Université Joseph Fourier — DEUG STPI-2
1999-2000 — IUP MIAGE-1

**Examen de Systèmes d'Exploitation et Réseaux**
5 mai 2000

Durée: 3 heures. Documents autorisés : notes personnelles, documents distribués en cours.

---

***N.B.*** *L'examen comporte 4 parties indépendantes. Prière de lire attentivement l'énoncé et de respecter les notations du texte. La longueur de l'énoncé n'est pas un signe de difficulté mais est nécessaire pour bien spécifier les problèmes posés. La* ***clarté*** *et la* ***concision*** *des réponses, ainsi que leur* ***présentation matérielle****, seront des éléments importants d'appréciation.*

---

**1. Synchronisation de processus (5 points)**

Soit *f* un fichier. On note *v-excl(f)* l'opération de verrouillage exclusif de *f* (verrouillage en écriture), *v-part(f)* l'opération de verrouillage partagé, et *dev(f)* l'opération de déverrouillage. On rappelle que lorsqu'un fichier *f* est verrouillé en mode exclusif, un nouveau verrouillage de *f* en mode exclusif ou partagé provoque le blocage du processus qui l'exécute ; lorsqu'un fichier *f* est verrouillé en mode partagé, un nouveau verrouillage de *f* en mode partagé n'est pas bloquant, mais un nouveau verrouillage de *f* en mode exclusif est bloquant.

Soient deux fichiers *f1* et *f2*, et deux processus *p1* et *p2*. Dans chacun des 3 cas ci-dessous, ont demande de répondre aux questions suivantes

1) L'exécution en parallèle de *p1* et *p2* peut-elle conduire à leur blocage définitif (interblocage) ? préciser dans ce cas l'ordre exact d'exécution des instructions conduisant à l'interblocage.
2) En cas de réponse positive à la question a), peut-on modifier le programme d'un des processus pour éviter ce risque de blocage, tout en conservant l'effet global du programme ?

**Cas 1 :**

| ***p1*** | ***p2*** |
|---|---|
| *v-excl(f1)* | *v-excl(f2)* |
| *v-excl(f2)* | *v-excl(f1)* |
| *...* | *...* |
| *dev(f2)* | *dev(f1)* |
| *dev(f1)* | *dev(f2)* |

**Cas 2 :**

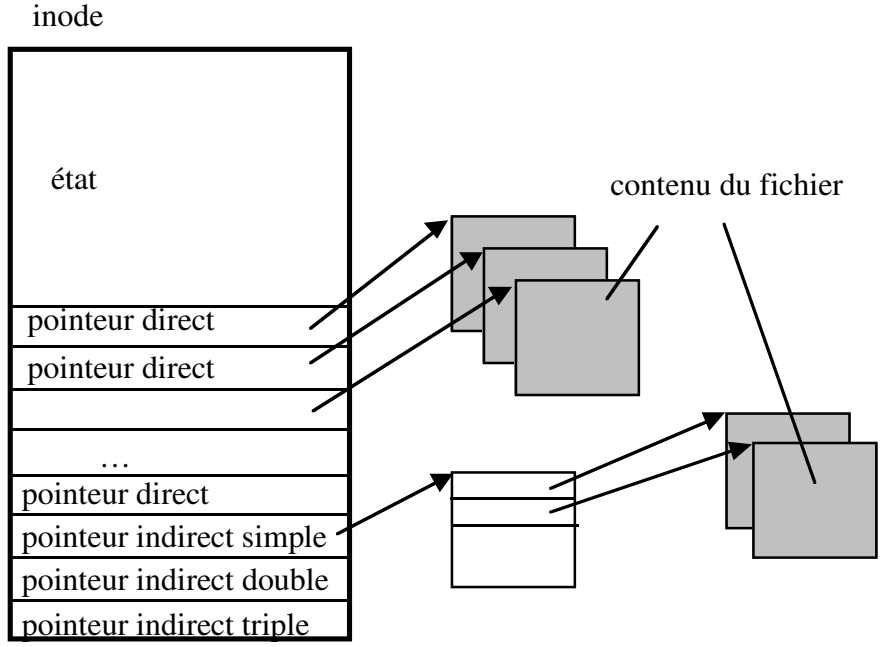
| ***p1*** | ***p2*** |
|---|---|
| *v-excl(f1)* | *v-excl(f2)* |
| *v-part(f2)* | *v-part(f1)* |
| *...* | *...* |
| *dev(f2)* | *dev(f1)* |
| *dev(f1)* | *dev(f2)* |

**Cas 3 :**

| ***p1*** | ***p2*** |
|---|---|
| *v-part(f1)* | *v-excl(f2)* |
| *v-excl(f2)* | *v-part(f1)* |
| *...* | *...* |
| *dev(f2)* | *dev(f1)* |
| *dev(f1)* | *dev(f2)* |

