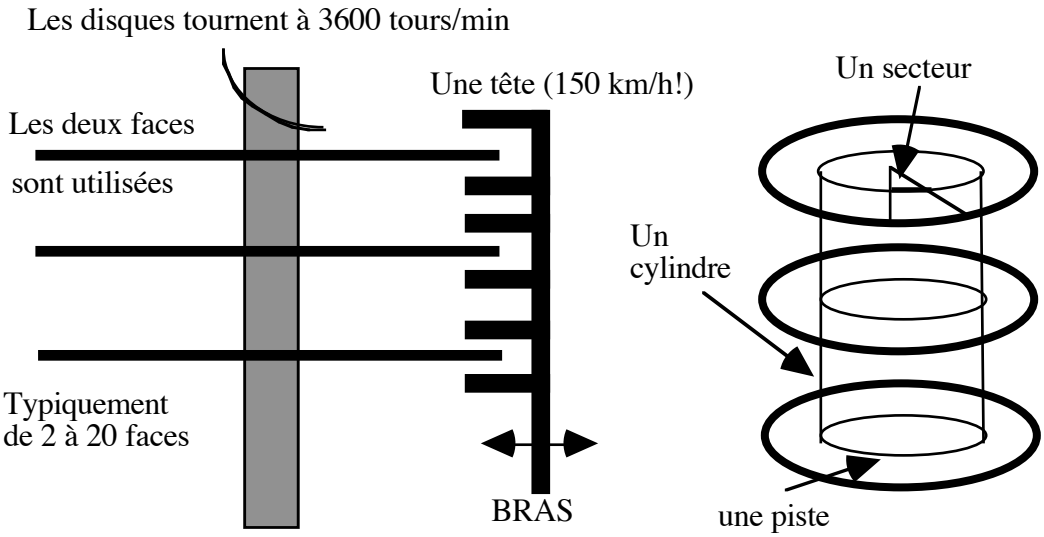
Corrigé ED 6

Exercice 1 :
Comparaison des politiques de gestion du disque

Question 1

On peut commencer par quelques rappels sur les disques :



- Opération de formatage: définit la taille d'un secteur (512 à 4096 octets) et le nombre de secteurs par piste(485)

- Capacité 36 Go

- Calcul du temps d'accès à un secteur :

Ta = Tps déplacement bras (seek time) + Délai rotationnel (un demi-tour en moyenne) + Tps transfert du secteur

Ta = quelques ms

Débit transfert = 500 Mo / seconde typiquement

Les politiques d'ordonnancement des requêtes disque:

Une requête = # cylindre, # piste, # secteur

On peut chercher à optimiser le débit, le temps moyen de réponse, la variance du temps de réponse (cf ED2)

FIFO : Un temps d'attente moyen prouvé plus long que pour les autres discipline

SSTF (Shortest Seek Time First): Les requêtes pour des secteurs au centre du disque sont mieux servies => bon débit, temps moyen faible, mais forte variance. Famine possible.

SCAN (Ascenseur): Voyage dans une même direction. Respect de "l'inertie" du disque pour limiter le mouvement de bras. Bon débit, temps moyen faible, faible variance mais les pistes médiannes sont atteintes plus souvent.

CSCAN (Circular Scan, Aller Simple): En bout de disque, il ya retour à la piste 0 sans consultation des pistes intermédiaires. Traite le disque comme s'il était un tore.

Signaler qu'il existe de nombreuses autres variantes. Il a par exemple été remarqué que la plupart des disques était inoccupés de 65% à 80% du temps. On peut profiter de ce temps mort pour ramener le bras dans une position intéressante.

