CY TECH	**Programmation Système et Réseau** **Examen Session Normale**
Mohamed Haddache	
ING2-GIA	Année 2024 - 2025

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 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. Quel est le rôle de la fonction exec() en C ?
2. Donner deux avantages et deux inconvénients d'utiliser des processus par rapport à des threads pour la gestion de tâches parallèles ?
3. Expliquez la notion de "processus zombie" et comment un système d'exploitation le gère.
4. Quel est le rôle principal des signaux dans un programme C ?
5. Quel est le comportement par défaut du signal SIGINT (généré par l'interruption du clavier, comme Ctrl+C) ?

Exercice 2 : (5 points)

Partie 01

Ecrire un programme C où le père envoie une chaîne de caractères à un processus fils via un Tube (pipe) anonyme. Le fils doit transformer cette chaîne en majuscules, puis renvoyer la chaîne modifiée à son père via un autre pipe anonyme. Le père affichera la chaîne modifiée

Partie 02

Deux villes A et B sont reliées par une seule voie de chemin de fer. Les trains peuvent circuler dans le même sens de A vers B ou de B vers A. Cependant, ils ne peuvent pas circuler dans les sens opposés. On considère deux classes de processus : les trains allant de A vers B (Train A vers B) et les trains allant de B vers A (Train B vers A). Leur comportement se définit comme suit :

Train A vers B	**Train B vers A**
Demande d'accès à la voie par A	Demande d'accès à la voie par B
Circulation sur la voie de A vers B	Circulation sur la voie de A vers B
Sortie de la voie par B	Sortie de la voie par B

En utilisant les sémaphores et les threads, écrire les deux programmes C qui simulent les actions des deux processus précédents

Exercice 3 : (5 points)

On suppose que l'on a un programme contenant la fonction main suivante

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
 int main(void)
{
   int i;
   int tab[100];

   for (i = 0; i < 100; i++)
   tab[i] = calculcomplese(i);
   return EXIT_SUCCESS;
}
```

1. La fonction **calculcomplexe** est une fonction nécessitant un temps d'exécution important. Modifier ce programme pour que l'utilisateur puisse interrompre le calcul en tapant **CTRL+C** (**SIGINT**); l'utilisateur peut alors choisir entre confirmer le calcul ou l'arrêter définitivement
2. Ecrire un programme **ping.c** qui, lorsqu'il reçoit le signal **SIGQUIT (CTRL+\)** affiche le message **ping !** et envoie le signal **SIGQUIT** à un numéro de processus donné par l'utilisateur. Le programme doit afficher indéfiniment le message **ping !**

Exercice 4 : (5 points)

On veut développer une application qui permet l'échange des messages texte entre un client et un serveur en utilisant le protocole TCP, on vous donne le code du client, ci-dessous.

1. Expliquer le rôle des deux fonctions: **inet_aton** et **getsockname**
2. Compléter le programme client TCP afin que le client envoie et reçoive des messages vers/à partir du serveur.
3. Ecrire le code serveur correspondant à cette communication et qui prend en considération plusieurs clients

```
#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);



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;
}
```

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL