Introduction au Shell

UV SR01
Dr. Hicham Lakhlef
Laboratoire Heudiasyc (UMR UTC-CNRS 7253)
Université de Technologie de Compiègne
France
hlakhlef AT utc.fr

A2022

Sommaire

Système d'exploitation, système d'exploitation Unix

- L'objectif d'un système informatique est d'automatiser le traitement de l'information.
- Un système informatique est constitué de deux entités: le matériel et le logiciel
- Côté matériel, un ordinateur est composé de:
  - L'**Unité Centrale** (UC) pour les traitements
  - La **Mémoire Central** (MC) pour le stockage
  - Les **Périphériques** : disque dur, clavier, souris, carte réseau... accessibles via des pilotes de périphériques

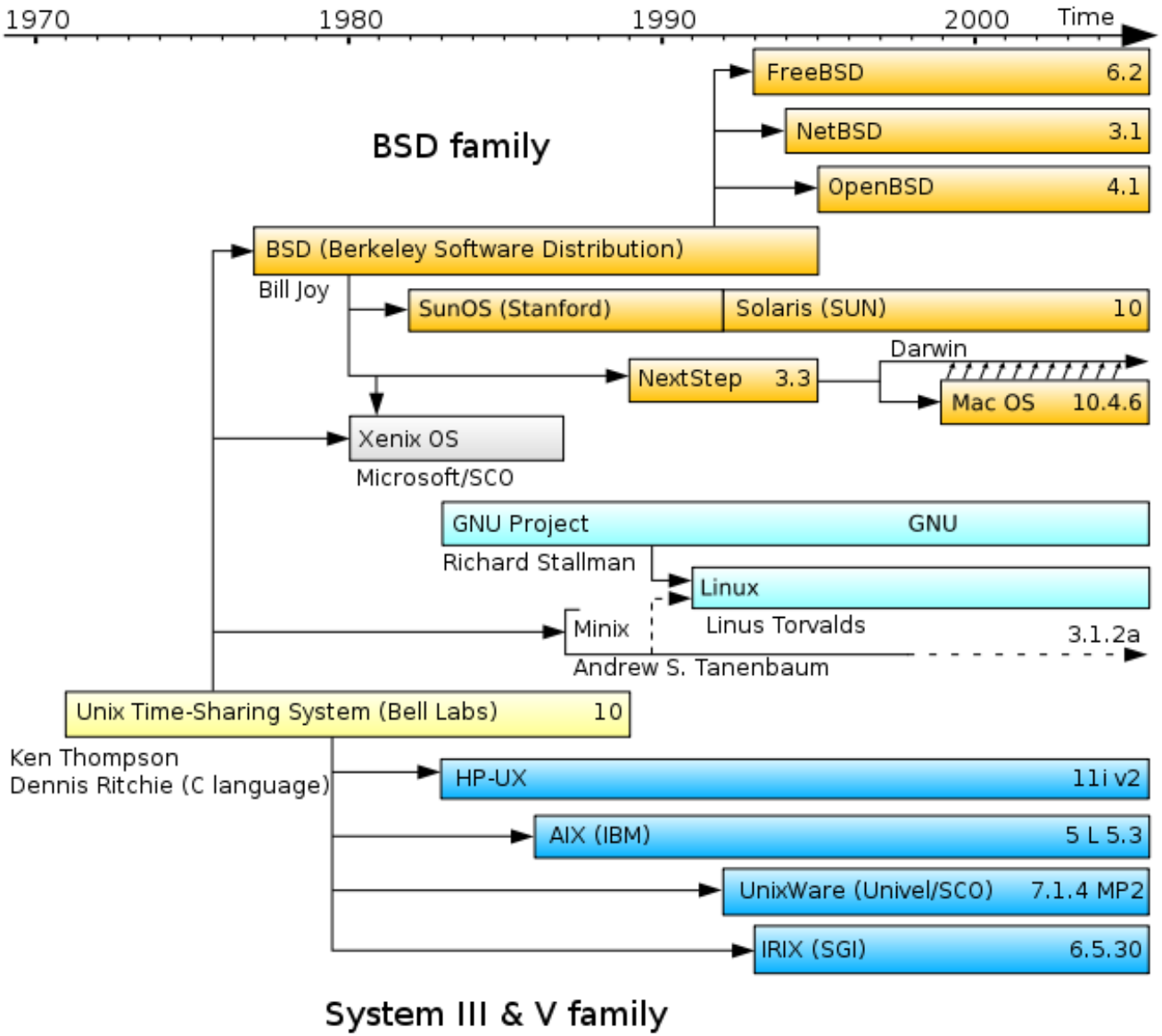
Historique d'Unix et Linux

- 1969 : création d'Unix - Ken Tompson (Laboratoires Bell)
- 1970 : adaptation au DEC PDP-11/20 par Thompson and Ritchie et naissance du premier langage portable : le langage C
