L3 Informatique
Université de Bourgogne

U.E. Systèmes & Réseaux I

Exercices Communication par Sockets

Exercice 1- *Mode non connecté*

1. Comment se fait la communication en mode non connecté ?
2. Expliquez comment se fait l'envoi d'un message ? Quelles peuvent être les raisons d'un échec ?
3. Expliquez comment se fait la réception d'un message ? Peut-il y avoir des situations d'interblocage (deadlock) ?
4. Faire le schéma des différentes étapes d'une communication (non connecté).
5. Commentez le programme en mode non connecté de l'annexe.

Exercice 2- *Mode connecté*

1. Dans le modèle client/serveur, quel processus demande la connexion ? Quelle fonction C permet de réaliser cette étape ?
2. Que doit-on créer avant la connexion ?
3. Comment se fait le dialogue ?
4. Quelle instruction met fin à la connexion ?
5. Quelle opération permet d'obtenir l'adresse de connexion ?
6. Quelles sont les opérations effectuées par le client et le serveur ? Faire le schéma correspondant détaillé.
7. Dans quels cas y a t-il une erreur lorsque la file d'attente est pleine ?
8. Commentez le programme en mode connecté de l'annexe.

```
// programme réseau en mode non connecté
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h> /* close() */
#include <string.h> /* memset() */
#include <stdlib.h>

#define FATAL(m) {perror(m); exit(1);}
#define LOCAL_SERVER_PORT 10001
#define MAX_BUF 10000
#define MAX_NOM 100

int main(int argc, char *argv[]) {
  int sd, rc, n, cliLen, fic;
  unsigned int taille;
  struct sockaddr_in cliAddr, servAddr;
  struct stat statfic;
  char nom[MAX_NOM];
  char buff[MAX_BUF];

  sd=socket(AF_INET, SOCK_DGRAM, 0); if(sd<0) FATAL("socket");

  servAddr.sin_family = AF_INET;
  servAddr.sin_addr.s_addr = htonl(INADDR_ANY);
  servAddr.sin_port = htons(LOCAL_SERVER_PORT);
  rc = bind (sd, (struct sockaddr *) &servAddr,sizeof(servAddr));
  if(rc<0) FATAL("bind");

  printf("%s: attente des clients sur le port UDP %u\n", argv[0],LOCAL_SERVER_PORT);

  while(1) {
    memset(nom,0x0,MAX_NOM);
    memset(buff,0x0,MAX_BUF);

    cliLen = sizeof(cliAddr);
    rc = recvfrom(sd, nom, MAX_NOM, 0, (struct sockaddr *) &cliAddr, &cliLen);
    if(rc<0) FATAL("recvfrom");
    fic=open(nom,O_RDONLY);
    if(fic==-1) { taille=0; rc = sendto(sd, &taille, sizeof(taille), 0,
(struct sockaddr *) &cliAddr, cliLen);}
    else { fstat(fic,&statfic); taille=htonl(statfic.st_size);
   rc = sendto(sd, &taille, sizeof(taille), 0, (struct sockaddr *) &cliAddr, cliLen);
```

```
        while((n=read(fic,buff,sizeof(buff)))>0)
     {sendto(sd, buff, n, 0, (struct sockaddr *) &cliAddr, cliLen);
    printf("envoi de %d octets\n",n);}
         }
    close(fic);
    printf("fin d'envoi\n");
  }
close(sd);
return 0;
}
```

```
// programme réseau en mode connecté
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/un.h>
void error(char *msg)
{
    perror(msg);
    exit(1);
}

void writeString(int newsockfd,char *buffer){
  int n;
  n = write(newsockfd,buffer,strlen(buffer));
  if (n < 0) error("ERROR writing to socket");
}

int main(int argc, char *argv[])
{
     int sockfd, newsockfd, clilen;
     int portno=8080;
     char buffer[256];
     char path[256];
     char fullpath[2560];
     int max;
     int r;
     struct sockaddr_in serv_addr, cli_addr;
     int n;

     sockfd = socket(AF_INET, SOCK_STREAM, 0);
     if (sockfd < 0)
        error("ERROR opening socket");
```

```
    bzero((char *) &serv_addr, sizeof(serv_addr));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
             error("ERROR on binding");
    while(1){
      listen(sockfd,5);
      clilen = sizeof(cli_addr);
      newsockfd = accept(sockfd,
 (struct sockaddr *) &cli_addr,
 &clilen);
      if (newsockfd < 0)
error("ERROR on accept");

      bzero(buffer,256);
      n = read(newsockfd,buffer,sizeof(buffer));
      if (n < 0) error("ERROR reading from socket");

      printf("Serveur : requêtes reçue : %s\n",buffer);
      sscanf(buffer,"GET %s %s\n",path);
      sprintf(fullpath,"/home/dgram/public_html%s",path);
      printf("Serveur : adresse demandée : %s\n",fullpath);

      writeString(newsockfd,"HTTP/1.1 200 OK\n");
      /*       writeString(newsockfd,"Date: Mon, 04 Dec 2006 15:09:25 GMT\n");
      writeString(newsockfd,"Server: Apache/2.2.3 (Debian) PHP/5.1.6-5\n");
      writeString(newsockfd,"Last-Modified: Tue, 04 Jul 2006 18:44:17 GMT\n");
      writeString(newsockfd,"ETag: \"ac259-1f19-e6eb0240\"\n");
      writeString(newsockfd,"Accept-Ranges: bytes\n");
      writeString(newsockfd,"Content-Length: 7961\n");
      writeString(newsockfd,"Connection: close\n"); */
      writeString(newsockfd,"Content-Type: text/html; charset=ISO-8859-1\n");
      writeString(newsockfd,"\n");

      FILE *f;
      if ((f=fopen(fullpath,"r"))==NULL) error("ouverture fichier");

      while(fgets(buffer,sizeof(buffer),f)!=NULL){
printf("lecture de %s",buffer);
writeString(newsockfd,buffer);
      }

      close(newsockfd);
    }
      return 0;
}
```