RAPPORT EXERCICE 2

Roussel Desmond Nzoyem Ngueguin

25 janvier 2020

On souhaite comparer l'efficacité des primitives systèmes de gestion des fichiers et des fonctions de la bibliothèque standard d'entrées-sorties. Pour cela, on mesure les temps d'exécution du programme « cesar » qui utilisent majoritairement des primitives systèmes ou des fonctions de bibliothèque standard en fonction des arguments qui lui sont fournis. Les informations importantes du test sont les suivantes :

Fichier utilisé : /usr/bin/doxygen
Taille : 14 924 104 octets (environ 15.6 Mo)
Nombre de mesures : 5
Valeurs mesurées : Temps utilisateur et Temps système
Commande exécutée : **time ./cesar -1 < /usr/bin/doxygen > /dev/null** (*exécutée dans un script Bash, voici la commande pour le cas d'utilisation des fonctions de bibliothèques*).

FONCTIONS DE BIBLIOTHEQUE STANDARD

Exécution (#)	Temps utilisateur (s)	Temps système (s)
1	0.282	0.048
2	0.29	0.012
3	0.287	0.02
4	0.269	0.008
5	0.306	0.02
Moyenne	**0.2868**	**0.0216**

PRIMITIVES SYSTEME, BLOCS DE TAILLE **1**

Exécution (#)	Temps utilisateur (s)	Temps système (s)
1	14.008	16.031
2	14.272	17.744
3	14.074	17.073
4	14.297	16.696
5	14.471	16.422
Moyenne	**14.2244**	**16.7932**

PRIMITIVES SYSTEME, BLOCS DE TAILLE **2**

Exécution (#)	Temps utilisateur (s)	Temps système (s)
1	6.951	8.527
2	7.318	8.979
3	6.797	7.957
4	7.632	8.267
5	7.476	8.431
Moyenne	**7.2348**	**8.4322**

PRIMITIVES SYSTEME, BLOCS DE TAILLE 4

Exécution (#)	Temps utilisateur (s)	Temps système (s)
1	3.638	3.95
2	3.692	4.128
3	3.792	4.459
4	3.511	4.316
5	3.511	4.115
Moyenne	**3.6288**	**4.1936**

PRIMITIVES SYSTEME, BLOCS DE TAILLE 8

Exécution (#)	Temps utilisateur (s)	Temps système (s)
1	2.021	2.188
2	2.013	2.256
3	1.94	2.265
4	1.968	2.188
5	1.944	2.287
Moyenne	**1.9772**	**2.2368**

PRIMITIVES SYSTEME, BLOCS DE TAILLE 16

Exécution (#)	Temps utilisateur (s)	Temps système (s)
1	1.153	1.093
2	1.164	1.055
3	1.116	0.936
4	1.087	1.107
5	1.168	1.008
Moyenne	**1.1376**	**1.0398**

PRIMITIVES SYSTEME, BLOC DE TAILLE 32

Exécution (#)	Temps utilisateur (s)	Temps système (s)
1	0.588	0.605
2	0.655	0.511
3	0.673	0.592
4	0.655	0.555
5	0.637	0.589
Moyenne	**0.6416**	**0.5704**

PRIMITIVES SYSTEME, BLOCS DE TAILLE 64

Exécution (#)	Temps utilisateur (s)	Temps système (s)
1	0.405	0.269
2	0.456	0.24
3	0.387	0.287
4	0.413	0.277
5	0.38	0.267
Moyenne	**0.4082**	**0.268**

PRIMITIVES SYSTEME, BLOCS DE TAILLE **128**

Exécution (#)	Temps utilisateur (s)	Temps système (s)
1	0.273	0.184
2	0.286	0.161
3	0.287	0.167
4	0.243	0.157
5	0.267	0.167
Moyenne	**0.2712**	**0.1672**

PRIMITIVES SYSTEME, BLOCS DE TAILLE **256**

Exécution (#)	Temps utilisateur (s)	Temps système (s)
1	0.217	0.1
2	0.217	0.096
3	0.271	0.056
4	0.216	0.092
5	0.264	0.032
Moyenne	**0.237**	**0.0752**

PRIMITIVES SYSTEME, BLOCS DE TAILLE **512**

Exécution (#)	Temps utilisateur (s)	Temps système (s)
1	0.202	0.04
2	0.203	0.052
3	0.2	0.044
4	0.204	0.04
5	0.188	0.037
Moyenne	**0.1994**	**0.0426**

