Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Introduction

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

PLAN

1. Le Bourne Shell
   - Commandes internes et externes
   - Ouverture de canaux
   - Groupement de commandes
2. Programmation shell
   - Structure et exécution d'un script
   - Les variables

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Le Bourne Shell

- Toutes les commandes que nous nous avons vu jusqu'à présent sont standards et ne sont pas propres à un shell particulier. Avant d'entamer la programmation shell elle-même, il nous faut voir quelques options détaillées du shell par défaut, le Bourne Shell.
- Fichier de configuration
- Le Bourne Shell dispose d'un fichier de configuration chargé et exécuté lors du lancement du shell. Il est généralement placé dans le répertoire par défaut de l'utilisateur et se nomme .profile.

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Commandes internes et externes

- Une commande peut être interne ou externe. Une commande interne est une commande directement incluse et interprétée par le shell, sans passer par un quelconque exécutable. Un exemple est la commande cd.
- Une commande externe est un exécutable que le shell recherchera dans une liste de chemins prédéfinis, le PATH, puis exécutera en tant que processus fils.
- D'autres mécanismes existent, comme les fonctions et les alias (hors Bourne Shell), que nous aborderons plus tard.

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

herescript

- La redirection " <<" n'a pas été abordée car elle est particulière. Elle permet l'utilisation des **Herescripts** (Script ici).
- Un **herescript** permet la saisie d'un texte jusqu'à un point donné et l'envoi de son résultat à une commande ou un filtre. Les redirections classiques sont aussi autorisées.
- Après le " << " on indique une chaîne définissant la fin de saisie, par exemple ici 'end'.

```
$ tr "[a-z]" "[A-Z]" << end
> bonjour les amis
> ceci est un exemple
> de herescript
> end
BONJOUR LES AMIS
CECI EST UN EXEMPLE
DE HERESCRIPT
```

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Ouverture de canaux

- Les canaux standards sont au nombre de trois et numérotés de 0 à 2.
- Ainsi 0< équivaut à < et 1 > à >.
- La commande exec permet d'ouvrir sept autres canaux numérotés de 3 à 9.
- On a donc en tout dix canaux. On peut envisager, dans le cadre de traitements, de sortir certains résultats par le canal 3, d'autres par le 4, et ainsi de suite. Les canaux ouvert le sont en entrée et en sortie.

  exec num_canal > fichier_ou_reunion

  $ **exec** 3>dump.log

  $ **ls** -l > &3

  essaie

  $ cat dump.log
- Tous ce qui sera écrit dans le canal 3 sera écrit dans le fichier dump.log. On peut ensuite fermer le canal en le

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Groupement de commandes

- Nous avons vu que le chaînage de commande est possible avec le ";". Il est aussi possible de grouper les commandes.
- Quand on exécute les commandes suivantes :
  $ **uname** -a ; **pwd** ; **ls** -l > resultat.txt &
- seule la dernière commande est exécutée en tâche de fond et seul son résultat est redirigé dans le fichier resultat.txt.
- Une solution serait
  $ **uname** -a > resultat.txt & ; **pwd** >> resultat.txt & ;
  ls -l >> resultat.txt &
  [1] 18232
  [2] 18238
  [3] 18135

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Groupement de commandes

- C'est une solution complexe et qui ne marchera pas toujours. De plus même si les commandes sont lancées séquentiellement, elles tournent toutes en parallèle et c'est la première finie qui écrira en premier dans le fichier. La solution consiste en l'utilisation des parenthèses "()".
  $ (uname -a ; pwd ; ls -l) > resultat.txt & [1] 18239
  $ [1] Done (uname -a; pwd; ls -l) > resultat.txt
- Dans ce cas, toutes les commandes placées entre les parenthèses sont lancées par un sous-shell, qui va ensuite exécuter les commandes précisées séquentiellement telles qu'indiquées. Ainsi la redirection concerne l'ensemble des commandes et rien n'empêche de lancer ce sous-shell en arrière-plan. On distingue bien d'ailleurs un seul PID 18239 lors de l'exécution des commandes.

