# Chapitre 10 : Entrées et sorties

10.1 Un numériseur traite une page de format lettre en couleur, avec une résolution de 600 pixels par pouce, en 5 secondes. L'image obtenue est compressée par un facteur 10 par rapport à l'image brute. Est-il possible de numériser un document et d'envoyer efficacement le résultat par réseau sur un réseau local à 100Mb/s?

Le numériseur produit 8.5 pouces * 600 p/pouce * 11 pouces * 600 p/pouce * 3 octets par pixel / 10 / 5s = 2019600 octet/s alors que la bande passante disponible est d'environ 12.5Mo/s. Cela ne pose donc pas de problème.

10.2 Un système utilise des entrées sorties calquées sur la mémoire et contient deux bus, un pour la mémoire et l'autre pour les entrées sorties. Le système envoie chaque accès mémoire sur le bus mémoire en premier et, en cas d'échec, l'envoie sur le bus des dispositifs d'entrée sortie. On vous propose de changer cette organisation afin de plutôt envoyer la requête sur les deux bus simultanément. Serait-ce avantageux?

L'idée est intéressante à première vue mais en fait le bus d'entrée sortie est normalement beaucoup plus lent et deviendrait le goûlot d'étranglement du système. En effet, il faudrait attendre après la demande d'accès courante avant de faire la demande d'accès suivante sur ce bus plus lent.

10.3 Un ordinateur peut accéder un mot en mémoire en 10ns. Lors d'une interruption, il doit sauver 32 registres tout usage, le compteur de programme et le registre de statut. Quel est le nombre maximal d'interruptions par seconde que ce système pourrait théoriquement soutenir?

Avec 34 registres à sauver puis à recharger à chaque interruption, il lui faut au moins 34 * 2 * 10ns = 680ns. Il est donc limité à un maximum de 1s / 680ns/Int. = 1.47MHz.

10.4 Un gestionnaire d'interruption doit envoyer le prochain caractère à l'imprimante et réactiver les interruptions. Est-ce qu'un ordre est préférable à un autre?

En envoyant le caractère immédiatement, cela occupe l'imprimante plus vite par rapport à si les interruptions étaient réactivées et une autre interruption en attente s'exécutait immédiatement. Par contre, le fait d'envoyer le caractère avant de réactiver les interruptions augmente légèrement la latence d'interruption.

10.5 Une imprimante peut traiter 6 pages de 50 lignes de 80 caractères par minute. L'envoi de chaque caractère sur le port parallèle par le CPU est très court. Est-ce efficace de faire ces transferts avec une interruption pour chaque caractère, considérant que l'interruption prend 50us?

Il y aura donc 6 p/m * 50 l/p * 80 c/l / 60 s/m = 400 c/s. Le temps passé en interruption sera donc de 400 * 50us = 20ms à chaque seconde, soit 2% du temps.

10.6 Dans quelle couche du système est-ce que chacune de ces opérations est normalement effectuée : a) calcul de la piste/secteur/tête d'une lecture, b) écriture d'une commande dans le registre d'un dispositif d'entrée sortie, c) vérification des droits d'accès pour un dispositif, d) conversion de données binaires en codes ACSII pour impression?

Pour a), ceci est effectué dans le pilote d'interface, possiblement dans une portion partagée entre plusieurs gestionnaires, si plusieurs types de disques ont une manière semblable de faire ce calcul. Le cas de b) est assurément dans le pilote d'interface, alors que c) est au niveau du système de fichiers virtuel dans le système d'exploitation. Finalement, d) est normalement effectué en mode usager.

10.7 L'envoi d'un paquet suit le chemin suivant sur un système : appel système, copie en mémoire système, copie vers la carte réseau, envoi à 10Mb/s, réception 1us après l'envoi, génération d'une interruption lorsque le paquet est complètement reçu, copie en mémoire système et copie en mémoire usager. Si chaque paquet fait 1024 octets, la copie de chaque octet prend 1us et l'interruption prend 1ms, quel est le débit maximal d'envoi entre deux processus si on attend l'accusé de réception (très petit) avant d'envoyer le paquet suivant?