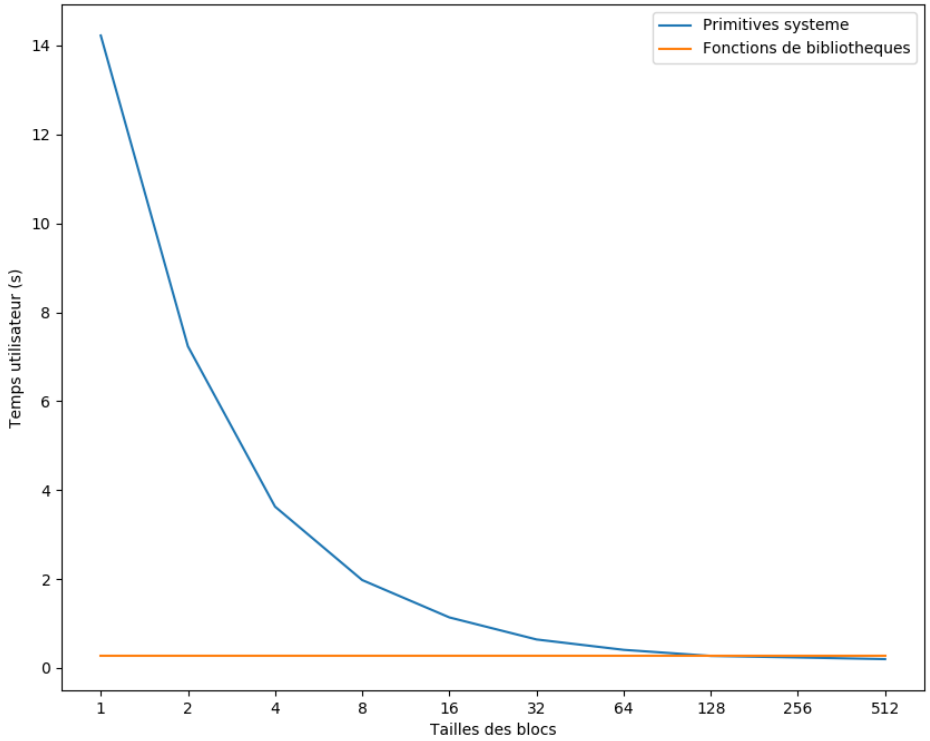
Pour faire les graphes, on place en abscisse les **tailles des blocs** (qui n'ont aucun effet dans le cas d'utilisation des fonctions de bibliothèque). On place en ordonnée les **moyennes des temps** qu'on a mesuré.

On regroupe les résultats précédents comme suit :

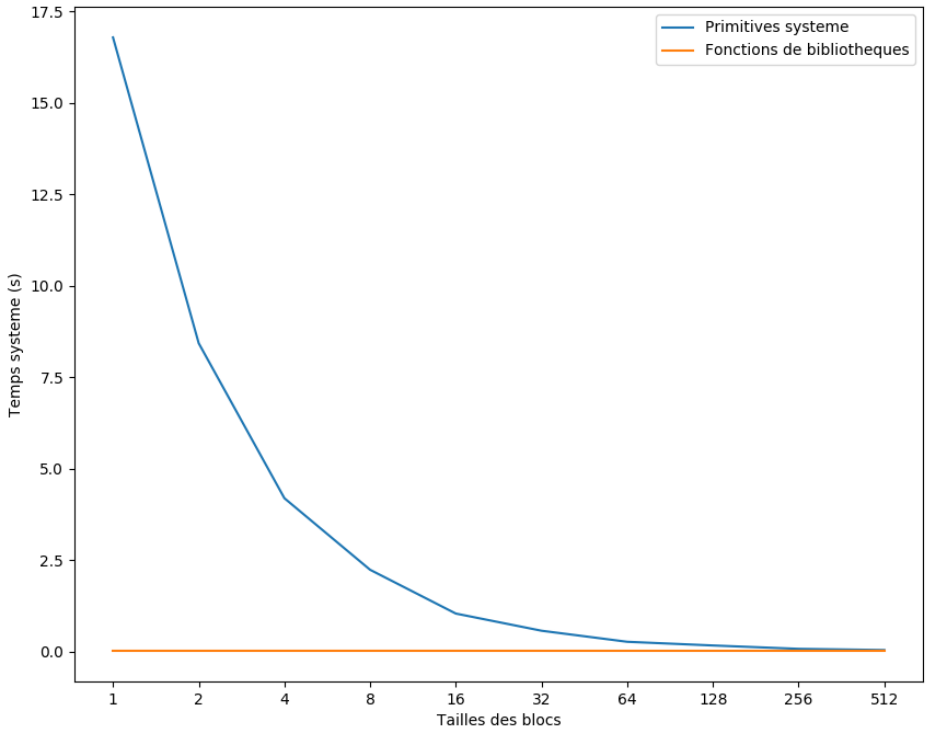
Temps_user_bibliotheque = 0.2826
Temps_sys_bibliotheque = 0.0216

Tailles_des_blocs = [1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
Temps_user_primitives = [14.2244, 7.2348, 3.6288, 1.9772, 1.1376, 0.6416, 0.4082, 0.2712, 0.237, 0.1994]
Temps_sys_primitives = [16.7932, 8.4322, 4.1936, 2.2368, 1.0398, 0.5704, 0.268, 0.1672, 0.0752, 0.0426]

