システムソフトウェア・試験問題

2021 年度 (2021 年 11 月 22 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のログ機構に関する記述について，以下の A〜I から正しいものをすべて選べ．

A) S をファイルシステムに変更を加えるような任意のシステムコールとする．S を実行中にシステムがクラッシュしたとする．ログ機構により，常に S による変更内容は失われることなくすべてファイルシステムに正常な形で反映される．

B) S をファイルシステムに変更を加えるような任意のシステムコールとする．S を実行中にシステムがクラッシュしたとする．ログ機構により，常に S による変更内容はすべて失われるが，ファイルシステムの一貫性は保たれる．

C) S をファイルシステムに変更を加えるような任意のシステムコールとする．S を実行中にシステムがクラッシュしたとする．ログ機構により，S による変更内容はすべて失われることなくファイルシステムに正常な形で反映されるか，変更内容はすべて失われるがファイルシステムの一貫性は保たれるかのいずれか一方になる．

D) S をファイルシステムに変更を加えるような任意のシステムコールとする．S ををを実行中にシステムがクラッシュしたとする．ログ機構により，S による変更が一部だけファイルシステムに反映されるが，ファイルシステムの一貫性は保たれるような状況が起こり得る．

E) xv6 では `begin_op` を実行し，いくつかの `log_write` を実行した後に `end_op` を実行することでひとつのトランザクションを構成している．いまあるトランザクションの `end_op` 実行直前にシステムがクラッシュした．このとき常に当該トランザクションによる変更内容は失われる．

F) xv6 では `begin_op` を実行し，いくつかの `log_write` を実行した後に `end_op` を実行することでひとつのトランザクションを構成している．いまあるトランザクションの `end_op` から呼び出された `install_trans` の実行中にシステムがクラッシュした．このとき当該トランザクションは常にコミットされたものとしてログに記録される．

G) xv6 では起動時に `install_trans` を実行してログに残っているコミット済みのトランザクションをファイルシステムに反映する．いま，xv6 起動時の `install_trans` 実行中にシステムの電源を切って（ホストから QEMU を強制停止して）しまった．この場合，xv6 を正常に起動できてもトランザクションの内容は常に失われる．

H) システムコール `write` を使って `char` 型の配列 `a` の内容を以下のようにしてファイルに書き出そうとしたが，実行中にシステムがクラッシュした．

```
write(fd, a, sizeof a);
```

OS を再起動して確認したところ，当該ファイルには `a` の内容が途中まで反映されていた．これはログ機構の正常な動作である．

I) 複数個のプロセスからファイル操作を行う場合，複数個のトランザクションが同時に発生し得る．xv6 ではそのようなことが起こらない（つまり任意の時刻において実行中のトランザクションが高々一つになる）よう，`begin_op` において同期を行っており，あるトランザクションの実行中は別のトランザクションを実行しようとしたプロセス（カーネルスレッド）は `SLEEPING` モードで待たされる．

2. 次ページのコード 1 は各 CPU コアが実行するスケジューラ，コード 2 はプロセスがスケジューラに制御を移す際に用いる関数である．これらについて以下の問に答えよ．

(a) xv6 のプロセスの状態は UNUSED, USED, SLEEPING, RUNNABLE, RUNNING, ZOMBIE のいずれかである．コード 1 の 15 行目において p->state が取り得る値はこれらのうちどれか．該当するものをすべて答えよ．

(b) コード 1 とコード 2 で用いられている関数 swtch を使ってコード 3 のようなユーザプログラムを書いた．4〜19 行目の構造体 context は xv6 のカーネルのソースコード proc.h で定義されているものと同じである．このプログラムを実行したとき，main: 3 と表示された次の行に表示されるものを以下の選択肢 a〜e からひとつ選べ．

a) foo: 2

b) foo: 3

c) foo: 6

d) foo: 8

e) main: 4

(c) コード 3 の 40 行目を削除したプログラムを実行したとき，main: 3 と表示された次の行に表示されるものを以下の選択肢 a〜e からひとつ選べ．

a) foo: 2

b) foo: 3

c) foo: 6

d) foo: 8

e) main: 4

(d) xv6 を起動するとシェルのプロンプトが表示されるが，この状態で何もコマンドを実行しないでいると，ホスト OS の CPU 負荷が非常に高くなることがある．この現象が発生する理由をコード 1 を参照して説明せよ．

3. RISC-V 版 xv6 のファイルシステムにおいて，inode ブロックに格納される dinode 構造体は以下のように定義されている．

```
struct dinode {
  short type;             // ファイルタイプ
  short major;            // 主デバイス番号
  short minor;            // 副デバイス番号
  short nlink;            // リンク数
  uint size;              // ファイルサイズ
  uint addrs[NDIRECT+1];  // ブロック参照
};
```

マクロ NDIRECT は 12 と定義されている．addrs[0] から addrs[NDIRECT-1] の 12 個がデータブロックへの直接参照で，addrs[NDIRECT] が間接参照である．ブロック番号を表す uint 型は 4 バイト（32 ビット）である．またブロックサイズは 1024 バイトである．

(a) 間接参照ブロックを使わずに作ることができるファイルサイズをバイト数で答えよ．

(b) 157192 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必要な場合はそれも含めて数えること．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) dinode 構造体のフィールド nlink の値は，当該構造体が表すファイルがディレクトリから参照されている数を表す．ここで xv6 において以下のようなコマンドを実行したとする（$はシェルのプロンプトである）．このときの，ファイル foo/hello.txt およびディレクトリ foo を表す dinode 構造体の nlink の値をそれぞれ記せ．

```
$ mkdir foo
$ mkdir foo/bar
$ echo Hello > foo/bar/hello.txt
$ mkdir foo/baz
$ ln foo/bar/hello.txt foo/hello.txt
```

```
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 1: scheduler

```
void
sched(void)
{
  int intena;
  struct proc *p = myproc();
  if(!holding(&p->lock))
    panic("sched p->lock");
  if(mycpu()->noff != 1)
    panic("sched locks");
  if(p->state == RUNNING)
    panic("sched running");
  if(intr_get())
    panic("sched interruptible");
  intena = mycpu()->intena;
  swtch(&p->context, &mycpu()->context);
  mycpu()->intena = intena;
}
```

コード 2: sched

```
#include "kernel/types.h"
#include "user/user.h"

struct context {
  uint64 ra;
  uint64 sp;
  uint64 s0;
  uint64 s1;
  uint64 s2;
  uint64 s3;
  uint64 s4;
  uint64 s5;
  uint64 s6;
  uint64 s7;
  uint64 s8;
  uint64 s9;
  uint64 s10;
  uint64 s11;
};

void swtch(struct context*, struct context*);

struct context main_context;
struct context foo_context;

#define SDEPTH 512
__attribute__ ((aligned (16)))
uint64 foo_stack[SDEPTH];

void foo_fun(uint64 *cp) {
    printf("foo: %l\n", *cp);
    swtch(&foo_context, &main_context);
    *cp += 2;
}

void foo() {
    uint64 c = 0;
    for (;;) {
        foo_fun(&c);
        foo_fun(&c);
    }
}

int main() {
    foo_context.ra = (uint64)foo;
    foo_context.sp = (uint64)(foo_stack + SDEPTH);
    uint64 c = 0;
    for (;;) {
        printf("main: %l\n", c);
        swtch(&main_context, &foo_context);
        c += 1;
    }
    return 0;
}
```

コード 3: swtest