システムソフトウェア・試験問題

2024 年度 (2024 年 12 月 2 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. 図 1 は 3 個のスレッド P_1, P_2, P_3 がバリア同期を行う様子を表している．

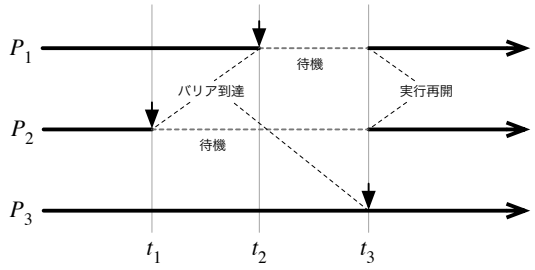

図 1: バリア同期

バリア同期とは，決められた数のスレッドがある特定の箇所（バリア）に到達するまで待機する同期手法である．図 1 では，時刻 t_1 に P_2 が，時刻 t_2 に P_1 が，時刻 t_3 に P_3 がそれぞれバリアに到達している．先に到達した P_1 と P_2 は P_3 がバリアに到達する時刻 t_3 まで待機し，t_3 に実行を再開する．

コード 1 は xv6 カーネルの関数 sleep と wakeup を用いたバリア同期機構の実装である．上で「バリアに到達」と表現しているのは関数 barrier_wait の呼び出しに相当する．N はバリア同期を行うスレッドの数である．コード 2 は barrier_wait の使用例であり，3 つのスレッドがそれぞれ関数 thread1, thread2, thread3 を実行するものとする．

(a) コード 1 の空欄 [A]〜[D] に当てはまる適切なものを以下の選択肢 (1)〜(14) から選べ．

(1) ++ (2) -- (3) = 0 (4) = 1 (5) = N
(6) < (7) <= (8) == (9) >= (10) >
(11) barrier_lock (12) &barrier_lock
(13) barrier_count (14) &barrier_count

```
1  #define N 3
2  int barrier_count = 0;
3  struct spinlock barrier_lock;
4
5  void barrier_wait(void) {
6      acquire(&barrier_lock);
7      barrier_count [ A ];
8      if (barrier_count [ B ] N) {
9          sleep( [ C ] , &barrier_lock);
10     }
11     else {
12         barrier_count [ D ];
13         wakeup(&barrier_count);
14     }
15     release(&barrier_lock);
16 }
```

コード 1: バリア同期機構の実装

```
1  int a = 0, b = 0;  // shared variables
2
3  void thread1(void) {
4      while (1) {
5          a = a + 2;
6          barrier_wait();
7          barrier_wait();
8      }
9  }
10
11 void thread2(void) {
12     while (1) {
13         b = b + 1;
14         barrier_wait();
15         barrier_wait();
16     }
17 }
18
19 void thread3(void) {
20     while (1) {
21         barrier_wait();
22         printf("%d\n", a - b);
23         barrier_wait();
24     }
25 }
```

コード 2: barrier_wait の使用例

(b) コード 1 の 11 行目と 14 行目をコメントアウトし，12, 13 行目を無条件に実行するよう変更するとどのような問題が発生するか説明せよ．

(c) コード 2 の実行結果を示せ．

(d) コード 2 の 7 行目，15 行目，23 行目の `barrier_wait` の呼び出しをすべてコメントアウトした場合，実行結果に影響はあるか．あるとしたらどのような影響か説明せよ．

2. 図 2 は xv6 のプロセスの状態遷移を表している．

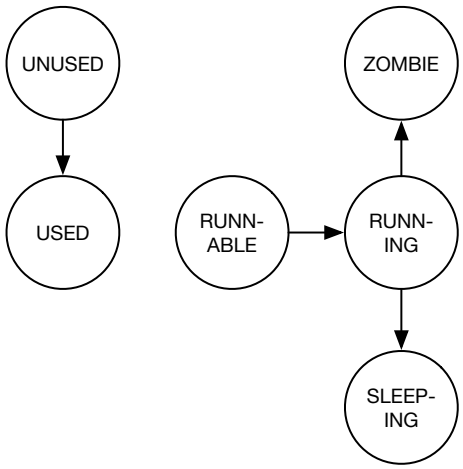

図 2: xv6 のプロセスの状態

(a) 適切な遷移を書き加えて状態遷移図を完成せよ．

(b) 各状態についての最も適切な記述を以下の選択肢 (1)〜(9) から一つずつ選べ．

(1) CPU がプログラムを実行している

(2) 起動に失敗したままメモリを占有している

(3) 親プロセスによる終了処理を待っている

(4) プロセス構造体が未使用である

(5) 入出力等が可能になるのを待っている

(6) 死んだはずのプロセスが実行を再開した

(7) プロセスの初期化を行っている

(8) 実行可能だが CPU が割り当てられていない

(9) 致命的なエラーが発生したため OS 自体の実行を停止しようとしている

(c) 以下は xv6 におけるプロセスの終了に関する記述である．

> あるプロセス P がシステムコール `exit` を実行すると [E] 状態になる．その後，P の [F] プロセスがシステムコール [G] を実行することにより，P のために使われていたプロセス構造体が解放される．

空欄 [E]〜[G] に当てはまる適切なものを以下の選択肢 (1)〜(9) から選べ．

(1) ZOMBIE	(2) SLEEPING	(3) RUNNABLE
(4) 子	(5) 親	(6) 姉妹
(7) `fork`	(8) `kill`	(9) `wait`

3. xv6 にはファイルシステムの一貫性を保つためのログ機構が実装されている．以下はログ機構に関する記述である．この記述の正誤について根拠とともに示せ．

> 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，S を実行しなかった場合と同じ状態を保持するかのいずれか一方になる．

4. xv6 のファイルシステムでは，1 個の i-node から直接参照できるデータブロックの数は最大 12 で，間接参照の数は 1 である．データブロック 1 個のサイズは 1024 バイトで，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．

(a) xv6 のファイルシステムで扱い得る最大のファイルのサイズをバイト数で答えよ．

(b) 181288 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．