Université de Caen Normandie
Département de mathématiques et d'informatique
L2 Informatique, 2017–2018
Unité : INF3B1

Système

L2 Système : Environnement de travail informatique

François Rioult

Table des matières

1 Corrections TP2

```
# 1.1
# générer 10 images .gif
for i in ‘seq 10‘; do touch $i.gif; done

# les renommer en .old et déplacer dans Images
for i in *.gif; do mv $i Images/‘basename $i .gif‘.old; done
# ou
for i in *.gif; do mv $i Images/$(basename $i .gif).old; done

# 1.2
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "error usage: $0 <input>" 1>&2
    exit
fi

input=$1

cat $input
exit 0

# 1.3
i=1; while [ $i -lt 10 ]; do echo $i; i=$(($i + 1)); done;
for i in ‘seq 100‘; do echo $i; done

# 1.4
#!/bin/bash

if [ $# -ne 1 ]; then
    echo "error usage: $0 <base num>" 1>&2
    exit
fi

input=$1

for i in ‘seq 0 10‘; do echo $(($i \* $input)); done
exit 0

# 1.4 solution sans boucle
seq 10 | xargs -l -I {} expr {} \* 7
```

```
# 1.5
#!/bin/bash

if [ $# -lt 2 ]; then
    echo "error usage: $0 <#repeat> <command ...>" 1>&2
    exit
fi

repeat=$1; shift
input=$*

for i in `seq $repeat`; do
    $input
done

exit 0

# 1.5 solution sans boucle
input="echo bonjour"; repeat=3; yes $input | head -$repeat | xargs -l -I {} $input
```

2 Corrections TP3

2.1 Question 3 : Script awk

On appelle le script suivant en précisant l'adresse (ici google) :

```
$ awk -f ip-to-country.awk ip-to-country 173.194.45.56
USA
```

```
BEGIN{
    if(ARGC != 3){
        print "erreur : usage " ARGV[0] " ip-to-country-table ip";
        exit;
    }
    ip = ARGV[ARGC - 1];
    print "ip " ip
    ipint = transforme(ip);
    ARGC --;
    FS = ",";
}

{
    if(transforme($1) <= ipint && ipint <= transforme($2)){
        print $3;
        exit;
    }
}

function transforme(ip){
    if (split(ip, tab, ".") != 4){
        print "erreur : format de l'adresse IP incorrect";
        exit;
    }
    return 256 * (256 * (256 * tab[1] + tab[2]) + tab[3]) + tab[4];
}
```

2.2 Question 4 :

Le script `dnstoip.sh` :

```
#!/bin/bash

if ‘test $# -ne 1‘ ; then
    echo "erreur : usage $0 domain" >&2
    exit 1
fi

host=‘host -t A $1 2>/dev/null | sed -n ’s/.*[^0-9]\([0-9][0-9]*\.\
[0-9][0-9]*\.[0-9][0-9]*\.[0-9][0-9]*\).*/\1/p’ | head -n 1 ‘

if test ${#host} -ne 0 ; then
    set $host
    awk -f ip-to-country.awk ip-to-country $host
fi

```

2.3 Question 5 :

Le script `histostat.sh` :

```
#!/bin/bash

if test $# -ne 1 ; then
    echo "erreur : usage $0 history"
    exit 1;
fi

sed -n ’s/.*   =http:\/\/\([^\/]*\).*/\1/p’ "$1" | sort -u
```

Le script `his2stat.sh` :

```
#!/bin/bash

if test $# -ne 1 ; then
    echo "erreur : usage $0 history"
    exit 1;
fi

./histostat.sh $1 | xargs -l ./dnstoip.sh | awk -f stat.awk
```

2.4 Question 6 :

```
#!/bin/awk

{
  if($1 in tab){
    tab[$1] ++;
  }else{
    tab[$1] = 1;
  }
}

END{
    for(i in tab){
        printf("%s %d : ", i, tab[i]);
        printf("\n");
    }
}
```

3 Corrections TP4

Le code du serveur :

```
#!/bin/bash

# communication ouverte int 16, fermeture int 15
# lecture dans le tube

# affichage processus courant, necessaire pour les clients
echo processus courant : $$ 1>&2

# detournement des signaux
trap 'echo "debut acces" 1>&2' 16
trap 'echo "fin acces" 1>&2' 15

# creation du tube
rm tube
mknod tube p

# effacement du verrou
rm -fr verrou

while : ; do
    read line < tube
    echo $line
    #cat tube
done

```

Le code du client :

```
#!/bin/bash

# accede periodiquement a la ressource

#-----------------------------------------------
# transmission au serveur
function transmet {
    echo $$ $@
```

```
}

#-----------------------------------------------
# trace de la transmission
function affiche {
    echo $$ $@ 1>&2
}

#-----------------------------------------------
# sommeil
function slepp {
    sleep $1
}

#-----------------------------------------------
# section critique
function sectioncritique {
    local delay=$1; shift
    transmet $delay
    slepp $delay
    rmdir verrou
    verrou=0
}

#-----------------------------------------------
# acces au serveur pendant un temps aleatoire
function acces {
    # accede un chiffre aleatoire
    delay=`./random.sh`
    affiche tente $delay

    # ouverture communication
    kill -16 $serveur

    # si verrou positionnable
    if mkdir verrou 2> /dev/null; then
        verrou=1
        sectioncritique $delay
    else
        affiche "acces reserve"
    fi

    # fermeture communication
    kill -15 $serveur
}

#-----------------------------------------------
# attend pendant un temps aleatoire
function attente {
```

```
        delay=`./random.sh`
        affiche attend $delay
        slepp $delay
    }


    #-----------------------------------------------
    #-------------- MAIN -----------------------------
    #-----------------------------------------------
    serveur=$1     # pid du processus observeur

    verrou=0         # indique si le verrou est positionne

    # detournement du signal ^C : repositionner la sortie standard et retirer le verrou
    trap 'exec 1>&3 3>&-; if [ $verrou -eq 1 ] ; then rmdir verrou; fi; exit' 2

    # informations de trace
    echo Processus accesseur $$
    echo Processus observeur $serveur

    # redirection de la sortie standard
    exec 3>&1
    exec 1>tube

    # boucle infinie attente - acces
    while : ; do
        attente
        acces
    done
```

Les scripts qui comptent en AWK :

```
#!/bin/awk

{
  if($1 in tab){
    tab[$1] ++;
  }else{
    tab[$1] = 1;
  }
}

END{
  for(i in tab)
    print i " " tab[i];
}
```

```
#!/bin/awk
```

```
{
  if($1 in tab){
    tab[$1] += $2;
  }else{
    tab[$1] = $2;
  }
}

END{
  taille = 0;
  for (i in tab) taille++ ;
  for(i in tab)
    print i " " tab[i] * 200 / taille / NR;
}
```