平成18年度オペレーティングシステム期末試験

2007年2月5日

問題は3問，7ページある．

1

スレッド間でデータ (簡単のため，int 型の整数とする) を受け渡しするキュー (Queue) を操作する二つの手続き get と put があるとする．

- get(q) はキュー q からデータを一つ取り出す．もちろん空のキューからデータを取り出すことはできない．その場合，データが put によって一つ挿入されるまでブロックする．
- put(q，x) はキュー q に一つのデータ x を挿入する．キューには一定の容量 c があり，満杯のキューにデータを挿入することはできない．つまり，q にデータがすでに c 個格納されていれば，一つのデータが get によって取り出されるまでブロックする．

以下の問いに答えよ．

(1) 簡単のために $c = 1$ とした上で，以下の構造体や関数定義の$\boxed{\cdots}$部分を補う形で，(容量1の) キューの実現方法を書け．スレッドの同期のための標準的な API (排他制御，条件変数など) を適宜用いよ．プログラムは C 言語風の擬似コードで書くことを想定しているが，厳密な文法や API の用法 (引数の数や順番など) にはこだわらないので，適宜文章で説明を補え．

```
typedef struct
{
  int n;        /* 格納されているデータ数．0 または 1 */
  int data;     /* n=1 のとき，格納されているデータ */
  ···
} * Queue;

void put(Queue q, int x) {
  ···
}

int get(Queue q) {
  ···
}
```

以下では一般の c に対するキューの実装が与えられているとし，それを用いて二つのスレッド A, B の間で多数のデータを**両方向**に送りあうことを考える．二つのキュー $Q_{A\to B}, Q_{B\to A}$ を用意し，前者は A から B へ，後者は B から A へデータを転送するのに用いる．スレッド A の送るべきデータ (n 個あるとする) を $a_0, \ldots, a_{n-1}$，スレッド B の送るべきデータ (m 個あるとする) を $b_0, \ldots, b_{m-1}$ と書く．

以下は，この目的を達成するために書かれたプログラムであるが，間違いがある (正しく動くとは限らない)．

```
thread_A() {
  for (i = 0; i < n; i++) {
    put(Q_{A→B}, a_i);
  }
  while (1) {
    get(Q_{B→A});
  }
}

thread_B() {
  for (i = 0; i < m; i++) {
    put(Q_{B→A}, b_i);
  }
  while (1) {
    get(Q_{A→B});
  }
}
```

(2) このプログラムには，デッドロックと呼ばれる状態に陥るという間違いがある．具体的にどういう状態に陥るのかを説明せよ．

(3) 正しいプログラムを書け．概要と擬似コードの形で示せ．ただし，以下に注意せよ．

- 両スレッドは自分が送るデータの個数は知っているが，受け取るべき個数は知らないものとする．すなわち，A は m を知らず，B は n を知らない．キューには，送るべきデータ $(a_0, \ldots, a_{n-1}, b_0, \ldots, b_{m-1})$ 以外のものを入れてはならず，A, B がそれ以外の手段で通信することも許されない．
- 正しい終了状態は，両スレッドが受け取るべきデータをすべて受け取った上で，空のキューに対して `get` を呼んでブロックしている状態であるとする．前項により，各スレッドはデータが何個来るかを知らないので,「これ以上データが来ない」ということを検出することはできない．従って両スレッドは通常の意味で終了 (`thread_A`, `thread_B` からリターン) することはできないことに注意せよ．

2

以下はC言語によるプログラミングの演習を行っている学生二人の会話である．読んで以下の問いに答えよ．

学生 A: さあできたぞ．

学生 B: よし，走らせよう!

A がプログラムを走らせると “segmentation fault” というメッセージを出してプログラムが終了した[1]

```
% ./tauthon draw3D.py
Segmentation fault
%
```

学生 A: Oh，shit!

学生 B: 欧米か! さっさとデバッグしろよ．

学生 A: だいたい segmentation faultって何だよ?

学生 B: Segmentation faultってのはな，要するに (a) な (b) をした時に出るものなんだよ．OS の授業でやっただろ?

学生 A: (a) だと，人聞きの悪い．いったい俺がどこで (a) をしたっていうんだ?

学生 B: 確かに (a) という言葉はまるで悪いことでもしたみたいでびっくりするけど，なぜかそういう言葉を使うんだよ．それはさておき，segmentation fault がどこで起きたかを調べるには，gdb とか，デバッガを使うのが早いんだよ．ほら，`gdb ./tauthon` ってやってみ．

A は言われたとおりタイプして gdb デバッガを立ち上げる．

```
% gdb ./tauthon
   ...
(gdb)
```

学生 B: 立ち上がったら次は，`run` コマンドだ．それでプログラムが走り出す．

A は言われたとおり入力してプログラムを走らせる．すると以下のように，segmentation fault がおきた旨のメッセージとともに，次のような表示が出た．

```
(gdb) run
Starting program: /home/tau/tauthon

Program received signal SIGSEGV, Segmentation fault.
0x08048364 in foo () at a.c:24
24   f(p, &q->x, r->next, s.x, t.next, u.next->x);
```

ただし，`p`，`q`，`r`，`s`，`t`，`u` はそれぞれ以下のように定義されている局所変数であるとする．`f` はこのプログラムの中で定義されている関数の名前である．

[1] プログラムや引数の名前はフィクションであり，実在のものとは一切関係ありません

```
int foo()
{
  list * p;
  list * q;
  list * r;
  list s;
  list t;
  list u;
   ...
   ...


}
```

list は以下のような構造体である.

```
typedef struct list {
  int x;
  struct list * next;
} list;
```

学生 B: どれどれ，なるほど，これはこの行`f(p, &q->x, r->next, s.x, t.next, u.next->x);`を実行している最中のどこかで，(a) な (b) を起こした，ということを表しているんだ.

