```
#!/bin/sh
# the next line restarts using tclsh \
exec tclsh "$0" "$@"
#
# Partie serveur de l'outil d'exploitation
#
# Eric BURGHARD 28/12/2001

package require aster 1.1
package require dp 4.0

source svrcfg.tcl

proc LoginClient { file adr } {
        return 1
}
proc Stat path {
        file stat $path st
        return [array get st]
}
proc GetDirContents path {
        return [glob -nocomplain [file join $path *]]
}
proc HandleFileInput { filefd cli } {
        if {[eof $filefd]} {
         fileevent $filefd readable {}
         close $filefd
         $cli Disconnect
        } else {
         puts [$cli GetSock] [read $filefd]
        }
}
proc NewUserConnection { sock adr port } {
        # Procédure traitant les nouvelles demandes de connections
        # - envois asynchrones des résultats d'execution des services
        # - transferts de fichiers
        #

        set cli [Client new cli[Client Seq] $sock]
        fconfigure $sock -blocking 0 -buffering line
        fileevent $sock readable [list HandleUserInput $cli]
}
proc HandleUserInput cli {
        set sock [$cli GetSock]

        if {[eof $sock]} {
         # supprime le client lorsque la fin de fichier est atteinte
         $cli destroy
        } else {
```

```
         set data [read $sock]
         puts "data: \"$data\""
         if {[regexp {^([A-Z]+) (.*$)} $data match key args]} {
          # execute la commande envoyée par le client ($key C {SERVICE,
GET, PUT})
          eval $key $args $cli
         }
        }
}
proc PUT { base_sid site_e site_d num cli } {
        # place une depeche dans le repertoire de reception
        # -> base_sid
        #    site_e
        #    site_d
        #    num

        set filename [file join [GetVal BALREC $base_sid] $base_d
X_${site_e}_${site_d}_${num}.dat]

}
proc GET { path cli } {
        set filefd [open $path r]
        fconfigure $filefd -blocking 0 -buffering line
        fileevent $filefd readable [list HandleFileInput $filefd $cli]

}
proc SERVICE { class id cli } {
        # Procédure pour signifier à un service d'envoyer son activité vers
$sock
        # -> class: Classe du type Service
        #    id:    identificateur du service
        #    sock:  socket client
        #
        [$class GetInst $id] Register [$cli GetSock]
}
# vérifie les arguments
switch $argc {
 0 {
  set port 5120
  set uport 5121
 }
 1 {
  set port [lindex $argv 0]
  set uport [expr $port + 1]
 }
 default {
  puts stderr "usage: [info script] \[port\]"
  exit 1
 }
}

# charge l'environnement global
GlobalSetEnv
```

```
# instancie les différents services
array set sites [GetVals SITE]
foreach base_sid_i [array names sites] {
 puts "scrutateur $base_sid_i"
 Scrut new scrut$base_sid_i $base_sid_i
}

# attend les RPCs sur le port $port
set rpc_srv [dp_MakeRPCServer $port LoginClient]
# attend les autres type de connection sur le port $uport
set usr_srv [socket -server NewUserConnection $uport]

# boucle indéfiniment
vwait forever

# ferme la connection vers les clients et la connexion RPC
foreach cli_i [Client allchildren] {
 $cli_i destroy
}
close $rpc_srv

# arrete tous les services
foreach serv_i [Service allchildren] {
 $serv_i destroy
}

exit
```