




Parcours des écoles d'ingénieurs Polytech (PeiP1)

Les Boitiers de Vote Electroniques (Clickers)

Pour dynamiser et rendre interactif un cours en amphi

Règles du grand jeu concours

- ▶ Même règles que la fois précédente:
 - ▶ Un entrainement en vue de la prochaine évaluation
 - ▶ Donc pas de note comptant pour la moyenne
 - ▶ Une compétition amicale
- ▶ Classement des 20 premiers à la fin du cours
 - ▶ Pas grand-chose à gagner, si ce n'est...
- ▶ Pour épicer un peu le tout 
 - ▶ Des questions qui comptent double

Configuration du Boitier de Vote

- ▶ Retirer les caches plastiques pour les piles
- ▶ Réglage du canal de communication :
 - ▶ Appuyer sur Channel
 - ▶ Composer le numéro de canal
 - ▶ Appuyer sur Channel
 - ▶ La LED du boitier devient verte
- ▶ Votre boitier est prêt ?
- ▶ Nous pouvons faire un premier test

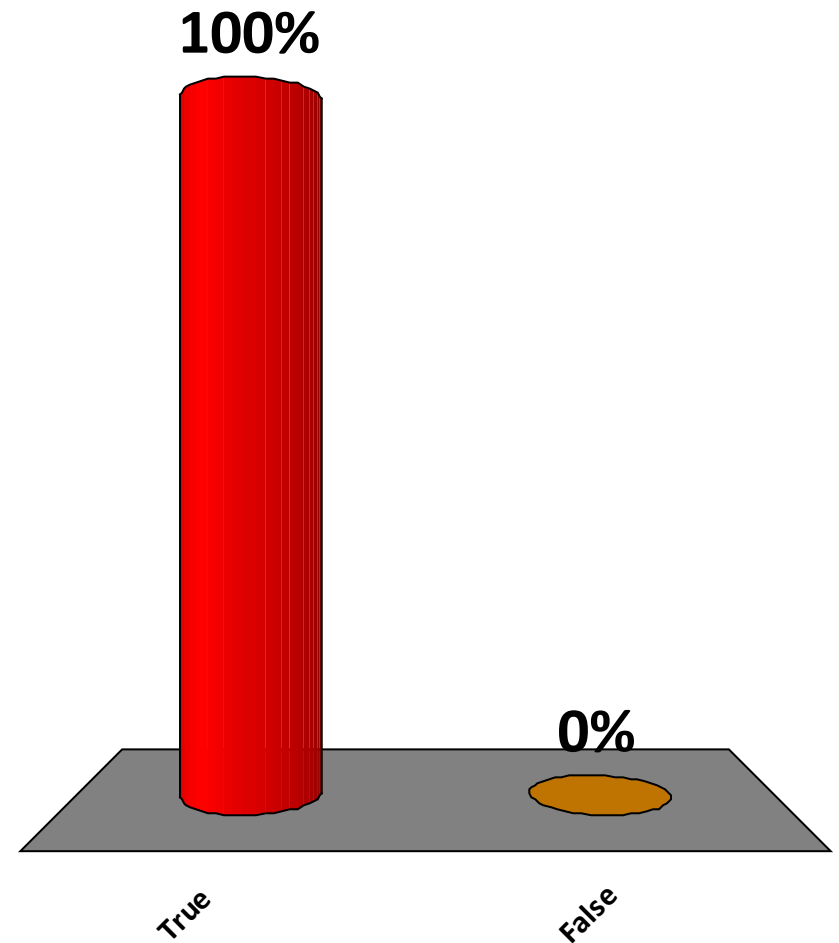


Est-ce que vous êtes prêts ?

► Etes-vous prêts ?

1. True
2. False

► Combien ont voté ?



Programmation Shell

Environnement Informatique 1

Faire un Script Shell

- ▶ Un script Shell c'est quoi ?
 - ▶ Le regroupement de commandes dans un fichier
- ▶ Un fichier « script Shell » doit
 - ▶ Indiquer le programme qui permettra d'exécuter les commandes
 - ▶ Contenir les commandes que l'on souhaite exécuter
 - ▶ Être un fichier exécutable
- ▶ On n'est pas obligé de le nommer `fichier.sh`
 - ▶ Il peut avoir le nom que l'on souhaite (même sans extension)

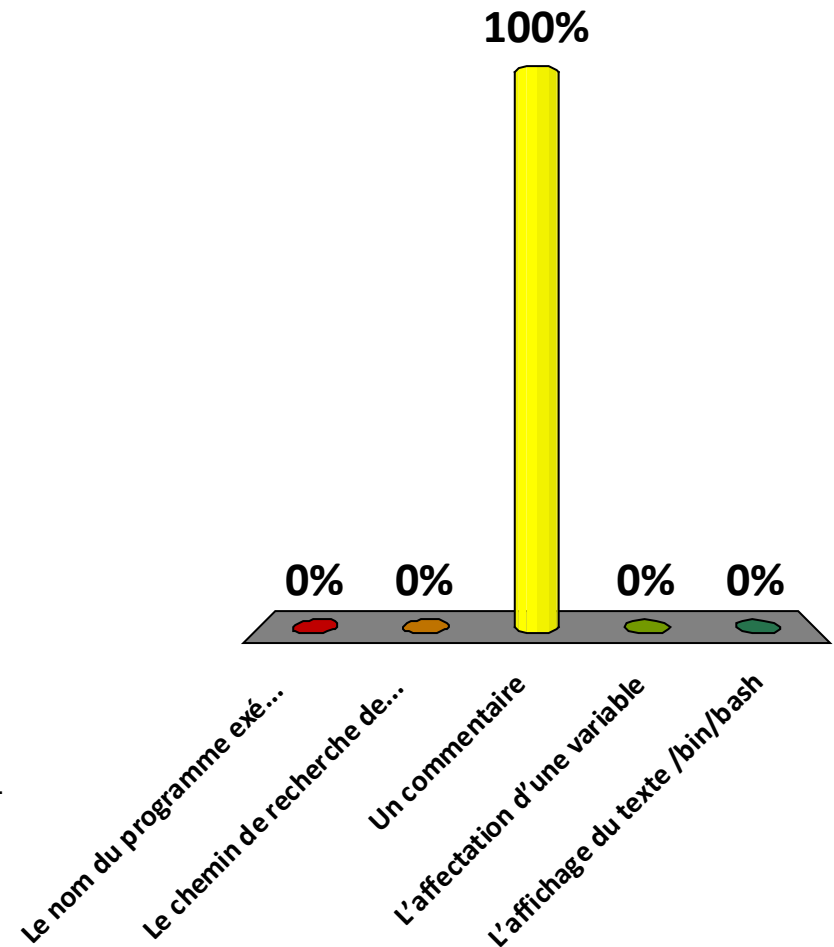


Entête d'un Script Shell

► Qu'est signifie la ligne suivante dans un script Shell:

► `#!/bin/bash`

1. Le nom du programme exécutant les commandes
2. Le chemin de recherche des commandes
3. Un commentaire
4. L'affectation d'une variable
5. L'affichage du texte `/bin/bash`



