Licence 2 — Info4B

Examen du Mercredi 4 Mai 2022

Durée 2h • Documents autorisés : une feuille A4 recto-verso

Le barème est donné à titre indicatif

Si un exercice vous conduit à faire des hypothèses, indiquez-les clairement sur votre copie.
Rédigez et justifiez précisément les réponses aux questions.

Exercice 1 - 6 pts

Les réponses aux questions de cet exercice sont limitées à 1/4 page de texte par question (hors schémas et/ou extraits de code).

1. Quel est l'intérêt d'utiliser un *buffer* (ou tampon) pour la lecture de fichiers ? Donner un exemple d'ouverture de flux sur un fichier en Java en lui associant un *buffer*.
2. Définir la notion de *socket*, expliquer son rôle et sa relation avec la notion de port. Montrez qu'avec un même port serveur et des threads qui gèrent les connexions de multiples clients, le noyau du système d'exploitation ou la machine virtuelle Java peuvent diffuser facilement les données envoyées par un client au thread concerné.
3. Plusieurs objets de la classe `MyThread` définie ci-dessous sont instanciés et démarrés en tant que threads. Le programmeur a choisi une construction de type `synchronized(o){...}` pour protéger le compteur `cpt`. Commenter les éventuels problèmes de concurrence que cela peut engendrer. Expliquer ce qu'il aurait dû éviter et en tirer une règle de type bonne pratique.

```
1  class MyObject {
       public Object lock;
3      public int cpt;
4
5      public MyObject() {
6          this.lock = new Object();
7          this.cpt = 0;
8      }
9  }
10
11 class MyThread extends Thread {
12     private MyObject myObject;
13
14     public MyThread(MyObject myObject) {
15         this.myObject = myObject;
16     }
17
18     @Override
19     public void run() {
20         for (int i = 0; i < 1000; i++) {
21             synchronized(myObject.lock) {
22                 myObject.lock = new Object();
23                 myObject.cpt += 1;
24             }
25         }
26     }
27 }
```

Exercice 2 - 6 pts

On considère une mémoire de 12 pages (numérotées de 0 à 11) comportant 6 pages de mémoire en RAM. Chaque page a une taille de 1024 octets ayant des adresses de 0 à 1023.

1. Faire un schéma de l'organisation de la mémoire, préciser la limite de la mémoire physique, numéroter les pages et donner les adresses des limites de chaque page.

2. On suppose que le mécanisme de *swap* utilise une stratégie LRU (*Least Recently Used*), c'est-à-dire moins récemment utilisée. Pour mettre en place cette stratégie le noyau du système d'exploitation possède une table qui, pour chaque page de la mémoire centrale, enregistre la date du dernier accès. Cette table a donc dans notre cas 6 éléments.

   Donner les différents états de la mémoire en fonction des demandes d'accès mémoire de la table 1. Si une valeur de page n'apparaît pas c'est qu'il n'y a pas eu encore de demande. On suppose que dans l'état initial la mémoire physique contient les pages présentes dans la fiche jointe, c'est-à-dire (2,4,1,3,5) et entrées dans cet ordre (2 est la page la plus anciennement utilisée).
3. Effectuer la simulation :
   (a) Remplir l'annexe, ajouter une structure de données qui permet d'enregistrer la date des accès (top) des pages chargées dans la RAM.
   (b) Pour chaque accès (table 1), remplir le tableau en annexe correspondant afin de donner le couple numéro de page, offset et le couple correspondant à la page placée en mémoire physique.
   (c) Quel est le nombre de défauts de page ?

top d'accès	1	2	3	4	5	6	7
accès adresse	815	1040	1 410	2 300	6 410	2 123	1530
top d'accès	8	9	10	11	12	13	14
accès adresse	1 603	6 400	7 200	410	2 538	6 146	8 436

