Nantes le 28 mars 2003

TP BASH

Abdeslam MOKRANI
Jérémy VALAYER
LIN Groupe 2

Dans ce rapport, nous allons expliquer le travail effectué pour réaliser une "boite à outils" qui permet de produire un document de format *XML* représentant l'état du système à un instant donné. Des fonctions permettant de réaliser, étape par étape, les différentes fonctionnalités demandées sont réalisée.

1. Mise en format XML

Fonctionnement

Nous avons réalisé une fonction *balise()* qui permet de mettre en format *XML* un texte donné en paramètre ou lu sur l'entrée standard. Le premier paramètre de cette fonction doit être le nom de la balise, le second doit être une chaîne contenant tous les attributs et tous les autres paramètres (s'ils existent) doivent représenter les lignes de la balise.

Si aucune ligne n'est passée en paramètre (nombre de paramètres inférieur à 3), alors les lignes de la balise sont lues sur l'entrée standard. Ceci permet de rediriger des fichiers ou les sorties standards de commandes ou de cette même fonction vers l'entrée standard de celle-ci, ce qui permet de réaliser des imbrications de balises. Des données volumineuses peuvent donc être transmises, par redirection, à cette fonction comme contenu de la balise à afficher.

Chacun des paramètres de cette fonction peut être nul (""). Ainsi, pour indiquer qu'il n'y a aucun attribut à afficher, il suffit d'évaluer le deuxième paramètre à nul. Dans le cas où une seule ligne est passée en paramètre (nb. de para. égale à 3), si celle-ci est vide, ou si aucune ligne n'est lue sur l'entrée standard, alors on utilise la forme spéciale *<nom attributs/>*.

Réalisation

On commence par afficher le nom de la balise et la chaîne de caractère "attributs" précédé d'un "<" en utilisant la commande d'affichage *echo* du *bash*. On test l'existence de lignes à l'aide de l'instruction *case*. Si aucune ligne, alors on essais de lire sur l'entrée standard (*read*). Si au moins une ligne est lue, alors on ferme l'ouverture de la balise par un ">", on affiche la ligne (la valeur lue étant mise dans la variable REPLY) et on affiche toutes les autres lignes éventuelles à l'aide d'une boucle *while*, on termine par fermer la balise par "</nom>". Sinon (aucune ligne lue), on ferme la balise par un "/>" pour respecter la forme spéciale *<nom attributs/>*.

Si les lignes sont passées en paramètres, alors on les affiche une par une à l'aide d'une boucle *while* en décrémentant le nombre de paramètres (*shift*) après avoir sauvegardé le nom de la balise dans une variable pour pouvoir fermer la balise. On peut indiquer que la balise est vide en passant une seule ligne vide, on utilise alors la notation *<nom attributs/>*. Voir annexe B page 10.

2. Arborescences de fichiers

Affichage d'un fichier

Une fonction *file()* permet d'afficher les propriétés d'un fichier ou d'un répertoire dont le nom (éventuellement avec son chemin d'accès) est passé en paramètre.

Réalisation

Le nom du fichier est extrait en supprimant le chemin d'accès à l'aide de la commande *basename*. La commande *ls* permet de récupérer la plupart des propriétés du fichier en utilisant l'option *-l* (*ls -l fichier)*. Pour un répertoire, on doit récupérer ses propriétés en cherchant la bonne ligne (commande *grep*) dans la liste des lignes de fichiers du répertoire parent en redirigeant la sortie standard de la commande *ls* (ls -l répertoire/.. | grep nonrépertoire).

La ligne de propriétés étant récupérée, on peut affecter ces propriétés respectivement aux variables 1 2 .. à l'aide de la commande *set*. On peut donc faire passer chaque propriété en paramètre de la fonction *balise()* pour générer un code *XML* représentant le fichier (voir annexe B page 12).

Affichage de fichiers d'un répertoire

On peut appliquer la fonction file() à chacun des fichiers d'un répertoire en utilisant l'instruction *for*. Les fichiers caché sont spécifiés par la chaîne de substitution de non de fichier ".*". Si aucun fichiers n'existe, alors la chaîne de substitution de non de fichiers sera considérée comme un nom de fichiers, mais elle ne sera pas affichée par la fonction *file()* car celle-ci vérifie d'abord que le fichier existe. La fonction *files()* permet de réaliser celà (voir annexe B page 12).

