MICS 実験第一 J4 課題レポート

学籍番号 2210342, 鈴木謙太郎

2024 年 7 月 21 日

1 課題 1

まず,write システムコールを直接用いるコード 1 のような mycp 関数を作成した.

Code 1: mycp 関数のソースコード

```c
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define BUFFER_SIZE 128

void mycp(const char *src, const char *dst, size_t buffer_size) {
  int src_fd = open(src, O_RDONLY);
  if (src_fd == -1) {
    perror("Error opening source file");
    exit(EXIT_FAILURE);
  }

  int dst_fd = open(dst, O_WRONLY | O_CREAT | O_TRUNC, 0644);
  if (dst_fd == -1) {
    perror("Error opening destination file");
    close(src_fd);
    exit(EXIT_FAILURE);
  }

  char *buffer = (char *)malloc(buffer_size);
  if (buffer == NULL) {
    perror("Error allocating buffer");
    close(src_fd);
    close(dst_fd);
    exit(EXIT_FAILURE);
```

```
28    }
29
30    ssize_t bytes_read;
31    while ((bytes_read = read(src_fd, buffer, buffer_size)) > 0) {
32      if (write(dst_fd, buffer, bytes_read) != bytes_read) {
33        perror("Error writing to destination file");
34        free(buffer);
35        close(src_fd);
36        close(dst_fd);
37        exit(EXIT_FAILURE);
38      }
39    }
40
41    if (bytes_read == -1) {
42      perror("Error reading from source file");
43    }
44
45    free(buffer);
46    close(src_fd);
47    close(dst_fd);
48  }
49
50  int main(int argc, char const *argv[]) {
51    if (argc != 3) {
52      fprintf(stderr, "Usage: %s <source> <destination>\n", argv[0]);
53      return 1;
54    }
55    char const *source = argv[1];
56    char const *destination = argv[2];
57    mycp(source, destination, BUFFER_SIZE);
58    return 0;
59  }
```

また, この mycp 関数の性能を評価する際の対照として, コード 2 のような標準ライブラリ関数を用いたコピー関数を作成した.

Code 2: 比較用のコピー関数のソースコード

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void copy_file_stdio(const char *src, const char *dst) {
5    FILE *src_file = fopen(src, "rb");
```

