CSI351 – Système d'exploitation
Tutorat 3 – Hiver 2016
Les fils (threads) - Solutionnaire

1. Quelles sont deux différences entre les fils utilisateurs et les fils noyaux? Dans quelles circonstances est-ce qu'un type de fil est meilleur que l'autre?

   (1) Les fils utilisateurs ne sont pas connus du noyau, tandis que le noyau reconnait les fils noyau. (2) Dans les systèmes qui utilisent les modèles soit M :1, soit M :N, les fils utilisateurs sont ordonnancés par une bibliothèque de fil et les fils noyaux sont ordonnancés par le noyau. (3) Les fils noyaux ne sont pas nécessairement associés à des processus tandis que tous fils utilisateurs font partie d'un processus. (4) Les fils noyaux sont plus couteux à entretenir en temps et ressources que les fils utilisateurs car les fils noyaux doivent faire partie de structures du noyau.

2. Quelles ressources sont utilisées à la création d'un fil? Comment ces ressources diffèrent de celles utilisées pour créer un processus ?

   Parce qu'un fil est plus petit qu'un processus, la création de fil utilise typiquement moins de ressources que la création d'un processus. La création d'un processus requière l'allocation d'un bloc de contrôle de processus (PCB – process control block), ce qui est un structure assez grande. Le PCB comprend un topogramme de mémoire (memory map), une liste de fichiers ouverts, et des variables d'environnement. L'allocation et la gestion du topogramme de mémoire est typiquement l'activité le plus demandant en temps. La création de fil utilisateur ou fil noyau ne demande que l'allocation de petite structure de donnée pour contenir un ensemble de registre, une pile, un état, et la priorité.

3. Décrivez la relation entres fils utilisateurs et fils noyaux pour chacun des modèles suivants : Plusieurs-à-un, Un-à-un, et plusieurs-à-plusieurs. En particulier indiquez où l'ordonnancement se fait pour chacun des modèles.

   Plusieurs à un:
   - **Les fils sont gérés dans un processus par une bibliothèque qui roule dans l'espace utilisateur. Le noyau n'est pas conscient des fils.**
   - **L'ordonnancement des fils: par la bibliothèque de fils dans l'espace utilisateur.**
   - **Avantage: Efficace quand le noyau n'est pas impliqué dans la gestion de fils. Le nombre de fils est limité par le développeur de la bibliothèque de fils et par l'application.**
   - **Désavantages : Lorsqu'un fil bloque sur un appel système fait au noyau, tous les fils seront bloqués aussi (c'est le processus au complet qui bloque).**

   Un à un:
   - **Le SE est conscient des fils dans le processus. Chaque fil utilisateur est associé à un fil noyau.**
   - **Ordonnancement : se fait par le noyau dans l'espace du noyau.**
   - **Avantage : un fil qui bloque ne bloque plus les autres fils, les fils d'un même processus peuvent rouler sur différents UCTs.**
   - **Désavantage : Le noyau est impliqué dans la gestion des fils est ainsi coûte plus cher en temps. Le nombre de fils par processus est normalement limité.**

   Plusieurs à plusieurs:
   - **Un nombre de fils noyaux sont alloué à un processus pour permettre l'exécution des fils utilisateurs. Les fils utilisateurs sont par la suite ordonnancés à ces fils noyaux (LWP) qui sont perçus comme des UCTs virtuels.**
   - **Ordonnancement : se fait par la bibliothèque et le noyau.**
   - **Avantage : Plusieurs fils utilisateurs peuvent être créés tandis que la concurrence est assurée par les fils noyaux.**
   - **Désavantages : La coordination entre la bibliothèque et le noyau est difficile.**

4. Le programme ci-dessous utilise les Pthreads. Quelle sera la sortie du programme à la LIGNE E et à la LIGNE P? Expliquez votre réponse.

```
#include <pthread.h>
#include <stdio.h>

int valeur = 0;
void *courreur(void *param); /* function pour le fil*/

int main(int argc, char *argv[])
{
   int pid;
   pthread_t tid;
   pthread_attr_t attr;

   pid = fork();

   if(pid == 0)  /* l'enfant */
   {
      pthread_attr_init(&attr);
      pthread_create(&tid,&attr,courreur,NULL);
      pthread_join(tid,NULL);
      printf(ENFANT: valeur = %d",valeur); /* LIGNE E */
   }
   else if(pid > 0) /* le parent */
   {
      wait(NULL);
      printf("PARENT: valeur = %d\n",valeur);  /* LIGNE P */
   }
}

void *courreur(void *param)
{
   valeur = 5;
   pthread_exit(0);
}
```