- 1974-77 : les sources d'Unix sont distribuées gratuitement aux Universités
- 1978 : Unix devient la propriété d'ATT et les sources deviennent payantes
- 1979 : création de BSD Unix pour l'Université de Californie à Berkeley
- 1987 : diffusion de X Window, interface graphique pour Unix développée par le MIT

Historique d'Unix et Linux

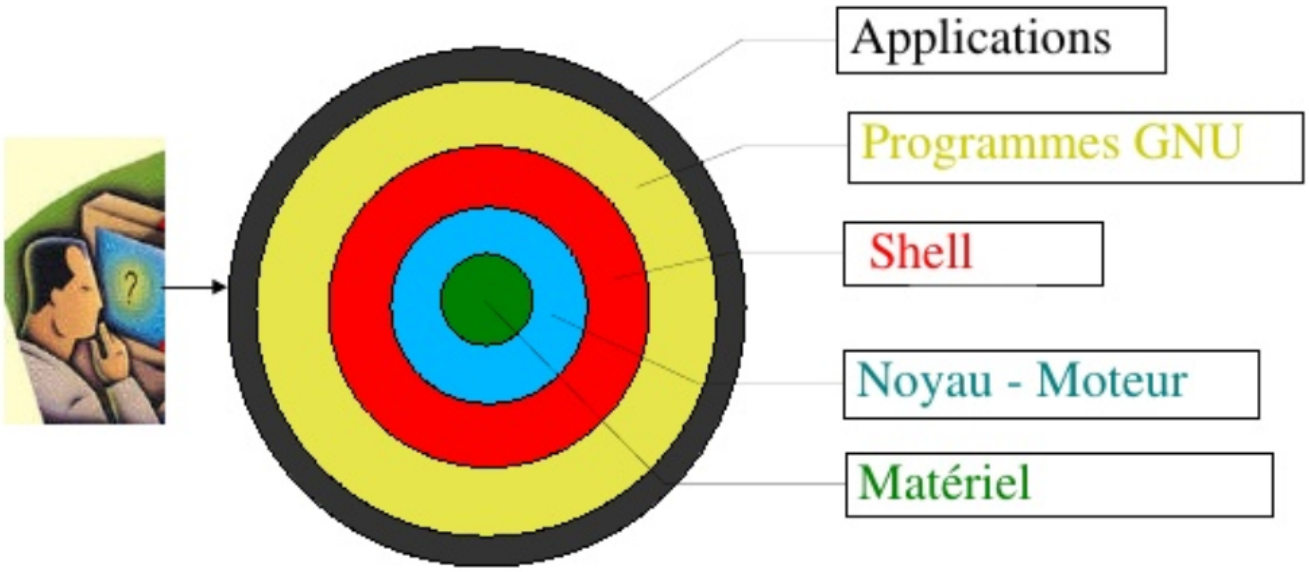
- 1987 : AIX d'IBM et HP-UX d'HP naissent
- 1991 : émergence de Linux
- 1992 : développement de Sun OS par Sun
- Linux a été écrit par Linus Torvalds, jeune étudiant finlandais, et a été amélioré par de nombreux développeurs dans le monde entier.
- 1991 : Linux 0.1 et diffusion du code source sur Internet
- 1993 : Linux 0.99
- 1994 : FreeBSD 1.0 basé sur BSD Unix
- 1995 : première distribution « commerciale » RedHat
- 2001 : Linux 2.4 pour l'USB, le Plug'n Play...
- 2004 : Linux 2.6, de l'embarqué aux gd systèmes
- 2011 : Linux kernel 3
- 2015 : Linux kernel 4

