Licence 2 — Info4B
Examen du Mercredi 3 Mai 2023
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

1. Quels sont les éléments nécessaires pour réaliser un système d'exploitation multi-tâches préemptif ? En citer au moins 3 (dont un de niveau matériel) et expliquer leur rôle.
2. Écrire un programme Java pour créer deux threads d'une même classe. Ces deux threads ont accès à une ressources et incrémentent, d'une valeur passée en paramètre, une variable compteur située dans la ressource. Il ne doit pas y avoir de problème de concurrence lors de l'exécution.

Exercice 2 - 7 pts

On considère une mémoire virtuelle composée de 16 pages (numérotées de 0 à 15) comportant 8 pages mémoire logées en RAM (pages 0 à 7). Chaque page a une taille de 1024 octets (adresses de 0 à 1023).

1. Expliquer le fonctionnement de la mémoire virtuelle en incluant le rôle de la MMU, du *swap* et du noyau du système d'exploitation.
2. Faire un schéma de l'organisation de la mémoire, préciser la limite de la mémoire physique, et identifier les pages stockées sur le disque. Numéroter les pages et donner les adresses des limites de chaque page (limite basse, limite haute).
3. On suppose que le noyau du système d'exploitation dispose d'une stratégie d'allocation de mémoire non contiguë. Il réalise les allocations mémoire par page : une page est allouée à un seul processus. Le mécanisme de *swap* utilise une stratégie LRU (*Least Recently Used*), c'est-à-dire qu'il décharge prioritairement la page la moins récemment utilisée.

   Pour la gestion de la mémoire virtuelle, le noyau du système d'exploitation possède une table qui, pour chaque page de la RAM, enregistre la date du dernier accès et une autre avec, pour chaque page, le processus pour qui elle a été allouée.

   Les deux tableaux suivants donnent les allocations des pages aux processus et les accès mémoire. Lorsqu'un processus essaye d'accéder à une page qui ne lui appartient pas, ou en dehors des adresses autorisées, alors un signal d'erreur de segmentation est envoyé au noyau, le processus s'arrête et ses pages mémoire sont libérées.

   Donner les différents états de la mémoire en fonction des demandes d'accès mémoire de la table 2. 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 (11,7,2,4,1,3,5,6) et entrées dans cet ordre (11 est la page la plus anciennement utilisée).

   (a) Effectuer la simulation : remplir les colonnes vides de l'annexe correspondant aux différents états de la mémoire virtuelle.

   (b) Pour chaque accès (table 2), remplir le tableau en annexe correspondant afin de donner le couple numéro de page, offset et l'adresse physique accédée correspondant à la page placée en RAM.

   (c) Quel est le nombre de défauts de page ?

Processus	P_1	P_2	P_3	P_4
Pages allouées	1,4,5,11,12	2,3,6	13,14	8,9,10

TABLE 1 – Pages allouées à chaque processus

Top d'accès	1	2	3	4	5	6	7
Accès adresse	4 099	13 001	5 530	13 320	3 001	1034	5 131
Processus	P_1	P_1	P_1	P_3	P_2	P_1	P_1
Top d'accès	8	9	10	11	12	13	14
Accès adresse	14 350	14 441	8 195	10 243	6 191	7 101	9 218
Processus	P_3	P_3	P_4	P_4	P_2	P_3	P_4