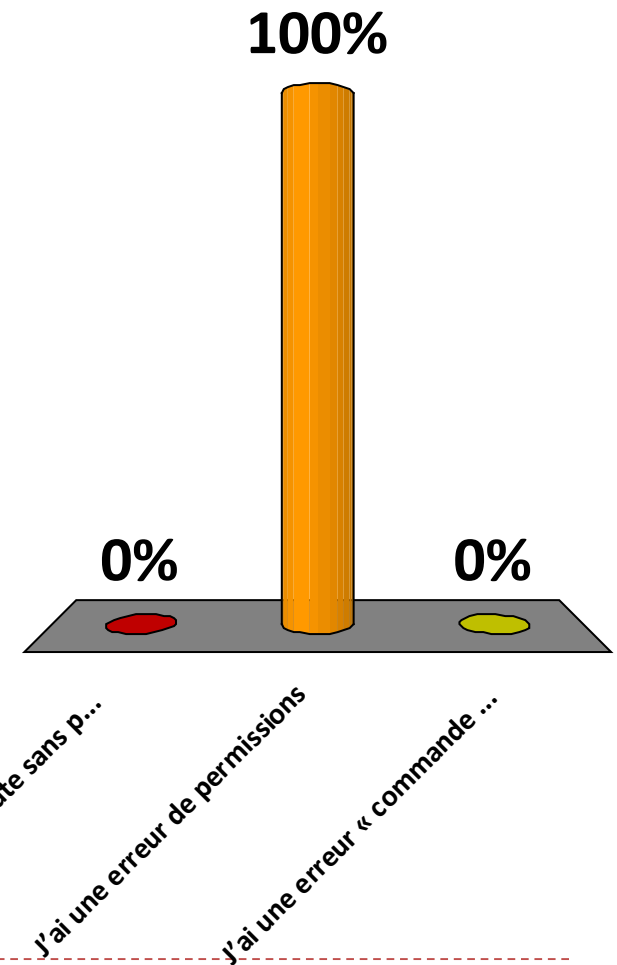
Exécution d'un script Shell

- ▶ Soit le script Shell créé de la manière suivante:

- ▶ `touch mon_script.sh`
- ▶ `gedit mon_script.sh`
- ▶ `./mon_script.sh`

- ▶ Qu'est ce qui se passe ?

1. Mon script s'exécute sans problème
2. J'ai une erreur de permissions
3. J'ai une erreur « commande introuvable »



Variables Shell

▶ Deux types de variables

▶ Variables utilisateur

- ▶ Pour vous et dans vos scripts
- ▶ En général en minuscule dans vos scripts

▶ Variables d'environnement

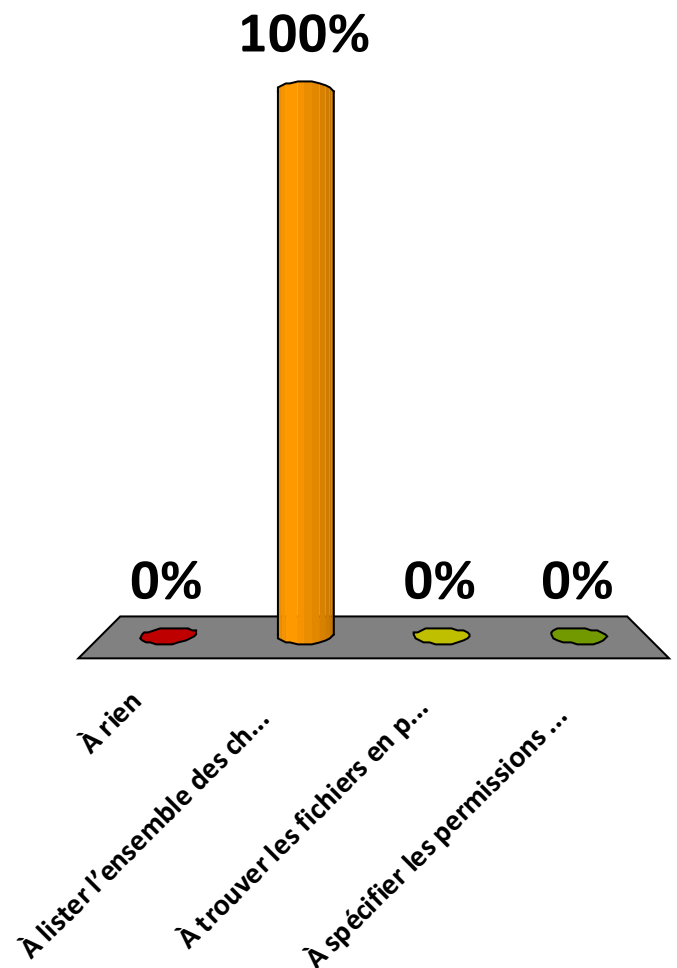
- ▶ Des variables « standards » utiles au fonctionnement du système
- ▶ En général définies en majuscule
- ▶ USER, HOME, PATH, ...

▶ Affectation et valeur d'une variable

- ▶ Attention pas d'espaces avant et après le =
- ▶ `X=22`
- ▶ `Y=$X`

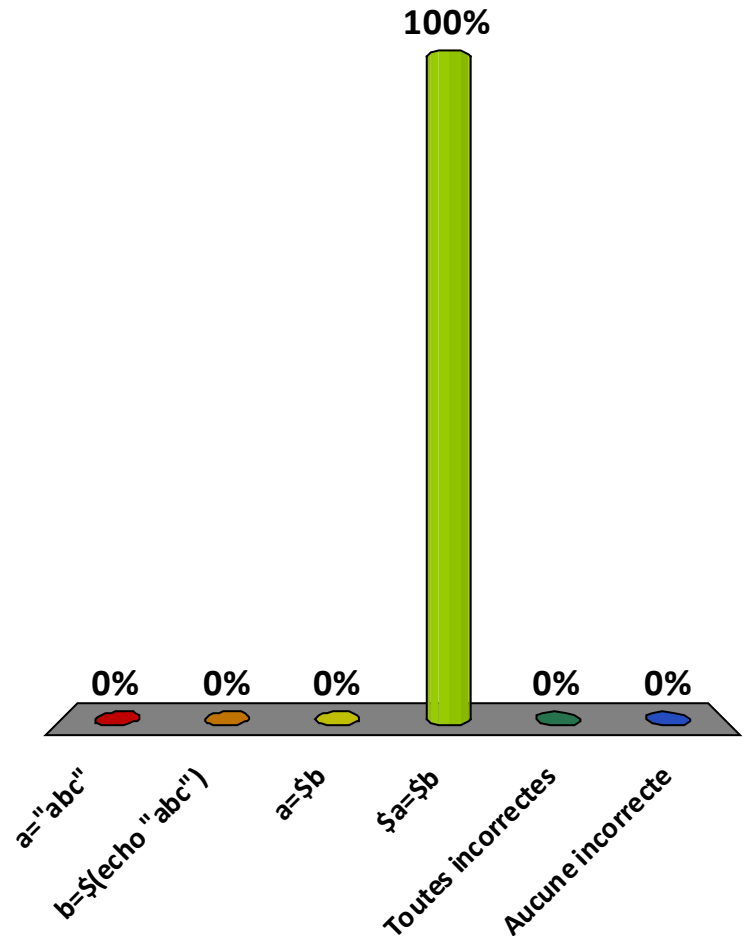
À quoi sert la variable d'environnement PATH ?

1. À rien
2. À lister l'ensemble des chemins où aller chercher les commandes
3. À trouver les fichiers en paramètre d'une commande
4. À spécifier les permissions des programmes



Quelle(s) est (sont) l'(les) initialisation(s) incorrecte(s) d'une variable ?