Familles UNIX, Projet GNU



Le noyau Unix

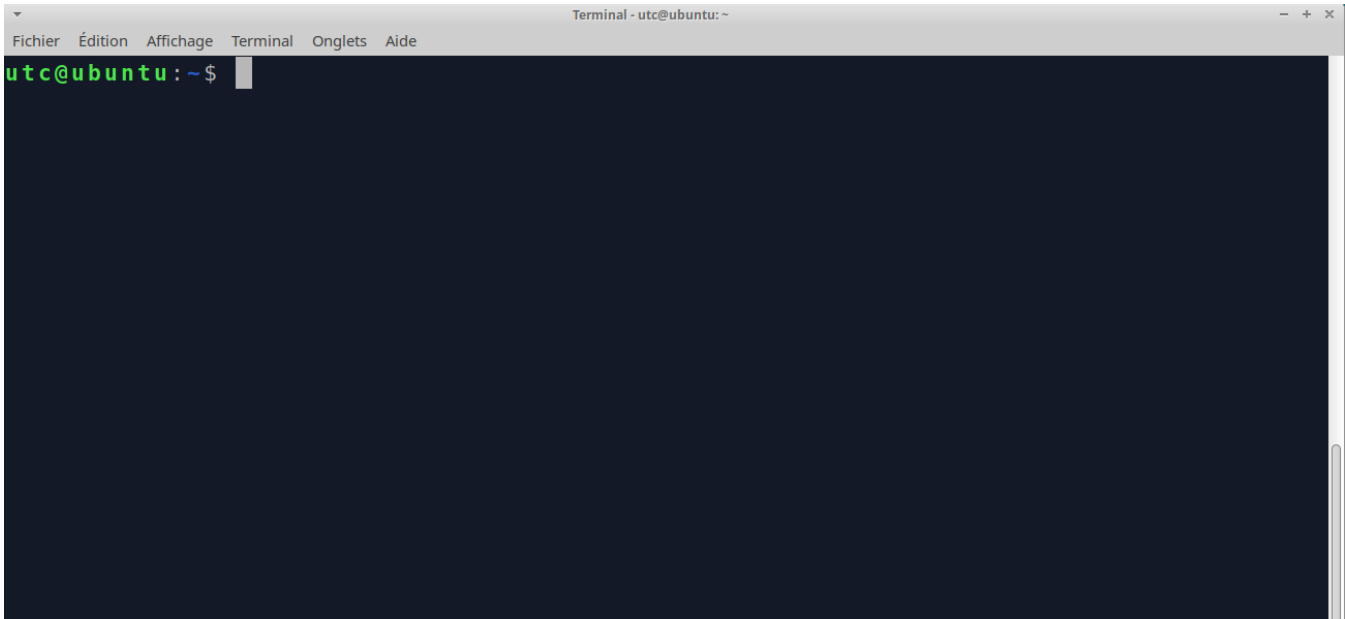
- **Le noyau** est un programme informatique (écrit en C) qui est au cœur du système d'exploitation d'un ordinateur, avec un contrôle complet sur tout ce qui se trouve dans le système.
- Il gère les ressources (processseur, mémoire, E/S...) du système Linux



Distributions Linux

Introduction au Shell

- À la connexion dans un terminal, l'utilisateur est mis en relation avec **un interpréteur de commandes** appelé shell en Unix.
- Shell est le pont entre l'utilisateur, les commandes exécutées par l'utilisateur et les commandes sous Unix qui sont déjà prédéfinies
- Le shell choisi dans ce cours est le bash: bourne again shell, apparu avec GNU/Linux.