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Groupement de commandes

- Une seconde possibilité est l'utilisation des accolades .... Dans ce cas aucun sous-shell n'est exécuté, et si une commande interne (cd ou autre) est exécutée, elle concerne le shell actif. L'accolade fermante doit être placée juste après un " ; ".
  $ uname -a; pwd; ls -l; > resultat.txt

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Groupement de commandes

- C'est une solution complexe et qui ne marchera pas toujours. De plus même si les commandes sont lancées séquentiellement, elles tournent toutes en parallèle et c'est la première finie qui écrira en premier dans le fichier. La solution consiste en l'utilisation des parenthèses " () ".
- $ (uname -a ; pwd ; ls -l) > resultat.txt &
  [1] 18239
  $ [1] Done (uname -a; pwd; ls -l) > resultat.txt

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Groupement de commandes

- Dans ce cas, toutes les commandes placées entre les parenthèses sont lancées par un sous-shell, qui va ensuite exécuter les commandes précisées séquentiellement telles qu'indiquées. Ainsi la redirection concerne l'ensemble des commandes et rien n'empêche de lancer ce sous-shell en arrière-plan.
- On distingue bien d'ailleurs un seul PID 18239 lors de l'exécution des commandes. Une seconde possibilité est l'utilisation des accolades {...}.
- Dans ce cas aucun sous-shell n'est exécuté, et si une commande interne (cd ou autre) est exécutée, elle concerne le shell actif. L'accolade fermante doit être placée juste après un " ; ".
  $ { uname -a; pwd; ls -l; } > resultat.txt

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Groupement de commandes

- En plus du chaînage classique, les commandées peuvent être liées et exécutées de façon conditionnelle. La condition d'exécution d'une commande est la réussite ou non de la précédente.
- Chaque commande une fois exécutée sort un code retour, généralement 0 si tout s'est bien passé, 1 ou 2 en cas d'erreur.
- Le shell peut récupérer cette valeur par la variable " $? ". (Variables abordées plus loin).

```
$ ls
...
$ echo $?
0
```

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Groupement de commandes

- Les caractères " && " et " || " permettent d'effectuer une exécution conditionnelle. commande1 && commande2 commande1 || commande2 La commande située après " && " sera exécutée uniquement si la commande précédente a retourné 0 (réussite). La commande située après " || " ne sera exécutée que si la commande précédente a retourné autre chose que 0.

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Groupement de commandes

```
$ cat liste
Produit objet prix quantites
souris optique 30 15
dur 30giga 100 30
dur 70giga 150 30
disque zip 12 30
disque souple 10 30
ecran 15 150 20
ecran 17 300 20
ecran 19 500 20
clavier 105 45 30
clavier 115 55 30
carte son 45 30
carte video 145 30
```

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Groupement de commandes

$ (grep "souris" liste && echo "Souris trouvee") || echo "Souris introuvable"

souris optique 30 15

Souris trouvee

$ (grep "memoire" liste && echo "Memoire trouvee") || echo "Memoire introuvable"

Memoire introuvable

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Groupement de commandes

- Le shell n'est pas qu'un simple interpréteur de commandes, mais dispose d'un véritable langage de programmation avec notamment une gestion des variables, des tests et des boucles, des opérations sur variables, des fonctions...

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Structure et exécution d'un script

- Toutes les instructions et commandes sont regroupées au sein d'un script. Lors de sont exécution, chaque ligne sera lue une à une et exécutée. Une ligne peut se composer de commandes internes ou externes, de commentaires ou être vide.
- Plusieurs instructions par lignes sont possibles, séparées par le " ; " ou liées conditionnellement par " && " ou "||". Le ";" est l'équivalent d'un saut de ligne.

