| | |
|---|---|
| CY TECH | **Programmation Système et Réseau**<br><br>**Examen Session Normale** |
| Thierry Garcia - Mohamed Haddache - Mariem Allouch Mahdi | |
| **ING2-GSI, GIA** | **Année 2022 - 2023** |

### Modalités

- Durée : 2 heures.
- Vous devez rédiger votre copie à l'aide d'un stylo à encre exclusivement.
- Toutes vos affaires (sacs, vestes, trousse, etc.) doivent être placées à l'avant de la salle.
- 2 feuilles recto verso manuscrites autorisées
- Aucune question ne peut être posée aux enseignants, posez des hypothèses en cas de doute.
- Aucune sortie n'est autorisée avant une durée incompressible d'une heure.
- Aucun déplacement n'est autorisé.
- Aucun échange, de quelque nature que ce soit, n'est possible.

# Exercice 1 : Questions de cours (5 points)

1. En utilisant le handler expliquer comment on peut ignorer un signal ?
2. Quelle est l'utilité des deux primitives : **dup** et **dup2** ? Expliquer la différence entre les deux.
3. Comment peut-on bloquer l'accès au clavier ? Illustrer la réponse par un programme **c**.
4. Quelle est la différence entre un processus et un thread ?
5. Quelle est l'utilité des sémaphores ?

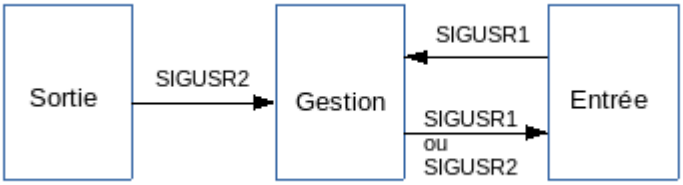
# Exercice 2 : (6 points)

L'accès à un ensemble de cabines d'essayage est réalisé de la façon suivante :

• Un premier processus « Gestion » récupère via la ligne de commande le nombre de cabines disponibles et affiche sur le terminal son numéro de processus.

• Il lance ensuite 2 processus fils, l'un pour l'entrée dans les cabines, l'autre pour la sortie.

• Le fonctionnement général du système est le suivant :




• Lorsqu'un client quitte une cabine, le processus Sortie envoie le signal SIGUSR2 au père qui met à jour le nombre de cabines disponibles (dans cette première phase la sortie est toujours autorisée)

• Lorsqu'un client désire entrer, le processus Entrée envoie le signal SIGUSR1 au processus Gestion qui lui répond SIGUSR2 si une cabine est disponible ou SIGUSR1 s'il n'y en a pas. Le processus Entrée se contentant d'afficher « Entrée autorisée » ou « Entrée interdite » en fonction de la réponse du processus Gestion. Le processus gestionnaire affichera également « Entrée autorisée » ou « Entrée interdite » en fonction du signal renvoyé au client.

1. Donner la fonction C de traitement de la réception du signal SIGUSR1 par le processus Gestion sachant que les variables globales ont été correctement initialisées (par exemple nbCabinesDisponibles).

2. Donner la fonction C unique de traitement de réception des deux signaux SIGUSR1 et SIGUSR2 par le processus Entrée. On rappelle que le processus Entrée se contente d'afficher « Entrée autorisée » ou « Entrée interdite » en fonction de la réponse du processus Gestion.
3. Quels moyens de communication proposeriez-vous si on voulait mettre en place une communication sans utiliser les signaux ? Citer plusieurs IPC vues en cours et donner quelques schémas expliquant leur fonctionnement.

## Exercice 3 : (3 points)

Faites jouer deux processus au jeu du plus grand, plus petit.

Pour rappel, le principe du jeu est le suivant :

-L'ordinateur tire au sort un nombre entre 1 et 100.

-Il vous demande de deviner le nombre. Vous entrez donc un nombre entre 1 et 100.

-L'ordinateur compare le nombre que vous avez entré avec le nombre « mystère » qu'il a tiré au sort. Il vous dit si le nombre mystère est supérieur ou inférieur à celui que vous avez entré. Et ainsi de suite, jusqu'à ce que vous trouviez le nombre mystère.

Le but du jeu, bien sûr, est de trouver le nombre mystère en un minimum de coups.

Dans cet exercice, l'ordinateur et le joueur sont chacun représentés par un processus. Afin de les faire communiquer, vous avez le choix entre les files de messages, la mémoire partagée, les sémaphores.

## Exercice 4 : (6 points)

## Partie 1

1. Compléter le programme client TCP afin que le client envoie et reçoive des messages vers/à partir du serveur.
2. Ecrire le code serveur correspondant à cette communication

**Client TCP**

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <unistd.h>

char* id=0;
short sport=0;
int sock=0; /* socket de communication */
int main(int argc, char** argv)
{
struct sockaddr_in client; /* SAP du client */
struct sockaddr_in serveur; /* SAP du serveur */
int ret,len;
if (argc!=4) {
fprintf(stderr,"usage: %s id serveur port\n",argv[0]);
exit(1);
}
id= argv[1]; sport= atoi(argv[3]);
if((sock = /*.....*/){
fprintf(stderr,"%s: socket %s\n",argv[0], strerror(errno));
exit(1);
}
serveur.sin_family = /*.....*/;
serveur.sin_port = htons(sport);
inet_aton(argv[2], &serveur.sin_addr);
if(connect(/*.....*/) {
fprintf(stderr,"%s: connect %s\n",argv[0],strerror(errno));
perror("bind");
exit(1);
}
len=sizeof(client);
getsockname(sock,(struct sockaddr *)&client, &len);
while(1){
char buf_read[256], buf_write[256];
printf("donner le message à envoyer: ");
scanf("%s", buf_write);
printf( "le message à envoyer par le client :  %s\n",  buf_write);
ret= (/*.....*/)
if(ret<=strlen(buf_write)) {
printf("\n%s: erreur dans write (num=%d, mess=%s)\n", argv[0],ret,strerror(errno));
continue;
}
ret= (/*.....*/)
if (ret<=0) {
```

