

Commencé le mardi 30 avril 2024, 14:07**État** Terminé**Terminé le** mardi 30 avril 2024, 15:49**Temps mis** 1 heure 42 min**Note** 19,78 sur 20,00 (98,91%)**Question 1**

Correct

Note de 1,00 sur 1,00

Quel est le nombre de processus créés par ce bout de code (ne pas inclure le processus principal) ?

```
int main() {  
    printf("message0\n");  
    if (fork()) {  
        printf("message1\n");  
        if (fork()==0) { printf("message2\n"); _exit(0); }  
    }  
    else { printf("message3\n"); _exit(0); }  
    while(wait(NULL)>0);  
    printf("message4\n");  
    _exit(0);  
}
```

Veuillez choisir une réponse.

- ☐ 3
- ☐ 5
- ☐ 4
- ☐ 1
- ☒ 2 ✓
- ☐ Aucune de ces réponses

La réponse correcte est :

2

Question 2

Correct

Note de 1,00 sur 1,00

Quels sont les ordres possibles des affichages de messages ?

```
int main() {  
    printf("message0\n");  
    if (fork()) {  
        printf("message1\n");  
        if (fork()==0) { printf("message2\n"); _exit(0); }  
    }  
    else { printf("message3\n"); _exit(0); }  
    while(wait(NULL)>0);  
    printf("message4\n");  
    _exit(0);  
}
```

☒ message0 ✓

message1

message2

message3

message4

☐ Aucune de ces réponses☒ message0 ✓

message1

message3

message2

message4

☐ message0

message3

message1

message4

message2

☐ message0

message2

message1

message3

message4

☐ message0
message3
message4
message1
message2

☒ message0 ✓
message3
message1
message2
message4

Les réponses correctes sont :

message0
message1
message2
message3
message4,

message0
message1
message3
message2
message4,

message0
message3
message1
message2
message4

Question 3

Correct

Note de 1,00 sur 1,00

Considérez le code suivant :

```
int main() { /*1*/
    /*2*/
    printf("message0\n");
    if (fork()) { /*3*/
        printf("message1\n");
        if (fork()==0) { /*4*/
            printf("message2\n"); _exit(0);
        }
    } else { /*5*/
        printf("message3\n"); _exit(0);
    }
    while(wait(NULL)>0);
    printf("message4\n");
    _exit(0);
}
```

Complétez le code pour que les messages des "printf" soient récupérés dans le fichier "data.txt".

/*1*/	<input type="text" value='int fd=open("data.txt", O_WRONLY O_CREAT);'/>	✓
/*2*/	<input type="text" value="dup2(fd,1); close(fd);"/>	✓
/*3*/	<input type="text" value="rien"/>	✓
/*4*/	<input type="text" value="rien"/>	✓
/*5*/	<input type="text" value="rien"/>	✓

La réponse correcte est :

```
/*1*/ → int fd=open("data.txt", O_WRONLY| O_CREAT);,
/*2*/ → dup2(fd,1); close(fd);,
/*3*/ → rien,
/*4*/ → rien,
/*5*/ → rien
```

Question 4

Correct

Note de 2,00 sur 2,00

Considérez le code ci-contre qui crée 3 threads T0, T1 et T2 qui exécutent chacun la fonction f.

Synchronisez les cycles (les itérations) des threads, à l'aide de sémaphores POSIX, de manière à ce que les cycles T0 et T2 s'exécutent dans l'ordre suivant :

(cycle de T0 || cycle de T1) ; cycle de T2 ; (cycle de T0 || cycle de T1) ; cycle de T2, etc. Le symbole "||" signifie exécution en concurrence.

```
#define N 3
```

```
/*0*/ sem_t S[N]; // S[j] sert à bloquer/débloquer Tj, pour j=0 à 2
```

```
void * f (void *x) {
    long c=0;
    long j= (long)x;
    while(1) { // un cycle de Tj
        if(j!=2) { /*1*/ }
        else { /*2*/ }
        printf("Cycle %lu de T%lu\n", c, j);
        c=c+1;
        if(j!=2) { /*3*/ }
        else { /*4*/ }
    }
    pthread_exit(NULL);
}

int main() {
    pthread_t T[N];
    long j;
    for(j=0; j<N;j++) {
        if (j!=2) { /*5*/ }
        else { /*6*/ }
    }
    for(j=0; j<N;j++)
        pthread_create (&T[j],NULL,f,(void *)j);
    sleep(1);
    for(j=0; j<N;j++)
        pthread_cancel (T[j]);
    for(j=0; j<N;j++)
        pthread_join (T[j],NULL);
    for(j=0; j<N;j++)
        sem_destroy (&S[j]);
    return 0;
}
```

