Intra - ift2245 Nom: Liam + 1 Matricule:

Examen Intra

IFT 2245

27 février 2019

Directives

— *Aucune* documentation autorisée.
— Pas de calculatrice, téléphone, ou autre appareil électronique autorisé.
— Répondre dans le cahier.
— Le pointage pour chaque question est entre parenthèses (total = 20).

Question 0. *Nom et prénom (1 point de bonus)*

Écrire son nom et prénom et son matricule en haut de chaque page et aussi sur la page couverture du cahier réponse.

Question 1. *Les processus et threads (5 points au total)*

fichiers +0.5
tas (thread) +0.5
variable globales

(a) (1 point) Nommez deux choses qui sont partagés entre un enfant *thread* et son parent (i.e., après *pthread_create*) qui ne sont pas partagés entre un enfant *processus* et son parent (i.e., après un *fork*) ?

(b) (1 point) Nommex deux stratégies de communication dans les systèmes "client-server" ?

Sockets +0.5
RPC +0.5
Tuyaux

(c) (3 points) Qu'est qui est imprimé quand ce code est exécuté :

```
int value = 0;
void *runner(void *param);
int main(int argc, char *argv[])
{
  pid_t pid;
  pthread_t tid;
  pthread_attr_t attr;
  pid = fork();
  if (pid == 0) {
    pthread_attr_init(&attr);
    pthread_create(&tid,&attr,runner,NULL);
    pthread_join(tid,NULL);
    printf("LINE A value = %d\n",value);
    char buffer[100];
    sprintf(buffer, "LINE B value = %d",value);
    execlp("echo","echo",buffer,NULL);
    printf("LINE C value = %d\n",value);
  }
  else if (pid > 0) {
    wait(NULL);
    printf("LINE D value = %d\n",value);
  }
  printf("LINE E value = %d\n", value);

}

void *runner(void *param) {
  value += 5;
  pthread_exit(0);
}
```

LINE A value = 5
LINE B value = 5
LINE D value = 0
LINE E value = 0

-0.5 every mistake.

Question 2. *Synchronisation (5 points au total)*

(a) (2 points) En considérant ce bloc de code :

```
int value = 0;
void *runner(void *param);
int main(int argc, char *argv[])
{
  pid t pid;
  pthread t tid1, tid2;
  pthread attr t attr;
  pthread attr init(&attr);
  pthread create(&tid1,&attr,power2,NULL);
  pthread create(&tid2,&attr,power2,NULL);
  pthread join(tid1,NULL);
  pthread join(tid2,NULL);

  printf("value = \%d\n",value);
}

void *power2(void *param) {
  int temp;
  temp = value;
  temp = temp * value;
  value = temp;
  pthread exit(0);
}
```

Quelles sont les différentes sorties pouvant être imprimées à l'écran ? (Vous pouvez supposer que les opérations d'association et multiplication dans la fonction `power2` sound individuellement atomiques)

(b) (1 point) Dans le problème "lecteur-écrivain", nous devons assurer que les auteurs ont un accès exclusif au système de fichiers, mais de nombreux lecteurs peuvent accéder au système de fichiers simultanément. Pourquoi un moniteur n'est pas un outil approprié pour résoudre le problème lecteur-écrivain ?

(c) (1 point) Nous avons vu deux définitions différentes d'un sémaphore et la manière dont elles implémentent la fonction `wait` :

Définition 1 :

```
typedef struct {
   int value;
   struct process *list;
} semaphore;

wait(semaphore *S) {
   S->value--;
   if (S->value < 0) {
     add this process to S->list;
     block();
   }
}
```

Définition 2 :

```
int S;

wait (S) {
    while (S <= 0);
    S--;
}
```

Quelle est la différence entre les deux (en ce qui concerne l'état du processus) ?

(d) (1 point) Nommez 2 (sur 3) des propriétés d'une solution au problème de la section critique.

Intra - ift2245 Nom:_____ Matricule:_____

Question 3. *L'ordonnancement (5 points au total)*

(a) (1 point) Dans le cas de l'algorithm d'ordonnancement "Round Robin", le choix de la "time slice (quantum)" impose un "tradeoff". Décris le.

Response time vs. # context switch.

Pour un ensemble de processus $P_{1..4}$, nous anticipons la situation suivante :

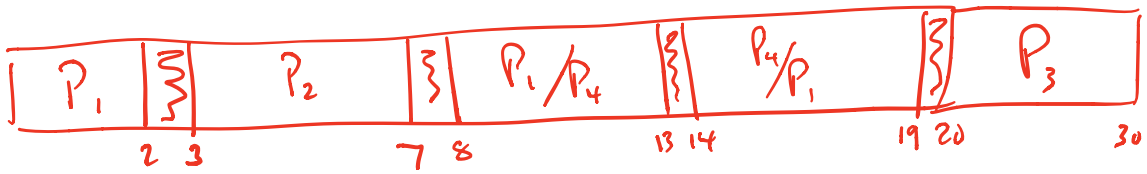
Processus	Arrivée* (ms)	CPU *burst time*(ms)
P_1	0	7
P_2	2	4
P_3	4	10
P_4	6	5

* indique le moment où le processus est d'abord prêt à être exécuté.

