ESIEE Engineering
IN3R21 Systèmes d'exploitation

Correction du TP n°4
Expressions régulières, redirections

Exercice 1 – Un serveur HTTP rudimentaire

1. 2. : non corrigées

3. `coproc x y` lance en parallèle les programmes x et y (avec leurs arguments éventuels) après avoir relié la sortie standard de x vers l'entrée standard de y, et la sortie standard de y vers l'entrée standard de x. Ceci équivaut à un pipe double entre x et y.

Il est impossible d'implémenter cette commande en shell standard : la seule chose que le shell propose de mettre en œuvre est le pipe simple, et tout ce qui suit un appel à `x | y` ne permet déjà plus de faire référence à x seul.

4. Une première version (paresseuse) consiste à lire seulement la première ligne et à extraire l'URL avec expr :

```
#!/bin/sh
#

read line
url=`expr "$line" : "GET \(.*\) HT"`
echo "L'URL demandée est : $url" 1>&2
echo "J'ai compris que vous vouliez l'URL $url"
exit 0
```

Une version plus rigoureuse consisterait à lire l'en-tête complet (fin de l'entête signalée par une ligne vide), et à filtrer le résultat avec grep pour ne retenir que la ligne GET :

```
#!/bin/sh
#

while read line; test "$line"; do
      echo "$line"
done | grep "^GET" | { read line

url=`expr "$line" : "GET \(.*\) HT"`
echo "L'URL demandée est : $url" 1>&2
echo "J'ai compris que vous vouliez l'URL $url"
exit 0
}
```

5. On a en fait intérêt à éliminer le premier '?' et tout ce qui suit le plus tôt possible, ce qui peut se faire simplement avec sed : on capture tout ce qui précède le '?' et on le remplace dans l'expression

substituée. Ce qui suit le '?' est lu jusqu'à la fin mais n'est pas remplacé :
```
echo "$url" | sed "s/^\([^?]*\)?.*$/\1/"
```

Pour remplacer les séquences d'échappement '%hh', on peut utiliser la commande printf : en effet, '\xhh' est remplacé par printf par le caractère ASCII dont hh est le codage hexadécimal. Exemple :

```
printf 'Mon\x20fichier'
Mon fichier
```

Il suffit donc de lancer sed et de remplacer '%hh' par '\xhh' où hh est inchangé, ce qui s'écrit ainsi :
```
sed 's/%\([0-9A-Fa-f]\{2,2\}\)/\\\\x\1/g'
```

Rappel de cours : le modifieur '/g' force le remplacement de toutes les occurrences du texte – sans lui, sed s'arrêterait à la première. Il suffit alors de faire afficher le chemin complet par printf, puis de relire le résultat dans une variable – l'utilisation de la backquote n'est pas forcément une bonne idée : s'il y a plus d'un espace dans le nom de fichier, on n'en retrouvera qu'un seul au final. Script complet :

```
#!/bin/sh
#

HTTPHOME=/opt/local/share/doc

while read line; test "$line"; do
      echo "$line"
      echo "j'ai lu $line" 1>&2
done | grep "^GET" | { read line

url=`expr "$line" : "GET \(.*\) HT"`
echo "L'URL demandee est : $url" 1>&2
echo "J'ai compris que vous vouliez l'URL $url"

# Eliminer le ? et la suite (1er pipe) puis
# substituer les sequences d'echappement (2eme pipe)
fichier=`echo "$url" | sed "s/^\([^?]*\)?.*$/\1/" | sed 's/%\([0-9A-Fa-f]\{2,2\}\
)/\\\\x\1/g'`

# Il suffit maintenant d'inserer le prefixe HTTPHOME devant le nom
# de fichier, de faire convertir les sequences par printf, et de
# relire le resultat
printf "$HTTPHOME$fichier" | {
      read fichier

      echo "Pour moi, cela correspond au fichier $fichier" 1>&2
}
}
```

