

**BCI – CC – Contrôle Partiel INF 104
Mars 2009 -1 heure - Sans document
INDICATIONS DE CORRECTION**

1 Questions de cours

1.1 Question 1 (2 points)

Dessiner le graphe d'état des processus et indiquer, pour chaque transition entre états, quels types d'événements peuvent la provoquer.

Corrigé -----

Transitions fausses -> 0,5.

Le graphe et les noms des transitions sans exemple : 1 point.

Tous les points si on donne des exemples de transitions : blocage/déblocage sur E/S, ou sur P et V sur sémaphore.

1.2 Question 2 (1 point)

Quels sont les avantages d'un sémaphore par rapport à un outil du type « test and set » ? Justifier.

Corrigé -----

0,5 point pour : pas d'attente active ou passage état bloqué, 0,5 pour file d'attente.

1.3 Question 3 (1 point)

Expliquer, de façon concise, quelle est la différence entre un programme et un processus.

Corrigé -----

0,5 point par mot clé : contexte, instance,...

1.4 Question 4 (1,5 point)

Que fait la fonction `exec` ? Exemple.

1.5 Question 5 (0,5 point)

Que veut dire multi-tâches ?

Que veut dire multi-processeur ?

1.6 Question 6 (2 points)

L'implémentation suivante du modèle lecteur/écrivain avec priorité aux lecteurs est-elle correcte ?

Si non, modifiez-la en justifiant vos modifications.

Variables partagées :

```
Init(S1, 0);
Init(S2, 1);
int N1=0;
```

Programme pour les lecteurs	Programme pour les écrivains
<pre>N1 = N1 + 1; si (N1 == 1) P(S2); Consulter(); N1 = N1 - 1;</pre>	<pre>P(S1); Modifier();</pre>

<code>si (N1 == 0) V(S1);</code>	<code>V(S2);</code>
----------------------------------	---------------------

Corrigé -----

Pour les lecteurs : ajouter un verrou sur N1 et remplacer par S1, initialisé à 1.

Pour les écrivains : mettre V(S1) après modification.

2 Exercices

2.1 Exercice 1 (2 points)

Soit les deux processus PA et PB qui partagent une variable `k` initialisée ainsi :

```
int k=1;
```

Processus A	Processus B
<pre>int main (){ int i; i=k; i=i+1; k=i; printf("k=%d\n",k); ... }</pre>	<pre>int main (){ int j; j=k; j=j+4; k=j; printf("k=%d\n",k); ... }</pre>

Les résultats suivants sont-ils possibles, ou non ? Pourquoi ?

	Processus A	Processus B
R1	<code>k=2;</code>	<code>k=2;</code>
R2	<code>k=2;</code>	<code>k=6;</code>
R3	<code>k=5;</code>	<code>k=2;</code>

Corrigé -----

Si on numérote I1, I2 les instructions de PA et J1, J2 celles de PB :

I1, J1, I2, J2, J3, I3, I4, J4 -> R1 (entrelacements se terminant par : J3, I3, I4, J4 ou , J3, I3, J4, I4)

I1, J1, I2, I3, I4, J2, J3, J4 -> R2

R3 impossible

2.2 Exercice 2

Nous allons compléter le programme ci-après en répondant aux questions suivantes.

```
int Var;
int main(){
int i, Pid;
void fonc();

Pid = fork();
if(Pid == 0){
for (i=0; i<200; i=i+1) {
Var=Var+1;
if (Var%7 == 0) kill(..., SIGUSR1);
}
exit(1);
}
if(Pid != 0){
}
return 0;
}
/***** fonc *****/
void fonc(){
```

```

printf("Var divisible par 7\n");
printf("Var=%d\n", Var);
}

```

Question 1 (4 points)

Compléter le programme pour que la fonction `fonc` soit appelée par le processus père à chaque émission du signal SIGUSR1. Justifier.

Corrigé -----(pour vérifier si les TP ont été faits correctement !)-----

```

int Var;
int main(){
    int i, Pid, Etat;
    void fonc();

    Pid = fork();
    if(Pid == 0){
        for (i = 0; i<200; i=i+1) {
            Var=Var+1;
            if (Var%7 == 0) kill(getppid(), SIGUSR1);
        }
        exit(1);
    }

    if(Pid != 0){
        signal (SIGUSR1, fonc);
        wait(&Etat);          /* !!!!!!!!!!!!! */
    }

    return 0;
}

/* Traitement de SIGUSR1 */
void fonc(){
    printf("Var divisible par 7\n");
    printf("Var=%d\n", Var);
    signal (SIGUSR1, fonc);
}

```

- 1 point pour l'ajout de `getppid` dans `kill`
- 2 (0,5 + 1,5) points pour (signal + wait) dans la branche « père »
- 1 point pour signal rappelé dans `fonc`

Question 2 (1 point)

Quelles sont les valeurs affichées pour `var` lors des appels à `fonc`? Justifier.

Corrigé -----
Toujours 0, Var est dupliquée...

2.3 Exercice 3

On veut réaliser un rendez-vous à N participants. Chaque processus participant va exécuter le programme suivant :

Variable partagée :
`int nb = 0;`

```

{
nb=nb + 1;
if (nb == N) {
/* L1 : débloquent tous les autres */
}
else {
/* L2 : se bloquer */
}
}

```

Question 1 (2,5 points)

Quel problème peut se poser lors de l'exécution de ce programme par plusieurs processus ? Modifier le programme pour résoudre ce problème.

(Pour le moment on ne se propose pas d'écrire les instructions remplaçant les lignes commentées L1 et L2)

Corrigé -----

Les 2,5 points ssi les appels sont bien placés, c'est à dire les V au début de chacune des branches du if, 2 si les V sont à la fin des branches.

P au début et V à la fin -> 1.

V dans une seule branche -> 1.

Protéger la variable nb :

`Init(Sem1, 1);`

```

{
P(Sem1);
nb=nb + 1;
if (nb == N) {
V(Sem1);
/* L1 : débloquent tous les autres L1 */
}
else {
V(Sem1);
/* L2 : se bloquer L2 */
}
}

```

Question 2 (2,5 points)

Proposer maintenant une programmation pour les deux lignes commentées L1 et L2.

Corrigé -----

Sémaphores bien placés, mais Sem2 pas bien initialisé -> 1,5.

`Init(Sem1, 1);`
`Init(Sem2, 0);`

```

{
P(Sem1);
nb=nb + 1;
if (nb == N) {
V(Sem1);
nb=nb - 1; /* pour faire (N-1) fois V */
while (nb > 0) {
V(Sem2);
nb=nb-1;
}
}
}

```

```
else {  
    V(Sem1);  
    P(Sem2);  
}
```