/*1*/ ✓

/*2*/ ✓

/*3*/ ✓

/*4*/ ✓

/*5*/ ✓

/*6*/ ✓

La réponse correcte est :

*/*1*/* → `sem_wait(&S[j]);`,

*/*2*/* → `sem_wait(&S[j]); sem_wait(&S[j]);`,

*/*3*/* → `sem_post(&S[2]);`,

*/*4*/* → `sem_post(&S[0]); sem_post(&S[1]);`,

*/*5*/* → `sem_init(&S[j],0,1);`,

*/*6*/* → `sem_init(&S[j],0,0);`

Question 5

Correct

Note de 1,50 sur 1,50

ATTENTION: Ne pas utiliser d'espaces vides.

Trois threads T0, T1 et T2 réalisent des appels à répétition à la fonction printf.

Complétez le moniteur Tour-a-Tour pour que les affichages s'exécutent dans l'ordre suivant :

printf de T0; printf de T1; printf de T2; printf de T0; printf de T1; printf de T2; etc.

```
#define N 3
```

```
Moniteur Tour-a-Tour {
```

```
    int tour=0; // à qui le tour ?
```

```
    boolc wq[N]; // pour attendre son tour
```

```
void wtour (int i) {
```

```
    if (  ✓ ) {
```

```
         ✓ ;
```

```
    }
```

```
}
```

```
void stour () {
```

```
     ✓ ;
```

```
     ✓ ;
```

```
}
```

```
}
```

```
Tour-a-Tour O;
```

```
T0 {
```

```
    while(1) {
```

```
        O.wtour(0);
```

```
        printf("Cycle de T0 \n");
```

```
        O.stour();
```

```
    }
```

```
}
```

```
T1 {
```

```
    while(1) {
```

```
        O.wtour(1);
```

```
        printf("Cycle de T1 \n");
```

```
        O.stour();
```

```
    }
```

```
}
```

```
T2 {
```

```
    while(1) {
```

```
        O.wtour(2);
```

```
printf("Cycle de T2 \n");  
O.stour();  
}  
}
```


Question 6

Partiellement correct

Note de 1,31 sur 1,50

ATTENTION: Ne pas utiliser d'espaces vides.

Trois threads T0, T1 et T2 réalisent des appels à répétition à la fonction printf.

Complétez le programme Tour-a-Tour pour que les affichages s'exécutent dans l'ordre suivant :

printf de T0; printf de T1; printf de T2; printf de T0; printf de T1; printf de T2; etc.

Comme le programme est en C, les moniteurs ne sont pas disponibles.

Utilisez plutôt les variables de condition POSIX.

```
#define N 3
```

```
int tour=0; // à qui le tour ?
```

```
pthread_cond_t wq[N]; // pour attendre son tour
```

```
pthread_mutex_t mutex; // pour assurer l'exclusion mutuelle en l'absence d'un moniteur
```

```
void wtour (int i) {
```

```
    pthread_mutex_lock(&mutex) ✓ ;
```

```
    if ( tour!=i ) {
```

```
        pthread_cond_wait(&wq[i], &mutex) ✗ ;
```

```
    }
```

```
    pthread_mutex_unlock(&mutex) ✓ ;
```

```
}
```

```
void stour () {
```

```
    pthread_mutex_lock(&mutex) ✓ ;
```

```
    tour=(tour+1)%N ✓ ;
```

```
    pthread_cond_signal(&wq[tour]) ✓ ;
```

```
    pthread_mutex_unlock(&mutex) ✓ ;
```

```
}
```

```
void * T0(void * arg) {
```

```
    while(1) {
```

```
        wtour(0);
```

```
        printf("Cycle de T0 \n");
```

```
        stour();
```

```
    }
```

```
}
```

```
void * T1(void * arg) {
```

```
    while(1) {
```

```
        wtour(1);
```

```
        printf("Cycle de T1 \n");
```

```
    stour();  
}  
}  
void * T2(void * arg) {  
    while(1) {  
        wtour(2);  
        printf("Cycle de T1 \n");  
        stour();  
    }  
}
```

Question 7

Correct

Note de 3,00 sur 3,00

1. Un système comporte 8 ressources d'un même type R partagées, en exclusion mutuelle, par 4 processus P1, P2, P3 et P4. Chaque processus peut demander, une à une, au maximum 3 ressources de type R qu'ils libèrent lui même, au bout d'un temps fini, à la fin de son exécution.

