - Syntaxe
  - Message d'erreurs

Que mettre dans un script ?

- **Première ligne (shebang) : chemin de interpréteur**
  - # !/bin/bash
- **Des commentaires**
  - # ceci est un commentaire
- **Des commandes et des variables**
- **Des fonctions**
- **Devermissage de script**
  - # !/bin/bash -x
  - Set -x ... set +x
  - echo

Variables

- Affectation d'une valeur
  - **Locale** (accessible au script seulement) :
    - `MYVAR= genotype.file`
    - `MYVAR= « Hello World »`
  - **Globale** (accessible par les processus fils)
    - `export MYVAR=genotype.file`
  - **Référence au contenu d'une variable** : $
    - `echo $MYVAR`

Variables systèmes

- **Lister les variables systèmes** : export
  - `$HOME,$PATH,$USERNAME,$PWD,$DISPLAY,...`
- **Variables scripts**
  - $# : nombre de paramètres
  - $0 : nom du programme
  - $1,$2,.. : arguments du scripts
  - $? : code retour de la dernière commande
  - $$: PID courant

Interprétation du contenu des variables

- **Simple quote ou apostrophe**
  - Les simples quotes délimitent une chaîne de caractères. Même si cette chaîne contient des commandes ou des variables shell, celles-ci ne seront pas interprétées.
    - `variable="secret" ;echo 'Mon mot de passe est $variable.'` => `Mon mot de passe est $variable`
- **Doubles quotes ou guillemets**
  - Les doubles quotes délimitent une chaîne de caractères, mais les noms de variable sont interprétés par le shell.
    - `variable="secret" ;echo "Mon mot de passe est $variable."` => `Mon mot de passe est secret`
- **Anti-quote**
  - Bash considère que les anti-quotes délimitent une commande à exécuter. Les noms de variable et les commandes sont donc interprétés.
  - `echo  \`variable="connu"; echo "Mon mot de passe est $variable."\`` => `Mon mot de passe est connu.`

Le branchement conditionnel

- **Si** <condition est vrai> **alors** <action1> **sinon** <action2>

```
IF [  "$MYVAR" = "genotype.file" ]; then
      echo "ok"
ELSE
      echo "not ok"
FI
```

- **Les tests possibles**
  - Sur les fichiers : -r , -w, -d, -f, ...
  - Sur les chaines : -n, -z, ...
  - Sur les valeurs numériques : -eq, -gt, -ge, …
- Plus d'infos ? : **man test**

La boucle for

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

```
REP=`ls`
for FILE in $REP
do
  echo "le nom du fichier est $FILE"
done
```

Mémo pour écrire un script

- Expressions arithmétiques
  - u=1
  - echo $((u*2 + 5))
  - 7
- Extraction des chemins et noms de fichier
  - basename /home/mylogin/myfile.txt
  - myfile.txt
  - dirname /home/mylogin/myfile.txt
  - /home/mylogin
- Gestion des fichiers/répertoires temporaires uniques
  - FILETMP=${ARGS}_$$;touch $FILETEMP
  - FILETMP=`mktemp`
  - FILETMP=`mktemp -d`
- Vérifier le nombre d'argument d'un script
  - if ["$#" -lt 1]; then
  - echo "Missing argument !" >&2
  - exit 1
  - fi
- Devermissage de script
  - # !/bin/bash -x
  - Set -x ... set +x
- Nettoyer/rafraichir l'écran
  - clear
  - watch -n 60 ./myscript.sh

Mon premier script

- Un script = un programme qui enchaîne des commandes shell
- Mettre des commandes dans un fichier
  - *nedit prog &*
- Donner les droits d'exécution
  - *chmod +x prog*
- Exécuter le script
  - *./prog*

Mon premier script

- Automatisation et planification
- Gain de temps (ré-utilisable)
- Modèles faciles à trouver sur le web
- Portable (fonctionne sur tous les Unix)
- *Attention toutefois aux différences de syntaxes suivant le shell utilisé (csh,bash...)*

Mon premier script

Script qui affiche le nombre de ligne des fichiers contenus dans un repertoire

```
#!/bin/bash
#positionnement de la variable REPERTOIRE avec la paramètre du script
REPERTOIRE=$1
#variable contenant la liste des fichiers du repertoire :
LISTE=`ls $REPERTOIRE`
echo $LISTE
#variable du répertoire de sortie
for FILE in $LISTE
do
  NBLINES=`wc -l $FILE | cut -d' ' -f1`
  echo "le fichier $FILE a $NBLINES lignes"
done
exit
```