IF-3-SYS : Systèmes d'exploitation 31 mai 2022

NOM Prénom :

Consignes

- L'examen dure 1h30. Prenez le temps de lire le sujet en entier (18 questions sur 9 pages)
- Écrivez lisiblement et surtout sans ratures. Utilisez un brouillon (vraiment).
- Les réponses seront à inscrire sur le sujet. Commencez par écrire votre nom ci-dessus.
- Documents et appareils interdits, sauf une feuille A4 recto-verso manuscrite.
- Pour le binaire et l'hexadécimal, aidez-vous des tableaux page 9.

1 Questions de cours

Dans cette première partie, les questions sont indépendantes les unes des autres. Attention, dans les questions vrai/faux, les erreurs sont décomptées : ne répondez pas au hasard.

Question 1 Pour chaque acronyme ci-dessous, donnez sa signification en toutes lettres :

INSA	Institut National des Sciences Appliquées
FCFS	
ISR	
MMU	
PID	

Question 2 Pour chacune des affirmations ci-dessous, entourez V si elle est correcte ou entourez F si elle est incorrecte et/ou absurde.

V F L'appel `exec()` permet de créer un nouveau processus pour exécuter un nouveau programme.
V F Pour savoir ce que l'utilisateur tape sur le clavier, le shell doit faire un *appel système*.
V F Seul le noyau (et aucun processus) est autorisé à exécuter des instructions dites *privilégiées*.
V F Un processus peut consulter sa propre table de pagination, mais seul le noyau peut la modifier.

Question 3 Donnez un exemple d'appel système qui se comporte comme une fonction dont on ne revient jamais :

Question 4 Si on exécute le programme ci-dessous, combien affichera-t-il de lignes, et avec combien de numéros distincts ?

```
1 void main() {
2    int n = 2;
3    while(n > 0) {
4       fork();
5       n = n-1;
6       printf("%d \n", getpid());
7       fork();
8    }
9 }
```

Ce programme affichera

[ ] numéros en tout, dont

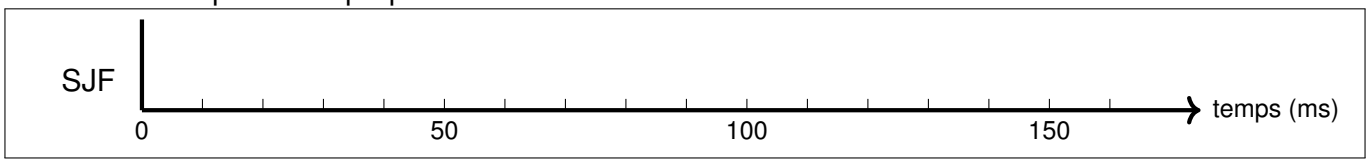
[ ] numéros différents.

Question 5 Il peut arriver, que ce soit par maladresse ou par malveillance, qu'un programme tombe dans une boucle infinie et qu'il exécute sans arrêt les mêmes instructions, sans plus jamais faire d'appel système. Dans ces conditions, comment peut-on l'empêcher de monopoliser le CPU ? Présentez les différents mécanismes matériels et logiciels nécessaires.

Question 6 On suppose dans cette question un ordonnanceur appliquant la stratégie SJF (Shortest Job First). On s'intéresse à quatre tâches purement calculatoires A à D décrites par le tableau ci-dessous. Les temps sont indiqués en millisecondes.

tâche	A	B	C	D
arrivée	0	20	30	70
durée	60	40	30	20

Au brouillon, dessinez un chronogramme indiquant la succession des tâches sur le processeur. Recopiez ensuite votre réponse au propre dans le cadre ci-dessous.



Question 7 Pour chacune des affirmations ci-dessous, entourez V si elle est correcte ou entourez F si elle est incorrecte et/ou absurde.

V F Grâce à `mmap()` il est possible de lire le contenu d'un fichier sans jamais invoquer `open()`.
V F Grâce à `mmap()` il est possible de lire le contenu d'un fichier sans jamais invoquer `close()`.
V F Grâce à `mmap()` il est possible de lire le contenu d'un fichier sans jamais invoquer `read()`.
V F Grâce à `mmap()` il est possible de lire le contenu d'un fichier sans jamais invoquer `write()`.

Question 8 On s'intéresse dans cette question à un système de mémoire virtuelle paginée. Au cours de l'exécution d'un programme, on observe la MMU faire les traductions d'adresses suivantes (en notation hexadécimale) :

3E1F→7E1F, 5454→9454, 2A8D→6A8D, E127→2127, C8BA→08BA, 9762→5762, 72CF→A2CF.

Complétez chaque phrase ci-dessous par une taille exprimée en kio, Mio, Gio (cf page 9). Si les informations données ne permettent pas de répondre, indiquez «inconnue».