Sommaire

Rôle du shell

- Le shell possède un double rôle:
  - C'est d'abord un interpréteur de commandes exécutant la boucle infinie suivante:
    1. Affichage de l'invite de commande ou prompt (« $ ») d'attente de lecture au clavier.
    2. Lecture d'une commande (validée par RETURN ou ENTRÉE).
    3. Analyse syntaxique (découpage en mots).
    4. Interprétation des caractères spéciaux.
    5. Exécution de la commande et retour au début.
  - Le shell est aussi un langage de programmation gérant des variables.
- Dans ce cours, nous étudions le shell en mode texte **bash** (Bourne-Again Shell)

Introduction au shell
Types de Shell

Shell	**Nom**	**Description**
Bourne Shell	sh	disponible sur toute plateforme UNIX
C shell	csh	Shell développé par BSD
Korn shell	ksh	Bourne Shell etendu par T
Bourne Again Shell	**bash**	Version améliorée de sh et csh. Fourni le plus souvent avec Linux
Zero Shell	zsh	shell avec beaucoup de fonctionnalitees
Tenex	tcsh	csh étendu
rc	rc	Implementation pour UNIX du shell de Plan 9
es	es	Extension de rc height

Syntaxe générale d'une commande

- La syntaxe générale d'une commande bash est la suivante: *nom [-options] [argument1...]*
  - *nom* est le nom de la commande;
  - *options* représente une ou plusieurs options;
  - *argument1* est le premier argument.
- Les options sont composées d'un seul caractère suivant un tiret.
- Il est parfois possible d'accoler plusieurs options (donc, plusieurs caractères). Par exemple, -asli pour les options -a -s -l -i.
- Si l'option demande un paramètre, il est séparé par un espace comme dans *-o fichier*.
- Dans une commande, chaque mot est séparé des autres par un espace ou une tabulation.

Introductio au shell
Historique des commandes

commande	Description
!!	Fait référence à la dernière commande
!n	La n-ième ligne de commande
!-n	La ligne de commande actuelle moins n
!**mot**	La commande la plus récente commençant par **mot**
!?**mot**?	La commande la plus récente contenant la chaîne **mot**
history n	Afficher les n dernières commandes
history -c	Effacer l'historique des commandes
^**mot1**^**mot2**^	Substitution rapide. Répétez la dernière commande en remplaçant **mot1** par **mot2**
Ctrl r **mot**	Rechercher dans l'historique

Principales commandes sur les fichiers

Caractère	Signification
file	Afficher le type d'un fichier
wc	Afficher la taille d'un fichier
cat	Afficher la totalité du fichier
more	Afficher page par page
head	Afficher les premières lignes de texte d'un fichier. Exemple : head -n 20
tail	Afficher les dernières lignes de texte d'un fichier. Exemple : tail -n 20
touch	Afficher en continue.
cp	Copier, exemple : cp fic1 fic2.
mv	Déplacer ou renommer, exemple : mv fic1 fic1.old
gzip	Compresser, exemple : gzip fic1
gunzip	Compresser, exemple : gunzip fic1.gz

Introduction à Bash

- Pour savoir quel shell utilise votre terminal :
  echo $SHELL
- Pour savoir quel shell actuellement utilisé :
  echo $0
- Il possible de déterminer les shells installés sur la machine :
  cat */etc/shells*
- bash est installé par défaut pour les utilisateurs. C'est le choix actuel dans la plupart des distributions Linux.

Redirection de la sortie standard

- À la connexion, le shell dispose de trois flots de communication:
  - Entrée standard : stdin (numéro 0)
  - Sortie standard : stdout (numéro1)
  - Erreur standard : et stderr (numéros 2)
- L'association par défaut de ces flots est l'écran pour stdout et stderr, et le clavier pour stdin.
- Une redirection est une modification de l'une ou de l'autre de ces associations.Par exemple :
  $ who > UV.txt
  $ cat UV.txt

Redirection double de la sortie standard

- Avec « > », si le fichier de redirection existe, son contenu initial est perdu. Par exemple :
  $ ls > UV.txt
  $ cat UV.txt
- La redirection double (commande » nom_fich) permet de ne pas détruire le fichier existant, mais ajoute le nouveau contenu en fin de fichier :
  $ ps » info
  $ cat info
- Dans le cas d'une redirection double , si le fichier n'existe pas, il est créé, comme pour une redirection simple.