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Structure et exécution d'un script

- Par convention les shell scripts se terminent généralement (pas obligatoirement) par " .sh " pour le Bourne Shell et le Bourne Again Shell, par " .ksh " pour le Korn Shell et par " .csh " pour le C Shell. Pour rendre un script exécutable directement :
  $ chmod u+x monscript
  Pour l'exécuter :
  $ ./monscript
  Pour éviter le ./ :
  $ PATH=$PATH:.
  $ monscript

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Structure et exécution d'un script

- Quand un script est lancé, un nouveau shell " fils " est créé qui va exécuter chacune des commandes. Si c'est une commande interne, elle est directement exécutée par le nouveau shell. Si c'est une commande externe, dans le cas d'un binaire un nouveau fils sera créé pour l'exécuter, dans le cas d'un shell script un nouveau shell fils est lancé pour lire ce nouveau shell ligne à ligne.
- Une ligne de commentaire commence toujours par le caractère " #". Un commentaire peut être placé en fin d'une ligne comportant déjà des commandes.
  # La ligne suivante effectue un ls
  ls # La ligne en question
- La première ligne a une importance particulière car elle permet de préciser quel shell va exécuter le script
  #!/usr/bin/sh
  #!/usr/bin/ksh

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Structure et exécution d'un script

- On en distingue trois types : utilisateur, système et spéciales. Le principe est de pourvoir affecter un contenu à un nom de variable, généralement un chaîne de caractère, ou des valeurs numériques.

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Nomenclature

Un nom de variable obéit à certaines règles :

- (1)Il peut être composé de lettres minuscules, majuscules, de chiffres, de caractères de soulignement
- (2)Le premier caractère ne peut pas être un chiffre
- (3)Le taille d'un nom est en principe illimité (il ne faut pas abuser non plus)
- (4)Les conventions veulent que les variables utilisateur soient en minuscules pour les différencier des variables système. Au choix de l'utilisateur.

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Déclaration et affectation

- Une variable est déclarée dès qu'une valeur lui est affectée. L'affectation est effectuée avec le signe "=", sans espace avant ou après le signe.
  var=Bonjour
  On peut aussi créer des variables vides. Celle-ci existera mais sans contenu.
  var=

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Déclaration et affectation

- $ chemin=/tmp/seb
  $ pwd /home/toto
  $ ls $chemin
  dump.log fic4 lien_fic1 liste_ls rep1
  seb2
  fic1 hardlink2_fic2 lien_fic2 ls.txt rep2
  toto.tar.gz
  fic2 hardlink3_fic2 liste mypass resultat.txt
  users
  fic3 hardlink_fic2 liste2 nohup.out seb1
  $ cd $chemin
  $ pwd /tmp/seb
  $ cd $chemin/rep1
  $ pwd /tmp/seb/rep1

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Accès et affichage

- Pour afficher la liste des variables on utilise la commande set. Elle affiche les variables utilisateur et les variables système, nom et contenu.
- La commande echo permet d'afficher le contenu de variables spécifiques.

```
$ a=Jules
$ b=Cesar
$ set
EDITMODE=emacs
HOME=/home/seb
LD_LIBRARY_PATH=/mor/app/oracle/product/8.1.7/lib
a=Jules
b=Cesar
$ echo $a $b a conquis la Gaule
Jules Cesar a conquis la Gaule
```

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Accès et affichage

- Une variable peut contenir des caractères spéciaux, le principal étant l'espace. Mais
  $ c=Salut les copains
  les: not found
  $ echo $c
- Ne marche pas. Pour cela il faut soit verrouiller les caractères spéciaux un par un, soit de les mettre entre guillemets ou apostrophes.
  c=Salut \ les\ Copains # Solution lourde
  c="Salut les copains" # Solution correcte
  c='Salut les copains' # Solution correcte

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Accès et affichage

- La principale différence entre les guillemets et les apostrophes est l'interprétation des variables et des substitutions. " et ' se vérouillent mutuellement.
- $ a=Jules
  $ b=Cesar
  $ c="$a $b a conquis la Gaule"
  $ d='$a $b a conquis la Gaule'
  $ echo $c
  Jules Cesar a conquis la Gaule
  $ echo $d
  $a $b a conquis la Gaule
  $ echo "Unix c'est top"
  Unix c'est top
  $ echo 'Unix "trop bien"'
  Unix "trop bien"