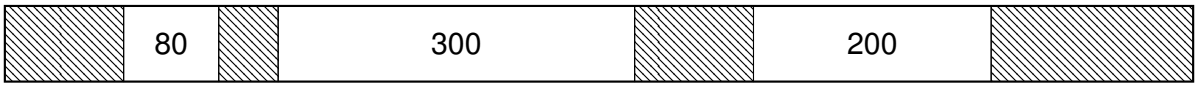
— La taille des pages virtuelles est

— La taille des pages physiques est

— La taille maximale d'un processus est

— La quantité de RAM disponible sur la machine est

Question 9 On s'intéresse dans cette question aux différentes stratégies d'allocation dynamique vues en cours, c'est à dire first-fit, best-fit et worst-fit. L'état initial du tas est illustré ci-dessous : les blocs occupés sont hachurés, la free-list comporte uniquement trois blocs libres de taille 80, 300, et 200 (dans cet ordre).



L'application demande successivement l'allocation de trois blocs : A de taille 150, puis B de taille 200, et enfin C de taille 50. Pour chaque stratégie, dessinez au brouillon l'état du tas après le traitement des trois requêtes. Indiquez en particulier l'emplacement de A, B, et C, ainsi que la taille des blocs encore libres. Si une stratégie ne permet pas de satisfaire les trois allocations, indiquez seulement «échec». Finalement, recopiez vos résultats au propre dans les cadres ci-dessous.

Hypothèses :

— On suppose qu'un bloc de taille N peut servir à allouer une zone de taille T $\leqslant$ N, laissant le cas échéant un bloc libre de taille $N - T$. Autrement dit, on néglige l'espace occupé par les méta-données.

— Les tailles de blocs libres ou occupés peuvent être des entiers quelconques : pas de restriction aux multiples de 8.

Réponses :

First-fit

Best-fit

Worst-fit

Question 10 Complétez le code de la fonction ci-dessous pour qu'elle renvoie le *pos*-ième bit de la représentation binaire de `value`. Le paramètre `pos` indique la *position* du bit souhaité, en comptant à partir de zéro par la droite. Par exemple, si `value=39` (qui s'écrit en binaire `100111`) vous devrez renvoyer 1 pour `pos=0,1,2` ou `5`, et renvoyer zéro partout ailleurs.

```
unsigned int getbit(unsigned int value, unsigned int pos)
{




}
```

Question 11 Pour chacune des affirmations ci-dessous, entourez V si elle est correcte ou entourez F si elle est incorrecte et/ou absurde.

V F Les différents threads d'un même processus ont la même pile d'exécution.
V F Les différents threads d'un même processus ont la même table de pagination.
V F Les différents threads d'un même processus ont le même CPU.
V F Les différents threads d'un même processus ont les mêmes sections critiques.

Question 12 Donnez trois exemples de commandes shell qui prennent typiquement en argument un chemin vers un répertoire.

Question 13 Comment reconnaît-on un *chemin relatif* d'un *chemin absolu* ? Donnez un exemple de chaque.

Question 14 On s'intéresse dans cette question à un disque dur d'une capacité de 4 Tio, formaté avec des secteurs de 4096 octets. Combien faut-il de bits, au minimum, pour encoder un numéro de secteur ? Pour les calculs en binaire, vous pouvez vous aider du tableau page 9.

2 Problème : synchronisation par sémaphores

Dans ce problème, on s'intéresse à un programme concurrent similaire à ceux vus en cours et en TD : plusieurs threads se partagent l'accès à une ressource commune qu'on appellera *le fichier*. Il y a deux catégories de threads : les *lecteurs* qui accèdent au fichier uniquement en lecture, et les threads *rédacteurs* qui peuvent également en modifier le contenu. Plusieurs lecteurs peuvent lire le fichier simultanément sans que cela pose de problème de synchronisation. Par contre, pendant qu'un rédacteur est en cours de modification, le contenu du fichier n'est pas cohérent et on veut donc empêcher tous les autres threads (lecteurs *et* rédacteurs) d'y accéder.

Autrement dit, on s'intéresse à un programme dans lequel une quantité arbitraire de threads exécutent chacun l'un ou l'autre des comportements ci-dessous, et on veut garantir les trois contraintes de synchronisation suivantes :

— il est permis à plusieurs lecteurs de `lire()` simultanément
— il est interdit à plusieurs rédacteurs de `modifier()` simultanément
— il est interdit aux lecteurs de `lire()` pendant qu'un rédacteur est en train de `modifier()`

threads lecteurs

```
while(true)
{
  // A?
  lire();
  // B?
}
```

threads rédacteurs

```
while(true)
{
  // C?
  modifier();
  // D?
}
```

Dans les questions suivantes, votre travail va consister à implémenter les synchronisations (points notés A, B, C et D) à l'aide de sémaphores et/ou de variables partagées.

Question 15 Une approche simpliste est de restreindre l'accès au fichier à un seul thread au maximum, sans tenir compte des deux catégories de threads. Implémentez ci-dessous les synchronisations nécessaires.

Remarques :

- n'oubliez pas de préciser la valeur initiale de vos sémaphores et/ou variables partagées.
- vous pouvez utiliser autant de code et/ou d'appels à P() et/ou à V() que vous jugerez utile.
- vous pouvez aussi laisser un cadre vide pour dire «aucune synchronisation n'est nécessaire ici».

