Question 1 (5 pts) : Généralités

a) **[1 pt]** Les fonctions CreateProcess et CreateFile de la librairie Win32 permettent de créer respectivement un processus fils et un fichier. Le processus père peut-il partager le pointeur de lecture / écriture d'un fichier ordinaire ? Si oui, expliquez comment ? Justifiez votre réponse.

b) **[1 pt]** Expliquez comment le système d'exploitation Linux (noyau 2.6) récupère-t-il l'information nécessaire à la vérification de la validité d'une adresse virtuelle d'un processus (la page référencée existe dans son espace d'adressage).

c) **[3 pts]** Un système gère 4 processus et 3 types de ressources. L'état courant du système est :

Processus	Alloc			Req			A (ressources disponibles)		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	1	0	2	0	1	0	1	1	x
P2	2	0	1	0	y	4			
P3	1	1	0	1	0	3			
P4	1	1	1	0	0	1			

Où Alloc indique les ressources allouées à chaque processus et Req donne pour chaque processus, les ressources nécessaires mais non encore obtenues.

1) Supposez que x=2 et y =1. À partir de cet état, donnez, s'il existe, un ordre des demandes d'allocation de ressources (non encore obtenues) qui mène vers un interblocage, dans le cas où l'évitement des interblocages n'est pas appliqué.

2) Supposez maintenant que l'algorithme du banquier est appliqué pour éviter les interblocages. Pour quelles plus petite valeur de x et plus grande valeur de y l'état courant est sûr ?

3) Supposez toujours que l'algorithme du banquier est appliqué pour éviter les interblocages, x = 2, y =1 et que le processus P3 demande 2 ressources de type R3. Est-ce que le système va accepter cette demande ? Justifiez votre réponse.

Question 2 (4.5 pts) : Moniteurs et variables de condition

a) **[2.5 pts]** On veut adapter le moniteur *ProducteurConsommateur*, vu en classe, au cas de deux producteurs *P1* et *P2* et d'un consommateur. Les producteurs doivent produire en alternance dans le tampon (une production de *P1* suivie d'une production de *P2*, etc.). Chaque producteur *Pi (i=1,2)* est supposé appeler sa propre fonction de production *produire_Pi.*

```
Moniteur ProducteurConsommateur1
{       const N=100 ;  int tampon[N] ;
        /*0*/
        void produire_P1 (int objet)
        {       /*1*/        }

        void produire_P2 (int objet)
        {       /*2*/      }

        int consommer ( )
        {       int objet ;  /*3*/  return objet ; }
}
```

Complétez le code du moniteur ci-dessus afin de permettre aux producteurs P1 et P2 de produire en alternance dans le tampon.
Attention : Comme mécanisme de synchronisation, vous devez vous limiter aux variables de condition (pour les deux questions a et b).

b) **[2 pts]** Supposez que la taille du tampon est égale à 1. Pour le cas d'un producteur et d'un consommateur, est-il possible de simplifier le code précédent (éliminer des variables de condition ou autres) ? Si oui, complétez le moniteur suivant. Sinon, justifiez votre réponse.

```
Moniteur ProducteurConsommateur2
{       int tampon ;
        /*0*/

        void produire (int objet)
        {       /*1*/        }

        int consommer ( )
        {       int objet ;  /*2*/  return objet ; }
}
```

Question 3 (4 pts) : Gestion de la mémoire

a) **[1 pt]** Dans un système paginé composé de 1024 cadres, la carte mémoire (core map) a des entrées sur 16 octets (chacune). Chaque entrée contient les informations sur un cadre de la mémoire physique. Si la taille d'un cadre est de 1024 octets, quel est en pourcentage l'espace mémoire occupé par la carte mémoire ?

b) **[1 pt]** Considérez un système de pagination avec des tables de pages à 4 niveaux et des pages de 4096 octets. Chaque adresse virtuelle est codée sur 48 bits répartis sur 5 champs : c1, c2, c3, c4 et d, où d est le déplacement dans la page (l'offset). Combien de bits sont réservés à l'offset d ? Le nombre maximal de pages de l'espace virtuel dépend-il de la répartition des bits restants sur les autres champs ? Justifiez votre réponse.

c) **[2 pts]** Supposez qu'un système paginé réserve 4 cadres physiques libres à chaque processus créés. Aucun autre cadre n'est alloué au processus. Durant son exécution, un processus référence dans l'ordre les pages : 0 1 7 2 3 2 7 1 0 3. Donnez pour chacun des algorithmes de remplacement de pages suivants, l'évolution de l'état des cadres 0, 1, 2 et 3 réservés au processus ainsi que le nombre de défauts de pages générés :
   - FIFO,
   - LRU.

