LE SHELL

L'exécution des commandes shell du fichier nomProc se fait avec la commande **bash** *(interprète du langage de commandes) :*

bash *nomProc [arguments...]*

*Si le fichier de commandes nomProc a été rendu exécutable (***chmod u+x** *nomProc), le mot* **bash** *devient facultatif.*

Commande

Une commande est une suite de mots séparés par des espaces : le premier étant le nom de la commande (argument 0), les autres étant passés comme paramètres (1, 2, 3, ...) à la commande. La commande renvoie un code de retour, 0 s'il n'y a pas d'erreur, ≠ 0 sinon.

Paramètres :

$0	: désigne le nom de la commande.
$1, $2, $3, ...	: désigne les paramètres de la commande.
$*	: désigne la liste des paramètres **$1 $2 $3**...

Canal (pipe)

C'est une suite de plusieurs commandes séparées par le symbole **|**. La sortie standard de chaque commande (sauf la dernière) devient l'entrée standard de la suivante.

Liste

C'est une séquence de commandes séparées par **;** ou **&&** ou **||** :

;	: exécution en séquence;
&&	: exécution si la commande précédente a renvoyé le code de retour 0;
||	: exécution si la commande précédente a renvoyé un code de retour ≠ 0;

Caractères spéciaux

Les caractères : **; & | ( ) < >** doivent être placés entre apostrophes ou bien précédés par un caractère \ pour ne pas être interprétés par le shell.

Commandes internes

break *[n]*	: sortie d'une boucle **for**, **until** ou **while** ; sortie de *n* niveaux, si *n* est précisé.
exit *[n]*	: sortie du processus shell, avec le code retour *n*, si *n* est précisé.
read *nom1 nom2*	: lit les variables *nom1*, *nom2*... à partir de l'entrée standard ; les mots de la ligne lue sont affectés aux variables *nom1* dans l'ordre.

Variables prédéfinies

Les variables suivantes sont automatiquement affectées par le shell :

#	: désigne le nombre de variables de position de la commande.
?	: désigne la valeur retournée par la dernière commande exécutée.
$	: désigne le numéro du processus courant.
!	: désigne le numéro du dernier processus lancé en parallèle.

Les structures d'enchaînement

boucle `for` :

```
for variable [ in mot... ]
do
     listeDeCommandes
done
```

sélection multiple `case` :

```
case valeur in
     modèle1 ) listeDeCommandes1 ;;
     modèle2 ) listeDeCommandes2 ;;
     ...
     * ) listeDeCommandesAutres ;;
esac
```

instruction conditionnelle `if` :

Forme générale

```
if listeDeCommandes1
then
     listeDeCommandes2
[ else
     listeDeCommandes3 ]
fi
```

Forme spécifique

```
if test expression
then
     listeDeCommandes1
[ else
     listeDeCommandes2 ]
fi
```

boucle `while` :

Forme générale

```
while listeDeCommande1
do
     listeDeCommandes2
done
```

Forme spécifique

```
while test expression
do
     listeDeCommandes
done
```

La commande d'évaluation de condition : `test`

Cette commande évalue une *expression* et renvoie un code de retour égal à 0 si expression est **vrai**, un code différent de 0 **sinon**.
Si la commande **`test`** est utilisée sans arguments, elle renvoie un code différent de 0.

Forme simplifiée :

`test` *expression* ⇔ `[` *expression* `]`

primitives utilisées pour construire *expression* :

- `-r` *fichier* : vrai si le *fichier* existe et s'il est accessible en lecture ;
- `-w` *fichier* : vrai si le *fichier* existe et s'il est accessible en écriture ;
- `-f` *fichier* : vrai si le *fichier* existe et si ce n'est pas un répertoire ;
- `-d` *fichier* : vrai si le *fichier* existe et si c'est un répertoire ;
- `-s` *fichier* : vrai si le *fichier* existe et s'il n'est pas vide ;

- `-z` *ch1* : vrai si la longueur de la chaîne *ch1* est zéro ;
- `-n` *ch1* : vrai si la longueur de la chaîne *ch1* n'est pas zéro ;
- *ch1* `=` *ch2* : vrai si les chaînes *ch1* et *ch2* sont égales ;
- *ch1* `!=` *ch2* : vrai si les chaînes *ch1* et *ch2* ne sont pas égales ;

- *n1* `-eq` *n2* : vrai si les entiers *n1* et *n2* sont égaux algébriquement ;
  `-eq` (=) peut être remplacé par les comparateurs suivants :
  `-ne` (≠), `-gt` (>), `-ge` (≥), `-lt` (<) ou `-le` (≤);

Ces primitives peuvent être combinées avec les opérateurs suivants :

- `!` : non
- `-a` : et
- `-o` : ou