Redirection de l'entrée standard

- Moins utilisée que la redirection de la sortie standard, la redirection de l'entrée standard (commande < nom_fich) permet à une commande d'utiliser comme données le contenu d'un fichier à la place d'une lecture clavier.
- Exemple avec la commande *write* :
  - Envoi d'un message depuis l'entrée standard:
    $ write » Paul # message à transmettre à Paul
    Bonjour Paul, ça va ?
    [CTRL-D]
  - Envoi d'un autre message cette fois à partir d'un fichier *message*:
    $ more message # fichier message contenant le message
    coucou
    $ write Paul < message # redirection de l'entrée standard, message reçu par Paul contenant "coucou"

Script bash

- Un programme bash est un *texte bash* dans un fichier texte
  - Interprétable par bash au lancement par l'utilisateur
  - Modifiable par un éditeur de texte (emacs,vi, gedit...)
  - Un programme bash doit être rendu exécutable avec : *chmod u+x mon_script.sh*
  - Par convention, les noms de script sont suffixés par l'extension « .sh »
- Invocation du script nommé mon_script.sh avec
  - ./mon_script.sh
  - Avec ses arguments : ./mon_script.sh arg1 arg2

Structure d'un script bash

- Première ligne : #!/bin/bash
  - #! : indique au système que ce fichier est un ensemble de commandes à exécuter par l'interpréteur dont le chemin suit
  - /bin/bash lance bash
- Puis séquence structurée de commandes shell

```
#! /bin/bash

commande1
commande2
...
```

mon_script.sh

- Sortie implicite du script à la fin du fichier
- Sortie explicite avec la commande exit

Variables bash

- Déclaration/affectation avec = (exemple var=valeur)
- Consultation en préfixant du caractère $ (exemple $var)
- Saisie interactive : read var1 var2 ... varn
  - Lecture d'une ligne saisie par l'utilisateur (jusqu'au retour chariot)
  - Le premier mot va dans var1
  - Le second dans var2
  - Tous les mots restants vont dans varn

Schéma algorithmique séquentiel (1/2)

- Suite de commandes les unes après les autres
- Sur des lignes séparées
- Ou séparés par point virgule (;)

Exemple 1

```
1   #!/bin/bash
2   isvalid=true
3   count=1
4   while [ $isvalid ]; do
5   echo $count
6   if [ $count -eq 5 ];
7   then
8   break
9   fi
10  ((count++))
11  done
```

Schéma algorithmique séquentiel (2/2)

Exemple 2

```
1  #!/bin/bash
2  echo "Entrez un numro valide"
3  read n
4  if [ $n -eq 101 ];
5  then
6  echo "Voici le premier numero"
7  elif [ $n -eq 510 ];
8  then
9  echo "Voici le deuxime numero"
10 elif [ $n -eq 999 ];
11 then
12 echo "Voici le troisime numero"
13 else
14 echo "Aucun numro ici"
15 fi
```

Tests sur les valeurs

- Tests sur des valeurs numériques

```
1 [ n1 -eq n2 ] : vrai si n1 est egal  n2
2 [ n1 ne n2 ] : vrai si n1 est diffrent de n2
3 [ n1 gt n2 ] : vrai si n1 suprieur strictement  n2
4 [ n1  ge n2 ] : vrai si n1 suprieur ou egal  n2
5 [ n1  lt n2 ] : vrai si n1 infrieur strictement  n2
6 [ n1 le n2 ] : vrai si n1 est infrieur ou egal  n2
```

- Tests sur des chaînes de caractères

```
1 [ mot1 = mot2 ] : vrai si mot1 est egale  mot2
2 [ mot1 != mot2 ] : vrai si mot1 nest pas gale  mot2
3 [ -z mot ] : vrai si mot est le mot vide
4 [ -n mot ] : vrai si mot nest pas le mot vide
```

L'instruction case

Syntaxe :

```
1  case expression in
2      pattern1 )
3          statements ;;
4      pattern2 )
5          statements ;;
6      ...
7  esac
```

- Le shell évalue la valeur de mot puis compare séquentiellement cette valeur à chaque modèle.
- Dès qu'un modèle correspond à la valeur de mot, la suite de commandes associée est exécutée, terminant l'exécution de la commande interne composée case.

