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	：	Mac mini M1 2020
実行コンピュータの OS	：	macOS Ventura 13.3.1
ファイルサイズ	：	5MiB(bs=1024, count=5120)

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	19.93	19.90	19.76	19.86
user	2.31	2.33	2.31	2.32
sys	17.56	17.49	17.40	17.48

2. 高水準 I/O(mycp)

	1 回目	2 回目	3 回目	平均
real	0.39	0.38	0.38	0.38
user	0.37	0.37	0.37	0.37
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	0.04	0.04	0.04	0.04
user	0.00	0.00	0.00	0.00
sys	0.03	0.03	0.03	0.03

考察

1. mycp2_1 と mycp2_1024 の比較

- システム時間
  mycp2_1 は mycp2_1024 よりシステム時間が約 600 倍になっている．システム時間は，システムコールの処理のためにカーネルが費やした時間である．２つのプログラムで `open()`，`close()` システムコールの使い方は同じなので，変化は `read()`，`write()` システムコールが原因だと考えられる．
  `read()`，`write()` システムコールの呼び出し回数は mycp_1 の方が約 1000 倍になっているはずなので，測定結果からシステムコール 1 回当たりの処理時間は 0.6 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大きく変化（1000 倍）しても，システムコール 1 回当たりの処理時間はあまり（1.7 倍程度）変化しない．
- ユーザ時間
  mycp2_1024 ではユーザ時間が短すぎて測定できなかった．mycp2_1 ではループを実行する回数が約 1000 倍なので，while 文の実行，`read()`，`write()` がシステムコールを発行の準備に費やす時間など，ユーザプログラム内で費やす時間が長くなったと考えられる．その結果，ユーザ時間が測定できたと考えられる．

2. mycp と mycp2_1024 の比較

- システム時間
  mycp は mycp2_1024 と比較してシステム時間が短くなっている．mycp のシステムコール発行回数は mycp2_1024 より少ないと推察できる．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を mycp2_1024 より少なくしていると考えられる。
- ユーザ時間
  mycp もループの実行回数が多いため while 文や `getc()`，`putc()` の実行回数が多く，計測できる程度のユーザ時間を費やしたと考えられる．`getc()`，`putc()` は FILE 構造体のバッファを操作する処理を 1 バイト毎に行っており，この処理時間がユーザ時間に算入される．

3. mycp と mycp2_1 の比較

- システム時間
  システム時間は高水準 I/O を用いる mycp の方が圧倒的に短い．FILE 構造体のバッファにデータをためてシステムコール回数を少なくしたお陰である．
- ユーザ時間
  mycp は mycp2_1 と比較してユーザ時間がおおよそ 6 分の 1 倍になっている．`read()`，`write()` の呼出し回数と `getc()`，`putc()` の呼出し回数は同じはずである．`getc()`，`putc()` は，`read()`，`write()` 関数中のシステムコールを呼び出す前処理（前処理までが user 時間に含まれる）より軽い．（これには驚いた．）
  macOS の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．`getc()`，`putc()` が，FILE 構造体のバッファを操作する処理には高速化するための工夫が凝らされている．