

17. La mise en mémoire cache et la pagination sont, en quelque sorte, des procédures très semblables, qui se caractérisent par deux niveaux de mémoire (la mémoire cache et la mémoire principale pour la première, et la mémoire principale et le disque pour la seconde). Dans ce chapitre, nous avons examiné des arguments en faveur de l'utilisation de petites et de grandes pages sur le disque. Les mêmes arguments s'appliquent-ils à la taille des lignes de la mémoire cache ?
19. Comparez les méthodes de la table de bits et de la liste d'espaces libres pour assurer le suivi de l'espace libre disponible sur un disque de 800 cylindres, dont chacun dispose de 5 pistes et de 32 secteurs. Combien d'espaces libres faut-il pour que la liste d'espaces libres atteigne la dimension de la table de bits ? Imaginez que l'unité d'allocation soit les secteur et que chaque espace libre implique une entrée de 32 bits dans une table.
20. Un modèle d'allocation d'espace de stockage aide à prédire les performances d'un disque. Supposez que le disque soit un espace d'adressage linéaire avec un très grand nombre  $N$  de secteurs composé d'une série de blocs de données, puis d'une autre, etc. Si les mesures empiriques montrent que les distributions de probabilités concernant la taille des espaces libres et des zones de données sont identiques, la probabilité qu'on les deux d'occuper  $n$  secteurs étant  $2^{-n} = (1/2)^n$ , quel est le nombre d'espaces libres attendu sur le disque ?

[Réponse détaillée] Le calcul utilise des approximations. On calcule d'abord la taille d'un trou moyen. Un trou a taille  $n$  secteurs avec probabilité  $(1/2)^n$  et donc la moyenne de taille des trous (en secteurs) peut être approximée par la somme infinie

$$\sum_{n=1}^{\infty} n(1/2)^n = 1/2 \sum_{n=0}^{\infty} n(1/2)^{n-1} = x \sum_{n=0}^{\infty} nx^{n-1}$$

avec  $x = 1/2$ . Sachant que  $\sum_{n=0}^{\infty} x^n = \frac{1}{1-x}$  on déduit en dérivant de chaque côté

$$\frac{d}{dx} \left( \sum_{n=0}^{\infty} x^n \right) = \sum_{n=0}^{\infty} nx^{n-1} = \frac{d}{dx} \left( \frac{1}{1-x} \right) = \frac{d}{dx} (1-x)^{-1} = \frac{1}{(1-x)^2}$$

et donc  $(1/2) \sum_{n=1}^{\infty} n(1/2)^{n-1} = (1/2) \frac{1}{(1-1/2)^2} = 2$ . Les trous font en moyenne 2 secteurs ; s'il y a au total  $T$  trous, ils occupent  $2T$  secteurs. Comme données et trous occupent le même espace, il y a au total  $4T$  secteurs et sachant qu'il y a en fait  $N$  secteurs, on a  $4T = N$  et  $T = N/4$  : le nombre de trous est  $N/4$ .

22. Plusieurs études sur les différents systèmes de fichiers ont démontré que plus de la moitié des fichiers ne dépassent pas quelques Ko, la grande majorité d'entre eux ne dépassant pas les 8 Ko. D'un autre côté, la frange des 10 % des fichiers les plus volumineux occupe environ 95 % de l'espace disque disponible. À partir de cette observation, quelle conclusion tirez-vous concernant la taille des blocs du disque ?

**Sémaphores** Rappel : un sémaphore est un nombre entier non négatif; haut incrémente le sémaphore de 1, bas le décrémente. Une fois qu'une instruction a été initiée sur un sémaphore, aucun autre processus ne peut accéder à ce sémaphore. Figure 6.27 de la page 497 :

Instruction	Semaphore = 0	Semaphore > 0
haut	Semaphore = Semaphore + 1 ; Si un autre processus est arrêté dans l'attente de l'exécution d'un bas sur ce sémaphore il peut terminer son bas et continuer son exécution	Semaphore = Semaphore + 1
bas	Le processus s'arrête jusqu'à ce qu'un autre processus fasse un haut sur ce sémaphore	Semaphore = Semaphore - 1

23. Considérez la méthode suivante, selon laquelle un système d'exploitation implémente des instructions de sémaphore. Lorsque l'UC est sur le pont d'exécuter un appel haut ou bas sur un sémaphore (une variable entière au sein de la mémoire), il règle d'abord les bits de priorité, pour désactiver toute les interruptions. Il charge ensuite le sémaphore, le modifie, puis réactive les interruptions. Cette méthode fonctionne-t-elle en présence
- D'un seul processeur qui alterne d'un processus à l'autre toutes les 100 milli-secondes ?
  - De deux processeurs qui partagent une mémoire commune contenant le sémaphore ?
25. À partir des informations qui suivent, produisez une table indiquant les processus actifs et ceux bloqués, et à quel moment entre 0 et 1000 milli-secondes. Les trois processus (P1, P2 et P3) exécutent des instructions haut et bas sur le même sémaphore. Lorsque deux processus sont bloqués et qu'un haut est exécuté, le processus au numéro le plus bas est activé : P1 a donc priorité sur P2 et P3, etc. À l'origine, les trois processus sont actifs et le sémaphore est réglé sur 1.
- Au temps  $t=100$ , P1 exécute un bas
  - Au temps  $t=200$ , P1 exécute un bas.
  - Au temps  $t=300$ , P2 exécute un haut.
  - Au temps  $t=400$ , P3 exécute un bas.
  - Au temps  $t=500$ , P1 exécute un bas.
  - Au temps  $t=600$ , P2 exécute un haut.
  - Au temps  $t=700$ , P2 exécute un bas.
  - Au temps  $t=800$ , P1 exécute un haut.
  - Au temps  $t=900$ , P1 exécute un haut.
26. Dans un système de réservation, lorsqu'un processus est occupé avec un fichier, aucun autre processus ne doit utiliser le même fichier. Dans le cas contraire, deux processus opérant pour deux agences de voyage différentes pourraient, par exemple, vendre le dernier billet disponible pour le même vol. Développez une méthode de synchronisation qui repose sur les sémaphores et assure que les fichiers ne puissent être accédés que par un seul processus à la fois (en supposant que les processus obéissent bien aux règles imposées).