**2. Unix (4 points)**

1) On rappelle que, dans le SGF d'Unix, un fichier est représenté de façon interne par une structure de données appelée *inode*. Cette structure a une taille totale de 128 octets et comprend deux parties : la représentation de l'état (68 octets) et un ensemble de pointeurs dont chacun représente l'adresse d'un bloc sur disque. La taille d'un bloc est de 8192 octets. La taille d'un pointeur est de 4 octets. Les pointeurs comprennent : un ensemble de pointeurs directs, qui pointent vers des blocs contenant les premières données du fichier ; un pointeur indirect simple, qui pointe vers un bloc contenant des pointeurs vers les blocs suivants du fichier ; un pointeur indirect double, qui pointe vers un bloc contenant des pointeurs vers des blocs de pointeurs vers les blocs suivants du fichier ; et enfin un pointeur indirect triple (un degré de plus d'indirection).

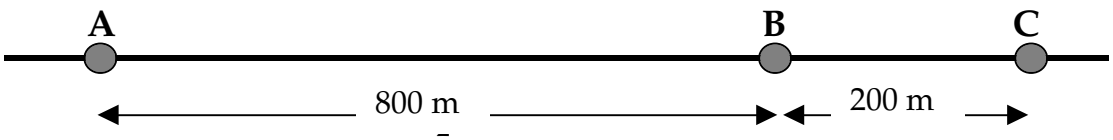



   a) Combien l'*inode* contient-il de pointeurs directs ?
   b) Quelle est la taille maximale de fichier que l'on peut représenter à l'aide de pointeurs directs uniquement ?
   c) Même question avec les pointeurs indirects simple, double et triple.

2) Rappeler ce qu'est un tube (*pipe*) dans le système Unix. Donner un exemple d'utilisation d'un tube. Indiquer les points communs et les différences entre un tube et un fichier ordinaire.

**3. Réseaux (6 points)**

1) On rappelle que dans le codage Manchester le bit 0 est représenté par une transition montante ⌊ et le bit 1 par une transition descendante ⌈
   a) Rappeler l'intérêt de cette représentation par raport à une représentation par niveaux.
   b) Représenter le codage de la suite 001111010.
2) On considère le réseau Ethernet représenté sur la figure ci-dessous. La vitesse de propagation sur le réseau est de $2.10^8$ m/s.



À l'instant t0, le nœud Cémet une trame à destination de A.

À l'instant t0 + 1 µs, le nœud A émet une trame à destination de B.

a) À quel moment et en quel point du réseau la collision aura-t-elle lieu ?

b) À quels moments la collision sera-t-elle respectivement détectée par A, par B et par C ?

c) Si on souhaite que la collision soit détectée alors que les stations concernées sont toujours en train d'émettre, quelle est la taille minimale de la trame, sachant que le débit est de 100 Mbit/s ?

3) On considère un anneau à jeton comprenant 10 stations. L'anneau a une longueur de 200 km, la vitesse de propagation est de $2.10^8$ m/s., et le débit est de 100 Mbit/s. On suppose que la durée de détention du jeton par une station est limitée à 100 ms.

a) Pourquoi limite-t-on la durée de détention du jeton ?

b) Quelle est la taille maximale du message que peut émettre une station avant de relâcher le jeton ?

c) Quel est le temps d'attente d'une station entre deux émissions successives, dans le cas d'une charge nulle (aucune autre station n'émet) et dans le cas d'une charge maximale (toutes les stations veulent émettre un message au moins égal à la taille maximale) ?

**4. Client-serveur (5 points)**

1) Expliquer ce qu'on appelle schéma client-serveur. Quel est l'intérêt de cette organisation ?

2) Expliquer le principe de l'organisation interne d'un serveur.

3) On rappelle brièvement le principe de la réalisation de l'appel de procédure à distance dans le système Tcl-DP. Le système fournit 3 procédures :

dp_MakeRPCServer numero_de_porte : exécuté sur la machine serveur, crée un processus serveur qui se met en attente sur la porte indiquée.

dp_MakeRPCClient machine_hote numero_de_porte : exécuté sur la machine client, crée une liaison depuis le processus client vers le serveur indiqué ; renvoie une valeur (soit server) qui est la désignation locale de la liaison.

dp_RPC $server commande : demande au serveur identifié par la désignation locale server d'exécuter la commande commande passée en paramètre.

Écrire avec Tcl-DP les programmes d'un client et d'un serveur qui réalisent les fonctions décrites ci-après (le serveur pouvant servir plusieurs clients). Le serveur fournit les procédures suivantes :

| | |
|---|---|
| ajouter n | ajoute le nombre n à un total maintenu par le serveur (ce total est initialement nul) |
| courant | renvoie la valeur courante du total |
| n-appels | renvoie le nombre de consultations (appels de courant) reçues par le serveur depuis sa mise en route |