Écrire un script

- saisir le texte dans un fichier

```
#
echo "bonjour !"
echo -n "aujourd'hui c'est "
date +%A
```

- rendre le fichier exécutable, exemples

```
chmod +x mon-script
chmod 755 mon-script
```

- exécution, exemples

```
mon-script
./mon-script
```


la variable PATH

- contient une liste de répertoires

```
$ echo $PATH
/usr/bin:/bin:/usr/games:.
```

qui indiquent où chercher les commandes

- Si "." (répertoire courant) est dans le PATH, on peut faire "mon-script"
- sinon, faire "./mon-script"

Scripts : commentaires

- les commentaires commencent par #
- en première ligne, commentaire spécial
  `#! /bin/bash`
  qui précise l'interpréte à utiliser.
- Avec un dièse seul en première ligne, c'est le shell par défaut du système (en général `/bin/sh`).

Travail

1. taper le script

```
001    #! /bin/bash
002    # Premier essai
003    clear
004    echo "Bonjour $USER"
005    echo -n nous sommes
006    date "+%A %d %B %Y"
007
008    echo -n Il y a
009    who | wc -l
010    echo connexions sur $HOSTNAME
011    exit 0
```


Variables, paramètres, expressions ...

Les variables du shell

- mémorisent des chaînes de caractères
- bash permet aussi les nombres et les tableaux
- variables système définies automatiquement
  - liste fournie par la commande set
  - HOME PWD SHELL USERNAME PATH LANG etc.
- l'utilisateur peut définir ses propres variables

Exercice

Quelles variables indiquent

- le nom de votre poste de travail,
- le type du processeur
- la version du système d'exploitation ?

paramètres positionnels

- Un script peut être invoqué avec des paramètres
- Exemple :
  `./mon-script laurel hardy`.
- Ces paramètres sont transmis par $1, $2, ...
- de plus
  - $# = nombre de paramêtres
  - $* = liste complète des paramètres

Observez

Tapez le script suivant, et essayez-le

```
#
echo il y a $# paramètres
echo '$0 contient ' $0
echo '$1 contient ' $1
echo '$2 contient ' $2
```

avec différentes combinaisons, dont des chaînes :

- `./mon-script un deux`
- `./mon-script "un deux"`
- `./mon-script "Nicolas S" "Segolene R".`

Exercice

Écrire un script

- à un paramètre (l'identifiant d'un utilisateur)
- qui indique sur quelle(s) machine(s) cette personne est connectée

Conseil : faire `rwho` et sélectionner les lignes qui la concernent.

Identifiants

Le nom (identifiant) de la variable peut contenir

- des lettres,
- des chiffres
- et des blancs soulignés

Attention

- Il ne peut pas commencer par un chiffre.
- Majuscules et minuscules sont différenciées

Affectations

Rôle : Met une valeur dans une variable.
Deux formes

- `nvar=chaine`, affectation simple
  - `reponse=ok`
  - `message='il fait froid'`
  - `PATH=/opt/games/bin:$PATH`
- `let var=expression`, affectation du résultat d'un calcul entier
  - `let quantite=12`
  - `let total=total+quantite`

Attention à la syntaxe : ne pas mettre d'espaces devant, ni à côté du signe "="

Essayez

```
a=12
b=42

c=a+b
echo $c

d=$a+$b
echo $d

let e=a+b
echo $e
```


Lecture

La commande `read v1 v2 ...`

- lit une ligne au terminal,
- affecte les mots dans les variables citées.

Exemple `read nom prenom`
Essayez : que se passe-t-il si on tape plus/moins de mots qu'il n'y a de variables ?

Exercice

Écrire un script qui

- demande l'année de naissance,
- calcule et affiche l'âge

```
$ ./quel-age
Votre année de naissance ?
1990
Vous êtes né en 1990, vous avez donc 19 ans.
$
```


Expansion

Expansion : développement d'une partie d'une expression (par exemple le nom d'une variable précédé par dollar) remplacée par autre chose.

1. expansion de variable :
   `echo bonjour $USER`
2. expansion numérique
   `echo le perimetre est $((2*l*h))`
3. expansion du résultat d'une commande :
   `echo il y a $(who | wc -l) connexions`

Remarque

- La dernière forme est notée historiquement avec des "anti-quotes".
- La forme `$(...)` évite les confusions Comparez
  - `echo il y a ‘who | wc -l‘ connexions`
  - `echo il y a ’who | wc -l’ connexions`
- Elle permet aussi l'emboîtement de commandes.

```
echo la réponse est $(grep $(traduction $mot) fichi
```


Exercice

Dans le script qui calcule l'âge, remplacez la constante 2009 par un appel à `date +%Y`

Exercice (à la maison)

Écrire un script qui écrit le nombre de processus qui vous appartiennent

- Résultat attendu :

  `Sur tuba, adupont a 45 processus`
- Méthode : comptez les lignes qui commencent par votre nom dans le résultat de `ps axu`).
- Améliorez en affichant le nom en clair "`Albert Dupont (etd1)`"
  Se baser sur le résultat de la commande
  - "`finger adupont`"
  - ou "`getent passwd adupont`"

Fonctions

- Objectif : découpage du code
  - pour simplifier (décomposition de problème)
  - éviter les répétitions de code
- Syntaxe :

```
function nom {
   instructions
}
```

- ou

```
nom ()  {
   instructions
}
```

- paramêtres positionnels $1, $2,...

Exemple

```
#
SAUVEGARDES=$HOME/Archives
mkdir $SAUVEGARDES 2>/dev/null

function ARCHIVER {
  T=$SAUVEGARDES/$2.tgz
  tar -czf $T $1
  echo le répertoire $1 est archivé dans $T
  ls -l $T
}

ARCHIVER ~/ASR1/TP1 scripts1
```


Boucles, Conditions, Tests

Tout ce qu'il faut pour programmer

la boucle "for"

- parcourt une suite de mots
- syntaxe

```
for variable in mot1 mot2 ...
do
  ... instructions ...
done
```


Application

La commande "astyle" reformate un fichier source. Reformater tous les fichiers du répertoire courant.

Example

```
for f in *.cc
do
   astyle $f
done
```

ou encore, en une seule ligne :

```
for f in *.cc ; do astyle $f ; done
```

Exercice

Écrire un script qui reçoit en paramètre une liste de noms, et affiche des messages de bienvenue numérotés
Résultat attendu :

```
$ ./bonjour John Paul Georges Ringo
-> 1 Hello John!
-> 2 Hello Paul!
-> 3 Hello Georges!
-> 4 Hello Ringo!
```


Aiguillage "case"

Etudie la ressemblance entre un "mot" et des modèles (*patterns* ou *templates*), exécute la suite d'actions correspondante

- Syntaxe :
  ```
  case MOT in [MODELE [| MODELE]...) COMMANDES ;;]...
  esac
  ```
- Exemple 1
  ```
  case $reponse in
  y* | o* )
      rm fichier
      echo fichier supprimé
      ;;
  *)
      echo fichier conservé
  ```


Aiguillage "case"

Exemple 2

```
for f in $* ; do
case $f in
*.cc | *.cxx )
    echo $f est écrit en C++
    ;;
*.tar)
    echo $f est une archive tar
    ;;
*)
    echo $f est inconnu
    ;;
esac
```


Exercice

Analysez le script suivant `pendule.sh`
Ce script est bâti sur le modèle des scripts de démarrage de services (voir répertoires `/etc/init.d` et `/etc/rc*.d`)

Exemple à analyser (1/3)

```
001  #! /bin/bash

002  PIDFILE=/tmp/pendule-$LOGNAME.pid
003  PROG=/usr/bin/xclock

004  function print_usage {
005    1>&2 echo usage: $0 '{start|stop|restart}'
006    exit 1
007  }

008  function do_start {
009      $PROG &
010      echo $! > $PIDFILE
```


Exemple à analyser (2/3)

```
012  function do_stop {
013      kill -9 $(cat $PIDFILE)
014      rm -f $PIDFILE
015  }

016  case $1 in
017  start)
018       do_start
019       ;;
020  stop)
021       do_stop
022       ;;
...
```