1. `a="abc"`
2. `b=$(echo "abc")`
3. `a=$b`
4. `$a=$b`
5. Toutes incorrectes
6. Aucune incorrecte



Nommage des variables

- ▶ En général les noms de variable de scripts Shell
 - ▶ sont en minuscule
 - ▶ mais j'ai le droit d'utiliser des majuscules si je ne redéfinit pas un nom de variable d'environnement existant
 - ▶ utilisent de préférence des caractères alphabétiques
 - ▶ mais je peux tout de même utiliser des chiffres et le caractère _
- ▶ On utilise les accolades pour définir sans ambiguïté le nom de variable
 - ▶ `${ }` à ne pas confondre avec `$ ()`

Nom de variables

► Soit les commandes suivantes:

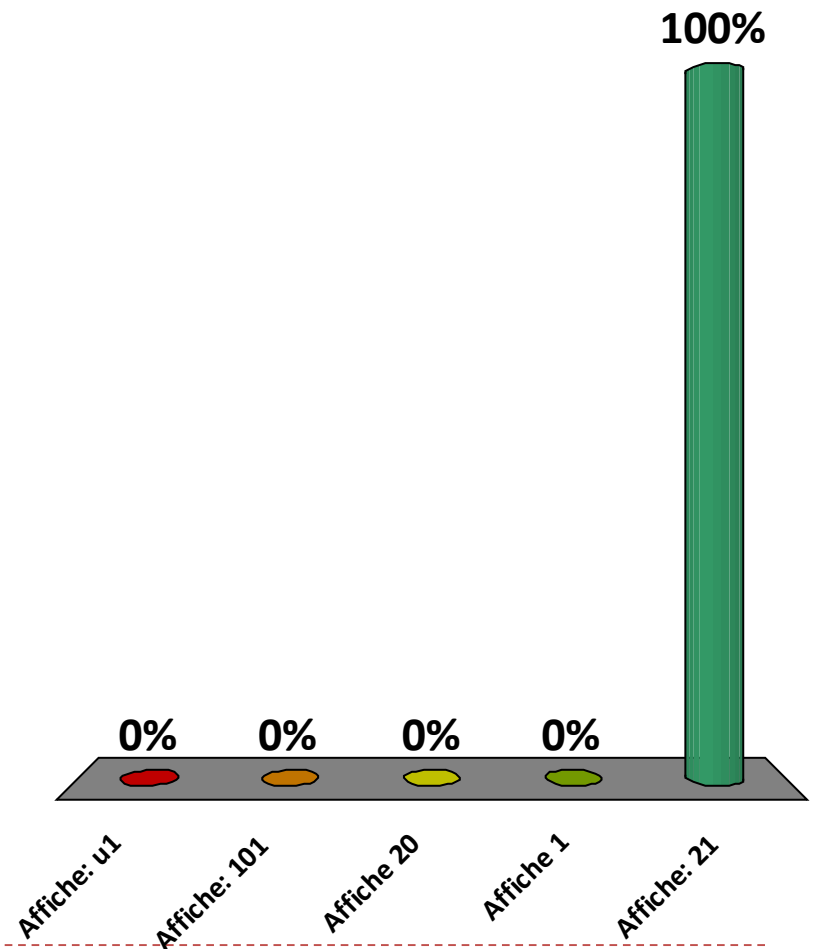
► `u1=10`

► `u=2`

► Quel sera le résultat de:

► `echo ${u}1`

1. Affiche: u1
2. Affiche: 101
3. Affiche 20
4. Affiche 1
5. Affiche: 21



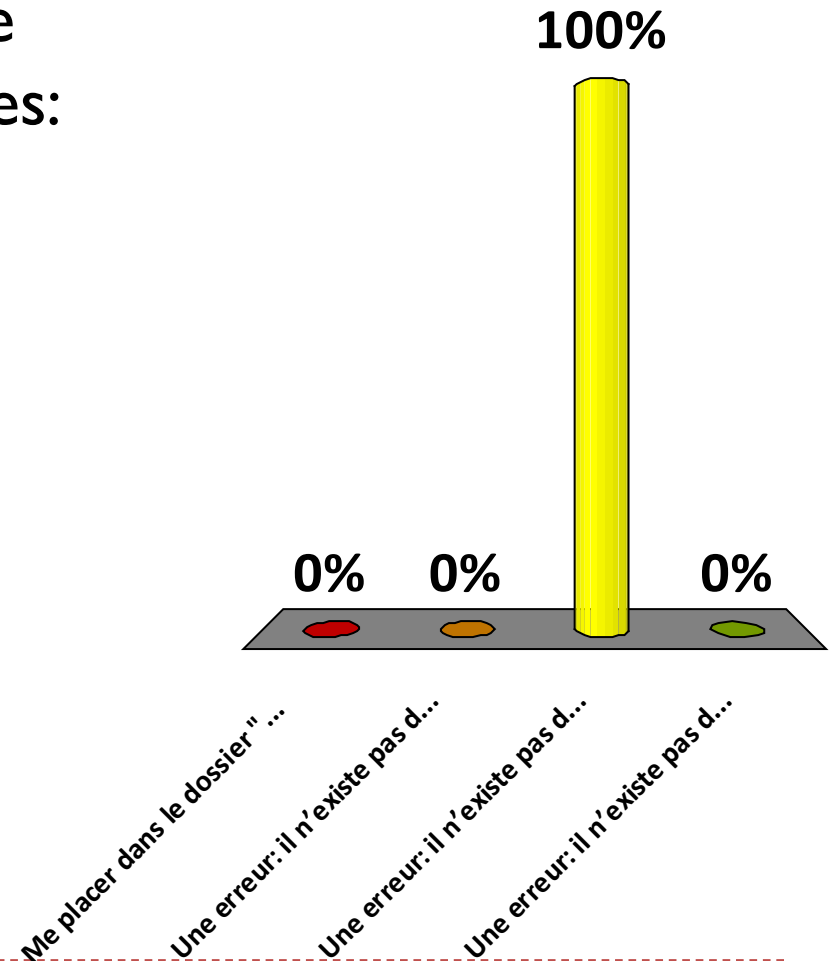


Utilisation des variables

- ▶ J'ai un seul dossier qui s'appelle "Mes Documents". Que va faire la suite de commandes suivantes:

- ▶ `dossier="Mes Documents"`
- ▶ `cd $dossier`

1. Me placer dans le dossier "Mes Documents"
2. Une erreur: il n'existe pas de dossier nommé "dossier"
3. Une erreur: il n'existe pas de dossier "Mes"
4. Une erreur: il n'existe pas de dossier "Mes Documents"



