システムソフトウェア・試験問題

2020 年度 (2020 年 11 月 26 日・試験時間 100 分)

この試験では，試験中に (1) 講義スライド，(2)xv6 のブックレット（xv6: a simple, Unix-like teaching operating system），および (3)xv6 のソースコードを参照してよい．また，解答にあたり白紙の計算用紙およびコンピュータ（あるいは電卓）を用いた計算（プログラムの作成と実行を含む）を行ってもよい．

1. xv6 のファイルシステムを構成するブロックは，ディスクの先頭からブートブロック（1 ブロック，未使用），スーパーブロック（1 ブロック），ログブロック（nlog 個），inode ブロック，ビットマップブロック，データブロックの順に配置される．ここで 1 ブロックの大きさは 1024 バイトであり，先頭ブロックの番号は 0，ブロック番号は 32 ビット（4 バイト）の符号なし整数で表現されるとする．また，ディスク内での inode を表す構造体 dinode の大きさは 64 バイトとする．

ここに xv6 のファイルシステムがひとつある．このファイルシステムのスーパーブロックに格納されている構造体 superblock を参照したところ，フィールド size の値は 2000，ninodes の値は 300，nlog の値は 40 であった．このファイルシステムの一貫性は保たれているとする．以下 **(a)**-**(d)** に挙げる superblock のフィールドの値をそれぞれ答えよ．

(a) `logstart`

(b) `inodestart`

(c) `bmapstart`

(d) `nblocks`

2. xv6 のスリープロック機構を実現する関数の定義[1] をコード 1〜コード 6 に示す．これらに関する以下の主張 **A**〜**L** のうち正しいものを全て選べ．ただし CPU コア数は 2 以上とする．

A. コード 1 の 3 行目における **`while`** を **`if`** に変更した場合，2 行目でスピンロックを確保しているので変更前と同様に動作する．

B. コード 1 の 3 行目における **`while`** を **`if`** に変更した場合，1 つのスリープロックに対して 2 つ以上のカーネルスレッドが 6 行目以降を実行し得る．

C. コード 3 の 4 行目で確保したスピンロックはコード 3 の 12 行目で開放される．

D. コード 3 の 4 行目で確保したスピンロックはコード 4 の 9 行目で開放される．

E. コード 3 の 4 行目で確保したスピンロックはコード 5 の 15 行目で開放される．

F. コード 3 の 10 行目における `p->state` の値は常に `RUNNING` である．

G. コード 4 の 4 行目における変数 `p` の値と 9 行目における変数 `p` の値は常に等しい．

H. コード 5 の 8 行目における変数 `p` の値と 15 行目における変数 `p` の値は常に等しい．

I. コード 5 の 13 行目における `p->state` の値は `RUNNABLE` になり得る．

J. コード 5 の 13 行目における `p->state` の値は `SLEEPING` になり得る．

K. コード 5 の 13 行目における `p->state` の値は `RUNNING` になり得る．

L. コード 6 をコード 7 のように変更しても変更前と同様に動作する．

[1] ページ数を抑えるために実際のソースコードからコメントやエラー処理を除いてフォーマットを変えている．

```
void acquiresleep(struct sleeplock *lk) {
    acquire(&lk->lk);
    while (lk->locked) {
        sleep(lk, &lk->lk);
    }
    lk->locked = 1;
    lk->pid = myproc()->pid;   // デバッグ用
    release(&lk->lk);
}
```

コード 1: acquiresleep

```
void releasesleep(struct sleeplock *lk) {
    acquire(&lk->lk);
    lk->locked = 0;
    lk->pid = 0;   // デバッグ用
    wakeup(lk);
    release(&lk->lk);
}
```

コード 2: releasesleep

```
void sleep(void *chan, struct spinlock *lk) {
    struct proc *p = myproc();
    if(lk != &p->lock){
        acquire(&p->lock);
        release(lk);
    }
    p->chan = chan;
    p->state = SLEEPING;
    sched();
    p->chan = 0;
    if(lk != &p->lock){
        release(&p->lock);
        acquire(lk);
    }
}
```

コード 3: sleep

```
void wakeup(void *chan) {
    struct proc *p;
    for (p = proc; p < &proc[NPROC]; p++) {
        acquire(&p->lock);
        if(p->state == SLEEPING &&
           p->chan == chan) {
            p->state = RUNNABLE;
        }
        release(&p->lock);
    }
}
```

コード 4: wakeup

```
void scheduler(void) {
    struct proc *p;
    struct cpu *c = mycpu();
    c->proc = 0;
    for (;;) {
        intr_on();
        for (p = proc; p < &proc[NPROC]; p++) {
            acquire(&p->lock);
            if (p->state == RUNNABLE) {
                p->state = RUNNING;
                c->proc = p;
                swtch(&c->context, &p->context);
                c->proc = 0;
            }
            release(&p->lock);
        }
    }
}
```

コード 5: scheduler

```
void sched(void) {
    int intena;
    struct proc *p = myproc();

    // エラーチェック用コードは省略

    intena = mycpu()->intena;
    swtch(&p->context, &mycpu()->context);
    mycpu()->intena = intena;
}
```

コード 6: sched

```
void sched(void) {
    int intena;
    struct proc *p = myproc();
    struct cpu *c = mycpu();

    // エラーチェック用コードは省略

    intena = c->intena;
    swtch(&p->context, &c->context);
    c->intena = intena;
}
```

コード 7: 変更したコード 6

3. xv6 が動作する RISC-V プロセッサでは，アドレスは 64 ビット中の下位 39 ビットが用いられている．ページテーブルは 3 段で，各ページテーブルのインデックスは 9 ビットである．

xv6 で以下を含むプログラムを実行したところ，2 行目の `printf` の呼び出しによって `0x0000000000003010` と表示された[2]．

```
char *p = malloc(1024 * 1024 * 4);
printf("%p\n", p);
```

(a) 1 ページの大きさ（バイト数）を答えよ．

(b) 1 行目の `malloc` によって確保され，ユーザプログラム中で使用可能になるメモリ領域の大きさ（バイト数）を答えよ．

(c) 1 行目の `malloc` によって確保されるメモリ領域の割り当てに必要なページ数を答えよ．ただし xv6 の `malloc` の実装では，使用可能になるメモリ領域の前に確保した領域の大きさを記録するためのヘッダ領域を必要とする．上記の場合，実は 16 バイトのヘッダが付随しており，`malloc` の返値から 16 を引いたアドレスがページ境界になっている．

(d) 1 行目の `malloc` によって確保されるメモリ領域を，物理メモリに割り当てるために使用されるページテーブルエントリの数を答えよ．

(e) `malloc` の実装に用いられている，メモリ領域の確保のためのシステムコールを以下から一つ選べ．

- `pipe`
- `mknod`
- `sbrk`
- `fstat`
- `dup`

[2] `printf` における `%p` はポインタの値を 16 進数で表示する