Affichage d'un répertoire

La fonction *dir()* permet d'imbriquer le résultat d'affichage de la fonction *files()* pour un répertoire donné dans une balise spécifiant le chemin d'accès de ce répertoire.

Affichage d'une arborescence de fichiers

La fonction récursive *subdir()* affiche tous les fichiers d'un répertoire donné, en utilisant la fonction *file()*, et tous les fichiers de ses sous répertoire en faisant un appel récursif.

L'affichage de cette fonction est imbriqué dans une balise portant le chemin d'accès du répertoire parent à l'aide de la fonction *tree()*.

Un exemple d'affichage d'une balise arborescence de fichiers et le code minimal qui permet de le faire est donné en annexe A page 7.

3. Volumes montés

Chaque affichée par la commande *mount* correspond à un volume monté. On lit ces lignes en redirigeant la sortie de cette commande sur une boucle *while read* . On affecte les différents champs de chaque ligne aux variables 1 2 .. à l'aide de la commande *set*, ce qui nous permet de récupérer la source physique du fichier dans la variable 1 et la cible ou il est monté dans variable 2 par exemple. Il ne reste plus qu'à utiliser ces variable comme paramètre de la fonction *balise()* pour réaliser l'affichage désiré. Le tout est imbriqué dans une balise nommé "mount", voir la fonction *lmount()* en annexe B page 12.

4. Les informations machine

Caractéristiques du CPU

Une machine peut posséder plusieurs processeurs (exemple : la machine *quad a 7* processeurs). Les propriété de chaque processeur sont indiquées dans le fichier */proc/cpuinfo*. La fonction *cpu()* permet d'afficher les propriété sous forme de balises toutes imbriqué dans une balise correspondant à un processeur dont le numéro est donné. Cette fonction utilise deux boucles *while* et la commande *cut* pour découper les propriétés désirées. Toutes les balises "processeurs" sont imbriquées dans une seule balise nommé *cpu*. Voir annexe A page 13.

Caractéristique de la mémoire

Le fichier */proc/meminfo* contient, à partir de la ligne 4, toutes les informations mémoires sous forme de couples type_mémoire-taille. Une commande *sed* appliquer à ce fichier pour transformer chaque ligne à une aux paramètres d'une balise de la forme : *type_mémoire valeur="taille"* permet donc de générer les paramètres qu'il faut passer à la commande balise pour avoir le code *XML* désiré. Il suffit donc de lire ligne par ligne la sortie standard de cette commande et d'exécuter : balise $ligne "". Voir la fonction *mem()* en annexe B page 13.

Caractéristique du système d'exploitation

On supprime les parenthèses de la ligne du fichier */proc/version* qui définit la version du système d'exploitation à l'aide de la commande *sed* ou la commande *tr* par exemple, on affecte les différents champs de la ligne obtenue aux variables 1, 2, .. à l'aide de la commande set. Le contenu de ces variables nous permettent de réaliser une balise *os* qui met en évidence les caractéristiques voulues. Voir annexe B page 13.

On peut maintenant réaliser une fonction *system* qui permet de d'imbriquer toutes les informations de celui-ci en faisant appel aux à toutes les fonctions vues précédemment dans une balise *system* qui indique en plus le nom de l'utilisateur et la date courante.

Remarques

- Le fichier /proc/module n'est pas accessible aux utilisateurs dans les salles d'informatique, nous n'avons donc par réalisé la fonctionnalité demandée se reportant à ce fichier.

- Tout le code a été testé sur la machine *quad* et les machines locales des dalles d'informatique et il marche correctement.

- Il suffit d'être utilisateur pour pouvoir exécuter la totalité du code.

Ce travail nous a montré la facilité de gestion de fichiers et de textes en utilisant les fonctionnalités du BASH et la commande sed. On remarque aussi que Linux nous permet d'accéder à toutes les informations du système soit par des commandes soit par des fichiers.