Ligne C: ENFANT: valeur = 5 **Puisque le fil change la valeur de la variable globale.**
Ligne P: PARENT: valeur = 0 **Aucun changement se fait à la variable global dans le parent, tout changement fait dans le processus enfant à aucun effet sur les variables globales dans le processus parent.**

5. Considérez que deux processus démarrent au temps=0. Le processus A roule avec un seul fil d'exécution, tandis que le processus B roule un programme multi-fils qui démarre trois fils. Le fil F1 commence au temps t=0, le fil F2 commence après que F1 ait roulé (i.e. affecté à l'UCT) pendant 190 unités de temps (u.t.), et le file F3 commence après que F1 ait roulé pendant 27 u.t. (i.e. que F1 démarre les fils F2 et F3). L'exécution du processus A et chaque fil du processus B est composé d'une séquence de rafales d'UCT séparés pare des opérations d'E/S demandé au SE. Prenez pour acquis que chaque opération d'E/S prend 52 u.t. pour compléter. L'exécution du processus A et de chaque fil du processus B se découle comme suit :

   Processus A (le multi-fil n'est pas utilisé)

      rafale d'UCT de 55 u.t., demande E/S, rafale d'UCT de 50 u.t., demande E/S, rafale d'UCT de 30 u.t., demande E/S

   Processus B (multi-fils)

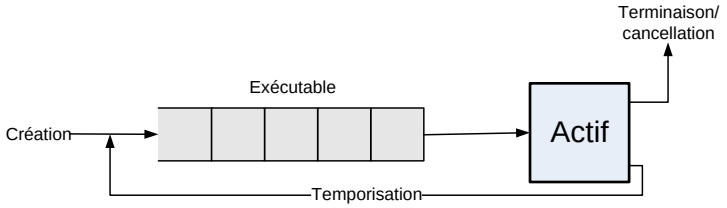
      T1: rafale d'UCT de 300 u.t. (tributaire de l'UCT)

      T2: rafale d'UCT de 12 u.t. suivie d'un demande E/S (répété 5 fois) (fil tributaire d'E/S)

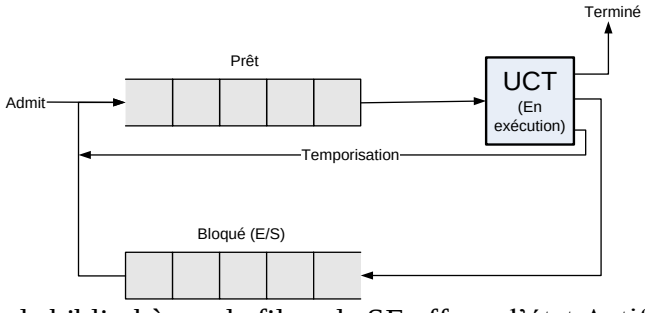
      T3: rafale d'UCT de 29 u.t., demande E/S, rafale d'UCT de 10 u.t., demande E/S, rafale d'UCT de 40 u.t., demande E/S

   Pour simplifier la question, prenez pour acquis que le temps d'exécution du code de gestion du SE et de la bibliothèque de fils est négligeable (i.e. le temps de commutation entres fils et/ou processus est traité comme étant 0).

a) **Modèle plusieurs à un**: En premier, considérez que les processus mutli-fils sont supportés avec un modèle plusieurs à un, c'est-à-dire, qu'une bibliothèque de fils est responsable de la gestion des fils dans un processus. Prenez pour acquis que la bibliothèque de fils utilise deux états une seule file d'attente tel que montré ci-dessous pour gérer les fils. Un fil est donné l'état Exécutable lorsqu'il attend de devenir actif et est donné l'état Actif lorsqu'il est permit d'exécuté (essentiellement affecté au fil noyau). D'autres états sont typiquement définis pour la gestion de fils utilisateurs, tel que Endormi et Arrêté. Mais pour garder les choses simples, ces états ne sont pas considérés pour ce problème. Prenez pour acquis que la bibliothèque de fils permet à un fil un maximum de 5 unités de temps dans l'état Actif avant de faire la commutation à un autre fil.