```
printf("\n%s: erreur dans read (num=%d, mess=%s)\n", argv[0],ret,strerror(errno));
continue;
}
printf("le message recu:  : %s\n", buf_read);
}
close(sock);
printf("j ai fini, au revoir\n");
return 0;
}
```

## Partie 02

1. Compléter les deux programmes client UDP et serveur UDP afin que chaque entité (client, serveur) envoie et reçoive des messages de l'autre côté.
2. Supposant que nous exécutons les deux programmes sur la même machine et que nous utilisons l'adresse 127.0.0.1 et le numéro du port 7000, pour les deux entités. Quels sont les messages affichés au niveau de chaque entité (client, serveur) ?
3. Modifier le client afin qu'il ferme la connexion quand l'utilisateur tape : **FIN**.

### Client UDP

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <unistd.h>
char* id=0;
short sport=0;
int sock=0; /* socket de communication */
int main(int argc, char** argv)
{
struct sockaddr_in ClientUDP; /* SAP du client */
struct sockaddr_in serveur; /* SAP du serveur */
int nb_message=0;
int ret, len;
int serveur_len= sizeof(serveur);
char buf_read[256], buf_write[256];
if (argc!=4) {
fprintf(stderr,"usage: %s id host sport\n",argv[0]);
exit(1);}
id= argv[1];
sport= atoi(argv[3]);
if ((sock = socket (/*.....*/)) == -1) {
fprintf(stderr,"\n%s: socket %s\n", argv[0], strerror(errno));
exit(1);}
```

```
len=sizeof(ClientUDP);
getsockname(sock,(struct sockaddr *)& ClientUDP, &len);
serveur.sin_family = AF_INET;
serveur.sin_port =/*.....*/
inet_aton(argv[2], &serveur.sin_addr);
while (nb_message<3) {
ret=recvfrom(/*.....*/);
if (ret<=0) {
printf("\n %s: erreur dans recvfrom (num=%d, mess=%s)\n", argv[0],ret,strerror(errno));
continue;
}
    printf("\n client %2s: (%s:%4d) recoit de ",id,
    inet_ntoa(ClientUDP.sin_addr),ntohs(ClientUDP.sin_port));
printf("(%s:%4d) : %s\n",inet_ntoa(serveur.sin_addr), ntohs(serveur.sin_port),buf_read);
sprintf(buf_write,"#%2s=%03d",id,nb_message++);
printf("\n client %2s: (%s:%4d) envoie a ", id, inet_ntoa(ClientUDP.sin_addr),ntohs(ClientUDP.sin_port));
printf("(%s:%4d): %s\n", inet_ntoa(serveur.sin_addr),ntohs(serveur.sin_port), buf_write);
ret=sendto(/*.....*/ );
if (ret<=0) {
printf("\n%s: erreur dans sendto (num=%d, mess=%s)\n", argv[0],ret,strerror(errno));
continue; }
}
printf("J'ai fini, au revoir\n");
return 0;
}
```

**Serveur UDP**

```
 #include <stdio.h>
#include <errno.h>
#include <netinet/in.h>
#include <string.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <unistd.h>

char* id=0;
short port=0;
int sock=0; /* socket de communication */
int nb_reponse=0;
char buf_read[256], buf_write[256];
int main(int argc, char** argv)
{
int ret;
struct sockaddr_in serveur; /* SAP du serveur */
    if (argc!=3) {
    fprintf(stderr,"usage: %s id port\n",argv[0]);
    exit(1);
    }
```

```
id= argv[1];
port= atoi(argv[2]);
if (/*.....*/ ) {

fprintf(stderr,"%s: socket %s\n",argv[0],strerror(errno));
      exit(1);
      }
      serveur.sin_family = AF_INET;
      serveur.sin_port = htons(port);
      serveur.sin_addr.s_addr = INADDR_ANY;
      if (bind (/*.....*/)<0) {
      fprintf(stderr,"%s: bind %s\n",argv[0],strerror(errno));
      exit(1);
      }
while (1) {
struct sockaddr_in client; /* SAP du client */
int client_len= sizeof(client);
ret=recvfrom( /*.....*/);
if (ret<=0) {
printf("%s: recvfrom=%d:%s\n",argv[0],ret,strerror(errno));
continue;
}
printf("serveur %s (%s:%d) recu le message %s\n",id, inet_ntoa(client.sin_addr),ntohs(client.sin_port),
buf_read);

sprintf(buf_write,"#%2s reponse%03d#",id,nb_reponse++);
ret=sendto(/*.....*/);
if (ret<=0) {
printf("%s: sendto=%d: %s\n",argv[0],ret, strerror(errno));
continue;
}
sleep(6);
}
return 0;
}
```