L'instruction case

- Cet exemple imprime le type d'un fichier (texte, Csource, etc.) en fonction de l'extension du nom de fichier.

```
1
2  #!/bin/bash
3  for nom_fichier in $(ls)
4  do
5          # Prendre l'extension de nom_fichiere
6          ext=${nom_fichier##*\.}
7          case "$ext" in
8          c) echo "$nom_fichier : Fichier source C";;
9          o) echo "$nom_fichier : Fichier objet";;
10         sh) echo "$nom_fichier : Shell script";;
11         txt) echo "$nom_fichier : Fichier Text" ;;
12         *) echo " $nom_fichier : Non trait";;
13 esac
14 done
```

Les boucles (while, for)

- Le schéma général d'une boucle while est le suivant :

```
1 #!/bin/bash
2 while cond; do
3     cmds
4 done
```

- Le schéma général d'une boucle for est le suivant :

```
1 #!/bin/bash
2 for var in list; do
3     cmds
4 done
```

Arguments d'une commande

- Chaque mot de la commande est stocké dans une variable numérotée :
  - "$0" : toujours le nom de la commande
  - "$1" ... "$9" : les paramètres de la commande
  - $# : nombre de paramètres de la commande
  - $? : récupération du code retour
  - "$@" : liste des paramètres : "arg1" "arg2" "arg3" "arg4" ...
  - shift : décale d'un cran la liste des paramètres
- Exemple : le script *mon_script.sh* suivant est à exécuter avec ./mon_script.sh UV SR01

```
1 #! /bin/bash
2 for i in "$@"; do
3   echo $i
4 done
```

Imbrication de commandes

- Pour récupérer le texte écrit sur le terminal par une commande dans une chaîne de caractères : $(cmd)
- Attention à ne pas confondre avec $cmd qui permet l'accès à la valeur de la variable *cmd*
- Exemple : *$ echo je suis sous le chemin $ (pwd)*
  - 1. Execution de la commande *pwd*
  - 2. Execution de la commande :*$ echo je suis sous le chemin résultat de pwd*

Exemples de Script
Obtenir la date actuelle

- Exemple : *date.sh*

```
1  #!/bin/bash
2  Annee=$(date +%Y)
3  Mois=$(date +%m)
4  Jour=$(date +%d)
5  Heure=$(date +%H)
6  Minute=$(date +%M)
7  Seconde=$(date +%S)
8  echo $(date)
9  echo "La date d'aujourd'hui: $Jour.$Mois.$Annee"
10 echo "L'heure actuelle est: $Heure:$Minute:$Second
```

Exemples de Script
S'authentifier

- Exemple : *connexion.sh*

```
1  !/bin/bash
2  echo " Entrer Nom_utilisateur"
3  read NomU
4  echo "Entrer mot de passe"
5  read pass
6  if [[ ( $NomU == "admin" && $pass == "utc" ) ]];
7  then
8  echo "connexion reussie"
9  else
10 echo "Echec de connexion"
11 fi
```

Exemples de Script
Fichier

- Que fait le programme suivant ?

```
1  #!/bin/bash
2  if [ "$#" -lt 1 ]; then
3      echo "Saisir le nom du fichier"
4      read -r fichier
5  else
6      fichier="$1"
7  fi
8  while read -r ligne; do
9      set $ligne
10     moyenne=$(($2+$3+$4)/3))
11     echo "L'eleve $1 a pour moyenne $moyenne"
12 done < "$fichier"
```

Sommaire

- Les recherches de modèles sont largement utilisées dans de nombreuses applications telles que les moteurs de recherche
- Une expression régulière est définie comme un modèle qui définit une classe de chaînes
- Étant donné une chaîne, nous pouvons alors tester si la chaîne appartient à cette classe de modèles
- Les expressions régulières sont utilisées par de nombreux utilitaires Unix comme grep, sed, awk ...

Grep

- **Grep** est un filtre, il peut trouver un mot dans un fichier, par exemple :
  *grep printf *.c*
- **Grep** supporte simplement les expressions régulières de base
- On peut l'utiliser (avec un tube) pour filtrer la sortie d'une commande :
  locate UVs | grep SR01
- **egrep** supporte les expressions régulières dites étendues et qui offre donc plus de puissance

Caractères spéciaux de egrep

Caractère	Signification
[...]	Plage de caractères permis.
[^...]	Plage de caractères interdits.
^	Début de ligne.
.	Un caractère quelconque, y compris un espace.
*	Répétition du caractère placé avant l'étoile.
$	Fin de ligne.
{...}	Répétition.
{Nbr}	Répétition de Nbr exactement.
{Nbr,}	Répétition de Nbr au minimum.
{Nbr1 Nbr2}	Répétition de Nbr1 à Nbr2.
+	Le caractère devant doit exister au min 1 fois.
?	Le caractère devant peut apparaître 1 ou 0 fois.
(a| b)	L'une ou l'autre des expressions sont autorisées.
(...)	Permettent de grouper des critères partiels.