Le SE utilise trois états et les files d'attente montré ci-dessous. Les états Prêt, Bloqué, et En exécution sont utilisé pour représenter respectivement un processus prêt à rouler, bloqué en attente d'une opération d'E/S, et alloué à l'UCT. Notez que les états tels que Nouveau, Terminé, et états pour processus suspendus ne sont pas inclus, encore pour garder le problème simple. Prenez pour acquis que le SE affecte l'UCT à un processus pour un maximum de 10 unités de temps, après lesquels l'UCT sera affecté au prochain processus à la tête de la file Prêt.



Prenez pour acquis que la bibliothèque de fils et le SE affecte l'état Actif/En exécution au fil/processus à la tête des files d'attentes Exécutable/Prêt. Tous processus/fils ajouté à ces files d'attentes sont placés à la fin de la file. Ceci est un algorithme d'ordonnancement premier-arrivé premier-servi (nous étudierons cet algorithme et d'autres algorithme d'ordonnancement bientôt dans le module 4).

Complétez le tableau suivant pour montrer comment les processus et fils sont déplacés entre les files d'attente et les états Actif/En exécution jusqu'à l'unité de temps 200. Notez qu'un processus est affecté à l'UCT lorsque l'état En exécution lui est donné, et que seulement lorsqu'un processus est affecté à l'UCT, un fil dans l'état Actif exécute. Il est possible qu'un fil soit dans l'état Actif quand un processus est dans l'état Prêt ou Bloqué.

Si un événement d'ordonnancement au niveau utilisateur et au niveau noyau on lieu en même temps, prenez pour acquis que seulement l'évènement d'ordonnancement du SE est complété, et que celui de la bibliothèque de fils sera complété la prochaine fois que l'UCT sera affecté au processus (voyez le temps 50 pour un exemple). Rappelez-vous que la bibliothèque de fils permet un fil d'avoir l'état Actif que pour un maximum de 5 unités de temps et que le SE affecte l'UCT (i.e. à l'état en exécution) à un processus un maximum de 10 unités de temps. Aussi prenez pour acquis que le temps avance pour un fil dans l'état Actif seulement lorsque le processus B est dans l'état En exécution, i.e., affecté à l'UCT. Ceci veut dire que lorsqu'un fil est dans l'état Actif et que le processus est soit dans l'état Prêt ou Bloqué, la bibliothèque ne considère pas que le temps avance pour le fil dans l'état Actif.

	UCT	**Système d'exploitation**		**Processus B**	
Temps	En exécution	File d'attente prêt	File d'attente bloqué	Actif	File d'attente exécutable
0	A(0)*	B(0)		F1(0)	
10	B(0)	A(10)		F1(0)	
20	A(10)	B(10)		F1(10)	
30	B(10)	A(20)		F1(10)	
39	B(19)	A(20)		F1(19)	F2(0)
40	A(20)	B(20)		F1(20)	F2(0)
50	B(20)	A(30)		F2(0)	F1(20)
55	B(25)	A(30)		F1(20)	F2(5)
60	A(30)	B(30)		F1(25)	F2(5)
70	**B(30)**	**A(40)**		**F2(5)**	**F1(25)**
75	**B(35)**	**A(40)**		**F1(25)**	**F2(10)**
77	**B(37)**	**A(40)**		**F1(27)**	**F2(10),F3(0)**
80	**A(40)**	**B(40)**		**F1(30)**	**F2(10),F3(0)**
90	**B(40)**	**A(50)**		**F2(10)**	**F3(0), F1(30)**
92	**A(50)**		**B(42)**	**F2(12)**	**F3(0), F1(30)**
97			**B(42), A(55)**	**F2(12)**	**F3(0), F1(30)**
144	**B(42)**		**A(55)**	**F2(12)**	**F3(0), F1(30)**
147	**B(45)**		**A(55)**	**F3(0)**	**F1(30), F2(15)**
149	**B(47)**	**A(55)**		**F3(2)**	**F1(30), F2(15)**
150	**B(50)**	**A(55)**		**F1(30)**	**F2(15), F3(5)**
154	**A(55)**	**B(52)**		**F1(32)**	**F2(15), F3(5)**
164	**B(52)**	**A(65)**		**F1(32)**	**F2(15), F3(5)**
165	**B(55)**	**A(65)**		**F2(15)**	**F3(5), F1(35)**
170	**B(60)**	**A(65)**		**F3(5)**	**F1(35), F2(20)**
174	**A(65)**	**B(62)**		**F3(7)**	**F1(35), F2(20)**
184	**B(62)**	**A(75)**		**F3(7)**	**F1(35), F2(20)**
185	**B(65)**	**A(75)**		**F1(35)**	**F2(20), F3(10)**
190	**B(70)**	**A(75)**		**F2(20)**	**F3(10), F1(40)**
194	**A(75)**	**B(72)**		**F2(22)**	**F3(10), F1(40)**
224	**B(72)**	**A(85)**		**F2(22)**	**F3(10), F1(40)**

Processus A accède l'UCT en premier F1 est le seul fil à t=0, et est affecté l'état Actif par la biblio. de fils.