6. Toutes ces conditions peuvent être réunies dans une seule ligne :
```
cat "$fichier" || cat "$fichier/index.html" || cat "$fichier/index.htm" || ls -l
"$fichier" || echo "404 NOT FOUND"
```

qui essaie toutes ces possibilités, dans l'ordre. Noter que cat échouera si fichier est un répertoire, raison pour laquelle cette possibilité est examinée en premier lieu. Seule le dernier bloc du script change donc :

```
printf "$HTTPHOME$fichier" | {
      read fichier

      echo "Pour moi, cela correspond au fichier $fichier" 1>&2
      echo "<html>"
      cat "$fichier" || cat "$fichier/index.html" || cat "$fichier/index.htm" || ls
-l "$fichier" || echo "404 NOT FOUND"
      echo "</html>"

      if [ - "$fichier" ]; then
            cat "$fichier"
      elif [ -d "$fichier" ];
}
```

7. Remarque : /bin/true est une commande externe qui ne fait rien d'autre que terminer avec le code de sortie 0.

```
# !/bin/sh
#

while true ; do
      coproc ./requete.sh "nc -l -p 8080"
done
```

Exercice 2 – Filtre anti-spam

1)

```
# Fonction sujet
#
# On doit d'abord isoler la bonne ligne avec grep et head. Détecter les '!'
# peut se faire avec expr et des chaînes arbitraires entre eux.
# Pour les majuscules, le test équivaut tout simplement a regarder s'il
# existe une seule lettre minuscule
#
function sujet () {
    local ligne=`grep -E "^Subject:" "$1" | head -n1 | sed -r "s/^Subject: //"`
    local r=0

    if expr "$ligne" : '^.*!.*!.*$' >/dev/null; then
        r=1
    elif ! expr "$ligne" : ".*[a-z].*"  >/dev/null; then
        r=1
    fi

    return $r
}
```

2)

```
# Fonction from
#
# Comme pour sujet, on s'appuie sur les bracket <> et l'arobase pour extaire
# le nom de domaine. Comme le code d'erreur de nslookup est 0 qu'il trouve
# ou non un nom de domaine, il faut chercher s'il écrit "can't find" ou
# non pour conclure.
#
function from () {
    local domain=`grep ^From: "$1" | head -n1 | sed -r "s/^From:[^@]*@([^>]*)
>$/\1/"`

    local invalide=`nslookup "$domain" | grep "can't find"`

    test "$invalide"
    return $((2*(1-$?)))
}
```

3)

```
# Fonction mois
#
# Elle ne fait que remplacer le mois, en notation RFC, par son numéro dans
# une date. On utilise pour cela une liste de mois, gérée manuellement.
#
function mois () {
    local
liste="Jan/01:Feb/02:Mar/03:Apr/04:May/05:Jun/06:Jul/07:Aug/08:Sep/09:Oct/10:Nov/11
:Dec/12:"
    local arg="$1"

    while [ "$liste" ]; do
        tete=${liste%%:*}
        mois=${tete%/*}
        num=${tete#*/}
        arg=`echo "$arg" | sed "s/$mois/$num/"`
        liste=${liste#*:}
    done

    echo $arg
}
```

4)

```
# Fonction pbdate
#
# L'idée ici est d'inverser l'année et le jour dans la date, puis d'appeler
# la fonction précédent pour convertir le mois en numérique. Exemple:
# 03 Apr 2008 -> 2008Apr03 -> 20080403
#
# En appliquant à la date de l'emil et à la date courante, on a un test
# suffisant pour savoir s'il y antidatage à au moins un jour.
#
function pbdate () {
    local datemail=`grep ^Date: "$1" | sed -r  "s/^[^0-9]*([^ ]*) ([^ ]*) ([^ ]*
).*$/\3\2\1/"`
```

```
    local date=`date -R | sed -r  "s/^[^0-9]*([^ ]*) ([^ ]*) ([^ ]*).*$/\3\2\1/"`

    local dm=`mois $datemail`
    local dr=`mois $date`

    echo "dm=$dm, dr=$dr"
    [ $dm -gt $dr ] && return 3
    return 0
}
```