システムソフトウェア・試験問題

2023 年度 (2023 年 11 月 30 日・試験時間 100 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. RISC-V 版 xv6 のカーネルを構成する関数を用いて有限長バッファ（キュー）を実装した（コード 1）．関数 buff_initを 1 回だけ実行した後，関数 buff_getおよび buff_putによってバッファにデータを出し入れする．バッファが空のときに buff_getを呼び出すと，呼び出したスレッドは他のスレッドが buff_putを呼び出してバッファにデータを入れるまで待たされる．また，バッファが満杯のとき buff_putを呼び出すと，呼び出したスレッドは他のスレッドが buff_getを呼び出してバッファからデータを取り出すまで待たされる．

バッファは配列 buffを用いたリングバッファとして実装されている．変数 buff_wの値を BUFSIZEで割った余りはバッファにデータを入れる位置を，変数 buff_rの値を BUFSIZEで割った余りはバッファからデータを取り出す位置となる．データを出し入れするたびに buff_rあるいは buff_wの値を 1 ずつ増やしているが，プログラムの簡略化のため，これらの変数ははオーバフローしないものとする．

構造体 spinlock および関数 acquire，release は xv6 におけるスピンロックの実装である．関数 sleep は適当なデータ（c とする）とそれに付随するスピンロックを引数とし，呼び出したカーネルスレッドの状態を SLEEPINGにしてその実行を中断する（コード 2）．このことを当該スレッドが c についてスリープ状態にあると呼ぶ．関数 wakeup は引数についてスリープ状態にあるカーネルスレッドの状態を RUNNABLE にする（コード 2）．

```
1  #define BUFSIZE 16
2
3  int buff[BUFSIZE];
4  uint64 buff_r, buff_w;
5  struct spinlock buff_lock;
6
7  void buff_init() {
8      buff_r = buff_w = 0;
9      initlock(&buff_lock, "buff");
10 }
11
12 void buff_put(int x) {
13     acquire(&buff_lock);
14     while ( A  ==  B  + BUFSIZE) {
15         sleep(& C , &buff_lock);
16     }
17     buff[ D  % BUFSIZE] = x;
18     E ++;
19     wakeup(& F );
20     release(&buff_lock);
21 }
22
23 int buff_get() {
24     acquire(&buff_lock);
25     while ( G  ==  H ) {
26         sleep(& I , &buff_lock);
27     }
28     int x = buff[ J  % BUFSIZE];
29     K ++;
30     wakeup(& L );
31     release(&buff_lock);
32     return x;
33 }
```

コード 1: 有限長バッファの定義

(a) コード 1 の空欄 A ～ L には，変数 buff_rおよび buff_wのいずれかが入る．これら 2 つの変数それぞれについて，入るべき空欄を A～L で答えよ．

(b) スピンロック buff_lock の役割を説明せよ．

(c) コード 2 の 8 行目を実行しているときに式 p->state が取り得る値を以下から選べ．

UNUSED, USED, SLEEPING, RUNNABLE, RUNNING, ZOMBIE

```
1  void sleep(void *chan, struct spinlock *lk) {
2      struct proc *p = myproc();
3      acquire(&p->lock);
4      release(lk);
5      p->chan = chan;
6      p->state = SLEEPING;
7      sched();
8      p->chan = 0;
9      release(&p->lock);
10     acquire(lk);
11 }
12
13 void wakeup(void *chan) {
14     struct proc *p;
15     for (p = proc; p < &proc[NPROC]; p++) {
16         if (p != myproc()) {
17             acquire(&p->lock);
18             if (p->state == SLEEPING &&
19                 p->chan == chan) {
20                 p->state = RUNNABLE;
21             }
22             release(&p->lock);
23         }
24     }
25 }
```

コード 2: sleep と wakeup の定義

(d) 10 個のスレッドが buff_put および buff_get をランダムに呼び出し続けているとする．以下の (A)〜(J) のうち常に成立するものをすべて選べ．

(A) `0 <= buff_w - buff_r`

(B) `10 <= buff_w - buff_r`

(C) `0 <= buff_r - buff_w`

(D) `10 <= buff_r - buff_w`

(E) `buff_w - buff_r <= BUFSIZE`

(F) `buff_w - buff_r <= BUFSIZE + 10`

(G) `buff_w - buff_r <= BUFSIZE - 10`

(H) `BUFSIZE <= buff_r - buff_w`

(I) `BUFSIZE + 10 <= buff_r - buff_w`

(J) `BUFSIZE - 10 <= buff_r - buff_w`

(e) コード 1 の 14 行目および 25 行目の **while** を **if** に変更した場合に起こり得る不具合を一つ挙げ，その理由を説明せよ．

2. 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) RISC-V 版 xv6 では最大何バイトまでの大きさのファイルを作ることができるか．ディスクは十分大きく，ディスクサイズによる制約はないものとする．

(b) 24000 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 RISC-V 版 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/salute.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
$ echo Hello > foo/hello.txt
$ ln foo/hello.txt foo/salute.txt
$ mkdir foo/bar
$ ln foo/salute.txt foo/bar/hi.txt
```

(d) nlink の値が正しい値より小さい場合に起こりうる不具合を挙げよ．

3. 図 1 は RISC-V 版 xv6 におけるユーザプロセスのメモリ空間である．

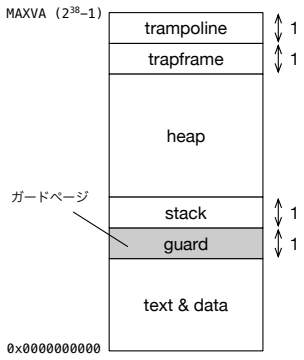

図 1: xv6 のユーザプロセスのメモリ空間

(a) テキスト・データ領域とスタック領域の間にガードページ (guard) と呼ばれる領域がある．以下から適切な用語を一つ選び，それを用いてガードページの役割について説明せよ．

トランポリン，カナリア，バッファキャッシュ，
ページフォルト，TLB，ページテーブル

(b) 図 1 におけるトランポリン (trampoline) と名付けられている領域には何が格納されているか．またその役割は何か．