CPE LYON

ÉCOLE SUPÉRIEURE DE CHIMIE PHYSIQUE ÉLECTRONIQUE DE LYON

Ex 1	Ex 2	Ex 3	Ex 4	Ex 5	Ex 6	Ex 7	Ex 8
3 points	2 points	2 points	2 points	2 points	3 points	3 points	3 points

Note 20 /20

Consignes relatives au déroulement de l'épreuve

Date : 10 Janvier 2019

Contrôle TP : PROGRAMMATION CONCURRENTE – SESSION 1 - 4IRC 2018/2019

Durée : **1 heure**

Professeur responsable : T. LIMANE

Documents Cours/TP : ☒ autorisées ☐ non autorisés Calculatrices : ☐ autorisées ☒ non autorisées

LES TELEPHONES PORTABLES ET AUTRES APPAREILS DE STOCKAGE DE DONNEES NUMERIQUES NE SONT PAS AUTORISES.
Les oreilles des étudiants doivent être dégagées.

Rappels importants sur la discipline des examens

- La présence à tous les examens est strictement obligatoire; tout élève présent à une épreuve doit rendre une copie, même blanche, portant son nom, son prénom et la nature de l'épreuve.
- Toute absence non justifiée est sanctionnée par un zéro.
- Toute fraude ou tentative de fraude avérée est sanctionnée par un zéro à l'épreuve et portée à la connaissance de la direction des études qui pourra réunir le Conseil de Discipline. Les sanctions prises peuvent aller jusqu'à l'exclusion définitive du (des) élève(s) mis en cause.
- **TOUTE SUSPICION SUR LA REGULARITE ET LE CARACTERE EQUITABLE D'UNE EPREUVE EST SIGNALEE A LA DIRECTION DES ETUDES QUI POURRA DECIDER L'ANNULATION DE L'EPREUVE; TOUS LES ELEVES CONCERNES PAR L'EPREUVE SONT ALORS CONVOQUES A UNE EPREUVE DE REMPLACEMENT A UNE DATE FIXEE PAR LE RESPONSABLE D'ANNEE.**

Exercice 1 [3 /3 points]

```
s = sem_create(100,0) ;
V(s) ;
printf("Ok") ;
exit(0) ;
```

Qu'affiche ce script?

☒ **Ok** ☐ **rien**

```
s = sem_create(100,0) ;
V(s) ;
P(s) ;
printf("Ok") ;
exit(0) ;
```

Qu'affiche ce script?

☒ **Ok** ☐ **rien**

```
s = sem_create(100,0) ;
P(s) ;
printf("Ok") ;
exit(0 ) ;
```

Qu'affiche ce script?

☐ **Ok** ☒ **rien**

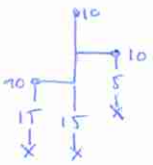
Exercice 2 [2 /2 points]

```
int main() {
    val = 10 ;
    if (fork() == 0) val = val-5 ; // fils
    else { // père
        fork() ;
        val +=5 ;
    }
    printf("%d " , val) ;
    exit(0 ) ;
}
```

Que peut afficher ce programme?

dans un ordre indéfini :

- 5
- 15
- 15



Exercice 3 [2 /2 points]

```
sem_t s ;
void* monThread (void* msg) {
    sem_wait(&s) ; // bloqué
    sem_post(&s) ;
    printf("%s ", (char*)msg) ;
    pthread_exit (0) ;

int main(){
    pthread_t th ;
    sem_init(&s, 0, 0 ) ; // sem init à 0
    pthread_create(&th, NULL, monThread, "monThread") ;
    sem_wait(&s) ; // bloqué
    printf("Père ") ;
    return 0 ;
}
```

Qu'affiche ce programme ?

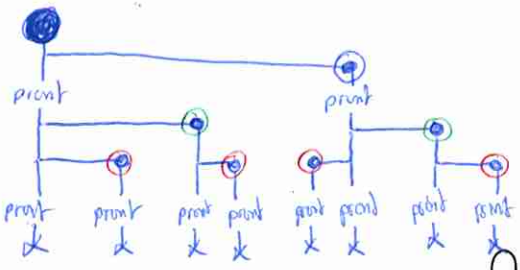
○ monThread ○ Père monThread ○ monThread Père ○ Père ⊗ Rien

Exercice 4 [9 /2 points]

Combien de lignes (4IRC 2019) affiche le programme suivant ?