Exemple à analyser (3/2)

```
...
023  restart)
023       do_stop
025       sleep 1
026       do_start
027       ;;
018  *)
029       print_usage
030       ;;
031  esac
```


Remarques

- La forme "`commande &`" lance l'exécution d'une commande en arrière-plan,
- Le numéro du processus correspondant est mis dans la variable `$!` du shell.

if-then-else-fi

En fonction du résultat d'une commande (succès ou échec), décide de la suite à donner
Un programme réussit quand il se termine par l'appel `exit(0)`
Exemple

```
if grep -q $mot fichier
then
   echo le mot $mot est dans le fichier
else
   echo pas trouvé le mot $mot
fi
```

Comment écrire des shell-scripts
Variables, paramètres, expressions...
Fonctions
Boucles, Conditions, Tests...
La boucle for
L'aiguillage "case"
Exercice
If - then - else - fi
La commande "test"
Enchaînement conditionnel de commandes
Boucle while
While-read
La boucle for numérique
select

if-then-else-fi

- La commande peut être un assemblage de plusieurs commandes (éventuellement un pipeline).
- Le bloc "else" est facultatif,
- on peut aussi mettre des blocs "elif" (else-if) pour une cascade de tests.

Syntaxe générale

```
 if COMMANDS; then COMMANDS;
[ elif COMMANDS; then COMMANDS; ]...
[ else COMMANDS; ] fi
```

Comment écrire des shell-scripts
Variables, paramètres, expressions...
Fonctions
Boucles, Conditions, Tests...
La boucle for
L'aiguillage "case"
Exercice
If - then - else - fi
La commande "test"
Enchaînement conditionnel de commandes
Boucle while
While-read
La boucle for numérique
select

Exercice

Écrire un script qui regarde si une personne est connectée sur une machine donnée

- en utilisant `rwho`).
- La nom de la personne et la machine sont en paramètres.

Comment écrire des shell-scripts
Variables, paramètres, expressions...
Fonctions
Boucles, Conditions, Tests...
La boucle for
L'aiguillage "case"
Exercice
If - then - else - fi
La commande "test"
Enchaînement conditionnel de commandes
Boucle while
While-read
La boucle for numérique
select

La commande "test"

La commande `test` assure un grand nombre de types de tests.
Exemples

- `test -f FICHIER` : teste si `FICHIER` est un fichier
- `test -d FICHIER` : teste si `FICHIER` est un répertoire
- `test CHAINE1 = CHAINE2` : comparaison de chaînes
- `test CHAINE1 != CHAINE2` : idem
- `test CHAINE1 < CHAINE2` : idem (ordre lexicographique)
- `test EXPR1 op EXPR2` : comparaison numérique, où op est `-eq` (equal) `-ne` (not equal) `-lt` (lesser than) `-le` (lesser or equal) `-gt` (greater than) `-ge` (greater or equal).

Comment écrire des shell-scripts
Variables, paramètres, expressions...
Fonctions
Boucles, Conditions, Tests...
La boucle for
L'aiguillage "case"
Exercice
If - then - else - fi
La commande "test"
Enchaînement conditionnel de commandes
Boucle while
While-read
La boucle for numérique
select

Exercices

1. écrire un script qui affiche le plus grand de ses deux paramètres (qui sont numériques).
2. écrire un script `fabriquer` à qui on donne le nom d'un exécutable
   exemple : `fabriquer mon-tp`
   et lance la compilation de `mon-tp.cc` si l'exécutable n'existe pas ou est plus ancien que le source.

Comment écrire des shell-scripts
Variables, paramètres, expressions...
Fonctions
Boucles, Conditions, Tests...
La boucle for
L'aiguillage "case"
Exercice
If - then - else - fi
La commande "test"
Enchaînement conditionnel de commandes
Boucle while
While-read
La boucle for numérique
select

Test (suite)

- Un point d'exclamation en premier paramètre inverse le test

```
if test ! -f $nomfichier
then
   echo le fichier $nomfichier est absent.
   exit 1
fi
```

- La commande `[` est un synonyme de `test`. On la termine alors par un crochet fermant. Exemples

```
if [  -d $r  ]
then
   echo le répertoire $r existe
fi
```

Comment écrire des shell-scripts
Variables, paramètres, expressions...
Fonctions
Boucles, Conditions, Tests...
La boucle for
L'aiguillage "case"
Exercice
If - then - else - fi
La commande "test"
Enchaînement conditionnel de commandes
Boucle while
While-read
La boucle for numérique
select

Application (1/2)

Remarques sur le script `pendule.sh` vu plus haut

1. on peut lancer plusieurs instances de `xclock`,
2. `pendule.sh stop` n'arrête que la dernière.