L'envoi demande deux copies (2 * 1024 * 1us = 2.048ms), et l'envoi à 10Mb/s (1024 octets * 8 bits / octet / 10Mbit/s = .8192ms). Le transit est 1us. La réception prend une interruption (1ms) et deux copies (2.048ms). Le total est donc 2.048 + .8192 + .001 + 1 + 2.048 = 5.9162ms. L'accusé de réception prend environ le temps de l'interruption, le reste étant négligeable étant donné sa petite taille. Ceci donne donc 1024 octets / (5.9162ms + 1ms) = 148058 o/s.

10.8 De combien devrait-on décaler le début de chaque piste si la tête se déplace d'une piste en 1ms, le disque tourne à 7200rpm et chaque piste contient 200 secteurs de 512 octets?

En 1ms, le disque peut faire : 7200 t/m / 60s/m * 1ms = .12 t, soit .12 * 200 = 24 secteurs. En sautant environ 24 secteurs, le disque devrait pouvoir lire séquentiellement, sans perte de temps, deux pistes consécutives.

10.9 Un disque a une probabilité p de faire défaut pendant 1 heure. Quelle est la probabilité qu'une unité de disque RAID, avec k disques et pouvant tolérer la défaillance d'un disque, fasse défaut pendant une heure donnée?

La probabilité que tous les disques soient opérationnels est $(1 – p)^k$ et la probabilité qu'un des disques soit brisé est $k\ p(1 – p)^{k-1}$. La probabilité de défaillance sera donc $1 - (1 – p)^k - k\ p(1 – p)^{k-1}$.

10.10 Les requêtes arrivent pour accéder les cylindres 10, 22, 20, 2, 40, 6 et 38. Si la tête prend 6ms par cylindre pour se déplacer, quel est le temps requis selon l'ordonnancement utilisé avec une position initiale à 20 : a) FIFO, b) le plus près, c) ascenseur qui est initialement en phase montante?

En a), nous aurons 20, 10, 22, 20, 2, 40, 6 et 38, soit des déplacements de 10 + 12 + 2 + 18 + 38 + 34 + 32 = 146 * 6ms = 876ms. Avec b), cela devient : 20, 20, 22, 10, 6, 2, 38, 40, soit un déplacement de 0 + 2 + 12 + 4 + 4 + 36 + 2 = 60 * 6ms = 360ms. Finalement, avec c) on obtient : 20, 20, 22, 38, 40, 10, 6, 2 soit un déplacement de 0 + 2 + 16 + 2 + 30 + 4 + 4 = 58 * 6ms = 348ms.

10.11 Le professeur louange la performance de l'algorithme de l'ascenseur utilisé dans le système d'exploitation Linux. Un étudiant fait un test avec un programme qui lit aléatoirement 10000 blocs bien répartis sur le disque. Il ne note aucune différence par rapport au même test fait avec un ordonnanceur de disque trivial, qui sert les requêtes dans leur ordre d'arrivée. Comment expliqueriez-vous cela?

Si le programme de test de l'étudiant utilise des lectures bloquantes, il n'est pas possible pour le système d'exploitation de voir plusieurs requêtes à la fois et d'en optimiser l'ordonnancement. Un bon test devrait soit utiliser plusieurs processus concurrents faisant des lectures ou un processus avec des lectures asynchrones.

10.12 Plusieurs versions de Unix utilisent 32 bits pour compter les secondes depuis le début de l'an 1970. Est-ce que cela posera problème? Quand?

Il y a environ 365.25 * 24 * 60 * 60 = 31557600s par an, soit 136 ans dans 4Gis ($2^{32}$secondes). Ceci mène en 1970 + 136 = 2106. Espérons que d'ici là les ordinateurs seront tous à 64 bits, ce qui devrait suffire pour la durée prévue du soleil et même bien au-delà.

