システムソフトウェア・試験問題の解答と解説

2019 年度 (2019 年 11 月 25 日・試験時間 90 分)

1. (a) 274432 バイト

解説 直接参照が $12 \times 1024 = 12288$ バイト，関節参照が $1024/4 \times 1024 = 262144$ バイトなので合計 274432 バイト．268K バイト（268Ki バイト）でも可．

(b) 21 個

解説 $\lceil 20000/1024 \rceil = 20$ なのでデータ格納に必要なデータブロックは 20 個．直接参照できる範囲（12 個）を超えているため間接参照ブロック 1 個が必要になり，合計 21 個となる．

(c) `foo/hello.txt`: 2, `foo`: 2

解説 `foo/hello.txt` に対応する inode はディレクトリ `foo` と `foo/bar` からリンクされている．ディレクトリ `foo` はその親ディレクトリに加え，`foo/bar` から `..` でリンクされている．

(d) 正しい値より小さい場合，ディレクトリからのリンクが残っているにも関わらずファイルの実体（inode およびそれから参照されているデータブロック）が未使用とみなされてしまう．正しい値より大きい場合，ディレクトリからのリンクが全て消去されてもファイルの実体が残ってしまう．

2. (a) スタックの溢れを検出する．

(b) (2)

解説 RISC-V 版 xv6 ではユーザプロセスへのガードページへのアクセスを禁止している．関数 `exec` の定義中（コード 1）で，2 ページを確保したのち，1 ページを関数 `uvmclear`（コード 2）を使ってアクセスを禁止してガードページとしている．こうすることで，スタックが溢れたときに物理ページの割り当てのないガードページにアクセスすることになり，その結果としてページフォルトが発生する．

```
66 // Allocate two pages at the next page boundary.
67 // Use the second as the user stack.
68 sz = PGROUNDUP(sz);
69 if((sz = uvmalloc(pagetable, sz, sz + 2*PGSIZE))
       == 0)
70   goto bad;
71 uvmclear(pagetable, sz-2*PGSIZE);
72 sp = sz;
73 stackbase = sp - PGSIZE;
```

コード 1: スタックとガードページの割り当て (exec.c)

```
344 // mark a PTE invalid for user access.
345 // used by exec for the user stack guard page.
346 void
347 uvmclear(pagetable_t pagetable, uint64 va)
348 {
349   pte_t *pte;
350
351   pte = walk(pagetable, va, 0);
352   if(pte == 0)
353     panic("uvmclear");
354   *pte &= ~PTE_U;
355 }
```

コード 2: ページのアクセス禁止 (vm.c)

3. (a) この変更により構造体 `semaphore` のフィールド `count` へのアクセスが排他的でなくなり，`count` の値が 0 より小さくなる可能性がある．

これに加えて，`acquire(&s->lock)` を実行せずに `sleep(s, &s->lock)` を実行した場合，確保していない `s->lock` をに対して `release` が呼ばれることになる．この場合 xv6 では関数 `panic` が呼ばれ，OS の実行が停止する．

解説　前半（第 1 パラグラフ）が説明できていればよい．後半もできた場合は加点.

(b)　K 個

(c)　$N_V - N_P$

(d)　$N_P - N_V$

(e)*　(b) と (c) の結果より

$$\texttt{sem.count} = K - N_{CS}.$$

一方 P と V の定義より，$\texttt{sem.count} \geq 0$ が常に成り立つ．よって $N_{CS} \leq K$ となる.