Question 2

File d'attente des requêtes :

N° de cylindre demandé	20	16	13	40	12	1	11
Ordre d'arrivée	1	2	3	4	5	6	7

Position initiale du bras : cylindre 15

• FIFO :

départ du cylindre #15

aller au #20 => 5 cyl.

aller au #16 => 4 cyl.

aller au #13 => 3 cyl.

aller au #40 => 27 cyl.

aller au #12 => 28 cyl.

aller au #1 => 11 cyl.

aller au #11 => 10 cyl. TOTAL = 88 cyl. parcourus

• SSTF

départ du cylindre #15

aller au #16 => 1 cyl.

aller au #13 => 3 cyl.

aller au #12 => 1 cyl.

aller au #11 => 1 cyl.

aller au #20 => 9 cyl.

aller au #1 => 19 cyl.

aller au #40 => 29 cyl. TOTAL = 73 cyl. parcourus

• SCAN

départ du cylindre #15

aller au #16 => 1 cyl.

aller au #20 => 4 cyl.

aller au #40 => 20 cyl.

on repart en arrière

aller au #13 => 27 cyl.

aller au #12 => 1 cyl.

aller au #11 => 1 cyl.

aller au #1 => 10 cyl. TOTAL = 64 cyl. parcourus

• Calcul des dates de délivrance (unité = nombre de cylindres parcourus)

ATTENTION: L'ordre de délivrance respecte les ordres d'arrivées (respect de la cohérence)

Requête	FIFO	SSTF	SCAN
20	5	15	5
16	5+4=9	15 (avant)	5
13	12	15	52
40	39	73	52
12	67	73	53
1	78	73	64
11	88	73	64
MOY.	**46,4**	**48,1**	**42,1**

Exercice 2 : Gestion de fichiers UNIX

1. Lors de la première demande de lecture, le descripteur de fichier contient en TABDIRECT[0] le numéro du premier bloc de données. Il faut donc le lire, et transférer les 256 premiers octets (disons ceux de numéro 0 à 255) de ce bloc dans une zone du programme. Lors de la deuxième demande, il s'agit du même bloc, mais comme il n'y a pas de conservation dans un tampon des blocs, il faut le relire, et délivrer les octets 256 à 511. Pour la cinquième demande, il s'agit des octets 0 à 255 du deuxième bloc de données, c'est-à-dire, le bloc de numéro TABDIRECT[1].

2. Les octets de la 41^{ième} demande sont situés dans le 11^{ième} bloc de données. Il faut donc lire le bloc INDIRECT_1 pour connaître le numéro du bloc de données qui nous intéresse (premier numéro de bloc dans ce bloc), et pouvoir lire celui-ci. Les octets de la 45^{ième} demande sont situés dans le bloc de données. Il faut donc aussi lire le bloc INDIRECT_1 pour connaître le numéro du bloc 12 de données qui nous intéresse (deuxième numéro de bloc dans ce bloc), et pouvoir lire celui-ci. Il s'ensuit que chaque demande de lecture, à partir de maintenant entraînera deux accès disque.

3. Les octets de la 1065^{ième} demande sont situés dans le 266^{ième} bloc de données. Il faut donc lire le bloc INDIRECT_2, puis celui dont le numéro est le premier dans le bloc venant d'être lu, pour avoir le numéro du bloc de données qui nous intéresse (premier numéro de bloc dans celui venant d'être lu), et lire enfin le bloc de données.

Les octets de la 1066^{ième} demande sont situés également dans le 266^{ième} bloc de données. En l'absence de mémorisation, il faudra relire les mêmes blocs que précédemment pour satisfaire la demande. On peut donc en conclure que dorénavant, 3 accès disque seront nécessaires pour satisfaire chaque demande.

4. Les octets de la 2089^{ième} demande sont situés dans le 522^{ième} bloc de données. Il faut donc lire le bloc INDIRECT_2, puis celui dont le numéro est le deuxième dans le bloc venant d'être lu, pour avoir le numéro du bloc de données qui nous intéresse (premier numéro de bloc dans celui venant d'être lu), et lire enfin le bloc de données. Les octets de la 2090^{ième} demande sont situés également dans le 522^{ième} bloc de données. En l'absence de mémorisation, il faudra relire les mêmes blocs que précédemment pour satisfaire la demande. On peut donc en conclure que 3 accès disque sont toujours nécessaires pour satisfaire chaque demande.