Options courantes de la commande egrep

Option	Signification
-c	Nombre de ligne trouvées (sans les afficher).
-i	Ne fait pas la différence entre majuscule et minuscule.
-n	Affiche le numéro de la ligne.
-l	Affiche le nom du fichier contenant la ligne (et pas la ligne).
-v	Affiche toutes les lignes qui ne contiennent pas le mot en question.
-h	Ne pas afficher le nom des fichiers dans les résultats lorsque plusieurs fichiers sont parcourus.
-s	Ne pas afficher les messages d'erreurs concernant les fichiers inexistants ou illisibles.

Classes de caractères prédéfinies

- Il existe certaines classes de caractères prédéfinies, avec des noms sont assez explicites : **[:alnum:]**, **[:alpha:]**, **[:cntrl:]**, **[:digit:]** (chiffres), **[:graph:]**, **[:lower:]** **(minuscules)**, **[:print:]** (affichables), **[:punct:]**, **[:space:]**, **[:upper:]** (majuscules), et **[:xdigit:]** (chiffres héxadécimaux).
- Par exemple, **[[:alnum:]]** correspond à [0-9A-Za-z], à la différence que le dernier dépend de l'encodage ASCII, alors que le premier est plus portable.
- Les crochets dans les noms de classes font partie intégrante du nom symbolique, et qu'ils doivent donc être inclus en plus des crochets encadrant la liste.
- Exemple
  head -30 /etc/services
  head -30 /etc/services | grep -e [[:alnum:]]
  head -30 /etc/services | grep -ve [[:alnum:]]

Caractères spéciaux

- Entre simple quotes (« ' ») les caractères spéciaux ne sont pas interprétés par le shell mais deviennent de simples caractères.
- Avec le caractère anti-slash « \» le caractère (spécial) qui le suit n'est pas interprété.
- Si on veut mélanger des caractères spéciaux, des variables, des commandes..., il faut utiliser les guillemets (« " »).
  - Seuls sont interprétés les méta-caractères « $ » (commandes et variables), « \» (annulation) et « ` » (commandes).
  - Exemple : *echo "Mon dossier est $(pwd)"*
  - Avec de simples quotes, $(*pwd*) ne serait pas interprété.

Classes de caractères prédéfinies

- Chercher toutes les lignes contenant au minimum une lettre en minuscule : *egrep [a-z]+*
- Chercher toutes les lignes contenant uniquement un nombre à 4 chiffres : *egrep ^[0-9]{4}*
- Toutes les lignes contenant des nombres de minimum 2 chiffres avant les deux points : *egrep :[0-9]{2,}:*
- Toutes les lignes commençant par des nombres de minimum 1 à 5 chiffres suivits par deux points : *egrep :^[0-9]{1,5}:*
- Cherche toutes les lignes contenant la chaîne "SR01 UV" ou "SR02 UV" : `egrep (SR01|SR02) UV`

Réutiliser les motifs

- Nommer un motif trouvé avec des parenthèses.
- La référence arrière $\backslash n$, où n est un chiffre unique, correspond à la sous-chaîne déjà mise en correspondance avec la n-ième sous-expression rationnelle entre parenthèses.
- Exemple : (exp1) exp2 (exp3) \1 \2 revient à (exp1) exp2 (exp3) exp1 exp3

Grep
Exercices

1. Donner la liste de fréquences des mots avec deux voyelles consécutives du fichier UV.txt
2. Donner la liste des numéros de cartes bancaires se trouvant dans un fichier donné
3. Donner la liste de fréquences des mots avec deux voyelles non-consécutives du fichier UV.txt
4. Trouver des lignes se terminant par une consonne

