オペレーティングシステム・試験問題

2013 年度 E・O クラス (2014 年 2 月 13 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. x86 の xchg 命令を使うと，メモリとレジスタに格納された 1 ワードの値をアトミックに交換できる．Xv6 ではこの命令を C の関数 xchg から呼び出してカーネル内の相互排除に用いている．

プログラム 1 は関数 xchg の動作を疑似コード[1] で表したものである．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 uint は **unsigned int** と定義された型である．この関数を呼び出すと引数 addr が指す場所に引数 newval の値が代入され，addr が指す場所に元々入っていた値が返値となる．

プログラム 2 は Xv6 のカーネル内相互排除に関係する関数定義を簡略化[2] したものである．以下のように関数 acquire と release を使ってロックの確保と解放を行う．

```
struct spinlock mylock;
initlock(&mylock);   // ロックの初期化
...
acquire(&mylock);    // ロックの確保
CS                   // クリティカルセクション
release(&mylock);    // ロックの解放
```

関数 cli を呼び出すと呼び出した CPU において割込みが禁止され，sti を呼び出すと割り込み禁止が解除される．変数 cpu の型は別途定義されている cpu 構造体へのポインタであり，一見大域変数のようであるが CPU 毎に異なる値を取る（CPU 間で共有されない）．この変数が指す構造体のフィールド ncli の初期値は 0 である．

いま Xv6 が 2 個以上の CPU を持つマルチプロセッサマシン上で動作しているとする．以下の問に答えよ．

[1]実際には asm で xchg 命令を呼び出している．
[2]エラー処理やデバッグ関連の部分を省略してある．

```
atomic uint xchg(uint *addr, uint newval){
    uint result = *addr;
    *addr = newval;
    return result;
}
```

プログラム 1: 関数 xchg（疑似コード）

```
struct spinlock {
    uint locked;
};

void initlock(struct spinlock *lk) {
    lk->locked = 0;
}

void acquire(struct spinlock *lk) {
    pushcli();
    while (xchg(&lk->locked, 1) != 0);
}

void release(struct spinlock *lk) {
    lk->locked = 0;
    popcli();
}

void pushcli(void) {
    cli();       // 割込み禁止
    cpu->ncli++;
}

void popcli(void) {
    --cpu->ncli;
    if (cpu->ncli == 0)
        sti();  // 割込み禁止の解除
}
```

プログラム 2: Xv6 のカーネル内相互排除関数

```
void acquire(struct spinlock *lk) {
    pushcli();
    while (lk->locked != 0);
    lk->locked = 1;
}
```

プログラム 3: 変更その 1

```
void acquire(struct spinlock *lk) {
    while (xchg(&lk->locked, 1) != 0);
}

void release(struct spinlock *lk) {
    lk->locked = 0;
}
```

プログラム 4: 変更その 2

```
void acquire(struct spinlock *lk) {
    cli();
    while (xchg(&lk->locked, 1) != 0);
}

void release(struct spinlock *lk) {
    lk->locked = 0;
    sti();
}
```

プログラム 5: 変更その 3

(a) 関数 acquire をプログラム 3 のように変更した．相互排除は正しく行われるか．理由とともに答えよ．

(b) 問 (a) の変更を施したまま CPU を 1 個にした．相互排除は正しく行われるか．理由とともに答えよ．

(c) 関数 acquire および release をプログラム 4 のように変更し，再び CPU を 2 個以上にした．相互排除は正しく行われるか．理由とともに答えよ．

(d) 問 (c) の変更を施したまま CPU を 1 個にした．相互排除は正しく行われるか．理由とともに答えよ．

(e) 関数 acquire および release をプログラム 5 のように変更し，再び CPU を 2 個以上にした．相互排除は正しく行われるか．理由とともに答えよ．

(f) 相互排除における活性の例を 2 つ挙げよ．その際クリティカルセクションという言葉を用いること．

(g) プログラム 2 のような，ロックが確保できるまで繰り返し試行を行う方式をスピンロックと呼ぶ．この方式は無駄に CPU 時間を消費するという欠点がある．一方，プロセスを wait モードにして（CPU を他のプロセスに明け渡すことで）待ち状態を作るロックの方式がある．このようなロックを何と呼ぶか．以下 (1)〜(4) の中から一つ選べ．

(1) ビジーロック (2) スリープロック
(3) マスロック (4) グローバルインタプリタロック

(h) スピンロックが利用されるのはどのような場合か．例を 1 つ以上挙げよ．

2. Xv6 のファイルシステムでは，inode 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．inode を表す構造体 dinode のサイズは 64 バイトとする．また第 0 ブロックは未使用，第 1 ブロックはスーパーブロックとする．

ここに Xv6 のファイルシステムを持つディスク（イメージファイル）がある．スーパーブロックを読んだところ，size が 4096，ninodes が 500，nlog が 20 であった．

(a) size はディスク全体のブロック数を表す値である．ディスク全体の大きさは何バイトか．ここでは 1K バイトは 1024 バイト，1M バイトは $1024^2 = 1048576$ バイトとする．

(b) ninodes は inode の個数を表す値である．inode ブロックの個数はいくつになるか．

(c) Xv6 の bitmap ブロックでは，データブロックを含む全てのブロックに対応するビットマップが作られている（処理を簡単にするため）．bitmap ブロックの個数はいくつになるか．

(d) nlog は log ブロックの個数である．ディスク全体のブロック数から，第 0 ブロック，スーパーブロック，inode ブロック，bitmap ブロック，log ブロックそれぞれの個数の合計を引いたものがデータブロックの個数になる．データブロックの個数はいくつになるか．

(e) dinode 構造体のフィールド nlink は，ディレクトリから参照されている数を表している．この値が実際に参照されている数と異なる場合に何が起きるか．