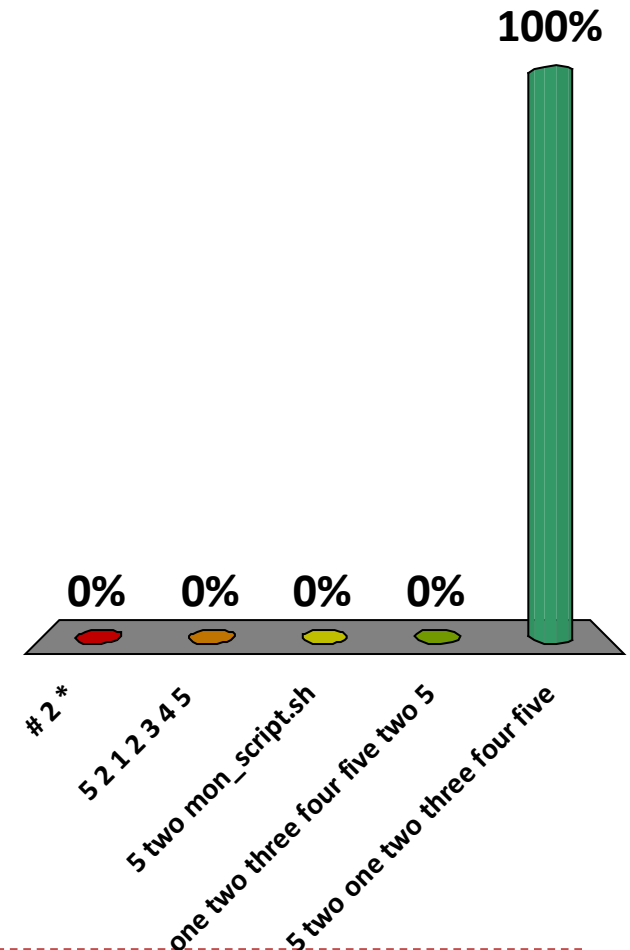
Les variables disponibles pour le script

- ▶ Des variables particulières sont créées automatiquement pour votre script Shell
 - ▶ \$0 \$1 \$2 \$3 ...
 - ▶ \$#
 - ▶ \$*
- ▶ Mais aussi
 - ▶ \$? : affiche le code d'erreur d'exécution de la commande précédente
- ▶ Mais à quoi correspondent ces variables ?

Comment récupérer un paramètre passé à mon script ?

- ▶ Soit le lancement de `mon_script.sh` de la manière suivante:
 - ▶ `mon_script.sh one two three four five`
- ▶ Si dans mon script j'ai la commande
 - ▶ `echo $# $2 $*`
- ▶ qu'est ce qui sera affiché ?

1. `# 2 *`
2. `5 2 1 2 3 4 5`
3. `5 two mon_script.sh`
4. `one two three four five two 5`
5. `5 two one two three four five`

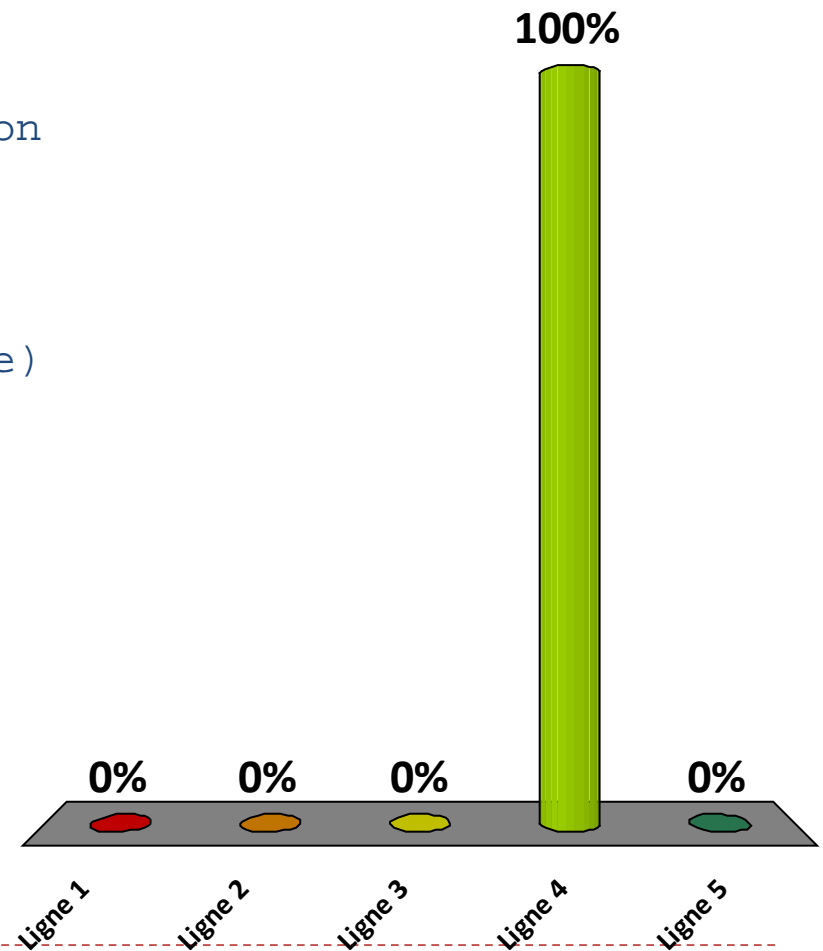


Exemple de script Shell. Où est l'erreur?

► Soit le script:

```
1. #!/bin/bash
2. read -p "Entrez un mot: " mot
3. echo "Le premier paramètre de mon
   script est: $1"
4. echo "Le mot saisi au clavier
   est: " mot
5. echo -n "Nous sommes le " $(date)
```

1. Ligne 1
2. Ligne 2
3. Ligne 3
4. Ligne 4
5. Ligne 5



Quel est le résultat obtenu ?

- ▶ Soit la suite de commandes suivantes:

- `date`

- `lundi 17 octobre 2016,16:44:02 (UTC+0200)`

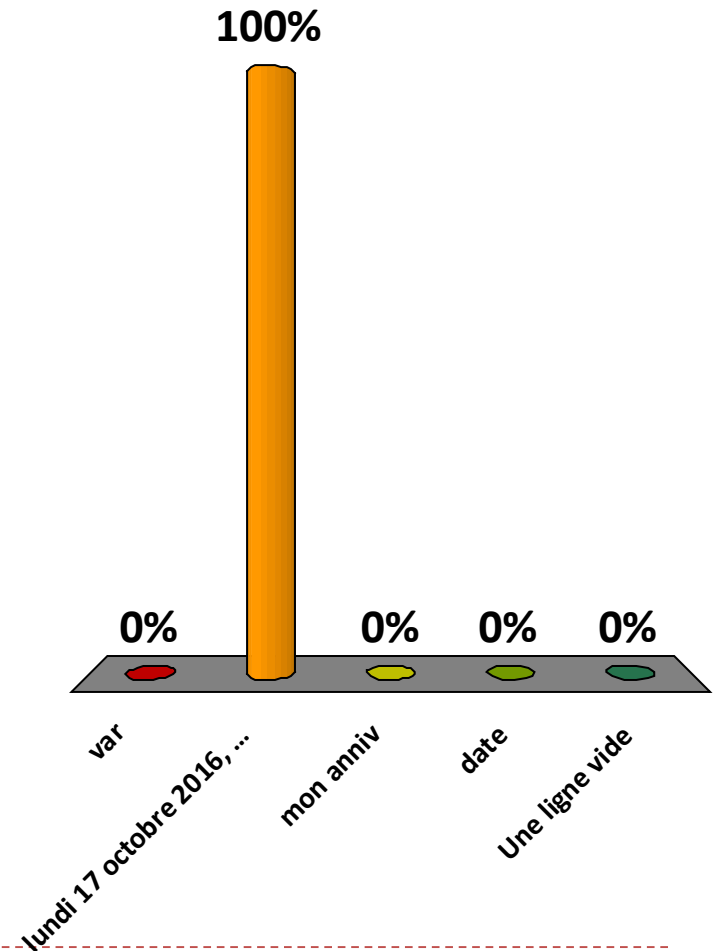
- `date="mon anniv"`

- `var=$(date)`

- ▶ Qu'est ce qui est affiché par:

