

Signaux: principaux signaux

nom du signal	No	cas usuel	comportement par défaut d'un processus le recevant	comportement modifiable ?
HUP	1	perte de connexion	Mort	Oui
INT	2	Ctrl-C sur le clavier	Mort	Oui
KILL	9	processus qui ne répond pas à kill -TERM	Mort	Non
TERM	15	chance de faire le ménage	Mort	Oui

gestion des signaux: TRAP

- TRAP: permet d'ignorer un signal ou de modifier le comportement associé à un signal
- utile pour garantir qu'un script fera le ménage avant de mourir élégamment
- Syntaxe:
 - trap 'liste de commandes' signal1 [signal2 [signal3 ...]]
- Exemples:
 - ignorer un signal: trap " signal
 - gérer un signal: trap 'liste de commandes » signal
 - remettre le traitement par défaut: trap - signal

TRAP: exemple

```
#!/bin/bash
fichierTemp=/tmp/$0.$$
#en cas de signal, on fait le ménage:
trap 'rm -f "$fichierTemp";exit 1'\
    SIGHUP SIGINT SIGTERM
#reste du script qui peut
#potentiellement modifier fichierTemp
commande1
commande2
```

analyse des options de ligne de commande

- 2 méthodes:
 - utiliser la commande interne `getopts` qui permet de gérer des options d'une lettre
 - à la main : boucle `while/case/shift` (avantage: on peut utiliser des noms longs pour les options)

getopt

- une option est composée d'une lettre précédée d'un + ou d'un -
- une option peut être suivie d'un argument
- exemple: `getopt "abc:d" var`
 - les options possibles sont -a, -b, -c, -d
 - le « : » après le « c » indique que l'option -c attend un paramètre
 - var est le nom de la variable dans laquelle sera mis le nom de l'option lue
 - l'argument éventuel sera mis dans la variable réservée OPTARG
 - getopt s'utilise dans une boucle while

getopt

- **Exemple:**

```
while getopt "ac:" option
do
    case "$option" in
        a) echo option a
            ;;
        c) echo option a avec l'argument
"$OPTARG"
            ;;
        \?) echo option invalide
            ;;
    esac
done
```

getopt: gestion des erreurs

- lorsque getopt rencontre une option invalide:
 - la variable var contient le caractère ?
 - un message est affiché à l'écran
- si : est en première position dans la liste des options:
 - aucun message ne sera affiché à l'écran
 - OPTARG contiendra la valeur de l'option fautive
 - exemple: getopt " : abc : d" var
- paramètre manquant:
 - var contient : et OPTARG contient le caractère de

gestion artisanale des arguments

```
while [ $# -gt 0 ]; do
  case "$1" in
    -Prefix) shift
              [ $# -gt 0 ] || { affErr; exit 1; }
              Prefix="$1"
              shift
              ;;
    -remoteComputer) shift
                    [ $# -gt 0 ] || { affErr; exit 1; }
                    remoteComputer="$1"
                    shift
                    ;;
    *) repASauver="$1"
      shift
      ;;
  esac
done
```

gestion d'un processus en arrière plan: wait

- `WAIT p`: attend la terminaison du processus dont le pid est p
- `WAIT`: attend la terminaison de tous les processus lancés en arrière plan à partir du shell courant
- Exemple: pour profiter d'une machine bipro:

```
gzip -9 fichier1&
```

```
gzip -9 fichier2&
```

```
wait
```

```
mv fichier1 fichier2 /archive/
```

commande find

- find permet de chercher récursivement les fichiers vérifiant une ou plusieurs conditions
- outre les expressions simples, l'expression que doit vérifier un fichier peut être de la forme (par priorité décroissante) :
 - (expression)
 - ! expression
 - expression1 -a expression2 : ET logique
 - expression1 expression2: ET logique
 - expression1 -o expression2: OU logique

commande find

- expressions élémentaires à argument numérique:
 - +n: toutes les valeurs supérieures ou égales à n
 - -n: toutes les valeurs inférieures ou égales à n
 - n: n exactement
- par la suite, partout où on verra un argument numérique n, on pourra utiliser +n, n ou -n
- exemples:
 - -size 1024k: les fichiers de taille égale à 1024 Ko
 - -size -1024k: les fichiers de taille inférieure égale à 1024 Ko
 - -size +1024k: les fichiers de taille supérieur ou égale à 1024 Ko

commande find: quelques expressions élémentaires

- quelques expressions élémentaires:
 - -name motifProtégé: les fichiers vérifiant le motif
 - -size n: les fichiers de taille n
 - -mtime n, -ctime n, -atime n
 - -perm p avec p ayant la forme numérique ou symbolique des arguments de chmod
 - -type c avec c=b,c,d (dossier),l (lien symbolique),p,f (fichier ordinaire),s
 - -user u
 - -group g
 - -link n : nombre de liens physique sur le fichier
 - -print: provoque l'affichage des noms des fichiers vérifiant l'expression (par défaut sur le Gnu find)
 - ...

commande find: -exec

- -exec commande;
 - pour chaque fichier trouvé, la commande est exécutée.
 - si {} apparaît parmi les arguments de la commande, il est remplacé par le nom du fichier trouvé
- -exec commande arguments {} +
 - syntaxe POSIX/SUSv3 (standard mais pas disponible sur toutes les plateformes)
 - les noms des fichiers trouvés sont accumulés dans une liste l
 - la commande est exécutée une seule fois, à la fin de la recherche et {} est remplacé par la liste des arguments

commande find : exemples

- fichiers ordinaire nommés core de plus de 1024Ko
 - `find -size +1024k -type f -name core -print`
- fichier ordinaires de l'utilisateur petit ou fichiers ordinaires de taille supérieure à 1024 Ko et de nom core
 - `find -type f \(-user petit -o \(-size +1024k -name core \) \) -print`
- fichier dont le nom commence par C
 - `find -name c* -print`

commande find: -exec et les espaces (&Co)

- rm est une commande qui ne lit pas sur son entrée standard. 3 méthodes pour effacer un ensemble de fichiers sélectionnés par find :
 - on lance un rm par fichier (lourd)
 - `find -name *.bak -exec rm -f {} \;`
 - syntaxe POSIX: un seul rm global est lancé :
 - `find -name *.bak -exec rm -f {} \+`
 - une solution avec les options spécifiques de Gnu find:
 - `find -name *.bak -print0 |xargs -0 rm -f`
 - ne marche pas avec les noms de fichiers contenant des caractères à problème (espace, saut de ligne, ', etc.):
 - `find -name *.bak -print |xargs rm -f`