1.1. Est-ce qu'il pourrait y avoir un interblocage dans un tel système ? ✓

1.2. Peut-on rendre un tel système exempt d'interblocage en ajoutant des ressources ? ✓

1.3. Si vous répondez oui à la question 1.2, donnez le nombre de minimal de ressources à ajouter ✓ de type R

2. Pour éviter les interblocages, le système utilise l'algorithme du banquier. Supposez que chacun des processus P1, P2 et P3 détient 2 ressources de type R et que le processus P4 détient 1 ressource de type R. Il reste donc une seule ressource libre de type R. Le système reçoit dans l'ordre une demande d'une ressource R de la part de P4 et une demande d'une ressource R de la part de P1. Le système va-t-il allouer la dernière ressource libre à P4, à P1 ou à aucun des deux ? ✓

Utilisez la page suivante pour justifier toutes vos réponses.

Question 8

Non répondue

Non noté

Justifiez vos réponses aux questions précédentes.

1.1 Si chaque processus détient 2 ressources et se met en attente d'une ressource, il n'y a aucune ressource de disponible. Chaque processus est en attente d'une ressource R allouée à un autre.

1.2 et 1.3 Oui, en ajoutant une ressource R. En effet, dans le cas où chacun détient déjà 2 ressources, il suffit d'allouer restante à l'un des processus. Ce dernier va pouvoir accomplir son exécution et libérer 3 ressources R1. Ces ressources peuvent être ensuite allouées aux 3 autres processus pour accomplir leurs exécutions.

2. Alloc :

R1
P12
P22
P32
P41

A : (1),

Req :

R1
P11
P21
P31
P42

L'état suivant si la demande de R1 par P4 est acceptée :

Alloc' :

R1
P12
P22
P32
P42

A' : (0),

Req'

R1
P11
P21
P31
P41

Cet état n'est sûr car $\text{Req}'(P1) > A'$, $\text{Req}'(P2) > A'$, $\text{Req}'(P3) > A'$, $\text{Req}'(P4) > A'$. \Rightarrow La demande de P4 est rejetée

L'état suivant si la demande de R1 par P1 est acceptée :

Alloc' :

R1
P13
P22
P32
P41

A' : (0),

Req'

R1
P10
P21

R1

P3 1

P4 2

Cet état est sûr car $\text{Req}'(P1) = A'$, $A' = A' + \text{Alloc}'(P1) = (3)$

$\text{Req}'(P2) \leq A'$, $A' = A' + \text{Alloc}'(P2) = (5)$

$\text{Req}'(P3) \leq A'$, $A' = A' + \text{Alloc}'(P3) = (7)$

$\text{Req}'(P4) \leq A'$, $A' = A' + \text{Alloc}'(P4) = (8)$

=> La demande de R1 par P1 est acceptée

Question 9

Correct

Note de 2,00 sur 2,00

Considérez un système de pagination pure avec des tables de pages à 3 niveaux et des adresses virtuelles codées sur 32 bits. La taille d'une page est 2KiO. Toutes les tables de pages, peu importe leurs niveaux, sont de même taille et ont le même nombre d'entrées. Veuillez entrer des réponses sans espace.

1) Donnez la taille maximale en nombre de pages de l'espace d'adressage virtuel d'un processus. (en Mi pages)

Réponse: ✓ Mi

2) Donnez le format d'une adresse virtuelle utilisé pour la convertir en adresse physique. (en bits)

offset: ✓ bits

niveau1: ✓ bits

niveau2: ✓ bits

niveau3: ✓ bits

3) Donnez le numéro de page qui correspond à l'adresse virtuelle 0x00110A10.

Réponse: ✓ (en base 10)

1) 1 page = 2KiO = 211 octets. Sur les 32 bits, 11 bits sont réservés au déplacement dans la page. Il reste donc 21 bits pour le numéro de page. 2^{21} pages = 2Mi pages.

2) Le format d'une adresse virtuelle : 3 champs de 7 bits chacun pour les 3 niveaux de tables de pages et 11 bits pour le déplacement (l'offset).

3) Le numéro de page de 0x00110A10 est donnée par les 21 bits de poids le plus fort 0x0221 c-à-d $2^9 + 2^5 + 1 = 512 + 32 + 1 = 545$. La page 545. Les 11 bits restants (0x210) donnent le déplacement dans la page.

Question 10

Correct

Note de 2,50 sur 2,50

Un système monoprocesseur avec une gestion de mémoire par pagination pure et des tables de pages à un niveau.

- La mémoire physique est composée de 4 cadres.
- La taille de chaque cadre est de 4 KiO. L'adresse virtuelle est codée sur 16 bits.
- Supposez que 2 processus P1 et P2, composés respectivement de 7 et 5 pages, arrivent dans le système, l'un à la suite de l'autre.
- Le système charge dans l'ordre, les pages 0 et 1 de P1 dans les cadres 1 et 2, et la page 1 de P2 dans le cadre 3, avant de commencer l'exécution des processus P1 et P2 (pré-pagination).