- `echo ${var}`

1. `var`
2. `lundi 17 octobre 2016, ...`
3. `mon anniv`
4. `date`
5. *Une ligne vide*



Redirections

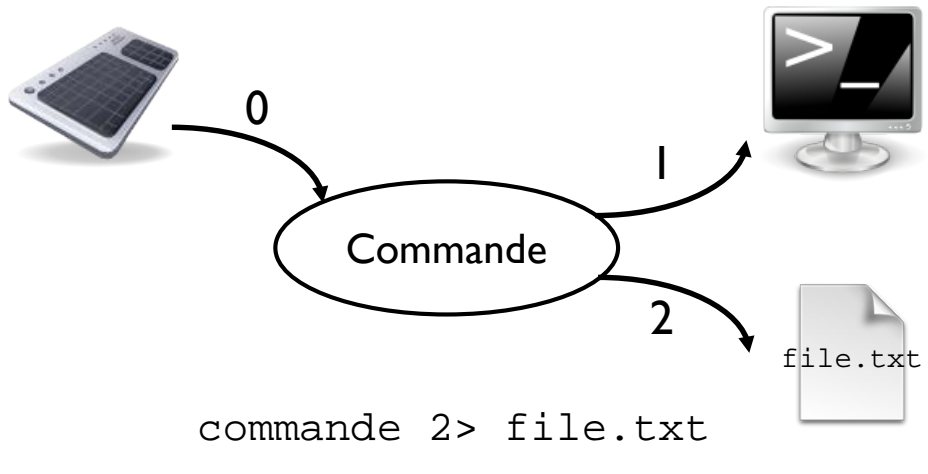
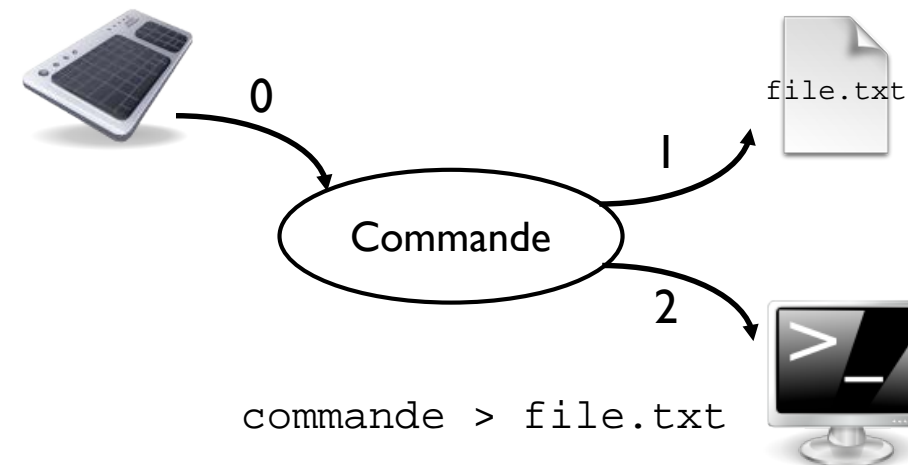
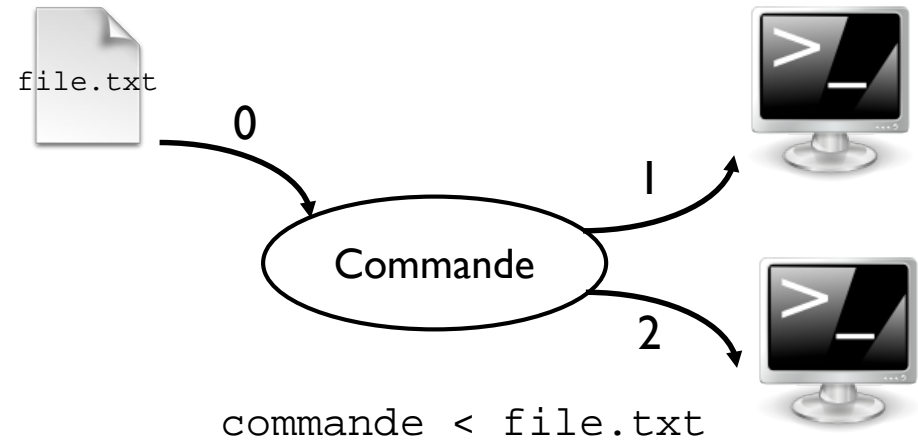
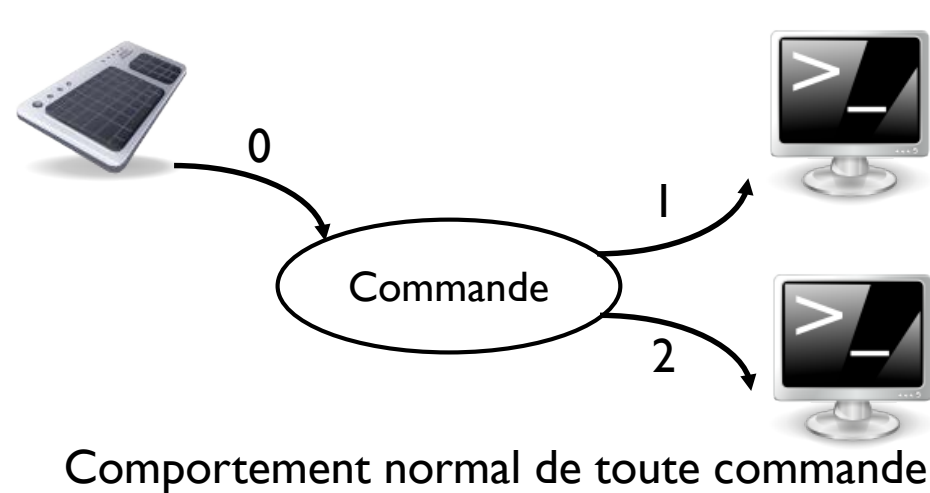
Comment gérer les flux de données grâce aux commandes
Unix

Les Redirections

- ▶ Chaque programme a des canaux de communication par défaut:
 - ▶ 0: Entrée standard (par défaut clavier)
 - ▶ 1: Sortie standard (par défaut le terminal)
 - ▶ 2: Sortie standard d'erreur (par défaut le terminal, donc écran)
- ▶ Il est possible de rediriger ces canaux de communication
 - ▶ Vers ou bien à partir d'un fichier (<, >, >>)
 - ▶ Vers un autre canal du même programme (>&)
 - ▶ Vers le canal d'un autre programme (|)
- ▶ Si on ne spécifie pas de numéro avec la redirection > ou >> c'est le canal 1 qui est celui par défaut

