

# 平成 21 年度オペレーティングシステム期末試験

2010 年 2 月 9 日

問題は 3 問, 4 ページある. 1 枚の解答用紙に 1 問解答すること.

## 1

10000000 未満の素数の個数を求める以下のプログラムを PRIME10M とする.

```
1: int isprime(int x) {  
    /* x が素数なら 1 */  
    int d;  
    for (d = 2; d * d <= x; d++)  
5:     if (x % d == 0) return 0;  
    return 1;  
}  
int main() {  
    int c = 0; int p;  
10: int N = 10000000;  
    for (p = 2; p < N; p++)  
        c += isprime(p);  
    printf("there are %d primes less than %d\n", c, p);  
}
```

また, PRIME10M の 10 行目:  $N = 10000000$ ; の代わりに  $N = 4000000$ ; とした, 4000000 未満の素数の個数を求めるプログラムを PRIME4M とする. さらに, 11 行目の for 文の終了条件を以下の様にし, 10 秒間で求められるだけの素数を求めるようにしたプログラムを PRIME10SEC とする.

```
for (p = 2; elapsed_time() < 10.0; p++)
```

`elapsed_time()` は (中身はここに書かないが), このプログラムが開始してからの経過時間 (実時間) を返す関数であるとする.

ある計算機で, PRIME10M を単独で (他に CPU を消費するプロセスがない状態で) 実行したところ, 約 20 秒かけて計算が終了した. 同じ条件で, PRIME4M は約 6 秒で終了した. この計算機は Linux, Windows などの PC/WS 向けのオペレーティングシステムを搭載している. また, 2 つの CPU (コア) を搭載しており, 2 つまでのスレッドを同時に実行することが出来る. 以下の問いに答えよ. すべて, 結果だけでなくその理由の説明とともに答えよ.

(次のページに続く)

- (1) PRIME10M と PRIME4M を 1 つずつ同時に開始したとする。PRIME10M は約何秒で終了するか。
- (2) PRIME10M を 1 つと PRIME4M を 5 つ、同時に開始した場合はどうか。
- (3) PRIME10M を 1 つと PRIME10SEC を 5 つ、同時に開始した場合はどうか。
- (4) PRIME10M と PRIME4M を 1 つずつに加え、さらにテキストエディタ 1 つを同時に開始した場合はどうか。テキストエディタに対しては、タイピングの速度にはやや自信のある少年が、1 秒 10 文字程度の速度で英文を入力しており、PRIME10M が終了するまでタイピングを行ったとする。

いずれの場合も、上記で実行したプログラム以外に、CPU を消費するプロセスはほぼ存在しないとする。

## 2

ある少年が、ファイルAの内容を逆順にしたファイルBを作るプログラムを書くことにした。Aの大きさはNバイトとする。少年が最初に書いたプログラムは以下である。

```
1: int main() {
    int fd_A = open(A, O_RDONLY); /* Aを開く */
    int fd_B = open(B, O_RDWR|O_CREAT|O_TRUNC, 0644); /* Bを開く(作る) */
    char * a = malloc(N); /* Aを読み込むための配列 aを確保 */
5:  read(fd_A, a, N);      /* Aをaに読み込む */
    char * b = malloc(N); /* 結果を格納する配列 bを確保 */
    int i;
    for (i = 0; i < N; i++) { /* aを逆にしてbへ書く */
        b[N - 1 - i] = a[i];
10: }
    write(fd_B, b, N);      /* Bに書き込む */
}
```

ヘッダファイルの読み込み (`#include`) や、エラー検査は省略している。また、`read/write` システムコールの仕様では、要求したバイト数未満の読み込みや書き込みでリターンする事がありうるが、プログラムの簡略化のため、ここではそれもないとしている (つまり、5 行目/11 行目の `read/write` の呼び出しは、必ずN バイト数の読み込み・書き込みをする)。

このプログラムの「速度」を、「 $N / \text{実行時間}$ 」(つまり、1 秒あたりに、逆順に出来るバイト数) で測定する。

- N を数 MB 程度とし、このプログラムを複数回連続して実行したところ、  
最初に実行した時と 2 度目以降では速度が大きく違った (a)。
- 次に、徐々に大きなN に対してこのプログラムを実行したところ、  
ある程度N が大きくなったところで、速度が大きく変化した (b)。

そこで少年は、このプログラムの速度を改善する手段を考えた。しばらく勉強すると、次の二つのシステムコールがあることを学んだ。

- `lseek` : ファイルの任意の位置を読み書きするシステムコール (c)
- `mmap` : ファイルをプロセスのアドレス空間にマップするシステムコール (d)

以下の問いに答えよ。

- (1) 下線部 (a) において、実行時間はどちら (最初と 2 度目以降) が速いか? それはなぜかを含めて答えよ。
- (2) 下線部 (b) において、「変化」はどちら向きの変化か? つまりN を増やしたときに遅くなったのか、速くなったのか? またこの「変化」はN がどのような値の時に起きるのか? その理由とともに答えよ。
- (3) 下線部 (c) の `lseek` を使う方針で上記のプログラムの速度を改善する方法の概要を示せ。なぜそれで速度が改善するのも合わせて答えること。
- (4) 下線部 (d) の `mmap` を使う方針で上記のプログラムの速度を改善する方法の概要を示せ。なぜそれで速度が改善するのも合わせて答えること。

### 3

- (1) 現代的な PC/WS 用オペレーティングシステムはプロセス間のメモリを分離している。つまり、バグのあるプログラムや悪意を持って書かれたプログラムでも、別のプロセスのメモリを読み書きする事は出来ないようになっている。メモリは所詮、CPU のメモリアクセス命令によって読み書きできる装置なのだから、これが「出来ない」のは決して自明なことではない。その仕組みを、ハードウェア (CPU) がどんな機能を提供しており、オペレーティングシステムがそれをどう使っているか、などを明確に区別しながら述べよ。
- (2) 現代的な PC/WS 用オペレーティングシステムでは、ユーザプログラムはシステムコールを介してファイルの読み書きを行う。これはファイルの読み書きを単純化するだけでなく、バグや悪意のあるプログラムから、ファイルを守るという意味を持っている。つまり、悪意を持ったプログラムがファイルを格納している装置 (例えばハードディスク) を直接アクセスしてデータを破壊したり盗み見たりすることは出来ないようになっている。ハードディスクは所詮コンピュータから読み書き出来る装置なのだから、これをユーザプログラムが「出来ない」というのは決して自明な事ではない。その仕組みを、ハードウェア (CPU) がどんな機能を提供しており、オペレーティングシステムがそれをどう使っているか、などを明確に区別しながら述べよ。

問題は以上である