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Suppression et protection

- On supprime une variable avec la commande unset. On peut protéger une variable en écriture et contre sa suppression avec la commande readonly. Une variable en lecture seule même vide est figée. Il n'existe aucun moyen de la replacer en écriture et de la supprimer, sauf à quitter le shell.

```
$ a=Jules
$ b=Cesar
$ echo $a $b
Jules Cesar
$ unset b
$ echo $a $b
Jules
$ readonly a
$ a=Neron
a: is read only
```

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Exportation

- Par défaut une variable n'est accessible que depuis le shell où elle a été définie.

```
$ a=Jules
$ echo 'echo "a=$a"' > voir_a.sh
$ chmod u+x voir_a.sh
$ ./voir_a.sh
a=
```

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Exportation

- La commande export permet d'exporter une variable de manière à ce que son contenu soit visible par les scripts et autres sous-shells.
- Les variables exportées peuvent être modifiées dans le script, mais ces modifications ne s'appliquent qu'au script ou au sous-shell.

```
$ export a
$ ./voir_a.sh
a=Jules
$ echo 'a=Neron ; echo "a=$a"' >> voir_a.sh
$ ./voir_a.sh
a=Jules
a=Neron
$ echo $a
Jules
```

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Accolades

- Les accolades de base " {} " permettent d'identifier le nom d'une variable. Imaginons la variable fichier contenant le nom de fichier 'liste'.
- On veut copier liste1 en liste2.
  $ fichier=liste
  $ cp $fichier2 $fichier1
  usage: cp [-fhip] [–] source_file destination_file
  or: cp [-fhip] [–] source_file ... destination_directory
  or: cp [-fhip] [$-R|-r$] [–] [source_file | source_directory]
  ... destination_directory

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Accolades

- Ça ne fonctionne pas car ce n'est pas $fichier qui est interprété mais $fichier1 et $fichier2 qui n'existent pas.
  $ cp ${fichier}2 ${fichier}1
  Dans ce cas, cette ligne équivaut à
  $ cp liste2 liste1
  Les accolades indiquent que fichier est un nom de variable.

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Accolades et remplacement conditionnel

Les accolades disposent d'une syntaxe particulière.
variable:Remplacement Selon la valeur ou la présence de la variable, il est possible de remplacer sa valeur par une autre.

Remplacement	*Signification*
{x:-texte}	si la variable x est vide ou inexistante, le texte prendra sa place. Sinon c'est le contenu de la variable qui prévaudra.
{x:=texte}	si la variable x est vide ou inexistante, le texte prendra sa place et deviendra la valeur de la variable.
{x:+texte}	si la variable x est définie et non vide, le texte prendra sa place. Dans le cas contraire une chaîne vide prend sa place.
{x:?texte}	si la variable x est vide ou inexistante, le script est interrompu et le message texte s'affiche. Si texte est absent un message d'erreur standard est affiché.

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Accolades et remplacement conditionnel

```
$ echo $nom
$ echo ${nom:-Jean}
Jean
$ echo $nom
$ echo ${nom:=Jean}
Jean
$ echo $nom
Jean
$ echo ${nom:+"Valeur définie"}
Valeur définie
$ unset nom
$ echo ${nom:?Variable absente ou non définie}
nom: Variable absente ou non définie
$ nom=Jean
$ echo ${nom:?Variable absente ou non définie}
Jean
```

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

variables système

En plus des variables que l'utilisateur peux définir lui-même, le shell est lancé avec un certain nombre de variables prédéfinies utiles pour un certain nombre de commandes et accessibles par l'utilisateur. Le contenu de ces variables système peut être modifié mais il faut alors faire attention car certaines ont une incidence directe sur le comportement du système.