Question 4 (6.5 pts) : Ordonnancement

Supposez trois processus P1, P2 et P3 avec les caractéristiques suivantes :

Processus	Date d'arrivée (Oi)	Durée d'exécution (Ci)
P1	0	10
P2	1	5
P3	2	7

Tableau 1

a) **[1.5 pt]** Donnez le digramme de Gant de l'exécution de ces processus dans le cas d'un ordonnancement circulaire de quantum de 4. Donnez les temps de séjour et d'attente moyens (TMS et TMA). Les temps de commutation de contexte sont supposés nuls.

b) Supposez maintenant que les processus P1 et P2 ont chacun une section critique (le processus P3 n'a pas de section critique). Les sections critiques de P1 et P2 sont exécutées en exclusion mutuelle. Les temps d'exécution des processus P1 et P2 donnés dans le tableau 2 n'incluent pas les temps d'attente avant l'accès aux sections critiques. Le triplet (xi, yi, zi) du temps d'exécution du processus Pi (pour i=1,2) signifie que :
   - Pi réalise un calcul de xi unités de temps CPU avant la demande d'entrer en section critique.
   - Pour exécuter et quitter sa section critique, yi unités de temps CPU sont nécessaires.
   - Après sa section critique, Pi se termine au bout de zi unités de temps CPU.

   L'ordonnancement de ces processus est toujours circulaire avec un quantum de 4 et les temps de commutation de contexte sont supposés nuls.

Processus	Date d'arrivée (Oi)	Durée d'exécution (Ci)
P1	0	10 (3, 6, 1)
P2	1	5 (2, 2, 1)
P3	2	7

Tableau 2

Donner le digramme de Gant de l'exécution des processus P1, P2 et P3, pour chacun des cas suivants :

1) **[2.5 pts]** une attente passive d'accès aux sections critiques.
2) **[2.5 pts]** une attente active d'accès aux sections critiques.

Vous devez indiquer sur les diagrammes les dates d'entrées et de sorties aux sections critiques ainsi que les attentes (passives ou actives). Donnez le temps de séjour moyen pour chaque cas.

Joyeuses fêtes

Le corrigé
Question 1

a) **Oui, car un fichier ouvert est un objet système qui a un jeton de sécurité. Parmi les attributs de ce jeton de sécurité, le champ bInheritHandle permet de préciser, lors de la création de l'objet, si cet objet est héritable par les processus fils du processus créateur de l'objet. Il suffit de positionner ce champ à « true » et de préciser lors de création du processus fils que tous les objets héritables sont partagés avec le fils (le paramètre *bInheritHandles* de CreateProcess doit être positionné à « true »).**

b) **Linux associe à chaque processus une structure task_struct (l'équivalent du bloc de contrôle du processus). La structure vm_area_struct, accessible via la structure task_struct par le champ mm, donne la liste des régions de l'espace virtuel du processus avec notamment leurs adresses virtuelles de début et de fin.**

c) 1) **P2 demande 2 R3 puis 1R2 A = 1 0 0**
P3 demande R1 puis R3 A = 0 0 0 puis bloque
P1 demande R2 bloque
P2 demande R3 bloque
P4 demande R3 bloque

2) **Req(P1) ≤ A , P1 est marqué A1 = A + Alloc(P 1) = (2 1 2+x)**
Req(P4) ≤A1, P4 est marqué A2 = A1 + Alloc(P4) = (3 2 3+x)
Req(P3) ≤ A2, P3 est marqué A3 = A2 + Alloc(P3) = (4 3 3+x)
Req(P2) ≤ A3 si x>=1 et y <=3.
x=1 et y =3

3) **Si 2R1 sont allouées à P3, l'état atteint serait :**

Processus	Alloc			Req			A (ressources disponibles)		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	1	0	2	0	1	0	1	1	0
P2	2	0	1	0	1	4			
P3	1	1	2	1	0	1			
P4	1	1	1	0	0	1			

Req(P1) ≤ A , P1 est marqué A1 = A + Alloc(P1) = (2 1 2)
Req(P3) ≤ A1, P3 est marqué A2 = A1 + Alloc(P3) = (3 2 4)
Req(P4) ≤ A2, P4 est marqué A3 = A2 + Alloc(P4) = (4 3 5)
Req(P2) ≤ A3 P2 est marqué.
Oui, la demande est acceptée car l'état atteint est sûr.