ATTENTION: NE PAS METTRE D'ESPACES DANS LES REPONSES

1) Donnez l'adresse physique de l'adresse virtuelle : 0001 0011 0111 1000, pour chacun des cas suivants :

- P1 référence cette adresse virtuelle et
- P2 référence cette adresse virtuelle.

Adresse physique pour P1 sur 14 bits : ✓

Adresse physique pour P2 sur 14 bits: ✓

Adresse physique pour P1 en hexadécimal (0x...): ✓

Adresse physique pour P2 en hexadécimal (0x...): ✓

2) Le processeur reçoit, dans l'ordre suivant, les accès aux pages des processus P1 et P2 :

(0, P1) (1, P1) (1, P2) (5, P1) (4, P2) (5, P1) (6, P1) (1, P1) (2, P1)

où (0, P1), par exemple, référence la page 0 du processus P1.

Supposez que lorsqu'un défaut de page se produit et qu'un retrait de page est nécessaire, le système effectue un remplacement global en utilisant LRU.

Donnez l'évolution de l'état de la mémoire, ainsi que le nombre de défauts de pages provoqués par chaque processus.

L'état de départ des cadres (0,1,2 et 3) est "vide,(0,P1),(1,P1),(1,P2)".

Pour répondre a cette question, complétez le tableau suivant (ATTENTION: NE PAS METTRE D'ESPACES DANS LES REPONSES) :

(0,P1): ✓

(1,P1): ✓

(1,P2): ✓

(5,P1): ✓

(4,P2): ✓

(5,P1): ✓

(6,P1): ✓

(1,P1): ✓

(2,P1): ✓

Réponse pour les défaut de page : ✓ défauts de page provoqués par P1

✓ défauts de page provoqués par P2

Question 11

Correct

Note de 2,00 sur 2,00

Considérez un système monoprocesseur et les 3 processus suivants :

Processus	Temps d'exécution	Date d'arrivée
A	6	0
B	3(2)5	2
C	5	3

Donnez le diagramme de Gantt dans le cas d'un ordonnancement circulaire de quantum 4. Supposez que les temps de commutation sont nuls.

0

A

✓

1

A

✓

2

A

✓

3

A

✓

4

B

✓

5

B

✓

6

B

✓

7

C

✓

8

C

✓

9

C

✓

10

C

✓

11

A

✓

12

A

✓

13

B

✓

14

B

✓

15

B

✓

16

B

✓

17

C

✓

18

B

✓

19

rien

✓

20

rien

✓

La réponse correcte est :
0 → A,

- 1 → A,
- 2 → A,
- 3 → A,
- 4 → B,
- 5 → B,
- 6 → B,
- 7 → C,
- 8 → C,
- 9 → C,
- 10 → C,
- 11 → A,
- 12 → A,
- 13 → B,
- 14 → B,
- 15 → B,
- 16 → B,
- 17 → C,
- 18 → B,
- 19 → rien,
- 20 → rien

Question 12

Partiellement correct

Note de 2,47 sur 2,50

Considérez un système d'exploitation monoprocesseur doté d'un ordonnanceur préemptif, à base de priorités (0 étant la plus faible priorité).
Supposez les processus suivants :

Processus	Date d'arrivée	Séquence d'exécution	Priorité
A	2	EERE	10
B	5	EEEE	7
C	0	ERRRE	3

Les processus A et C partagent la ressource R (en exclusion mutuelle).

- a) Donnez le diagramme de Gantt de l'ordonnancement des processus entre les instants 0 et 13. Sur le diagramme, indiquez les accès des processus à la ressource R
- b) Y a-t-il une inversion de priorités? Si oui, précisez les processus concernés, l'instant de début, ainsi que la durée de l'inversion de priorités.
- c) Donnez le diagramme de Gantt de l'ordonnancement des processus entre les instants 0 et 13, dans le cas où le protocole PIP est utilisé pour traiter les inversions de priorités.
- d) Y a-t-il une inversion de priorités? Si oui, précisez les processus concernés, l'instant de début, ainsi que la durée de l'inversion de priorités.

Utilisez uniquement des lettres majuscules. Identifiez les cases inutilisées par un I (i majuscule).

a)

Diagramme de Gantt a)



b) OUI , si OUI, il y a une inversion de priorités de 6 unités de temps entre les processus:

- ☒C
- ☐B
- ☒A

La réponse correcte est :

- A
- C

c)

Diagramme de Gantt c)



d) ✓ , si OUI, il y a une inversion de priorités de ✓ unités de temps entre les processus:

- ☒ B ✗
- ☐ A
- ☒ C ✓

La réponse correcte est :

- A
- C