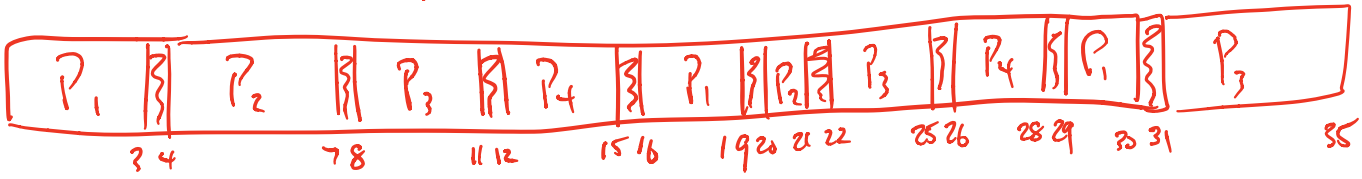
En plus, supposons que le "overhead" du "dispatcher" (temps gaspillé a faire un changement de contexte) est **1ms**

(b) Dessinez le diagramme de Gantt pour l'algorithme "shortest remaining time first" (préemptif)
(c) Dessinez le diagramme de Gantt pour l'algorithme "Round Robin" avec quantum = 3ms
(d) Quelle est l'utilisation du processeur ("CPU utilization") pour (b) et (c) ?
(e) Quelle est le délai moyen ("average turnaround time") pour (b) et (c) ?

(b)



(c)



(d) (i) $\frac{7 + 4 + 10 + 5}{30} = \frac{26}{30}$

(ii) $\frac{26}{35}$

(e) (i) $\frac{1}{4}(13 + 5 + 26 + 13) = 57/4 = 14.25$

(ii) $1/4(30 + 19 + 31 + 22) = 102/4 = 25.5$

without overhead : $\frac{1}{4}(16 + 4 + 7 + 22) = 45/4$

$\frac{1}{4}(22 + 14 + 15 + 22) = 73/4$

Question 4. *Interblocage (5 points au total)*

Le tableau ci-dessous indique les ressources actuelles et maximales requises pour la ressource de type R_j.

Processus	Maximum	Actuel
P_1	10	4
P_2	5	2
P_3	8	2

← Needs 3 → returns 2 for 5 but P_1 and P_3 need 6 so 4 more than what's already allocated

4+4+2+2 = 12

(a) (1 point) Quel est le nombre total minimum d'instances de ressource de type R_j qui doit exister pour que le système soit dans un état sûr ("safe state") ?

Pour (b) - (e), considérons un système avec trois types de ressources, chacune avec un nombre différent d'unités totales :

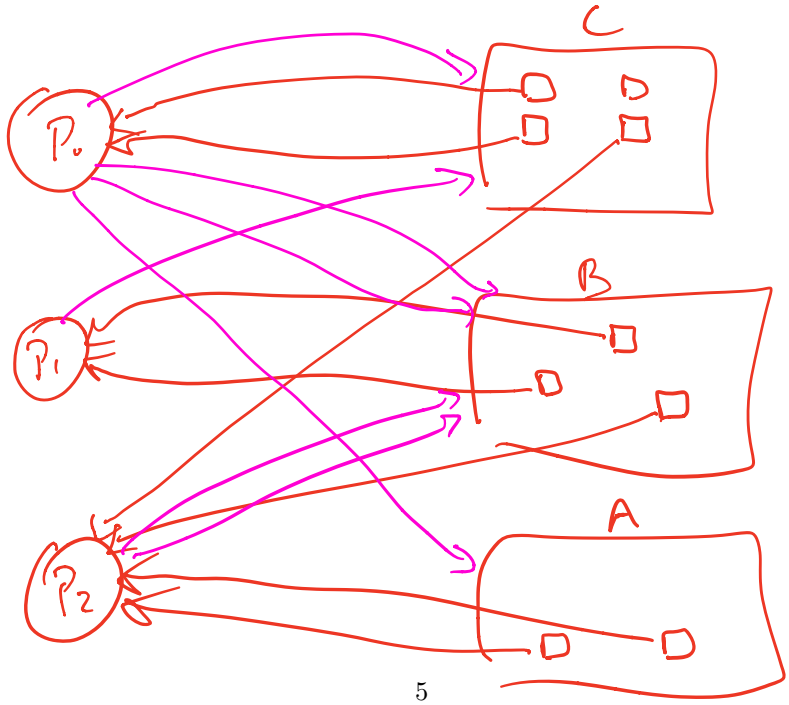
A - 2, B - 3, C - 4

À un instant donné, il existe trois processus dont les allocations de ressources actuelles et maximales sont indiquées dans le tableau ci-dessous :

Processus	Alloué	Max
	A, B, C	A, B, C
P_0	$0, 0, 2$	$1, 2, 3$
P_1	$0, 2, 0$	$0, 2, 1$
P_2	$2, 1, 1$	$2, 3, 1$

(b) (1 point) dessine le graphe de ressources (avec les arcs futurs).
(c) (1 point) le système est-il dans un état sûr ("safe state") ? Pourquoi ou pourquoi pas ?
(d) (1 point) P_1 demande 2 ressources de type C, devrait-il être accordé ? Pourquoi ou pourquoi pas ?
(e) (1 point) P_0 demande 1 ressource de type C, devrait-il être accordé ? Pourquoi ou pourquoi pas ? (supposons que le système est toujours dans l'état indiqué dans le tableau ci-dessus, pas après l'allocation en (d))

(b)



(c) order $P_1 - P_2 - P_0$

(d) request > Max so NO

(e) No -> not in safe state.