情報処理システム Ⅱ ・中野・定期試験問題

2026 年 7 月 31 日

注意　問題は 3 問あります。解答はすべて別紙の解答用紙に記入しなさい。

I　Unix 系 OS 環境にログインし、端末エミュレータを開き、下図右のような C プログラム `foo.c` を、下図左のようにコンパイルして実行した。a ～ g の文章の空欄を埋めるのに最も適切な語句を次ページの候補の中から選んで解答用紙に書きなさい。ただし、`double` 型のデータの大きさは 8 B である。(51 点)

```
$ cc -o foo foo.c
$ ./foo
a[] = 32.3 -12.0 -50.3 36.2 77.1
77.1
36.2
-50.3
-12.0
32.3
a = 0x4114e0
&i = 0x7fffbffff59c
$
```

foo.c

```
#include <stdio.h>

double a[5];

int main()
{
    int i;

    printf("a[] = ");
    for (i = 0; i < 5; i++)
        scanf("%lf", &a[i]);
    for (i = 4;i >= 0; i--)
        printf("%.1f\n", a[i]);
    printf("a = %p\n", a);
    printf("&i = %p\n", &i);
    return 0;
}
```

a. 端末エミュレータでキーボードから入力された「`cc -o foo foo.c`」や「`./foo`」という文字列を読み取っているのは一般にシェルと呼ばれるプログラムのプロセスである。このプロセスは、(1) を介して端末エミュレータへ接続されているファイル記述子 (2) 番から (3) というシステムコールを使ってこれらの文字列を読み込む。

b. 「`./foo`」は (4) パス名と呼ばれる書き方で、プロセスの (5) ディレクトリを起点として特定のファイルを指定するものである。

c. シェルが「`./foo`」という文字列を読み込むと、自らのプロセスを複製してできた新しいプロセスの仮想アドレス空間に、(6) というシステムコールを利用して `foo` の機械語プログラムなどを取り込み、特定の番地からその実行を開始する。この時点では、`printf` 関数や `scanf` 関数の機械語プログラムは、このプロセスのアドレス空間には取り込まれておらず、(7) リンクが行なわれた結果として、(8) システムコールなどによって初めてプロセスのアドレス空間に現われることとなる。

d. プログラム中の配列 `a` と変数 `i` は、実行時には、それぞれ、プロセスのアドレス空間の (9) 領域と (10) 領域に割り当てられる。

e. `foo` のプロセスが起動された後、キーボード (標準入力) から「`32.3 -12.0 -50.3 36.2 77.1`」という文字列が入力されているが、`foo` の実行結果をみると、末尾の「`77.1`」の 4 文字は、標準ライブラリ関数 (11) によって `double` 型のデータに変換されて、0x (12) 番地から始まる 8 B の領域に書き込まれているはずである。

f. このようなアドレスは、実際には、CPU のアドレス変換機構によって、(13) アドレスに変換されて使用される。この変換をどのように行なうかは、カーネルの (14) 管理機能が担っている。

g. `foo` のプロセスが、キーボードからの入力を待っているとき、このプロセスは (15) 待ち状態となっており、カーネルは CPU 資源を他のユーザープロセスの実行に割り当てることができるが、Unix 系 OS では (16) マルチタスク方式が採用されているため、それ以外のタイミングでも、強制的に CPU 資源の使用権が奪われ、プロセスが (17) 待ち状態になることがある。

(次ページに続く)

空欄の候補

-1、0、1、2、3、4、4114e0、4114e8、4114f0、4114f8、411500、411508、7fffbffff59c、7fffbffff5a4、7fffbffff5ac、7fffbffff5b4、7fffbffff5bc、7fffbffff5c4、ALU、BSS、CPU、close、execve、fork、link，main、mmap，printf、read、scanf、unlink，write、アイドル、アドレス、イベント、カーネル、カレント、キャッシュ、スタック、テキスト、データ、ディレクトリ、デバイス、パイプライン、ヒープ、ファイル、ブートストラップ、プリエンプティブ、プロセス、メモリ、ルート、レジスタ、親、階層的、仮想、疑似端末、協調的、子、実行、数理、静的、絶対、相対、動的、特権、非特権、物理

(裏面に問題 II)

Ⅱ Unix 系 OS 上の端末エミュレータにおいて、下のようにコマンドを実行した。a ～ g の文章の空欄を埋めるのに最も適切な語句を問題末尾の候補の中から選んで解答用紙に書きなさい。ただし、同じ候補を複数回使っても構いません。(24 点)

端末エミュレータでのコマンド実行の様子

```
$ pwd
/home/y250000/src
$ cd ..
$ cc -o bar src/bar.c
$  ./bar 3 in.dat >out.dat
$ ls -l
合計 48
-rwxr-xr-x   1 y250000  math       10680 7月 30日  09:07 bar
-rw-r--r--   2 y250000  math          27 7月 30日  08:37 in.dat
-rw-r--r--   1 y250000  math          81 7月 30日  09:08 out.dat
drwxr-xr-x   2 y250000  math         179 7月 30日  09:07 src
$
```

a. この実行例では、pwd、cd、cc、./bar、ls の 5 つのコマンドを実行しているが、このうちシェルの内部コマンドとして実行されたのは [(1)] である。

b. ./bar 3 in.dat >out.dat の実行時に、bar.c の main 関数へ第 1 引数として渡された整数値は [(2)] である。

c. ./bar 3 in.dat >out.dat の out.dat の部分は、[(3)] と書いても同じ動作となるはずである。

d. 一方、./bar 3 in.dat >out.dat の ./bar の部分を bar と書くと実行できなくなってしまう。これは、通常、[(4)] には [(5)] というパス名を含めないためである。

e. ./bar 3 in.dat >out.dat の実行時に、out.dat を open したプログラムは [(6)] である。

f. ls -l の実行で表示されたファイルやディレクトリの名前は、そのファイルやデイレクトリの [(7)] の一部として記録されているわけではなく、それらが置かれているディレクトリ (この例では /home/y250000) に記録されている。

g. ls -l の実行結果によると、in.dat というファイルは、/home/y250000 というディレクトリ以外にも [(8)] リンクされていることが分かる。