学生 A: 俺には何がなんだか.(といって，肩をすくめる動作をする)

学生 B: 欧米か! 大事なのは，並んでいる引数，p とか，`&q->x` とか，... の中で (a) な (b) を起こしたのはどの式か，ということだ．もちろん君は segmentation fault が何かを知らなかったくらいだから，このプログラムの中で mprotect とかなんとか，難しいシステムコールは使ってないよね.

学生 A: 後半が何を言っているのかは全然わかんなかったけど要はあまり深く気にしなくていいってことで.じゃあ，どの式かって言うと，うーん，俺にはわからないから，B 君どうぞ.

学生 B: しょうがないなぁ. (c) と (d) あたりかな．C 言語の仕組みをちょいとわかってればわかるはずなんだけどな.

学生 A: で，実際に確かめるにはどうすればいいの?

学生 B: デバッガには，式の値を表示する`print`ってコマンドがあるんだ．とりあえずこれは試験問題だからあまり余計なヒントにならないよう「全部の変数の値を表示してみよ」と言っておこう.

A は次のようにコマンドを打ち込んで以下のような出力を得る.

```
(gdb) print p
$1 = (list *) 0x0
(gdb) print q
$2 = (list *) 0x100
(gdb) print r
$3 = (list *) 0x200
(gdb) print s
$4 = {x = 300, next = 0x400}
(gdb) print t
$5 = {x = 500, next = 0x600}
```

```
(gdb) print u
$6 = {x = 700, next = 0x800}
```

学生 B: ま，初歩的なミスをしたということだね．これを見る限り僕の言ったとおり (c) と (d) の両方とも，segmentation fault をおこすね．それは上の`print` の結果を見れば明らかだね．

学生 A: 俺にはなーんにも明らかじゃありません．

学生 B: 基本から説明するからよく聞くように． (e) ．ちなみに，こういうことになってしまった原因は大方， (c) や (d) へちゃんとした値を代入してないことじゃないかと思うけどね．どれどれ (プログラムを見る), あーやっぱり．

学生 A: えーとつまり，そいつらに何かを代入すれば言い訳ね．じゃ，こんなのはどう?

```
foo() {
  ...

  ...
  (c) = 10000;
  (d) = 20000;
  ...
}
```

学生 B: そんな滅茶苦茶な... ひょっとしたら segmentation fault はおきなくなるかもしれないが，適当な数を代入すればいいというものじゃないんだよ!

学生 A: あー，そういえば，C には`&`とかいう演算子があったっけ．そうか．えーっと，じゃ，まずはこういう関数を作って,

```
list * alloc_list() {
  list l;
  return &l;
}
```

こういう風にしたら?

```
foo() {
  ...

  ...
  (c) = alloc_list();
  (d) = alloc_list();
  ...
}
```

学生 B: これで確かに segmentation fault はおきなくなるかもしれないが，残念ながらそれも駄目．その理由は， (f) .

学生 A: うーん，じゃ，l を大域変数にして，これは?

```
list l;
list * alloc_list() {
  return &l;
}


foo() {
  ...

  ...
  (c) = alloc_list();
  (d) = alloc_list();
  ...
}
```

学生 B: うーん，これも駄目だね．その理由は，(g)．要するに君は C 言語のメモリ管理というものについて，まったくわかっとらんようだねぇ．

学生 A: わかったから．で，結局どうすればいいの?

学生 B: この場合は結論から言うと，C 言語の標準的なライブラリに含まれる (h) というを使えばよい．この関数は要するに，(i)．わかったかな．

学生 A: わかりました．これまでは segmentation faultっていうのがおきると，ともかくプログラムがどっかで間違っているという以外にはわからずに闇雲にプログラムを眺めていたんだが，要するにどういうときに segmentation fault がおきるのかってのをちゃんと理解すると，だいぶプログラミング能力っていうか，プログラムの問題解決能力が向上するんだね．

学生 B: 少しは教えた甲斐があった．じゃあ，お疲れってことでなんか食いに行こうぜ．何にしようか?

学生 A: チェリーパイ!

学生 B: 欧米か!

以下の問いに答えよ．

(a), (b) に入る言葉を答えよ．

(c), (d) には，`f(p, &q->x, r->next, s.x, t.next, u.next->x);` に並ぶ引数のうちのどれかが入る．それらを答えよ (順不同).

(e) には，(c) または (d) が segmentation fault を起こす原因であるということを B が結論付けるための推論が入る．それは，C 言語の基本的知識として，どんな式がどのような時に segmentation fault を起こしうるのか・起こしえないのか，に関する説明が入る．自分なりの説明を数行で書け.

(f), (g) には，それぞれ直前で示された`alloc_list` 関数が正しくない理由が入る．自分なりの説明を数行で書け.

(h) にはよく使われる C 言語の標準的なライブラリ関数の名前，(i) にはその動作の基本的な説明，これまでの間違った修正方法と何が違うのかの説明が入る．(h) に適切な名前と，(i) に入る説明を書け.

3

OS は，CPU が備えている MMU (メモリ管理ユニット) を用いて，様々な機能や，性能の向上を実現している．そのうち，以下であげるものについて説明し，その実現方法の概要を述べよ．さらに，ここで述べられて**いない**機能，または性能の向上を一つあげ，同様の説明を行え．

- プロセスの論理アドレス空間の分離 (メモリの保護)
- 要求時 (demand) ページング
- mmap システムコールによるファイルの読み書き

問題は以上である