ANNEXE A

- Exemple d'un code

```
t='   ' # tabulation pour le décalage des lignes d'une balise

#-------------------------------------------------------------------------------+
# affichage d'une balise                                                       |

# balise [nom [attributs [ligne1] [ligne2] .. ]]

# nom                   : nom de la balise
# attributs             : chaîne contenant tous les attributs
# ligne1, ligne2, ..    : contenu de la balise

# si aucune ligne, alors lecture du contenu de la balise à partir de
# l'entrée standard

balise()
{
    echo -n "<$1"
    if [ "$2" != "" ]; then
        echo -n " $2" # affichage des attributs précédés d'un espace
    fi
    case $# in
    0|1|2)
        # aucune ligne, lecture à partir de l'entrée standard
        if read; then
            echo ">"
            echo "$t$REPLY" # on affiche la ligne précédée d'une
                            # tabulation
            while read; do
                echo "$t$REPLY"
            done
            echo "</$1>"
        else
            # si rien n'est lu, alors balise vide de type :
            # <nom attributs/>
            echo "/>"
        fi
    ;;
    *)
        if [ $# -gt 3 -o "$3" != "" ]; then
            echo ">"
            name="$1"
            while [ $# -gt 2 ]; do
                echo "$t$3" # on affiche la ligne précédée
                            # d'une tabulation
                shift
            done
            echo "</$name>"
        else
            # si contenu vide, alors balise vide de type :
            # <nom attributs/>
            echo "/>"
        fi
    ;;
    esac
}
```

```
file()
{
    name=`basename "$1"`
    if [ -f "$1" ]; then
        type=file
        set a `ls -l "$1"` # le "a", c'est  pour éviter d'interpréter le -wrx ..
                           # comme option de la commande set
    elif [ -d "$1" ]; then
        type=directory
        set a `ls -la "$1"/.. | grep "$name"$` # -a; pour répertoires cachés
    else
        exit 1
    fi
    (
        balise creation "date=\"$8/$7 $9\"" ""
        balise protection "rights=\"$2\"" ""
    )|
        balise file "name=\"$name\" type=$type size=$6 owner=$4 group=$5"
}


subdirs()
{
    name=`basename "$1"`
    ( for fic in "$1"/* "$1"/.[!.]*; do
        if [ -f "$fic" ]; then
            file "$fic"
        elif [ -d "$fic" ]; then
            subdirs "$fic" # appel récurssif
        fi
     done
    )|
    balise subdir name="\"$name\""
}


tree()
{
    subdirs "$1" | balise tree path="\"$1\""
}


# création d'une arborescence pour le test
mkdir toto; cd toto
mkdir reptoto juju bla juju/lulu bla/rep juju/lulu/one juju/lulu/two \
      "bla/[ le rep ]" # caractères spéciaux
touch fichier vide file bla/fileone bla/filetwo juju/lulu/unfichier \
      "un fichier" 'bla/[ le rep ]/le fichier$ $*-#' # caractères spéciaux
touch .cache .invisible bla/rep/.notvisible juju/lulu/two/.hide # fichiers cachés

# affichage de l'état du système
tree $PWD

# suppression de l'arborescence
cd ..; rm -rf toto
```

- Affichage à l'exécution

```
<tree path="/home/CIE/mokrani/BASH/xml/code/toto">
      <subdir name="toto">
         <subdir name="bla">
            <file name="fileone" type=file size=0 owner=mokrani group=linfo>
               <creation date="28/mar 17:57"/>
               <protection rights="-rw-r--r--"/>
            </file>
            <file name="filetwo" type=file size=0 owner=mokrani group=linfo>
               <creation date="28/mar 17:57"/>
               <protection rights="-rw-r--r--"/>
            </file>
            <subdir name="[ le rep ]">
               <file name="le fichier$ $*-#" type=file size=0 owner=mokrani group=linfo>
                  <creation date="28/mar 17:57"/>
                  <protection rights="-rw-r--r--"/>
               </file>
            </subdir>
            <subdir name="rep">
               <file name=".notvisible" type=file size=0 owner=mokrani group=linfo>
                  <creation date="28/mar 17:57"/>
                  <protection rights="-rw-r--r--"/>
               </file>
            </subdir>
         </subdir>
         <file name="fichier" type=file size=0 owner=mokrani group=linfo>
            <creation date="28/mar 17:57"/>
            <protection rights="-rw-r--r--"/>
         </file>
         <file name="file" type=file size=0 owner=mokrani group=linfo>
            <creation date="28/mar 17:57"/>
            <protection rights="-rw-r--r--"/>
         </file>
         <subdir name="juju">
            <subdir name="lulu">
               <subdir name="one"/>
               <subdir name="two">
                  <file name=".hide" type=file size=0 owner=mokrani group=linfo>
                     <creation date="28/mar 17:57"/>
                     <protection rights="-rw-r--r--"/>
                  </file>
               </subdir>
               <file name="unfichier" type=file size=0 owner=mokrani group=linfo>
                  <creation date="28/mar 17:57"/>
                  <protection rights="-rw-r--r--"/>
               </file>
            </subdir>
         </subdir>
         <subdir name="reptoto"/>
         <file name="un fichier" type=file size=0 owner=mokrani group=linfo>
            <creation date="28/mar 17:57"/>
            <protection rights="-rw-r--r--"/>
         </file>
         <file name="vide" type=file size=0 owner=mokrani group=linfo>
            <creation date="28/mar 17:57"/>
            <protection rights="-rw-r--r--"/>
         </file>
         <file name=".cache" type=file size=0 owner=mokrani group=linfo>
            <creation date="28/mar 17:57"/>
            <protection rights="-rw-r--r--"/>
         </file>
         <file name=".invisible" type=file size=0 owner=mokrani group=linfo>
            <creation date="28/mar 17:57"/>
            <protection rights="-rw-r--r--"/>
         </file>
      </subdir>
   </tree>
```