Sed

- Comment faire pour renommer 1500 documents d'un coup ou encore modifier du texte dans des centaines de fichiers à la fois
- **Sed** est un éditeur de flux. Un éditeur de flux est utilisé pour effectuer des transformations de texte de base sur un flux d'entrée
- **Sed** peut faire des choses qui prendraient des heures à faire avec une interface graphique

Sed
Fonctionnement de SED

sed repose-t-il sur le mécanisme suivant :

1. Lecture d'une ligne sur le flux d'entrée (jusqu'à ce qu'il rencontre un caractère de saut de ligne) ;
2. Traitement de cette ligne en la soumettant à toutes les commandes rencontrées dans le fichier script ;
3. Affichage de la ligne résultante sur la sortie standard, sauf si sed est invoqué avec l'option – n ;
4. Passage à la ligne suivante, et ainsi de suite jusqu'à la fin du flux d'entrée standard.

Sed
Suppression

- sed travaille sur un flux de données et pas directement sur un fichier, il ne s'agit pas d'une véritable suppression, mais plutôt d'un abandon.
- Supprimer une ligne selon son numéro
  - Pour supprimer il faut utiliser l'option *d*
  - Exemple : *sed '1d;4d; 5d' SR01.txt* est une commande pour supprimer les lignes 1, 4 et 5.
- Supprimer un intervalle de lignes
  - On peut également spécifier un intervalle en utilisant la virgule : *sed '1,4d' test.txt* supprimera les lignes 1 à 4.

Sed
Filtrage

- C'est exactement l'inverse de ce que nous venons de faire.
- Ici on choisit de ne rien afficher par défaut.
- On veut par exemple n'afficher que les lignes qui commencent par dièse, on va utiliser l'option silencieux -n, avec la commande print p : sed -n '/^#/p' SR01.txt
- On peut choisir de n'afficher qu'une ligne : sed -n 5p SR01.txt , ou une intervalle : sed -n 1, 5p SR01.txt

Sed
La substitution

- Il s'agit de la tâche principale des commandes sed , car les autres fonctions les plus utilisées (d et p) peuvent souvent être assurées par d'autres utilitaires (grep, tail...) de manière moins directe, mais souvent moins effrayante pour l'utilisateur courant.
- Forme usuelle de la commande sed : **commande/avant/après/option**
- La substitution permet de remplacer un motif par un autre
- Par exemple, pour remplacer toutes les occurrences de "UV" par "SR01" dans le fichier A2020.txt:
  sed 's/UV/SR01/' A2020.txt
- Si vous ne spécifiez pas le fichier d'entrée ou si le fichier d'entrée vaut -, sed filtre le contenu de l'entrée standard.

Sed
La substitution (options)

- Les options possibles à la fin de la commande de substitution sont les suivantes :

Option	Signification
g	Remplacer tous les motifs rencontrés dans la ligne en cours. Cette option est presque toujours employée.
i	Ne remplacer que la i-ième occurrence du motif dans la ligne. Cela peut surtout servir lorsqu'on manipule des fichiers qui représentent des lignes d'enregistrements, contenant des champs séparés par des délimiteurs.
p	Afficher la ligne si une substitution est réalisée.
w	Suivie d'un nom de fichier, cette option permet d'y envoyer le résultat de la substitution. Cela sert généralement à des fins de débogage.

Sed
La translitération

- Prototype : sed 'y/ source-chars/dest-chars/'
- Translittérer tous les caractères de l'espace de motif qui correspondent à l'un des *source* – *chars* source avec le caractère correspondant dans *dest* – *chars*
- Exemple : Translittérer 'a-j' à '0-9' :
- $ echo hello world | sed 'y/abcdefghij/0123456789/'
  74llo worl3

Sed
Exercices

1. Une (seule) commande Sed pour substituer tout les mots "Michel" dans le fichier «prenom1» avec «Julien» et envoyer le résultat dans le fichier «prenom2»
   sed 's/chaine1/chaine2/g' chemin1 > chemin2
2. Transformer les caractères a, b et c du fichier /etc/passwd par un @ de la ligne 11 à 20
   sed -e '11,20s/[abc]/@/g' /etc/passwd
3. Transformer les mots du fichier /usr/share/dict/french qui commencent par une consonne et se terminent par un z par le mot Ubuntu