Les Redirections en Images

Vers des fichiers



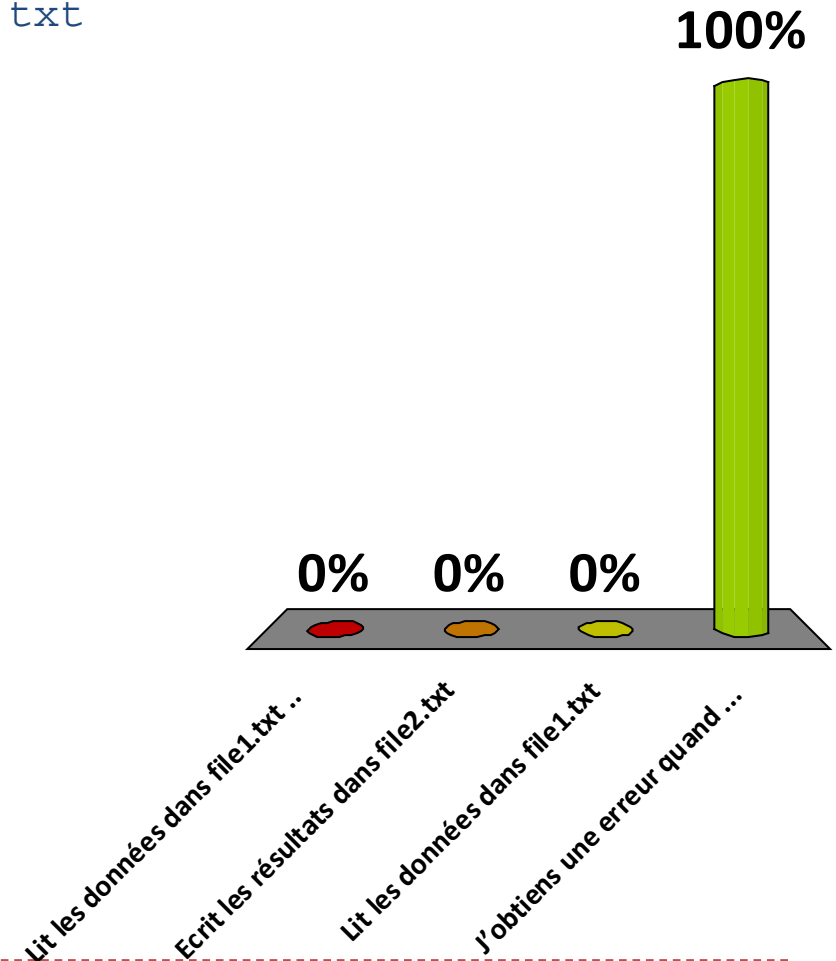
Quel est le résultat de la commande suivante ?

► Soit l'exécution suivante :

```
file1.txt < mon_script.sh > file2.txt
```

Que fait ce script avec ces deux redirections ?

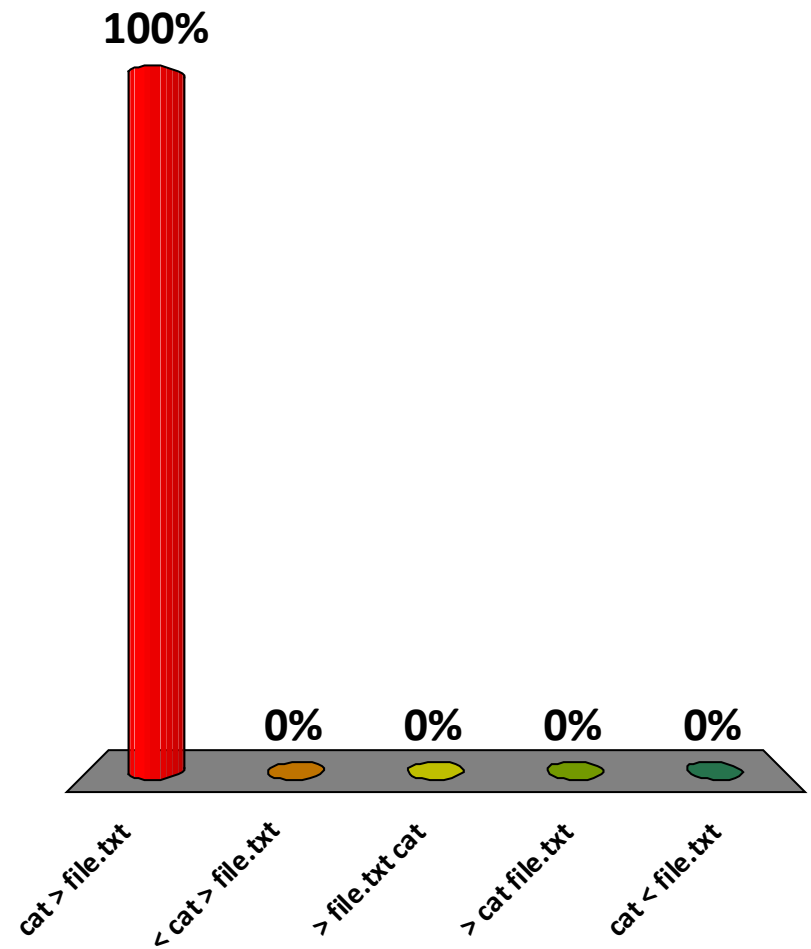
1. Lit les données dans `file1.txt` et écrit les résultats dans `file2.txt`
2. Écrit les résultats dans `file2.txt`
3. Lit les données dans `file1.txt`
4. J'obtiens une erreur quand j'exécute cette commande





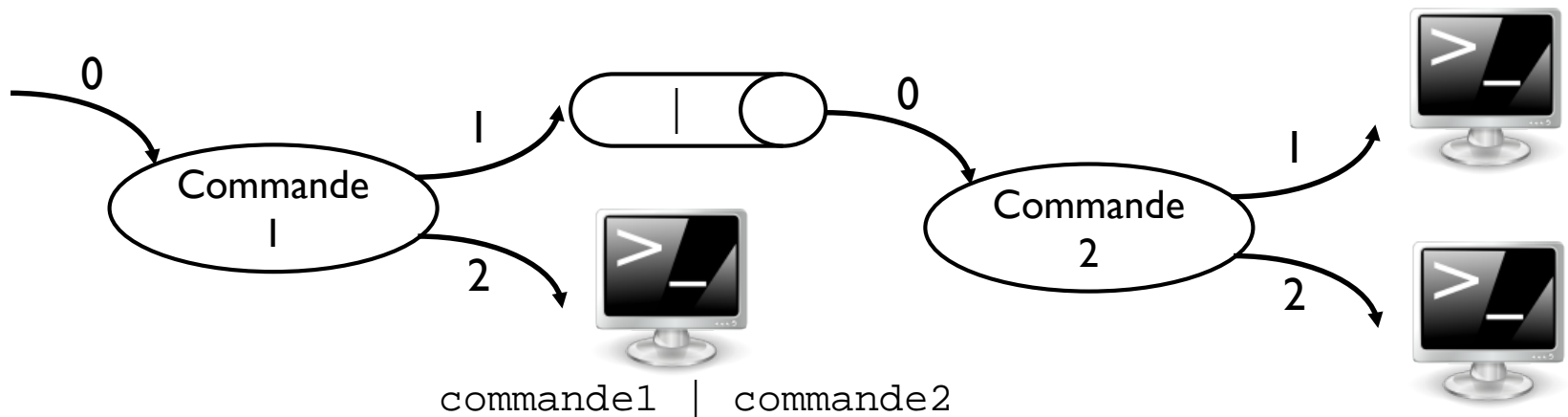
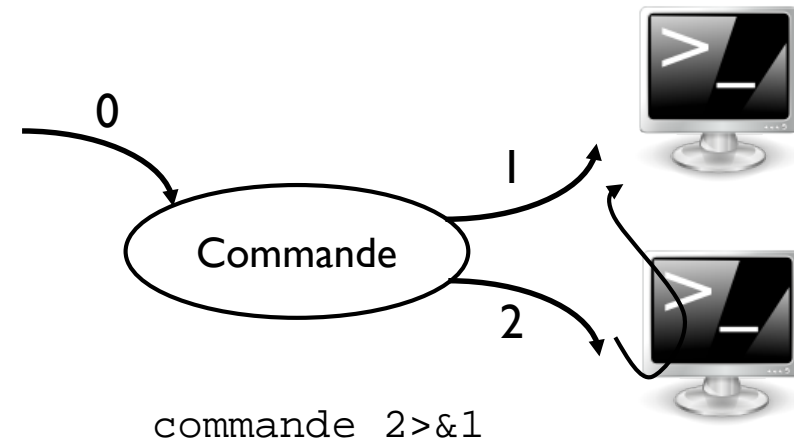
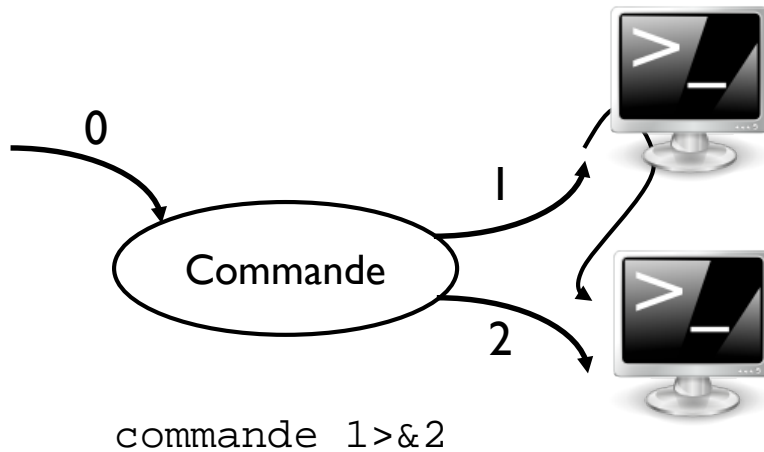
Quelle commande permet de remplir le contenu d'un fichier à partir du clavier ?

