オペレーティングシステム・試験問題

2012 年度 E・O クラス (2013 年 2 月 15 日・試験時間 90 分)

書籍，配布資料およびノート等は参照してはならない．ただし，最大一枚までのメモ（手書きに限る．A4 両面使用可）を参照できるものとする．

1. オペレーティングシステム Xv6 では，カーネル内のスレッド間の同期に x86 の `xchg` 命令を用いている．この命令は共有変数とスレッド内の局所変数の値をアトミックに交換するもので，その動作を C 風の疑似コードで書くと以下の関数のようになる．

```
atomic void xchg(bool *m, bool *r) {
    bool tmp = *r;
    *r = *m;
    *m = tmp;
}
```

ここで **atomic** はこの関数の実行がアトミックに行われることを表す．また **bool** は真偽値型で，その値は **true** あるいは **false** のいずれかとする．この関数を使って **bool** 型の 2 つの変数 `x` と `y` の値を交換するには，`xchg(&x, &y);` とすればよい．

さて，昨年度の期末試験ではプログラム 1 を穴埋め問題として出した．これは `xchg` を使った相互排除の一方式であり，実際に Xv6 のカーネルで使われているものを簡略化したものである．関数 `P` はひとつのスレッドの動作を表すもので，複数個のスレッドがこの関数を実行する．変数 `in_use` は共有変数，`r` はスレッド毎の局所変数である．*CS* および *NC* はそれぞれクリティカルセクションと非クリティカルセクションを表しており，これらの中では変数 `in_use` および `r` の読み書きは行われないものとする．

(a) 相互排除における安全性とは何か．クリティカルセクションという言葉を用いて説明せよ．

(b) プログラム 1 は 3 個以上のスレッドでも安全性をみたすか．証明の必要はないので結果だけ答えよ．

```
1  bool in_use = false;      // shared
2  void P(void) {
3    bool r;             // thread local
4    while (true) {
5        NC
6        r = true;
7        do {
8            xchg(&in_use, &r);
9        } while (r);
10       CS
11       in_use = false;
12   }
13 }
```

プログラム 1: `xchg` を用いた相互排除

(c) プログラム 1 のような相互排除方式では，*CS* に入る条件がみたされるまでループによって繰り返し検査が行われる．このような待ち状態の実現方法によるロックを何と呼ぶか．以下 (1)〜(4) の中から一つ選べ．

(1) ジャイアントロック　(3) アームロック
(3) スピンロック　(4) ポストロック

(d) スリープロックでなく，問題 **(c)** のようなロックの方式が使用されるのはどのような場合か．1 つ以上の例を挙げよ．

(e) プログラム 2 はプログラム 1 の改良版として作られたものである．空欄 A, B に入る変数をそれぞれ記せ．A と B に入る変数は同じであってもよい．

(f) Xv6 では，共有メモリマルチプロセッサでも動作するよう，実際に `xchg` 命令を実行する際にバスにロックをかけている．あるプロセッサがバスにロックをかけている間は，他のプロセッサはバスを使うことはで

```
1  bool in_use = false;
2  void P(void) {
3    bool r;
4    while (true) {
5        NC
6        r = true;
7        while (true) {
8            xchg(&in_use, &r);
9            if (! A ) break;
10           while ( B );
11       }
12       CS
13       in_use = false;
14   }
15 }
```

プログラム 2: xchg を用いた相互排除（改良版）

きない．このことをふまえ，プログラム 2 がプログラム 1 の改良版になっていることの理由を説明せよ．(ヒント：共有変数の読み出しだけであればバスをロックする必要はないものとする)

2. 図 1 はバディシステムを用いて以下の順にメモリの割当と解放を行ったときの様子を表している．

(1) A: 6KB の割当, (2) B: 3KB の割当,
(3) C: 7KB の割当, (4) D: 5KB の割当,
(5) C を解放, (6) B を解放.

(a) 上に続けて以下の順にメモリの割当と解放を行ったときの様子を図 1 にならって解答用紙に記入せよ．

(7) E: 10KB の割当, (8) A を解放,
(9) F: 10KB の割当, (10) G: 3KB の割当,
(11) D を解放, (12) G を解放.

記憶領域の合計は 64KB で，最小割当単位は 4KB とする．バディシステムによる領域の分割は実線で明記すること．解放後に複数の領域が併合された場合，その間には線をひかないこと．同サイズの空き領域が複数あるときは下位（小さい番地）から使うものとする．図では上が下位としている．

(b) バディシステムで生じることがあるのは (A) 外部断片化と (B) 内部断片化のどちらか．

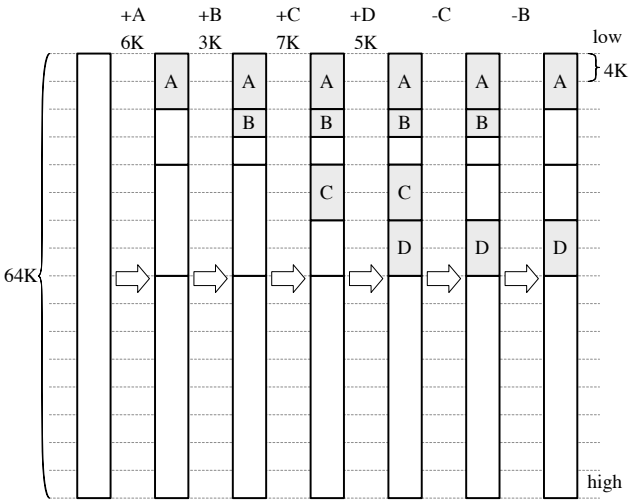

図 1: バディシステムによるメモリ割当て

3. Xv6 のファイルシステムでは，i-node はブロックへの直接参照 12 個と 1 段の間接参照 1 個を持つ．ここでブロックサイズは 512 バイトであり，ブロック番号は 4 バイト（32 ビット）の符号なし整数で表される．

(a) Xv6 では最大何バイトまでのファイルを扱うことができるか．ただしディスクは十分大きいものとする．

(b) 4000 バイトのファイルが占めるデータブロックの数はいくつか．間接参照ブロックがある場合はそれも数えること．i-node，ビットマップ，ログのためのブロックは数えなくてもよい．

(c) 8000 バイトのファイルが占めるデータブロックの数はいくつか．間接参照ブロックがある場合はそれも数えること．i-node，ビットマップ，ログのためのブロックは数えなくてもよい．

(d) Xv6 のソースコードを見ると，ディスクの i-node ブロック数を `ninodes / IPB + 1` で計算している．ここで `ninodes` は（スーパーブロックで指定されている）総 i-node 数，`IPB` は 1 ブロックあたりに格納できる i-node（`dinode` 構造体）の数である．実はこのように i-node ブロック数を計算すると無駄な（使用されない）i-node ブロックが生じることがある．`ninodes` が 200 のときはそのような無駄が生じているか否かを答えよ．ここで `dinode` 構造体の大きさは 64 とする．