課題 No.2 の解答例

高水準 I/O を使用した場合，バッファサイズの異なる低水準 I/O を使用した場合について，ファイルをコピーする時間を比較した．

実行条件

実行したコンピュータ	：	MacBook Pro M1 2020
実行コンピュータの OS	：	macOS 11.3 (Big Sur)
ファイルサイズ	：	12MiB(bs=1024, count=12288)

実行結果

実行時間は遅い方から以下の順であった．

1. 1 バイトの write システムコール (mycp2_1)

	1 回目	2 回目	3 回目	平均
real	20.37	19.77	19.55	19.90
user	1.61	1.61	1.63	1.62
sys	18.67	18.04	17.85	18.19

2. 高水準 I/O(mycp)

	1 回目	2 回目	3 回目	平均
real	0.44	0.44	0.44	0.44
user	0.42	0.42	0.42	0.42
sys	0.01	0.01	0.01	0.01

3. 1,024 バイトの write システムコール (mycp2_1024)

	1 回目	2 回目	3 回目	平均
real	0.05	0.05	0.05	0.05
user	0.00	0.00	0.00	0.00
sys	0.04	0.04	0.04	0.04

考察

1. mycp2_1 は mycp2_1024 よりシステム時間が約 400 倍になっている．システムコールの呼び出し回数は約 1000 倍になっているはずなので，システムコール 1 回当たりの処理時間は 0.4 倍程度と考えられる．システムコール 1 回当たり読み書きするデータの量が 1000 倍になっても，システムコール 1 回当たりの処理時間は 2.5 倍 (1/0.4 倍) 程度で済んでいる．

2. mycp は mycp2_1024 と比較してシステム時間が短くなっている．このことから高水準 I/O が用いるバッファは 1024 バイトより大きく，mycp のシステムコール呼び出し回数は mycp2_1024 より少なくなっていると考えられる。

3. mycp は mycp2_1 と比較してユーザ時間が 4 分の 1 倍に短くなっている．高水準 I/O の関数 (`getc()`，`putc()`) は，システムコールを呼び出すための前処理より軽い．（これには驚いた．）オンラインマニュアル (`man getc`) で調べてみると「`getc()` はマクロでありインラインに展開される」と書いてある．高速化するための工夫が凝らされている．