10.13 Les concepteurs d'un système ont estimé que la souris peut se déplacer au maximum à 20cm/s. Si la résolution (déplacement minimal rapporté) de la souris est de .1mm et chaque déplacement est codé sur 3 octets, quel est le débit maximal que peut générer la souris?

Si la souris envoie 3 octets à chaque .1mm pour 20cm en une seconde, cela donne 200mm/s / .1mm * 3 octets = 6000 octets/s, ce qui ne présente pas une charge importante.

10.14 La consommation de puissance est une contrainte importante sur les ordinateurs portatifs. Ils essaient de sauver sur leur consommation dès que possible en éteignant certains dispositifs comme le disque dès qu'ils sont inutilisés un certain temps. Un utilisateur note que la consommation est sensiblement réduite lorsqu'il utilise son ordinateur en mode console (pas d'environnement graphique, seulement l'affichage de texte comme sur un terminal) plutôt qu'avec X windows et GNOME ou KDE. Comment expliquez-vous cela?

L'affichage graphique demande beaucoup plus de calcul du CPU ainsi que du GPU. De plus, les logiciels graphiques consomment beaucoup plus de mémoire, ce qui demande des lectures supplémentaires du disque, principalement lors du démarrage de ces grosses applications.

10.15 Sous Linux, l'utilitaire « file » retrouve le format d'un fichier. Comment fonctionne cet utilitaire? Que se passe-t-il en changeant l'extension du fichier?

L'utilitaire « file » lit le début du fichier et le compare avec une liste d'entêtes connus. Contrairement à l'explorateur Windows qui se base sur l'extension du fichier, l'utilitaire « file » retourne le type du contenu du fichier, peu importe son extension.

10.16 Dans un shell Linux, la variable d'environnement $PWD contient le chemin d'accès du répertoire courant. Utilisez cette variable pour construire un chemin absolu pour accéder au fichier « foo » se situant dans le répertoire parent.

« $PWD/../foo » : La variable $PWD contient le chemin absolu du répertoire courant. À partir de ce répertoire, la chaine « .. » indique le répertoire parent. Ensuite, le nom de fichier « foo » est ajouté. Chaque élément du chemin est séparé par « / ».

10.17 Pour renommer un fichier, il est possible d'utiliser l'opération « rename », ou de copier le fichier et ensuite effacer le fichier précédent. Quelle est la différence entre les deux méthodes?

La méthode basée sur la copie requiert le double de l'espace disque du fichier à renommer de manière temporaire. Tout le contenu du fichier est lu et réécrit. L'utilisation de « rename » change uniquement la chaine du nom du fichier et/ou son répertoire parent, ce qui ne nécessite pas d'espace supplémentaire ni de recopie, le contenu reste sur place, ce qui est beaucoup plus efficace.

10.18 Un même fichier est ouvert depuis deux processus. L'un deux ferme le fichier et le supprime. Que se passera-t-il dans l'autre processus? À quel moment l'espace occupé sera-t-il libéré?

Le fichier existe tant qu'il est référencé par au moins un processus. Ainsi, même si le fichier n'a plus de nom dans aucun répertoire après la suppression, son contenu n'est pas libéré et il continue d'être disponible pour les processus qui l'utilisent encore. Lorsque le fichier n'est plus ouvert par aucun processus, alors l'espace est libéré. Il faut faire attention à cet aspect, car il peut manquer d'espace sur un disque à cause de gros fichiers temporaires qui restent ouverts et qui ne sont pas visibles dans l'arborescence.

10.19 Sous Windows, l'outil système « Defrag » compacte les blocs de fichiers sur un système de fichier FAT ou NTFS. Pourquoi est-ce utile sur un disque dur? Est-ce que cela diminue la fragmentation interne et/ou externe?

Le compactage améliore la contiguïté des fichiers et diminue le nombre de déplacements de la tête de lecture d'un disque dur, ce qui diminue la latence d'accès et augmente le débit de lecture. Cependant, le compactage ne diminue pas la fragmentation externe, car le système de fichier est de type bloc, ni la fragmentation interne, car même si le fichier est formé de plusieurs blocs, seul le dernier est incomplet et cette situation ne change pas en réorganisant les blocs de manière contiguë.