Idée : tester l'existence du PIDFILE pour ne lancer qu'une instance

Comment écrire des shell-scripts
Variables, paramètres, expressions...
Fonctions
Boucles, Conditions, Tests...
La boucle for
L'aiguillage "case"
Exercice
If - then - else - fi
La commande "test"
Enchaînement conditionnel de commandes
Boucle while
While-read
La boucle for numérique
select

Application (1/2)

```
function do_start {
    if [ -f $PIDFILE ]
    then
        2>&1 echo Pendule déjà lancée
        exit 1
    fi
    $PROG &
    echo $! > $PIDFILE
}
```

Application (2/2)

```
function do_stop {
    if [ ! -f $PIDFILE ]
    then
        2>&1 echo Pendule déjà arrêtée
        exit 1
    fi
    kill -9 $(cat $PIDFILE)
    rm -f $PIDFILE
}
```

Ennchainement conditionnel

- `commande1 && commande1` exécute la première commande, puis la seconde si la première a réussi. Exemple

```
if [ -f $nomfichier ] && grep -q critere $nomfichier
then
    ....
fi
```

- `commande1 || commande1` exécute la première commande, puis la seconde si la première a échoué. Exemple

```
[ -d /tmp/rep ] || mkdir /tmp/rep
```

Exercice

La variable $? contient le code de retour de la dernière commande.
Simplifiez la séquence

```
001   grep $critere1 $nomfichier
002   code=$!
003   if [ ! $code1 = 0 ]
004   then
005       grep $critere2 $nomfichier
006       code=$!
007       if [ ! $code2 != 0 ]
008       then
009           action1
010       else
011           action2
```

while-do-done

- Syntaxe : `while COMMANDES1 ; do COMMANDES2 ; done`
- Exercice : écrire un script qui affiche la factorielle d'un entier, calculée avec une boucle.

while-read

La tournure "while read" traite un texte ligne par ligne

```
while read ligne
do
      commande
done  < fichier
```

Exercice while-read

Partant d'un fichier de prénoms et d'âges, du genre

```
julie 12
marcel 87
emilie 9
```

faire afficher le prénom et l'âge du plus vieux.

read, séparateur IFS

On peut lire un fichier dont les champs sont délimités par " :" (par exemple) en précisant le séparateur dans la variable IFS qui est utilisée par read. Exemple

```
getent passwd | while IFS=: read user pass uid gid \
                      full home shell
              do
                 echo "compte $user, nom en clair  $full"
              done
```

until-do-done

- `until COMMANDES1 ; do COMMANDES2 ; done`
- une boucle `while` avec test inversé.

boucle for numérique

Ressemble davantage à la boucle de C, mais est d'un usage plus rare dans les shell scripts, où on fait rarement des calculs.
Exemple :

```
for (( i = 1; i <= 5; i++ ))
do
   echo $i
done
```


Exercice

Écrire un script qui affiche la table de multiplication par *n* (passé en paramètre)

```
multiplication par 3
   1   2   3   4   5   6   7   8   9  10
  x3  x3  x3  x3  x3  x3  x3  x3  x3  x3
 --- --- --- --- --- --- --- --- --- ---
   3   6   9  12  15  18  21  24  27  30
```

Conseil : au lieu d'`echo`, utilisez `printf` pour écrire un nombre sur une largeur donnée, exemple `printf "%4d" $n`

select

Propose une boucle avec menu interactif. On en sort par `break`.
Exemple :

```
select action in manger dormir partir
do
  case $action in
    1) echo miam ;;
    2) echo zzzz;;
    3) echo au revoir ; break ;;
  esac
done
```

La variable `$REPLY` contient le mot correspondant à la sélection (par exemple `dormir`)