Réponse : 10

```
void maFonction() {
  fork() ;
  printf("4IRC 2019\n") ;
  fork() ;
  fork() ;
  return ;
}
int main() {
  maFonction() ;
  printf("4IRC 2019\n") ;
  return 0 ;
}
```



Exercice 5 [9 /2 points]

```
int p=0;
void F(int s) {
   if (p==0)
      printf("Non\n") ;
   else
      printf("Oui\n") ;
   exit(0) ;

signal(SIGCHLD,F) ;
p = fork() ;
if (p==0) exit(0) ;
while(1) sleep(1) ;
printf("while") ;
```

L'exécution de ce programme provoque l'affichage de :

☒ **Oui** ○ **Non**

Pourquoi ? [Justifiez votre réponse]

car SIGCHLD est envoyé au père lorsqu'il a un nouveau fils

SIGCHLD → "child status has changed"
↳ je ne sais pas qui est notifié, père ou fils. je pars du principe que c'est le père qui est notifié

Exercice 6 [2 / 3 points]

```
void maFonction() {
      printf ("%d\n",getpid()) ;
}
int main () {
  signal (SIGINT , maFonction) ;
  if(fork()==0) {
      sleep(10) ;
      exit(0) ;
  }
  sleep(15) ;
  kill(getpid(),SIGINT) ;
  printf (" Fin ") ;
  return 0 ;
}
```

On lance ce programme ("**prg**") en ligne de commande. On suppose que le processus père possède le pid **500** (pid du fils est égal à **501**).
On lance alors la commande suivante une seconde après le lancement de "**prg**" :

$ kill -2 500

Que peut-on voir s'afficher à l'écran immédiatement après exécution de cette commande?

- ☐ **500 501**
- ☐ **500 501 501**
- ☐ **500**
- ☐ **500 501 Fin**
- ☒ **500 Fin** → j'aurai mis dit "500\n500\n Fin"

Exercice 7 [2 /3 points]

```
sem_t sem ;
void* F (void* arg) {
    sem_wait(&sem) ;
    sem_wait(&sem) ; //?
    sem_post(&sem) ;
    sem_post(&sem) ;
    return NULL ;
}
int main( ){
    pthread_t th1, th2 ;
    sem_init(&sem,0,2) ; // sem init à 2
    pthread_create(&th1, NULL, F, NULL) ;
    pthread_create(&th2, NULL, F, NULL) ;
    pthread_join(th1, NULL) ; // on attend la fin des threads
    pthread_join(th2, NULL) ;
    return 0;
}
```

Peut-on être sûr que les deux threads lancés vont se terminer ?

○ Oui ⊗ Non

Justifiez votre réponse

il n'y a que 2 "jetons" à prendre.
si un thread prend les 2 avant l'autre tout ira bien mais s'ils en prennent chacun un, ils vont se bloquer mutuellement (à la ligne avec "//?")

Exercice 8 [3 /3 points]

```
int main( ) {
   int *v;
   shmid=shmget(123, sizeof(int), IPC_CREAT|IPC_EXCL|0666);
   v = shmat(shmid, 0, 0) ;
   *v=0 ;
   for(int i=0; i<10; i++) {
      if (fork()==0) { // fils
         for(int k=0 ; k<50 ; k++) {
            *v = *v+1 ; lock unlock
            sleep(1) ;
         }
         exit(0) ;
   }
   while(wait(NULL) != -1) ;
   fprintf("%d" , *v) ;
}
```

On a exécuté 5 fois ce programme et le résultat affiché est différent d'une exécution à l'autre.

```
$ ./a.out
497
$./a.out
470
$./a.out
500
$./a.out
499
$./a.out
492
```

Expliquez de façon claire ces résultats.

REPONSE : la variable (v) n'est pas protégée par un mutex alors qu'elle est modifiée simultanément par plusieurs programmes. on ne garantis pas lire la bonne valeur, et la ligne [*v = *v+1] demande plus d'une instruction processeur.