```
6   if (src_file == NULL) {
7     perror("Error opening source file");
8     exit(EXIT_FAILURE);
9   }
10
11  FILE *dst_file = fopen(dst, "wb");
12  if (dst_file == NULL) {
13    perror("Error opening destination file");
14    fclose(src_file);
15    exit(EXIT_FAILURE);
16  }
17
18  int ch;
19  while ((ch = fgetc(src_file)) != EOF) {
20    if (fputc(ch, dst_file) == EOF) {
21      perror("Error writing to destination file");
22      fclose(src_file);
23      fclose(dst_file);
24      exit(EXIT_FAILURE);
25    }
26  }
27
28  fclose(src_file);
29  fclose(dst_file);
30 }
31
32 int main(int argc, char const *argv[]) {
33   if (argc != 3) {
34     fprintf(stderr, "Usage: %s <source> <destination>\n", argv[0]);
35     return 1;
36   }
37   char const *source = argv[1];
38   char const *destination = argv[2];
39   copy_file_stdio(source, destination);
40   return 0;
41 }
```

これらの関数を用いて, コピー対象とするファイルのサイズを $16^1 \sim 16^9$ バイトの間で変えながら, それぞれの関数の性能を評価した. 性能評価には,GNU time コマンドを用い, 実行時間を 10^{-2} 秒単位で測定した.

まず,mycp 関数の測定結果は図 1 のようになった.

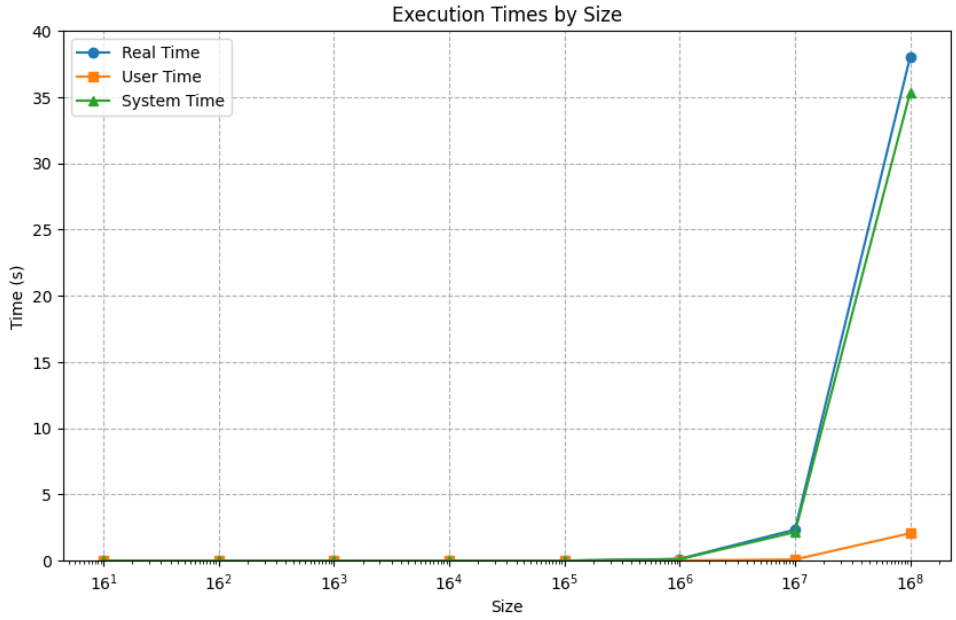

図 1: mycp 関数の性能測定結果

次に, 標準ライブラリ関数を用いたコピー関数の測定結果は図 2 のようになった.

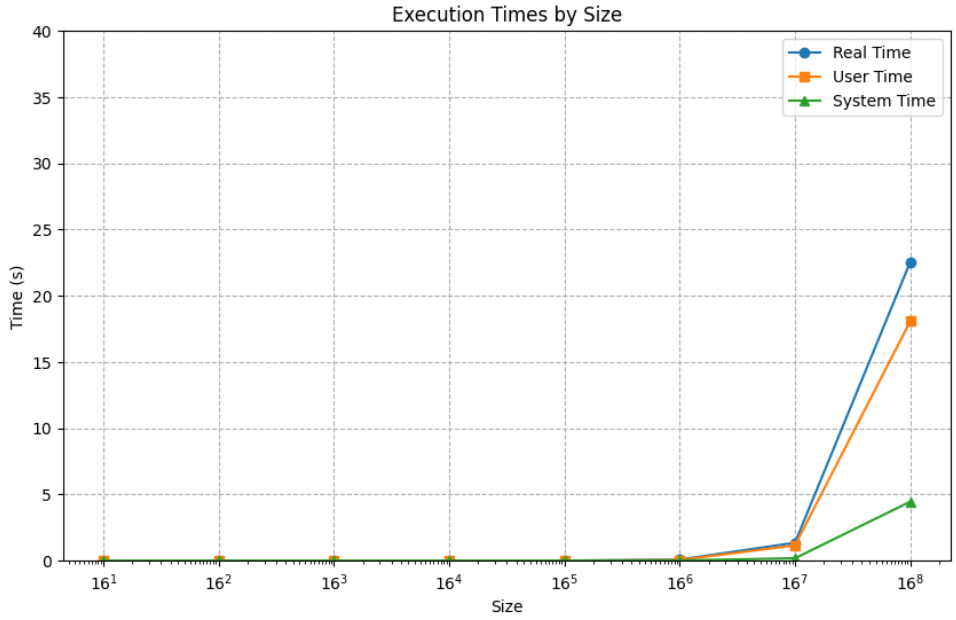

図 2: 標準ライブラリ関数を用いたコピー関数の性能測定結果

2 つの関数の測定結果で共通している点は, ファイルサイズが 16^6 バイト (1 メガバイト) 程度までは 10^{-2} 秒単位での実行時間がほぼ一定であることである. また, ファイルサイズが 16^7 バイト (10 メガバイト) 程度を超えると, どちらの関数も実行時間が time コマンドで測定できる程度になることがわかる.

一方で, 実行時間が増えてきてからの傾向については,mycp 関数と標準ライブラリ関数を用いたコピー関数で異なる点がみられる. 独自に作成した mycp 関数の実行時間は system time が占める割合が大きいが, 標準ライブラリ関数を用いたコピー関数の実行時間は user time が占める割合が大きい.

この違いとしては, システムコールの回数の違いが考えられる. mycp 関数は,read と write のシステムコールを用いてファイルのコピーを行っているため, システムコールの回数が多くなる. 一方で, 標準ライブラリ関数を用いたコピー関数は, ファイルの読み書きを行う際にバッファリングを行っているため, システムコールの回数が少なくなる. fgetc/fputc は 1 バイトずつ読み取りと書き込みを行っているが, 内部的にはある程度大きなバッファで読み書きを行っているため, システムコールの回数が少なくなっていると推測される.

このような結果, システムコールの回数が異なっているので,2 つの関数において実行時間の内訳が異なると考えられる.

2 課題 2

3 課題 3