10.20 Considérez une structure d'inode avec 4 pointeurs directs, un pointeur indirect de niveau 1 et un pointeur indirect de niveau 2. La taille des blocs est de 512 octets et celle des pointeurs est de 64 bits. Quelle est la proportion de l'espace qui est utilisé pour les données par rapport à l'ensemble des blocs destinés au fichier dans le cas d'un fichier de 64 Kio? Considérez que l'inode occupe un bloc.

Le nombre de blocs de données nécessaires pour stocker le fichier est 64 Kio / 0,5 Kio = 128 blocs. Les pointeurs directs stockent 4 blocs, ce qui reste 124 blocs à placer dans les autres niveaux. Le nombre de blocs adressables par le pointeur de niveau 1 est de 512 / 8 = 64, il reste donc 124 – 64 = 60 blocs à placer au niveau 2. Pour adresser 60 blocs depuis le pointeur indirect de niveau 2, il faut un bloc ayant une entrée vers un bloc qui contient 60 pointeurs vers les blocs de données. L'inode occupe 1 bloc, le premier niveau indirect nécessite un bloc et le second niveau indirect nécessite 2 blocs, pour un total de 4. La proportion utilisée pour les données est donc 128 / (128 + 4) = 97%.

10.21 Considérez la structure d'inode de la question précédente. Calculez le temps moyen d'accès à chacun des blocs de données pour un fichier de taille maximale et des accès aléatoires dans le cas où il n'existe aucune cache des blocs, et que la lecture d'un bloc nécessite un délai d = 6 ms.

Les pointeurs directs référencent 4 blocs. Le premier niveau indirect référence 64 blocs. Le second niveau indirect référence 64 * 64 = 4096 blocs. Ainsi, la taille maximale est de 4164 blocs. Le temps d'accès pour les blocs directs nécessite la lecture de l'inode puis le bloc de données (2*d), le premier niveau indirect consiste en l'inode, le bloc indirect puis le bloc de données (3*d), et le second niveau indirect consiste à un accès de plus que le précédent (4*d). Pour la taille maximale et un accès aléatoire, le temps moyen d'accès est donc : (2*d*4/4164) + (3*d*64/4164) + (4*d*4096/4164), ce qui donne environ 23,9 ms.

10.22 Décrivez les avantages et les inconvénients d'un lien symbolique par rapport à un lien dur. Dans quels cas chacun d'eux est-il utilisé?

Un lien symbolique permet de créer des liens entre des fichiers sur deux systèmes de fichiers ayant leur propre liste d'inode tandis qu'un lien dur ne peut franchir la partition du fichier lié. Le lien dur a l'avantage de ne jamais être brisé, contrairement à un lien symbolique qui le deviendra si le fichier lié est supprimé. Les liens symboliques sont généralement utilisés pour les liens entre des noms de librairies sur le système (voir /usr/lib), ce qui permet de déplacer une librairie

sur une autre partition. Les liens durs peuvent être utilisés pour les sauvegardes, comme dans le système TimeMachine d'Apple.

10.23 Si un système de fichiers utilise des blocs de 2Kio mais sur un certain disque tous les fichiers ont exactement 1Kio, quelle est la taille perdue par fragmentation interne? Discutez de la perte selon la distribution de la taille des fichiers? Dans quelle condition la perte sera plus grande ou plus petite que celle obtenue précédemment?

Dans un tel cas, 50% de l'espace est perdu en fragmentation interne. Si la taille typique des fichiers est plus petite, cette fragmentation sera plus grande alors qu'elle diminuera pour une taille typique plus grande. Si la longueur des fichiers varie aléatoirement de manière uniforme, l'espace moyen perdu par fichier sera de 2Kio / 2 (la moyenne entre 0 et 2Kio). Toutefois, en pourcentage, cela sera plus petit si c'est 1Kio par fichier de plusieurs Kio en moyenne.