ANNEXE B

```
#!/bin/sh

t='   ' # tabulation pour le décalage des lignes d'une balise

#-------------------------------------------------------------------------------+
# affichage d'une balise                                                        |

# balise [nom [attributs [ligne1] [ligne2] .. ]]

# nom                   : nom de la balise
# attributs             : chaîne contenant tous les attributs
# ligne1, ligne2, ..    : contenu de la balise

# si aucune ligne, alors lecture du contenu de la balise à partir de
# l'entrée standard

balise()
{
    echo -n "<$1"
    if [ "$2" != "" ]; then
        echo -n " $2" # affichage des attributs précédés d'un espace
    fi
    case $# in
    0|1|2)
        # aucune ligne, lecture à partir de l'entrée standard
        if read; then
            echo ">"
            echo "$t$REPLY" # on affiche la ligne précédée d'une
                            # tabulation
            while read; do
                echo "$t$REPLY"
            done
            echo "</$1>"
        else
            # si rien n'est lu, alors balise vide de type :
            # <nom attributs/>
            echo "/>"
        fi
    ;;
    *)
        if [ $# -gt 3 -o "$3" != "" ]; then
            echo ">"
            name="$1"
            while [ $# -gt 2 ]; do
                echo "$t$3" # on affiche la ligne précédée
                            # d'une tabulation
                shift
            done
            echo "</$name>"
        else
            # si contenu vide, alors balise vide de type :
            # <nom attributs/>
            echo "/>"
        fi
    ;;
    esac
}
```

```
#-------------------------------------------------------------------------------+
# balise des propriétés d'un fichier                                           |

# file fichier

file()
{
    name=`basename "$1"`
    if [ -f "$1" ]; then
        type=file
        set a `ls -l "$1"` # le "a", c'est  pour éviter d'interpréter le -wrx ..
                           # comme option de la commande set
    elif [ -d "$1" ]; then
        type=directory
        set a `ls -la "$1"/.. | grep "$name"$` # -a; pour répertoires cachés
    else
        exit 1
    fi
    (
        balise creation "date=\"$8/$7 $9\"" ""
        balise protection "rights=\"$2\"" ""
    )|
        balise file "name=\"$name\" type=$type size=$6 owner=$4 group=$5"
}


#-------------------------------------------------------------------------------+
# affichage des balise de tous les fichier d'un répertoire (fichiers visibles  |
# puis fichiers cachés)                                                        |

#file répertoire

files()
{
    for fic in "$1"/* "$1"/.[!.]*; do file "$fic"; done
    # si pas de fichiers (visibles ou cachés), fic prend pour valeur "$1"/\*
    # ou "$1"/.[!.]\* (l'étoile n'est pas substitué), mais fic ne sera pas
    # affiché comme fichier grace à la condition au début de la fon. file()
}


#-------------------------------------------------------------------------------+
# balise de contenu d'un répertoire                                            |

# dir répertoire

dir()
{
    files "$1" | balise dir path="\"$1\""
}
```

```
#-------------------------------------------------------------------------------+
# balise d'un répertoire imbriquant tous ses sous répertoires                   |

# subdir répertoire

subdirs()
{
    name=`basename "$1"`
    ( for fic in "$1"/* "$1"/.[!.]*; do
        if [ -f "$fic" ]; then
            file "$fic"
        elif [ -d "$fic" ]; then
            subdirs "$fic" # appel récurssif
        fi
     done
    )|
    balise subdir name="\"$name\""
}


#-------------------------------------------------------------------------------+
