システムソフトウェア・試験問題の解答と解説

2018 年度 (2018 年 11 月 26 日・試験時間 90 分)

1. (a) 9 回

(b) 5 回

解説 アクセスの様子を以下のような表で表す．表の各列は，上からアクセスのあったページの番号，そのアクセスが生じた時点で物理ページフレームに割り当てられたページを新しく割り当てられた順に並べたもの，そしてページフォルトの有無（P がページフォルトの発生を表している）である．例えば最初の表の第 3 列目（左から数えて 4 列目）は，アクセスのあったページの番号が 3，そのときページフレームに割り当てられたページは新しい順に 3,2,1 で，ページフォルトが生じたことを表している．

物理ページフレーム数が 3 の場合

	0	1	2	3	0	1	4	4	4	2	3	3
新	0	1	2	3	0	1	4	4	4	2	3	3
		0	1	2	3	0	1	1	1	4	2	2
旧			0	1	2	3	0	0	0	1	4	4
	P	P	P	P	P	P	P			P	P	

物理ページフレーム数が 4 の場合

	0	1	2	3	0	1	4	4	4	2	3	3
新	0	1	2	3	3	3	4	4	4	4	4	4
		0	1	2	2	2	3	3	3	3	3	3
			0	1	1	1	2	2	2	2	2	2
旧				0	0	0	1	1	1	1	1	1
	P	P	P	P			P					

多くのページ置換アルゴリズムでは，物理ページフレーム数が増えればページフォルトの回数は減るが，FIFO アルゴリズムでは逆に増えるケースが生じる (Bélády の例外)．この問題の例は Bélády の例外の例ではないことに注意．

(c) LRU を実現するためには各ページのアクセス時刻（タイムスタンプ）を記録する必要があるが，メモリアクセス毎にタイムスタンプを記録するのは非現実的である．

2. (a) ブロックが未使用にも関わらず対応するビットが 1 の場合，そのブロックは使用されずディスクの肥やしとなる．逆にブロックが使用されているにも関わらず対応するビットが 0 の場合，上書きされて思いがけない内容になることがある．

(b) `foo/baz`: 2，`bar`: 1，ルートディレクトリ: 3

解説 `ln` コマンドはファイルへのリンクを作成する．問題のように実行した場合，ファイル `foo/baz` へのリンクをもつ `boo` という名前のエントリをディレクトリ `bar` に作成する．inode の `nlink` の値は，その inode が表しているファイルへのリンクの個数を表しているため，ファイル `foo/baz` では 2 となる．また，ディレクトリ `bar` はルートディレクトリからの参照 1 つだけなので 1 となる．ルートディレクトリの場合，`foo` と `bar` からの `..` による参照がそれぞれ 1，加えてルートディレクトリ自身からの `..` による参照が 1 となり，計 3 となる．

(c) 3

解説 xv6 のログ機構は，`begin_op()` の実行から `end_op()` の実行に間に行われたファイルシステムへの変更（トランザクション）が，完全に実行された全く実行されなかったかのいずれかになることを保証している．トランザクションが完全に行われた後に，その内容をなかったことにすることはできない．また，ファイルシステム以外（プロセスの状態やメモリの内容）については関与しない．したがって解答は 3 のみである．

3. (a) CPU 数が 1 の場合は正しく相互排除が行われる．CPU 数が 1 より大きい場合は行われない．

解説 CPU 数が 1 の場合，`pushcli` によって（タイマ割り込みを含む）割込みが禁止されているため，最初にロックを獲得したカーネルスレッドがそのロックを解放するまで他のスレッド（および割り込みハンドラ）は動作できず，結果として相互排除が行われる．CPU 数が 1 より大きい場合は，`*var` の読み出しと書き込みの間に他の CPU によるアクセスが生じる可能性があるため，相互排除は行われない．

(b) デッドロック

解説 あるカーネルスレッド T がロック L を獲得し，それを解放する前に同じ CPU 上で割り込みが発生したとする．その割り込みのハンドラ H が L を獲得しようとした場合，H は L が解放されるまで待つ必要があるが，割り込まれた T は H の終了を待つことになり，結果としてデッドロックになる．

具体例として，`sys_sleep` 内で `tickslock` を獲得したのちにタイマ割り込みが発生すると，割り込みハンドラから呼ばれる `trap` 内で `tickslock` を獲得しようとする．

4. (a) ディスクに関連する操作では，ロックを獲得してから解放するまでの時間が長いため，スピンロックを用いるとロックの解放を待つスレッドが無駄に CPU 時間を消費することになるため．

(b) `wakeup` によって複数個のスレッドが再開する可能性があるので，再開後に最初に `lk->locked` に 1 を代入した（ソースコード 2 の 10 行目）スレッドのみがロックを獲得することを保証するため．