オペレーティングシステム・試験問題

2015 年度 E・O クラス (2016 年 2 月 12 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. xv6 起動直後にルートディレクトリにおいて以下の 6 つのコマンドをこの順で実行した．

```
mkdir a
mkdir a/b
echo Hello > foo
cat foo > a/bar
ln foo a/b/baz
echo GoodBye > foo
```

これらのコマンドがエラーなく実行できたとする．実行後のファイルシステム [1] について以下の問 **(a)**–**(g)** に答えよ．

(a) ルートディレクトリにおいてコマンド `cat foo` を実行すると `GoodBye` が表示される．ルートディレクトリにおいてコマンド `cat a/bar` および `cat a/b/baz` を実行して表示される内容をそれぞれ記せ．

(b) 上記コマンドの実行によって，新たに割り当てられる有効な `dinode` 構造体の個数を答えよ．

(c) `dinode` 構造体が割り当てられるのは以下のどのブロックか．1〜6 の番号で答えよ．

1. ブートブロック　　2. スーパーブロック
3. ログブロック　　4. inode ブロック
5. ビットマップブロック　　6. データブロック

(d) 上記コマンドの実行によって新たに使用済みとなるデータブロックの個数を答えよ．ブロックサイズは 512 バイト，`dirent` 構造体の大きさは 16 バイトとする．またルートディレクトリの変更では新たにデータブロックの割り当ては生じないものとする．

(e) ファイル `/foo`，`/a/bar`，`/a/b/baz` を表す `dinode` 構造体それぞれの `nlink` の値を答えよ．

(f) ディレクトリ `/a`，`/a/b` を表す `dinode` 構造体それぞれの `nlink` の値を答えよ．

(g) ディレクトリ `/a`，`/a/b` を表す `dinode` 構造体それぞれの `size` の値を答えよ．

2. 以下は xv6 のスーパーブロックに格納される構造体 `superblock` の定義である．ここでログ開始，inode 開始，ビットマップ開始とあるのは，それぞれの役割をもつブロック群の最初のブロック番号である．

```
struct superblock {
  uint size;        // 総ブロック数
  uint nblocks;     // データブロック数
  uint ninodes;     // inode数
  uint nlog;        // ログブロック数
  uint logstart;    // ログ開始
  uint inodestart; // inode開始
  uint bmapstart;   // ビットマップ開始
};
```

(a) デフォルトで作成されるファイルシステムイメージ fs.img では，フィールド `size` の値は 1000，`ninodes` の値は 200，`nlog` の値は 30 である．フィールド `nblocks`，`logstart`，`inodestart`，`bmapstart` それぞれの値を答えよ．ただしディスク上ではブートブロック，スーパーブロック，ログブロック，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レイアウトされるものとし，ブートブロックとスーパーブロックのブロック数は 1 で，それぞれのブロック番号は 0 および 1 である．またブロックサイズは 512 バイト，`dinode` 構造体の大きさは 64 バ

[1] バッファキャッシュおよびログの内容はディスクに反映されているものとする．

イトとし，ビットマップはデータブロックだけでなく全ブロック分が確保されるものとする．

(b) ビットマップブロックの内容に誤りがある場合に生じ得る不具合を 2 つ挙げよ．

3. 以下は 2 つのスレッド p, q の相互排除を目的としたアルゴリズムを C 風の疑似コードで表したものである．NC および CS はアプリケーションのクリティカルセクション以外の部分およびクリティカルセクションを表している．このアルゴリズムでは，共有変数の 1 回の読み出しおよび書き込みがアトミック（不可分）であると仮定している．またこの疑似コードにおいて，文はコードに書かれた通りの順に全て実行される [2] ものとする．

```
// 共有変数
bool wantp = false, wantq = false;
```

```
// スレッドp
while (true) {
 p0: NC
 p1: wantp = true;
 p2: while (wantq) {
 p3:  wantp = false;
 p4:  wantp = true;
     }
 p5: CS
 p6: wantp = false;
}
```

```
// スレッドq
while (true) {
 q0: NC
 q1: wantq = true;
 q2: while (wantp) {
 q3:  wantq = false;
 q4:  wantq = true;
     }
 q5: CS
 q6: wantq = false;
}
```

(a) スレッド p では，ラベル p3 において共有変数 wantp に **false**を代入したあと，直後のラベル p4 において同じ変数に **true**を代入している．またスレッド q においても同様のことをしている．このようなことをする意図を簡潔に説明せよ．

(b) このアルゴリズムについて正しく述べているものを以下からひとつ選んで 1〜4 の番号で答え，またその理由を述べよ．

1. 正しく相互排除が行われる（安全性，活性ともにみたされる）．
2. 安全性がみたされない．
3. デッドロックが生じることがある．
4. 飢餓が生じることがある．

(c) 上記 **(b)** における安全性とは具体的にどのような性質か．

[2] コンパイラの最適化による文の削除や，アウトオブオーダ実行やキャッシュの影響といった現象は考えない