# de la bonne utilisation de la commande test et des valeurs des variables

- "$var" vs $var
- les bonnes pratiques concernant l'utilisation de la commande test :
  - [ -n "$a" ] && [ "$a" = "$b" ] au lieu de [ -n "$a" -a "$a" = "$b" ]
  - toujours mettre les variables entre apostrophes ": "$var" et pas $var
  - utiliser les operateurs arithmetiques pour les nombres et literaux pour les chaines
  - utiliser [ "x$A" = "aexpression" ] pour comparer la valeur de variables (pour éviter des problèmes si $A est vide)
  - Les operateurs -e, -nt, -ot, -N, -L, -h, >, <, ne sont pas portables. -e, -h et -L sont POSIX toutefois. -h et -L sont raisonnablement portables

# substitution de variables

- attribuer une valeur par défaut aux variables:

| | valeur si la variable | | |
|---|---|---|---|
| expression | est non nulle | est nulle | effet de bord |
| ${variable:-valeur} | $variable | valeur | |
| ${variable:=valeur} | $variable | valeur | la variable prend la valeur valeur |
| ${variable:+valeur} | valeur | $variable | |

- afficher un message d'erreur si la variable est vide:
  - ${variable:?message}: si la variable est vide, le shell affiche le message et provoque la terminaison du script shell. Si la variable n'est pas vide, l'expression vaut $variable

# substitution de variables (2)

- longueur de la valeur d'une variable: ${#variable}
- suppression de fragments :

| | | | |
|---|---|---|---|
| à gauche | | | |
| | # | plus petit | ${variable#modele} |
| | ## | plus grand | ${variable##modele} |
| à droite | | | |
| | % | plus petit | ${variable%modele} |
| | %% | plus grand | ${variable%%modele} |

- le modèle doit être conforme à ceux utilisés pour la substitution de noms de fichiers du shell utilisé
- avec bash, il faut positionner l'option extglob par « shopt -s extglob » pour pouvoir utiliser les expressions complexes dans les modèles.

# substitution de variables : exemples

```
f=/usr/local/Photos/reveDeVacancesderevealamon
  tagne.jpg
$ echo ${f#/}
usr/local/Photos/reveDeVacancesderevealamontag
  ne.jpg
$# suppression du premier / du nom
$ echo ${f#*/}
usr/local/Photos/reveDeVacancesderevealamontag
  ne.jpg
$ # suppression du chemin
$ g=${f##*/}
$ echo $g
reveDeVacancesderevealamontagne.jpg
```

# substitution de variables : exemples (2)

```
$# la variable f n'a pas été modifiée
$ echo $f
/
  usr/local/Photos/reveDeVacancesderevealamont
  agne.jpg
$# pour supprimer l'extension d'un fichier
$ echo ${g%.*}
reveDeVacancesderevealamontagne
$#obtenir le chemin
$ echo ${f%/*}
/usr/local/Photos
$ echo ${g%%ve*}
re
$ echo ${g%ve*}
reveDeVacancesdere
```

# break et continue

- break: permet de forcer la sortie d’une boucle for, while ou until. Il peut prendre un argument strictement supérieur à 1 afin de spécifier le nombre de boucles imbriquées dont il faut sortir. Si la valeur spécifiée est supérieure au nombre de boucles imbriquées, on sort de toutes les boucles imbriquées.
- continue: Il permet de passer `a la prochaine it´eration d’une boucle for, while ou until. Le mot clef continue peut aussi prendre un argument strictement supérieur à 1 afin de spécifier le nombre d’itérations à ”sauter”. Si la valeur spécifiée est supérieure au nombre d’itérations possibles, la dernière itération est toutefois exécutée.

# initialisation des paramètres positionnels avec set

- set: suivie d'arguments permet d'affecter ces arguments aux paramètres positionnels
- les caractères présents dans la variable IFS séparent les arguments
- Exemples:

```
$ set a b c
$ echo $1
a
$set $(date)
$echo $2
dèc
```

# Interprétation de la ligne de commande

- séparation en mots
- expansion des accolades
- expansion du ~
- expansion des paramètres, des variables
- expansion des commandes
- évaluation des expressions arithmétiques
- découpage des mots
- développement des noms de fichiers
- suppression des " \ ' non protégés ne résultant pas d'un développement

# découpage des mots

- découpage de tout ce qui n'est pas entre guillemets
- les caractères séparateurs sont dans la variable IFS (par défaut: espace, tab,  retour chariot)
- les arguments explicitement nuls (" ou "") sont conservés. Ceux résultant du développement d'un paramètre sans valeur sont éliminés

# Développement

- des accolades :
  - {mot1, mot2, mot3, ...}
  - file{2,34,IMP} donne file2 file34 fileIMP
- du tilde:
  - ~nomLogin
  - ~petit
- paramètres/variables:
  - ${parametre}ou $parametre
  - $!var: on remplace !var par la valeur de var avant de faire l'expansion (indirection)

# Développement  des commandes

- $(commande)
- ou `commande` (ancienne forme déconseillée)
- la sortie standard de la commande remplace l'expression
- ls -l $(grep -il toto *.c)

# développement des expressions arithmétiques

- $(( expression))
- l'expression est traitée comme si elle se trouvait entre guillemets (sans traitement spécifique pour le guillemet)
- l'expression subit le développement des paramètres/variables et des commandes
- x=1; echo nouvelle valeur $((x=x+3))

# développement des noms de fichiers

- *: n'importe quelle séquence
- ?: un caractère quelconque
- [abcdef]: un caractère parmi tous les caractères entre crochet
- [a-f]: tous les caractères compris entre a et f
- [- : le caractère -
- [^ ou [!: tout sauf les caractères qui suivent ^ ou !