

## FICHE 15 – Scripts: variables

### 1. pidof.sh:

```
#!/bin/bash  
ps -ef | grep "$1" | grep -v grep | grep -v "$0" | tr -s ' ' | cut -d " " -f 2,8,9
```

OU

```
ps -C "$1" -o pid=,cmd=
```

### 2. echo \$PATH

PATH=

echo \$PATH

### 3. cd .. → fonctionne car c'est une *shell builtin command*

- pour lister les *builtin commands* : *help*
- pour connaître le statut de la commande cmd : *type cmd* OU *type -t cmd*

<http://aral.iut-rodez.fr/fr/sanchis/enseignement/bash/ch01s03.html>

### 4. ls → « Aucun fichier ou dossier de ce type » car ls est une commande trouvée via le PATH (*shell builtin*)

### 5. PATH=/usr/bin:/bin ; ls → pas de souci : commande ls trouvée

### 6. echo "\$PATH contient \$PATH"

### 7.

- a. **mkdir -p ~/.local/bin**
- b. **mv ~/pidof.sh ~/.local/bin**
- c. **PATH=\$PATH:~/.local/bin**
- d. **cd ~ ; pidof.sh init**

### 8.

- a. **nano ~/.bashrc**
- b. y ajouter : **PATH=\$PATH:~/.local/bin**

### 9. info.sh :

en utilisant les variables d'environnement

```
#!/bin/bash  
output="$HOSTNAME"  
output="$output\n$USER"  
output="$output\n$HOME"  
output="$output\n$PWD"  
output="$output\n$(date) "  
echo -e $output
```

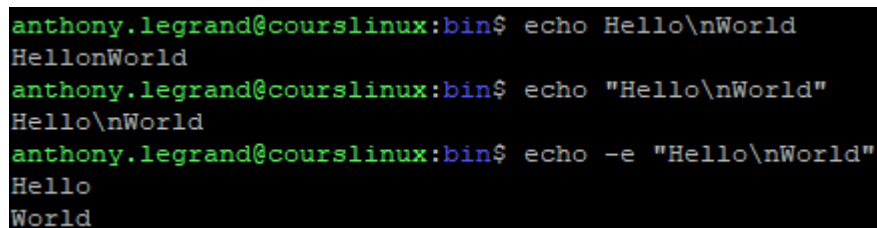
OU en utilisant des commandes

```
#!/bin/bash
output="$(hostname) "
output="${output}\n$(whoami) "
output="${output}\n$(ls -d ~) "
output="${output}\n$(pwd) "
output="${output}\n$(date) "
echo -e $output
```

#### Explications :

Les passages à la ligne doivent être encadrés par des guillemets. En effet, rappelons que le backslash sert à « échapper » (*escape*) un caractère spécial (càd. ayant une fonctionnalité associée) pour le traiter littéralement. Dans le cas du `\n`, la lettre `n` n'a aucune fonctionnalité associée et le `\` n'aura aucune effet : la lettre `n` sera simplement affichée, sans le `\`. Pour conserver la séquence d'échappement (*escape sequence*) `\n`, il faut la mettre entre guillemets.

L'option `-e` de la commande `echo` permet d'activer l'interprétation des séquences d'échappement, telles que `\n` qui représente un passage à la ligne (cf. `man echo`). Sans cette option, `echo` affiche les séquences d'échappement comme de simples suites de caractères.



```
anthony.legrand@courslinux:bin$ echo Hello\nWorld
HellonWorld
anthony.legrand@courslinux:bin$ echo "Hello\nWorld"
Hello\nWorld
anthony.legrand@courslinux:bin$ echo -e "Hello\nWorld"
Hello
World
```

## FICHE 16 – Scripts : entrées

1.

```
#!/bin/bash
#####*/
# entete.sh
# genere un entete dans une variable
#####*/

ligne="#####"

entete='#!/$SHELL
entete="$entete\n$ligne"
entete="$entete\n# $1"
entete="$entete\n# $(whoami) "
entete="$entete\n# $(date +%d %B %Y) "
entete="$entete\n$ligne"

echo -e $entete
```

echo -e → interpréter les caractères échappés (i.e. suivis de \ ; ex: \n, \t)

2.

```
#!/bin/bash
#####*/
# entete.sh
# genere un entete dans une variable, comprenant une description
# saisie au clavier
#####*/

ligne="#####"

# lecture de la description
echo "Donnez la description du script <Ctrl-D> pour terminer"
read l1
read l2
read l3

entete='#!/$SHELL
entete=$entete"\n$ligne"
entete="$entete\n# $1"
entete="$entete\n#\t$l1"
entete="$entete\n#\t$l2"
entete="$entete\n#\t$l3"
entete="$entete# $USER"
entete="$entete\n# $(date +%d %B %Y) "
entete=$entete"\n$ligne"

echo -e $entete
```

OU version avec while (mais les boucles ne seront vues qu'au chapitre 19).

```
# lecture de la description
echo "Donnez la description du script <Ctrl-D> pour terminer"
while read desc; do
    maDesc="$maDesc#\t$desc\n"
done

entete='#!'$SHELL
entete=$entete"\n$ligne"
entete="$entete\n# $1"
entete="$entete\n#$maDesc"
entete="$entete# $USER"
entete="$entete\n# $(date +%d %B %Y)"
entete=$entete"\n$ligne"

echo -e $entete
```

3.

```
#!/bin/bash
#####*/
# entete.sh
# genere un entete dans un script, comprenant
# une description saisie au clavier et rend le script executable
#####*/

ligne="#####"

# lecture de la description
echo "Donnez la description du script <Ctrl-D> pour terminer"
read l1
read l2
read l3

entete='#!'$SHELL
entete=$entete"\n$ligne"
entete="$entete\n# $1"
entete="$entete\n#\t$l1"
entete="$entete\n#\t$l2"
entete="$entete\n#\t$l3"
entete="$entete\n# $USER"
entete="$entete\n# $(date +%d %B %Y)"
entete=$entete"\n$ligne"

echo -e $entete > "$1"
chmod u+x "$1"
```

Variante avec plusieurs affichages (pas de concaténation dans une variable) :

```
#!/bin/bash

read -p "ligne 1: " l1
read -p "ligne 2: " l2
read -p "ligne 3: " l3
```

```
echo "#!$SHELL" > "$1"
echo "#####" >> "$1"
echo "# $1" >> "$1"
echo "# $11" >> "$1"
echo "# $12" >> "$1"
echo "# $13" >> "$1"
echo "# $USER" >> "$1"
echo "# $(date +%d %B %Y)" >> "$1"
echo "#####" >> "$1"

chmod +x "$1"
```