COMPARAISON DES **TEMPS UTILISATEURS** EN UTILISANT LES FONCTIONS DE BIBLIOTHEQUE ET LES PRIMITIVES SYSTEMES

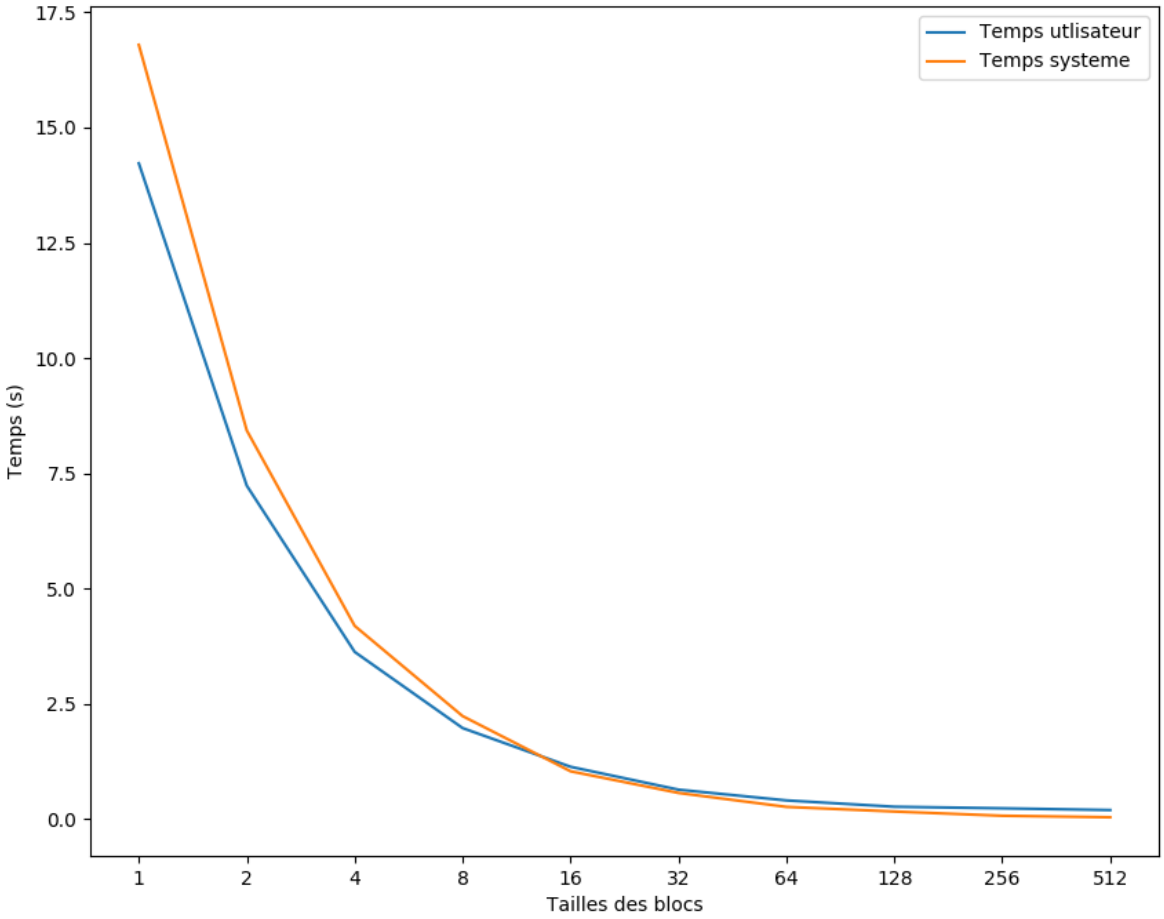


COMPARAISON DES **TEMPS SYSTEMES** EN UTILISANT LES FONCTIONS DE BIBLIOTHEQUE ET LES PRIMITIVES SYSTEMES



Les deux graphes précédents montrent que l'utilisation des primitives systèmes est plus coûteux en temps, surtout lorsque la taille des blocs est petite. Ceci est dû aux multiples appels de primitives systèmes qui ont lieux, vu que lorsque la taille des blocs est relativement faible, le nombre total de blocs nécessaire est grand. Lors de chacun de ces appels, le système d'exploitation doit passer la main au noyau pour exécuter la primitive système et puis repasser la main à notre programme après. Ces interruptions systèmes sont à l'origine de la différence de temps observée.

COMPARAISON DES TEMPS UTILISATEUR ET TEMPS SYSTEME DANS LE CAS D'UTILISATION DES **PRIMITIVES SYSTEMES**



Ici on constate que le temps système est supérieur au temps utilisateur jusqu'à un certain niveau, ou le temps système devient plus petit. Ceci car lorsque la taille des blocs est petite, plusieurs opérations demandées par notre programme (plusieurs « read » et plusieurs « write ») sont exécutées dans le noyau, ce qui augmente considérablement le temps système. Mais quand la taille des blocs devient élevée, le système d'exploitation passe très peu de temps dans le noyau, ce qui réduit le temps système. Quant au temps utilisateur, il est lié au nombre d'itérations qu'il faut effectuer pour parcourir entièrement l'entrée standard. Lorsque la taille du bloc est élevée, une seule itération peut suffire, c'est rapide mais pas autant que l'exécution des deux primitives système « read » et « write ».