Conditions initiales

lecteur : synchro A

rédacteur : synchro C

lecteur : synchro B

rédacteur : synchro D

N'hésitez pas à ajouter des commentaires pour expliquer votre démarche :

Question 16 Vous remarquez que cette solution simpliste est certes correcte en termes de synchronisation, mais qu'elle aura probablement des performances assez mauvaises. Proposez une nouvelle solution qui autorise des lecteurs multiples à accéder simultanément au fichier.

Conditions initiales

lecteur : synchro A

rédacteur : synchro C

lecteur : synchro B

rédacteur : synchro D

N'hésitez pas à ajouter des commentaires pour expliquer votre démarche :

Question 17 Vous remarquez que cette nouvelle solution souffre d'un risque de famine. En réalité, ce sont nos trois contraintes de synchronisation (cf page 5) qui, à elles seules, sont insuffisantes pour garantir un comportement «raisonnable» du système.
Décrivez un scénario d'exécution dans lequel une situation de famine se produit.

Question 18 Proposez une nouvelle solution évitant ce risque de famine.

Conditions initiales

lecteur : synchro A

rédacteur : synchro C

lecteur : synchro B

rédacteur : synchro D

Annexe : aide pour les calculs en binaire

Les premiers nombres entiers, notés en décimal, hexadécimal, et binaire :

Dec	Hex	Bin
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100

Dec	Hex	Bin
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001

Dec	Hex	Bin
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110

Dec	Hex	Bin
15	F	1111
16	10	10000
17	11	10001
18	12	10010
19	13	10011

Les premières puissances de 2, notées en décimal :

$2^0 = 1$	$2^{16} = 65\,536$	$2^{32} = 4\,294\,967\,296$	$2^{48} = 281\,474\,976\,710\,656$
$2^1 = 2$	$2^{17} = 131\,072$	$2^{33} = 8\,589\,934\,592$	$2^{49} = 562\,949\,953\,421\,312$
$2^2 = 4$	$2^{18} = 262\,144$	$2^{34} = 17\,179\,869\,184$	$2^{50} = 1\,125\,899\,906\,842\,624$
$2^3 = 8$	$2^{19} = 524\,288$	$2^{35} = 34\,359\,738\,368$	$2^{51} = 2\,251\,799\,813\,685\,248$
$2^4 = 16$	$2^{20} = 1\,048\,576$	$2^{36} = 68\,719\,476\,736$	$2^{52} = 4\,503\,599\,627\,370\,496$
$2^5 = 32$	$2^{21} = 2\,097\,152$	$2^{37} = 137\,438\,953\,472$	$2^{53} = 9\,007\,199\,254\,740\,992$
$2^6 = 64$	$2^{22} = 4\,194\,304$	$2^{38} = 274\,877\,906\,944$	$2^{54} = 18\,014\,398\,509\,481\,984$
$2^7 = 128$	$2^{23} = 8\,388\,608$	$2^{39} = 549\,755\,813\,888$	$2^{55} = 36\,028\,797\,018\,963\,968$
$2^8 = 256$	$2^{24} = 16\,777\,216$	$2^{40} = 1\,099\,511\,627\,776$	$2^{56} = 72\,057\,594\,037\,927\,936$
$2^9 = 512$	$2^{25} = 33\,554\,432$	$2^{41} = 2\,199\,023\,255\,552$	$2^{57} = 144\,115\,188\,075\,855\,488$
$2^{10} = 1024$	$2^{26} = 67\,108\,864$	$2^{42} = 4\,398\,046\,511\,104$	$2^{58} = 288\,230\,376\,151\,711\,744$
$2^{11} = 2048$	$2^{27} = 134\,217\,728$	$2^{43} = 8\,796\,093\,022\,208$	$2^{59} = 576\,460\,752\,303\,423\,488$
$2^{12} = 4\,096$	$2^{28} = 268\,435\,456$	$2^{44} = 17\,592\,186\,044\,416$	$2^{60} = 1\,152\,921\,504\,606\,846\,976$
$2^{13} = 8\,192$	$2^{29} = 536\,870\,912$	$2^{45} = 35\,184\,372\,088\,832$	$2^{61} = 2\,305\,843\,009\,213\,693\,952$
$2^{14} = 16\,384$	$2^{30} = 1\,073\,741\,824$	$2^{46} = 70\,368\,744\,177\,664$	$2^{62} = 4\,611\,686\,018\,427\,387\,904$
$2^{15} = 32\,768$	$2^{31} = 2\,147\,483\,648$	$2^{47} = 140\,737\,488\,355\,328$	$2^{63} = 9\,223\,372\,036\,854\,775\,808$
			$2^{64} = 18\,446\,744\,073\,709\,551\,616$

On rappelle également que :

- 1 kio = 1024 octets,
- 1 Mio = 1024 Kio,
- 1 Gio = 1024 Mio,
- 1 Tio = 1024 Gio,
- etc. (avec dans l'ordre : Pio, Eio, Zio, Yio)

En cas de doute sur ces unités, n'hésitez pas à demander des précisions.