TABLE 2 – Demandes d'accès à la mémoire

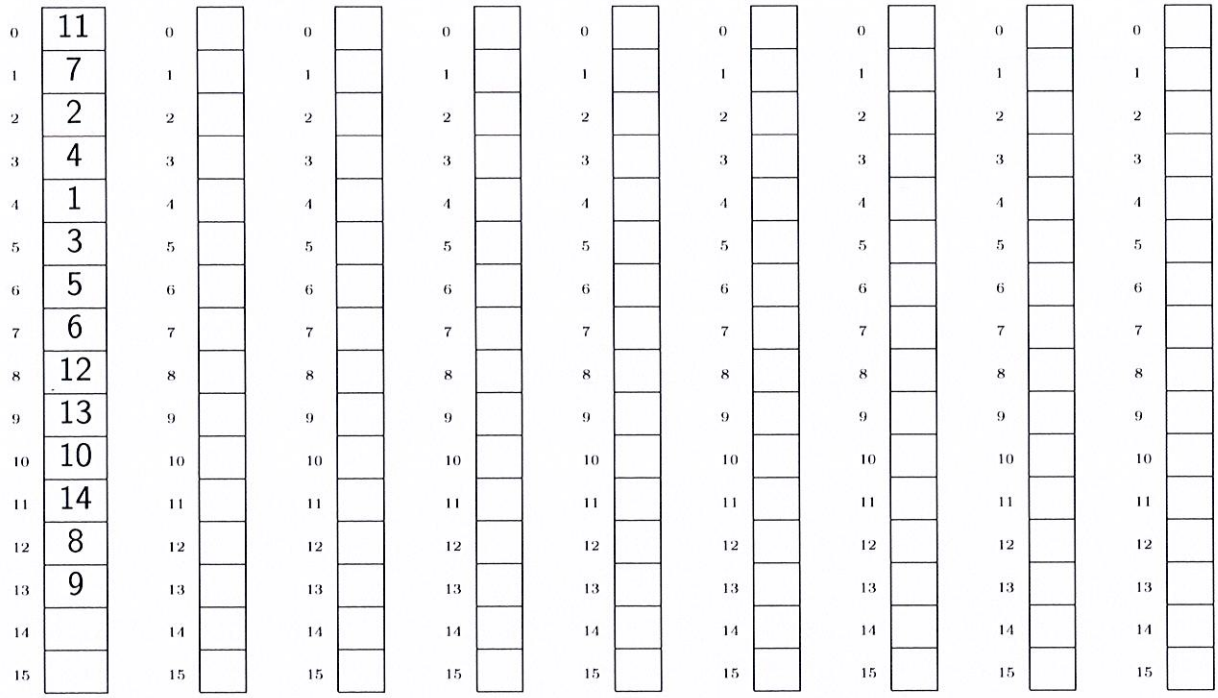
Exercice 3 - 7 pts

On souhaite réaliser une application de type coffre-fort numérique. Les clients, qui ont un compte utilisateur identifié par une adresse email et un login unique, envoient au serveur des fichiers que ce dernier sauvegarde sur ses disques. On ne s'intéresse pas à la partie chiffrement (cryptage) ni à l'authentification des utilisateurs. Les fichiers sont enregistrés sur le serveur dans des répertoires différents identifiés par le login de l'utilisateur. Tous les types de fichier peuvent être pris en charge. Un client peut demander à télécharger un fichier sauvegardé précédemment.

1. Faire un schéma du système faisant apparaître le serveur, plusieurs clients, les adresses IP et les ports utilisés. Quel type de socket allez-vous utiliser pour envoyer ou recevoir des fichiers ? Justifier votre réponse.
2. Écrire la portion de code du serveur qui attend les connexions des clients.
3. Écrire la méthode `run` du thread coté serveur qui reçoit le login, puis réceptionne un fichier et l'enregistre sur le serveur.
4. Écrire la portion de code du client qui permet de se connecter au serveur, lire le fichier dont le nom est passé en paramètre et envoyer son contenu au serveur.
5. Sur le serveur on enregistre dans deux hashtables les noms des fichiers et leur propriétaire. L'une doit permettre de rechercher les fichiers d'un propriétaire dont on connaît le login. L'autre doit permettre de vérifier qu'un nom de fichier appartient bien à la personne connectée. Définir une classe `Index` contenant les hashtables sachant qu'un objet de cette classe doit pouvoir être sauvegardé dans un fichier et qu'il faut protéger cette ressource des accès concurrents. Pour la classe `Index`, on demande la déclaration des membres de la classe (attributs et méthodes). Le corps des méthodes n'est pas à fournir.

Numéro d'anonymat :

Compléments à remplir pour l'exercice 2.



Correspondances mémoire virtuelle et adresses réelles

Demande	Processus	Adresse	Page	Offset	Page RAM	Adresse réelle
1		4 099	4	3	3	
2		13 001				
3		5 530				
4		13 320				
5		3 001				
6		1 034				
7		5 131				
8		14 350				
9		14 441				
10		8 195				
11		10 243				
12		6 191				
13		7 101				
14		9 218				