1. `cat > file.txt`
2. `< cat > file.txt`
3. `> file.txt cat`
4. `> cat file.txt`
5. `cat < file.txt`



Les Redirections en Images

Vers des canaux



Que fait la redirection suivante ?

► Soit le script `mon_script.sh`:

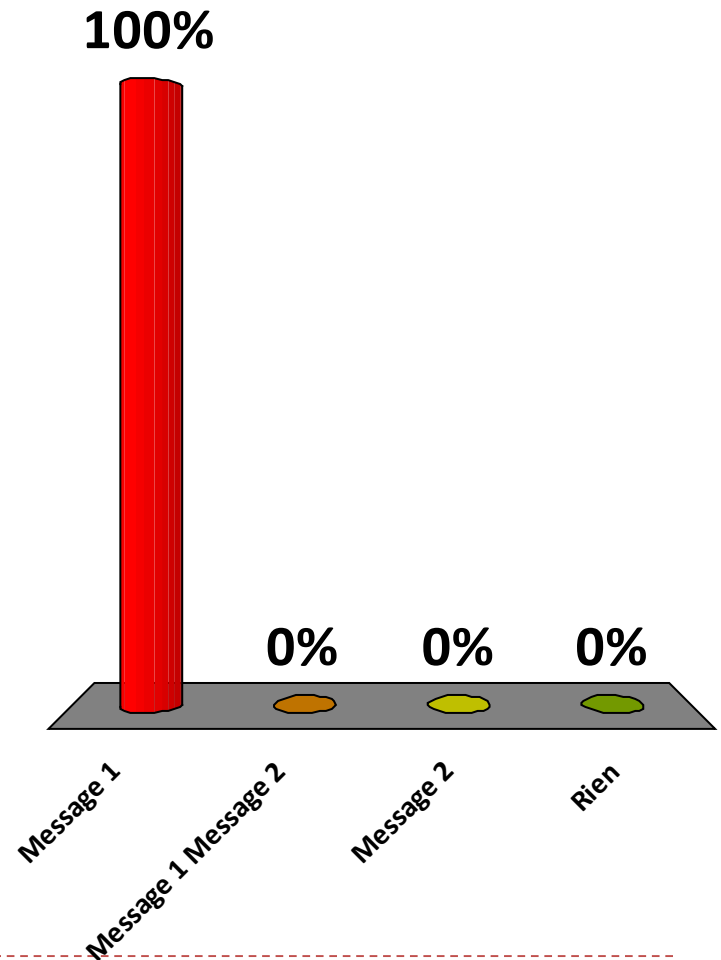
```
#!/bin/bash  
echo -n "Message 1 "  
echo "Message 2" >&2
```

et l'exécution suivante:

```
mon_script.sh > file.txt
```

Que contient le fichier `file.txt` ?

1. Message 1
2. Message 1 Message 2
3. Message 2
4. Rien





Que fait la redirection suivante ?

► Soit le script `mon_script.sh`:

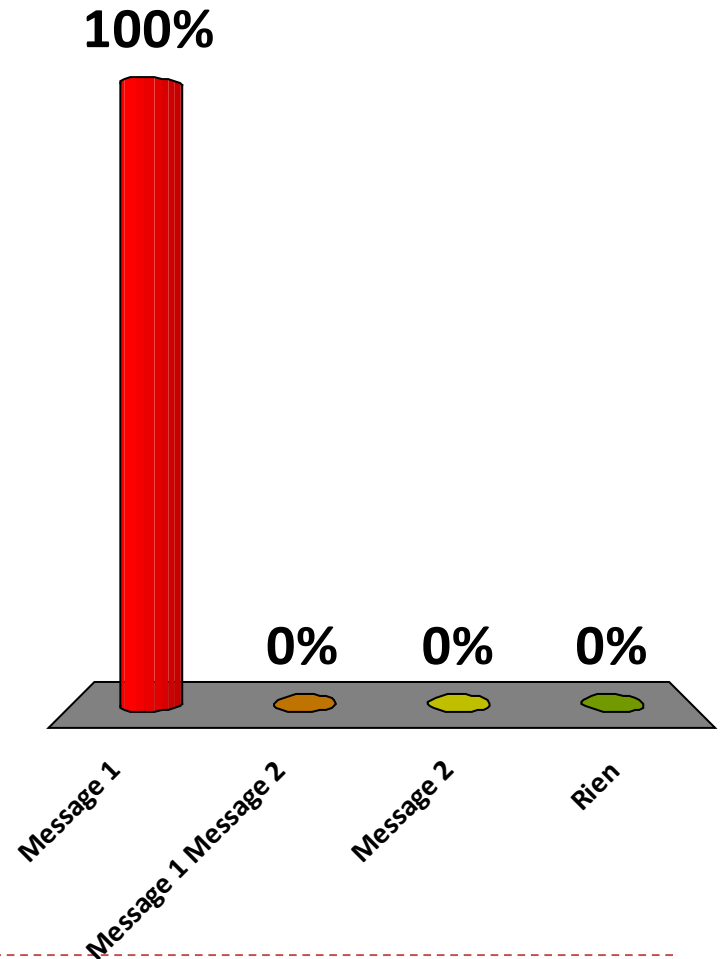
```
#!/bin/bash  
echo -n "Message 1 "  
echo "Message 2" >&2
```

et l'exécution suivante:

```
mon_script.sh 2>&1 > file.txt
```

Que contient le fichier `file.txt` ?

