Rapport TP3 - Antoine Maendly

Voici la sortie obtenu en executant le programme sur un fichier de 10MB :

```
Copied (read/write) "bigfile" into "tmp" (9765KB in chunks of 1 bytes) in 19.099603 ms
Copied (read/write) "bigfile" into "tmp" (9765KB in chunks of 32768 bytes) in 0.006221 ms
Copied (fread/fwrite) "bigfile" into "tmp" (9765KB in chunks of 1 bytes) in 0.438382 ms
Copied (fread/fwrite) "bigfile" into "tmp" (9765KB in chunks of 32768 bytes) in 0.004278 ms
```

Différence dans la taille du buffer

On voit premièrement que, dans les deux cas, avoir un buffer plus grand permet de copier le fichier nettement plus rapidement.

- Ceci n'est pas une surprise car avec le buffer de 1 bytes, on devra, pour chaque bytes, faire un appel système pour Read et un appel pour Write alors que avec un buffer plus grand le nombre d'appel système est considérablement réduit. Comme chaque appel a un cout en temps, forcément plus on fait d'appel, plus notre programme sera lent.
- De plus, il est plus efficace de lire des blocks de données dans le disque de taille raisonablement grand alors que pour la lecture avec un buffer de 1 byte, les blocks sont très petits.

Différence entre read/write et fread/fwrite

On voit aussi que quand on utilise les fonctions fread/fwrite, le programme est plus rapide que quand on utilise les fonction read/write.

- Les fonctions fread/fwrite sont optimisé pour limité les appels système en fonctionnant avec un système de buffer et attendent avant de faire un appel système. Alors que avec les fonctions read/write, elles vont faire un appel système à chaque fois ce qui ralenti passablement le programme.
- Ceci explique aussi pourquoi, quand le buffer augmente, la différence du temps d'execution est diminué entre read/write et fread/fwrite. Comme on a quand même une taille de buffer limité, l'optimisation de fread/fwrite est moins remarquable qu'avec un peit buffer.

Strace

En utilisant strace, on voit clairement le nombre d'appels. Ce test est fait sur un très petit fichier avec le buffer de 1 byte.

En se concentrant sur les appel d'écriture et de lecture, on voit que read/write fait un appel pour chaque byte alors que fread/fwrite en a fait que 4 !

---

read/write

% time	seconds	usecs/call	calls	errors	syscall
60.59	0.192919	19	10001		write
38.98	0.124114	12	10002		read
100.00	0.318426	15	20038	1	total

---

fread/fwrite

% time	seconds	usecs/call	calls	errors	syscall
13.91	0.000280	70	4		close
7.05	0.000142	35	4		write
100.00	0.002013	42	47	1	total