Le SE commute les processus à tous les 10 u.t.

F2 arrive après que F1 roule pour 19 u.t. et est place dans la file d'attente exécutable

La biblio. de fils commute entre fils à tous les 5 u.t. d'exécution sur l'UCT.

F3 arrive après que F1 à rouler pour 27 u.t.

Processus B est bloqué quand F2 demande de l'E/S.

Processus A est bloqué quand il demande de l'E/S.

Processus B est affecté à UCT quand l'opération d'E/S pour F2 est complétée.

Processus A est place dans la file d'attente prêt lorsque son opération E/S est complétée.

* Les numéros entre parenthèses à côté de A, B, F1, F2, et F2 représentent le temps d'exécution relative (le temps que le processus/fil exécute sur l'UCT). Notez que ces temps ne sont incrémentés que lorsque le processus/fil est affecté à l'UCT.-

b) **Modèle un à un:** Maintenant considérez que le SE soit conscient des fils et appliqué les trios états non seulement aux processus mais aussi au fils dans un processus multi-fils. Aucune bibliothèque de fils n'est impliquée dans l'ordonnancement et la gestion des fils d'un processus. Prenez pour acquis que le SE affecte l'UCT à un processus/fil pendant un maximum de 10 u.t. avant de commuté l'UCT à un autre processus/fils. Complétez le tableau suivant pour montré comment les fils du processus B et le processus A sont déplacés entre les files d'attente Prêt et Bloqué et l'état En exécution. Comme avant, une discipline de file d'attente premier-arrivé premier-servi est utilisé avec les files d'attente Prêt et Bloqué. Comme dans le cas (a), le fil F2 arrive après que F1 ait exécuté 19 u.t. et F3 arrive après que F1 ait exécuté 27 u.t. (i.e. F1 démarre les fils F2 et F3).

	UCT	**Système d'exploitation**	
Temps	En exécution	File d'attente Prêt	File d'attente Bloqué
0	A(0)	F1(0)	
10	F1(0)	A(10)	
20	A(10)	F1(10)	
30	F1(10)	A(20)	
39	F1(19)	A(20),F2 (0)	
40	A(20)	F2(0), F1(20)	
50	F2(0)	F1(20), A(30)	
60	F1(20)	A(30), F2(10)	
67	**F1(27)**	**A(30), F2(10), F3(0)**	
70	**A(30)**	**F2(10), F3(0), F1(30)**	
80	**F2(10)**	**F3(0), F1(30), A(40)**	
82	**F3(0)**	**F1(30), A(40)**	**F2(12)**
92	**F1(30)**	**A(40), F3(10)**	**F2(12)**
102	**A(40)**	**F3(10), F1(40)**	**F2(12)**
112	**F3(10)**	**F1(40), A(50)**	**F2(12)**
122	**F1(40)**	**A(50), F3(20)**	**F2(12)**
132	**A(50)**	**F3(20), F1(50)**	**F2(12)**
134	**A(52)**	**F3(20), F1(50), F2(12)**	
137	**F3(20)**	**F1(50), F2(12)**	**A(55)**
146	**F1(50)**	**F2(12)**	**A(55) , F3(29)**
156	**F2(12)**	**F1(60)**	**A(55) , F3(29)**
166	**F1(60)**	**F2(22)**	**A(55) , F3(29)**
176	**F2(22)**	**F1 (70)**	**A(55) , F3(29)**
178	**F1(70)**		**A(55) , F3(29), F2(24)**
188	**F1(80)**		**A(55) , F3(29), F2(24)**
189	**F1(81)**	**A(55)**	**F3(29), F2(24)**
198	**A(55)**	**F3(29), F1(90)**	**F2(24)**

Processus A accède l'UCT en premier F1 est le seul fil à t=0, et est placé dans la file d'attente Prêt.

Fil F2 est créer après que F1 ay roulé pour 10 u.t. et est place dans la file d'attente.