# balise d'une arborescence de fichiers                                         |

# tree répertoire
tree()
{
    subdirs "$1" | balise tree path="\"$1\""
}
```

```
#-------------------------------------------------------------------------------+
# balise des volumes montés                                                     |

# lmount

lmount()
{

    ( mount | while read; do
        set $REPLY
        ( balise source "value=\"$1\"" ""
          balise target "value=\"$3\"" ""
          balise type  "$5 value=$6" ""
        )|
        balise file_system ""
      done
    )|
    balise mount
}
```

```
#-------------------------------------------------------------------------------+
# balise des caractéristiques du CPU                                           |

# cpu

cpu()
{
    while read; do # pour lire les infos de tous les processeurs de la machine

    no=`echo $REPLY | cut -d" " -f3` # pour récupérer le numéro du processeur
    declare -i i=1 # compteur (chaque processeur a 17 lignes de propriétés)

    while [ $i -le 17 ] && read; do
        name=`echo $REPLY | cut -d: -f1 | tr -d " "` # nom d'une propriété si
                                                     # plusieurs mots, on les concatène
        value=`echo $REPLY | cut -d: -f2`  # valeur de la propriété
        balise $name "value=\"${value:1}\"" "" # de format <nom valeur/>
        i=i+1
    done |
    balise processor "no=$no"

    if read; then true; fi  # lire la ligne vide si elle existe
    done < /proc/cpuinfo  | # redirection du fichier
    balise cpu ""
}
```

```
#-------------------------------------------------------------------------------+
# balise des caractéristiques de la mémoire                                    |

# mem

mem()
{
    # on récupère les deux premiers champs de toutes les lignes
    # à partir de la ligne 4 du fichier /proc/meminfo
    sed -n '4,$s/\(.*\): *\(.*\) .*/\1 value=\2/gp' /proc/meminfo |
    while read; do
    balise $REPLY ""
    done | balise mem ""
}
```

```
#-------------------------------------------------------------------------------+
# balise des caractéristiques du système d'exploitation                        |

# os

os()
{
  set `sed 's/[()]//g' /proc/version`
  ( balise admin adress="$4" ""
    balise gcc "$6=\"$7 $8\" distr=\"$9 $10\"" ""
   ) |
  balise os "name=\"$1\" $2=\"$3\" date=\"${12} ${14} ${13} ${17} à ${15} ${16}\""

}
```

```
#-------------------------------------------------------------------------------+
# balise des caractéristiques du système                                        |

# system

system()
{
    ( echo -e "\n.....................::Etat du système::......................."
      echo -e "\n\n"
      balise REPERTOIR_COURANT "" ""
      echo
      dir $PWD
      echo -e "\n\n"
      balise REPERTOIR_COURANT_ARBORESCENCE "" ""
      echo
      tree $PWD
      echo -e "\n\n"
      balise VOLUMES_MONTES "" ""
      echo
      lmount
      echo -e "\n\n"
      balise CPU "" ""
      echo
      cpu
      echo -e "\n\n"
      balise MEMOIRE "" ""
      echo
      mem
      echo -e "\n\n"
      balise SYSTEME_D_EXPLOITATION "" ""
      echo
      os
    )|
    balise system "user=$USERNAME date=\"`date`\""
}
```

```
# création d'une arborescence pour le test
mkdir toto; cd toto
mkdir reptoto juju bla juju/lulu bla/rep juju/lulu/one juju/lulu/two \
      "bla/[ le rep ]" # caractères spéciaux
touch fichier vide file bla/fileone bla/filetwo juju/lulu/unfichier \
      "un fichier" 'bla/[ le rep ]/le fichier$ $*-#' # caractères spéciaux
touch .cache .invisible bla/rep/.notvisible juju/lulu/two/.hide # fichiers cachés

# affichage de l'état du système
system

# suppression de l'arborescence
cd ..; rm -rf toto
```