CY TECH	**Programmation Système et Réseau** **Examen de Rattrapage**
Mohamed Haddache	
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.
- 1 feuille recto verso manuscrite autorisée
- 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.

Questions de cours (5 points)

1. Donner la différence entre **wait** et **waitpid** ? **(1 point)**
2. Citer trois mécanismes de communication entre les processus **(1 point)**
3. Donner la définition d'un sémaphore ? donner les primitives associées à un sémaphore ? **(2 points)**
4. Quelle est la différence entre un processus et un thread ? **(1 point)**

Exercice 1 (6 points)

1. Ecrire un programme **fork1.c** qui utilise la primitive **fork ()** pour créer un processus fils :
   - Dans le processus père vérifier que le fork s'est bien passé (code de retour ! = -1).
   - Le processus père devra afficher son PID, le PID de son propre père et celui de son fils, attendre la terminaison du fils et quitter. Vous utiliserez pour cela la fonction **wait.**
   - Le processus fils devra afficher son PID, le PID de son propre père.
2. Ecrire un programme **fork2.c**, modification de **fork1.c**, où la fonction **wait** est remplacée par la fonction **waitpid**.
3. Modifiez le code pour que le père affiche le code de retour du fils en utilisant les macros : **WIFEXITED** et **WEXITSTATUS**.

Exercice 2 (4 points)

Ecrire un programme ayant le comportement suivant :

1. Des threads sont créés (leur nombre étant passé en paramètre lors du lancement du programme)
2. Chaque thread affiche un message (par exemple ‘’hello world !’’)
3. Le thread principal attend la terminaison des différents threads crées.

Exercice 3 (5 points)

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;
}
```