1. Message 1
2. Message 1 Message 2
3. Message 2
4. Rien



Que fait l'enchainement de commandes suivantes ?

► Soit le fichier `/etc/passwd` suivant:

```
user2:x:11:21::/home/user2:/bin/bash
```

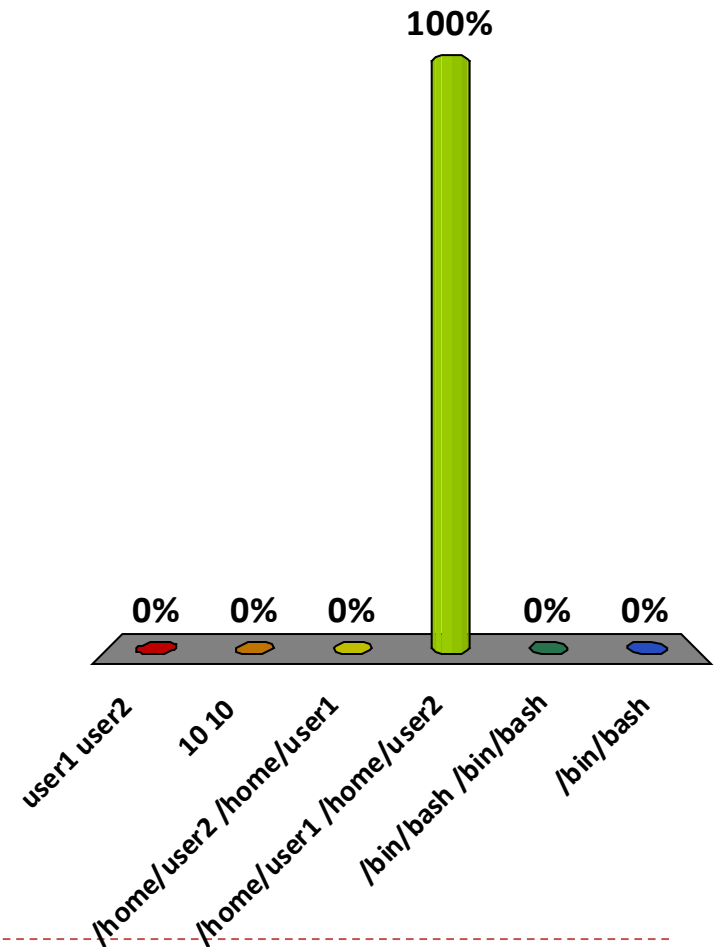
```
user1:x:10:10::/home/user1:/bin/bash
```

et l'exécution de commandes suivantes:

```
cat /etc/passwd | cut -d ":" -f 6 | sort |  
uniq
```

Qu'est ce qui sera affiché ?

1. user1 user2
2. 10 10
3. /home/user2 /home/user1
4. /home/user1 /home/user2
5. /bin/bash /bin/bash
6. /bin/bash





Utilisation des variables

► Soit les commandes:

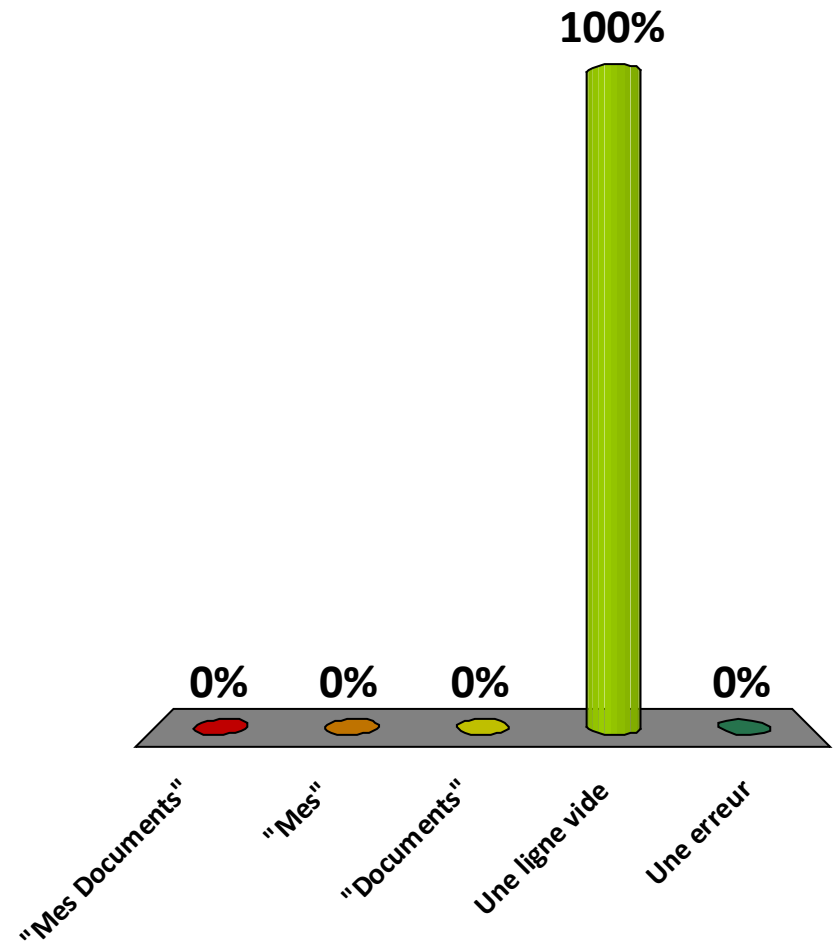
► `var=" Mes Documents "`

► `echo $var | cut -d " " -f 5`

2 espaces

► Le résultat affiché sera:

1. "Mes Documents"
2. "Mes"
3. "Documents"
4. Une ligne vide
5. Une erreur



Des suites de commandes pour « tout faire »!

- ▶ Des traitements complexes peuvent être réalisés grâce à la redirections de flots de données
 - ▶ Compter le nombre de fichiers (ou dossiers) à partir d'un dossier
 - ▶ `ls -alR /etc | grep '^-' | wc -l`
 - ▶ Créer la liste des utilisateurs (ou des groupes) de votre machine
 - ▶ `cat /etc/passwd | cut -d ":" -f 1 | sort`
 - ▶ Connaître le nombre de comptes qui ont bien un mot de passe
 - ▶ `sudo cat /etc/shadow | cut -d ":" -f 2 | grep -v "^[^*!]+$" | wc -l`
 - ▶ Compter le nombre de mots dans un document
 - ▶ ... vous le ferez en TD, je ne vais pas vous donner la solution !

Et pour la suite du cours...

La théorie de l'information (par la pratique)

Plan du cours pour la suite

- ▶ Codage de l'information
 - ▶ Codage binaire de toute information
 - ▶ Transformation d'une information binaire
 - ▶ Applications pratiques sur des fichiers de type
 - ▶ Texte
 - ▶ Son
 - ▶ Images
- ▶ Les Réseaux
 - ▶ Introduction
 - ▶ Internet et Protocoles
 - ▶ Internet et Sécurité

Merci et RdV au prochain cours

Pensez à rendre votre TD de cette semaine