5. Les 40 premières demandes de lecture demanderont chacune un accès disque. Les 1024 suivantes en demanderont chacune deux. Les demandes restantes (32768 - 1024 - 40 = 31704) en demanderont chacune trois. On a donc un total de 97200 accès disque, et un temps d'attente d'entrées-sorties de 3888 secondes, soit un peu plus d'une heure. Notons que l'indirection de niveau 3 n'est pas utilisée puisque 2 niveaux permettent de mémoriser 65 Mo, alors que nous n'en avons que 8 Mo.

Exercice 3 : Duplication de descripteurs d'entrée-sortie

L'explication est donnée dans le programme à travers les commentaires.
Le principe est d'utiliser un fichier temporaire temp entre un processus et son père: le fils écrit dans temp en ayant redirige la sortie standard sur temp, le père lit dans temp en ayant redirige l'entrée standard sur temp.
Tout processus crée possède une table de descripteurs qui contient au départ les entrées 0, 1 et 2 qui pointent respectivement vers l'entrée standard (stdin), la sortie standard (stdout) et l'erreur standard (stder).

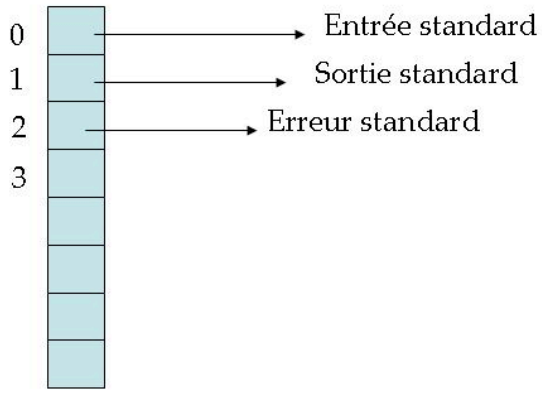
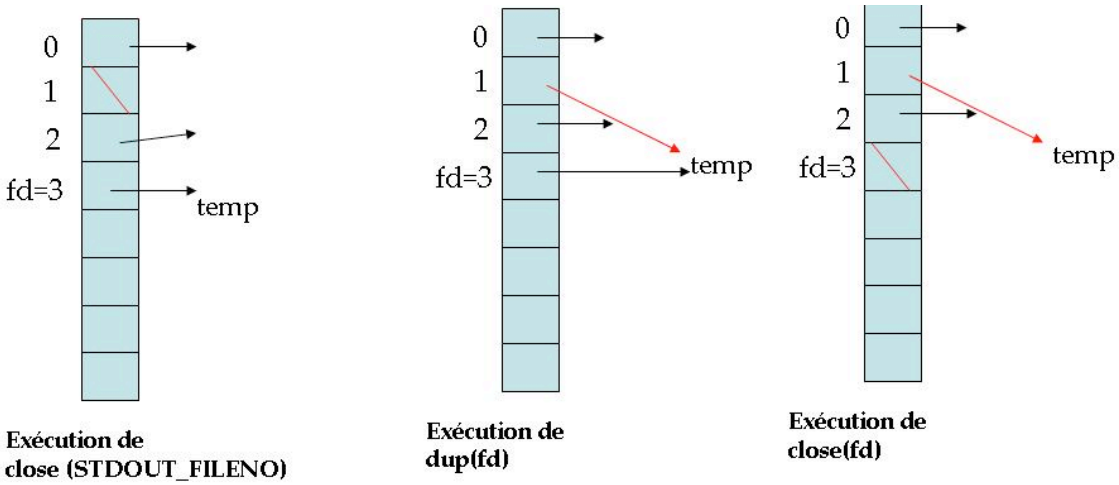


Table des descripteurs
d'un processus

Voici comment les différentes opérations effectuées par le processus fils se traduisent dans sa table de descripteurs.



La table de descripteurs du processus père est similaire sauf qu'on agit sur l'entrée 0 (entrée standard).