Question 2

a) *Moniteur ProducteurConsommateur1*

```
{       const N=100 ;
        int tampon[N] ;
/*0*/   boolc nplein, nvide ; // variables de condition
        int compteur =0, ic=0, ip=0, tour =1 ; // pour chacun son tour
        boolc  wtour ;
        void produire_P1 (int objet)
        {/*1*/  if (tour !=1) wait(wtour) ;
                if (compteur==N) wait(nplein) ;
                tampon[ip] = objet ; ip = (ip+1)%N ;compteur++ ;
                if (compteur==1) signal(nvide) ;
                tour = 2 ;
               signal (wtour) ;
        }
        void produire_P2 (int objet)
        {/*2*/  if (tour !=2) wait(wtour) ;
                if (compteur==N) wait(nplein) ;
                tampon[ip] = objet ; ip = (ip+1)%N ; compteur++ ;
                if (compteur==1) signal(nvide) ;
                tour = 1 ;
               signal (wtour) ;
        }
        int consommer ( )
        {/*3*/  int objet ;
                if (compteur ==0) wait(nvide) ;
                objet = tampon[ic] ; ic = (ic+1)%N ;  compteur -- ;
                if(compteur==N-1) signal(nplein) ;
                return objet ;
        }
}
```

b) **Oui, car cela revient à forcer l'ordre suivant : une production, une consommation, une production, etc.**

```
Moniteur ProducteurConsommateur
{       int tampon, tour =1 ;
        boolc  wtour ;
        void produire (int objet)
        {       if (tour !=1) wait(wtour) ;
                tampon = objet ;
                tour = 2 ;
```

```
            signal (wtour) ;
        }
        int consommer ( )
        {       int objet ;
                if (tour !=2) wait(wtour) ;
                objet = tampon ;
                tour = 1 ;
                signal (wtour) ;
                return objet ;
        }
}
```

Question 3

a) **16 octets sont nécessaires pour 1024 octets pour 100 octets : (16 x 100)/1024 = 1,56 %**

b) **d est sur 12 bits. Non, il ne dépend pas de la répartition des bits. Il est égal à 2^{48-12} = 64 Gi pages.**

c)

FIFO	0	1	7	2	3	2	7	1	0	3
0	0	0	0	0	3	3	3	3	3	3
1		1	1	1	1	1	1	1	0	0
2			7	7	7	7	7	7	7	7
3				2	2	2	2	2	2	2

FIFO 6 défauts de pages

LRU	0	1	7	2	3	2	7	1	0	3
0	0	0	0	0	3	3	3	3	0	0
1		1	1	1	1	1	1	1	1	1
2			7	7	7	7	7	7	7	7
3				2	2	2	2	2	2	3

LRU 7 défauts de pages.

Question 4

a) **0 P1 4 P2 8 P3 12 P1 16 P2 17 P3 20 P1 22**
TMA = ((22-10) + (16-5) + (18-7)) / 3 = 11.33
TMS = ((22) + (17-1) + (20-2)) / 3 = 18.66

b) 1) **0 P1 3 P1 sc 4 P2 6 P3 10 P1 sc 14 P3 17 P1 sc 18 P1 19 P2 sc 21 P2 22**
P2 bloque à 6, il est débloqué à 18. TMS = (19 + (22-1) + (17-2)) / 3 = 18,33

2)
0 P1 3 P1 sc 4 P2 6 P2 aa 8 P3 12 P1 sc 16 P2 aa 20 P3 23 P1 sc 24 P1 25 P2 sc 27 P2 28
(aa pour attente active). TMS = (25 + (28 -1) + (23-2)) / 3 = 24,33