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

variables système

Variable	*Contenu*
HOME	Chemin d'accès du répertoire utilisateur. Répertoire par défaut en cas de cd.
PATH	Liste de répertoires, séparés par des « : » où le shell va rechercher les commandes externes et autres scripts et binaires. La recherche se fait dans l'ordre des répertoires saisis.
PS1	Prompt String 1, chaîne représentant le prompt standard affiché à l'écran par le shell en attente de saisie de commande.
PS2	Prompt String 2, chaîne représentant un prompt secondaire au cas où la saisie doit être complétée.
IFS	Internal Field Separator, liste des caractères séparant les mots dans une ligne de commande. Par défaut il s'agit de l'espace de la tabulation et du saut de ligne.
MAIL	chemin et fichier contenant les messages de l'utilisateur
MAILCHECK	intervalle en secondes pour la vérification de présence d'un nouveau courrier. Si 0 alors la vérification est effectuée après chaque commande.

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

variables système

SHELL	Chemin complet du shell en cours d'exécution
LANG	Définition de la langue à utiliser ainsi que du jeu de caractères
LC_COLLATE	Permet de définir l'ordre du tri à utiliser (si LANG est absent)
LC_NUMERIC	Format numérique à utiliser
LC_TIME	Format de date et d'heure à utiliser
LC_CTYPE	Détermination des caractères et signes devant ou ne devant pas être pris en compte dans un tri.
USER	Nom de l'utilisateur en cours.
LD_LIBRARY_PATH	Chemin d'accès aux bibliothèques statiques ou partagées du système.
MANPATH	Chemin d'accès aux pages du manuel.
LOGNAME	Nom du login utilisé lors de la connexion.

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Variables spéciales

Il s'agit de variables accessibles uniquement en lecture et dont le contenu est généralement contextuel.

Variable	*Contenu*
$?	Code retour de la dernière commande exécutée
$$	PID du shell actif
$!	PID du dernier processus lancé en arrière-plan
$-	Les options du shell

```
$ echo $$
23496
$ grep memoire liste
$ echo $?
1
$ grep souris liste souris optique 30 15
$ echo $?
0
$ ls -lR >toto.txt 2<&1 & 26675
$ echo $!
26675
```

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Paramètres de position

Les paramètres de position sont aussi des variables spéciales utilisées lors d'un passage de paramètres à un script.

Variable	*Contenu*
$0	Nom de la commande (du script)
$1-9	$1,$2,$3... Les neuf premiers paramètres passés au script
$#	Nombre total de paramètres passés au script
$*	Liste de tous les paramètres au format "$1 $2 $3 ..."
$@	Liste des paramètres sous forme d'éléments distincts "$1" "$2" "$3" ...

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Paramètres de position

```
$ cat param.sh
#!/usr/bin/sh
echo "Nom : $0"
echo "Nombre de parametres : $#"
echo "Parametres : 1=$1 2=$2 3=$3"
echo "Liste : $*"
echo "Elements : $@"
$ param.sh riri fifi loulou Nom : ./param.sh
Nombre de parametres : 3
Parametres : 1=riri 2=fifi 3=loulou
Liste : riri fifi loulou
Elements : riri fifi loulou
```

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Paramètres de position

La différence entre $@ et $* ne saute pas aux yeux. Reprenons l'exemple précédent avec une petite modification : $ param.sh riri "fifi loulou"
Nom : ./param.sh
Nombre de parametres : 2
Parametres : 1=riri 2=fifi loulou 3=
Liste : riri fifi loulou
Elements : riri fifi loulou

Le Bourne Shell
Commandes internes et externes
Ouverture de canaux
Groupement de commandes
Programmation shell
Structure et exécution d'un script
Les variables

Paramètres de position

Cette fois-ci il n'y a que deux paramètres de passés. Pourtant les listes semblent visuellement identiques. En fait si la première contient bien
"riri fifi loulou"
La deuxième contient
"riri" "fifi loulou"
Soit bien deux éléments.
Dans les premier exemple nous avions
"riri" "fifi" "loulou"