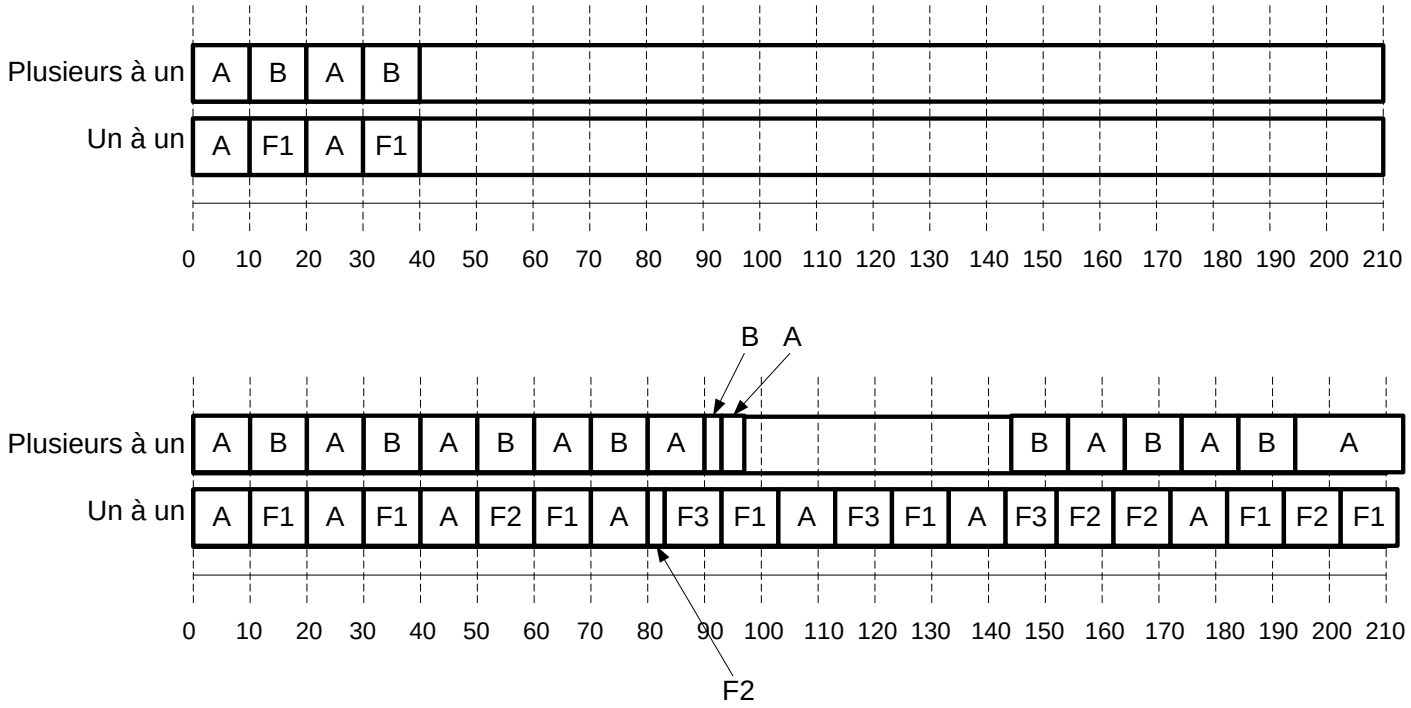
F2 est place dans la file d'attente Bloqué à sa demande d'E/S

F2 est place dans la file d'attente Prêt quand son opération E/S est complétée

est place dans la file d'attente Bloqué à sa demande d'E/S.

A est replacé dans la file prêt lorsque l'entrée/sortie est complétée.

c) Sur le diagramme suivant, montrez quels processus ou fils sont alloué à l'UCT pour les deux cas de (a) et (b). Notez que la perspective présenté est celle du SE – et donc, dans le cas du modèle plusieurs à un, indiquez quelles processus sont affecté à l'UCT, tandis que dans le cas du modèle un à un, montrez comment les fils du processus B sont affectés à l'UCT. Commentez sur la différence de niveau de concurrence entre les deux modèles.



Le modèle un à un offre une meilleur concurrence puisque l'UCT est occupé durant toute la période d'observation. Ceci est du au fait que le SE peut faire rouler d'autres fils du Processus B lorsque le modèle un à un est utilisé. Dans le modèle plusieurs à un, un fil qui bloque, bloque aussi tout le processus, et ainsi tous les fils dans le processus.