TABLE 1 – Demandes d'accès à la mémoire

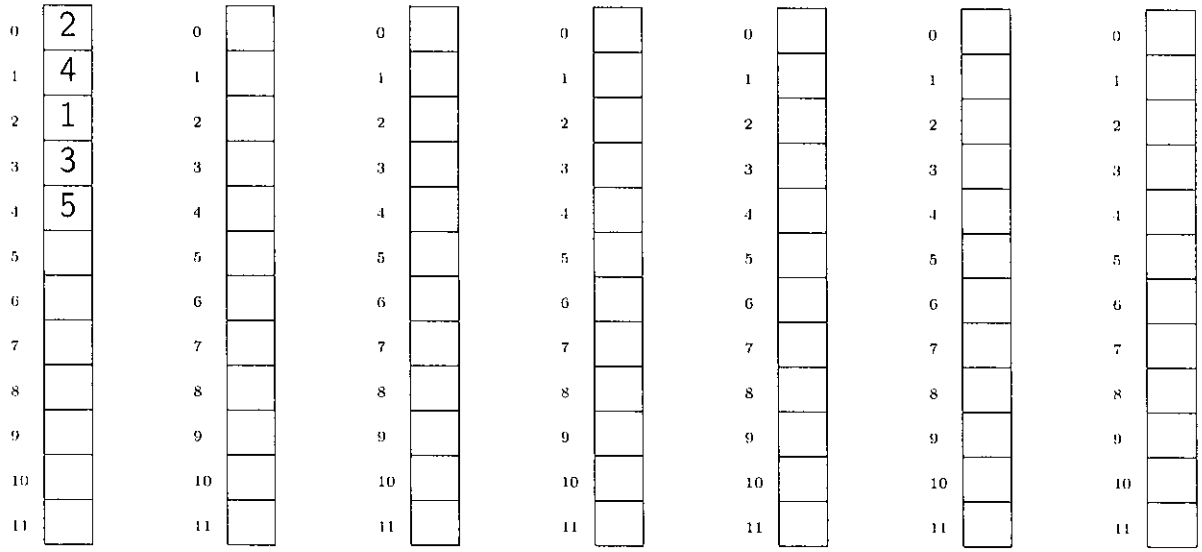
Exercice 3 - 8 pts

On souhaite réaliser une application collaborative pour le recensement d'oiseaux. Pour cela, les utilisateurs envoient leurs observations au serveur au moyen de la partie cliente de l'application. Les données d'observation possèdent plusieurs attributs : la date, les coordonnées GPS sous forme de chaîne de caractères, l'espèce de l'oiseau et le nombre d'individus observés. Le serveur met également les données globales à disposition de tous les utilisateurs. Ainsi, depuis l'application cliente, les utilisateurs peuvent accéder aux données à l'aide de requêtes (par exemple, connaître l'espèce des oiseaux qui sont le plus vus, ceux qui sont rares, leur localisation, restreindre sur une date ou sur une espèce d'oiseau). Le serveur doit pouvoir supporter plusieurs enregistrements d'observations ainsi que plusieurs requêtes simultanément. De plus, il doit être capable de fournir rapidement les informations aux utilisateurs.

1. Quel type de socket allez-vous utiliser pour envoyer des observations ? Même question pour que les clients récupèrent des données ? Justifier votre réponse.
2. Écrire le squelette de la classe `Observation` ainsi que le corps de la méthode du client qui permet d'envoyer un objet `Observation` au serveur.
3. Définir la stratégie utilisée par le serveur pour gérer les données des observations, les organiser, les sauvegarder. Utiliser des tables de hachage : les déclarer et montrer comment insérer et rechercher des données.
4. Écrire la classe du serveur qui permet de réceptionner une demande de données d'un client, de rechercher les données concernées en utilisant les structures définies à la question précédente, puis de renvoyer ces données au client. Le serveur doit pouvoir prendre en charge plusieurs clients et on suppose que les recherches sont prédéfinies.
5. Pour prendre en compte de nombreux utilisateurs un seul serveur ne suffit plus. Comment modifier le système pour gérer plusieurs serveur mais toujours permettre d'avoir une vision globale des données ? Proposer deux pistes de solution et expliquer comment les serveurs communiqueront.

Numéro d'anonymat :

Compléments à remplir pour l'exercice 2.



Correspondances mémoire virtuelle, mémoire et adresses réelles

Demande	Adresse	Page	Offset	Page RAM	Adresse réelle
1	815	0	815	5	5 935
2	1 040				
3	1 410				
4	2 300				
5	6 410				
6	2 123				
7	1 530				
8	1 603				
9	6 400				
10	7 200				
11	410				
12	2 538				
13	6 146				
14	8 436				