空欄の候補

-1、0、1、2、3、4、10680、27、81、179、.、./bar、..、../out.dat、../src/out.dat、/home/out.dat、/home/y250000/out.dat、/home/y250000/src/out.dat、/out.dat、cc、cd、ls、pwd、y250000、math、アドレス、カーネル、カレントディレクトリ、コマンドサーチパス、シェル、シンボリック、タイマー、ハード、ブートストラップ、プリエンプティブ、プロセス、メタデータ、ルートディレクトリ、ライブラリ、親、階層的、書き込み、仮想、協調的、環境変数、子、実行可能、静的、動的、物理、読み出し、割り込み

(次ページに問題 Ⅲ)

Ⅲ ある計算機システムで、1 ページの大きさが 2 KiB であるようなデマンドページング方式の仮想記憶システムを持ったオペレーティングシステムが稼働しており、右の C プログラムをコンパイルして得られる機械語プログラムのプロセスが動いている。このプロセスは、配列 a のための領域として 1000 KiB までの物理メモリを使用することが可能で、配列 a へのアクセス以外でページフォルトが起こることはない。ページフォルトが起こると、(必要なら) 最後にアクセスされた時刻が最も遠い過去であるような物理ページを (補助記憶装置に) ページアウトし、(必要なら) アクセスしたい仮想ページの内容を (補助記憶装置から) その物理ページにページインする。int 型のデータの大きさは 4 B で、配列 a は、ページ番号が 10 進表記で 1000 であるような仮想ページの先頭から割り当てられている。バイトアドレッシング方式で、キャッシュメモリの影響は考えないものとして次の問いに答えなさい。(25 点)

```
#define N (1000*1024)

int a[N];

int main() {
    int i, sum = 0, step = [      ];

    while (1) {
        for (i = 0; i < N; i += step)
            sum += a[i]; // (*)
    }
    return sum > 0;
}
```

(1) 配列 a のために必要とされる仮想ページ数を求めなさい。

(2) a[1000] が置かれる仮想ページ番号を求めなさい。

(3) int 型の正の定数 k で空欄を補うとする。(*) の印を付けた行の a[i] へのアクセスでページアウトが起こらないような k の最小値を求めなさい。

(4) 空欄を 512 で補うと、(*) の印を付けた行で a[i] へアクセスする際に、配列 a に使用されている仮想ページのページアウトが発生した。このときの i の値が 512000 であったとして、ページアウトした仮想ページのページ番号を求めなさい。

(定期試験問題終り)

情報処理システム Ⅱ ・定期試験・解答用紙

I (1) 疑似端末　(2) 0

(3) read　(4) 相対

(5) カレント　(6) execve

(7) 動的　(8) mmap

(9) BSS　(10) スタック

(11) scanf　(12) 411500

(13) 物理　(14) メモリ

(15) イベント　(16) プリエンプティブ

(17) 実行

II (1) cd　(2) 3

(3) /home/y250000/out.dat　(4) コマンドサーチパス

(5) .　(6) シェル

(7) メタデータ　(8) ハード

III — (2)、(3)、(4) の解答欄は裏面

(1) 4 B の大きさの要素が 1000×1024 個並んだものであるから、4 B×1000×1024 = 4000 KiB、つまり、2000 ページとなる。

学籍番号　氏名

(裏面に続く)

情報処理システム II ・定期試験・解答用紙

III — (1) の解答欄は表面

(2) `a[1000]` のアドレスは `a[0]` と比べて $4 \times 1000 = 4000$ 番地分大きい。1 ページは 2048 B であるから、`a[0]` がページ番号 1000 の先頭にあることを考えると、`a[1000]` が置かれるページ番号は 1001 となる。

(3) 使用できる物理ページは 1000 KiB、つまり 500 ページしかないので、2000 ページ必要な配列 `a` 全体は割り当てることができない。つまり、ページアウトしないためには、全体の 4 分の 1 のページのみにアクセスする必要がある。このため、`i += step` で、`a[i]` のアドレスが 4×2 KiB 分進まなければならない。よって、求める最小の k は $4 \times 2048 \div 4 = 2048$。

(4) `step` が 512 であるため、`a[i]` のアドレスは、$4 \times 512 = 2048$ ずつ、つまり 1 ページ分ずつ進む。このため、配列 `a` に使用されている 2000 ページのすべてにアクセスすることになるので、`a[i]` へのアクセスの度にページフォルトが起こる。このときページアウトされるのは、500 回前にアクセスしたページとなる。

`a[0]` のページ番号が 1000 であるから、`a[512000]` は、その $4 \times 512000 \div 2048 = 1000$ ページ先となり、そのページ番号は、$1000 + 1000 = 2000$ である。よって、ページアウトするページ番号は $2000 - 500 = 1500$ となる。

(解答用紙終り)