システムソフトウェア　小課題１：レポート

提出日：2018 年 10 月 26 日

系／学科／類：情報工学系

学籍番号：16B13354

ログイン名：hoshino.s.af

氏名：星野　シンジ

1　sleep_sec() システムコールの動作概要

まず、最初に cmostime を用いて現在の rtcdate 構造体を取得する。その上で、得られた rtcdate の構造体から引数秒後の rtcdate 構造体を計算する。この rtcdate 構造体は変数として持っておく。

次に、実際にプロセスをスリープロックさせる部分だが、最初に tickslock を acquire してスピンロックの確保している。そして、スピンロックを確保した後は、while 文が４０回ループするごとに cmostime で現在の時刻を確認し、先に計算した引数秒後の rtcdate と等しい、またはそれよりも未来になれば、while 文を脱するように実装した。また、ループする時に毎回 sleep を呼び出している。

最後に、while 文を抜けた後は、release で tickslock を解放している。

ソースコードの一部を図 1 に示しておく。

```
1   int sys_sleep_sec(void) {
2     int n;
3     struct rtcdate dp0, dp;
4
5     // returns error if the int argument was negative
6     if (argint(0, &n) < 0)
7       return -1;
8
9     cmostime(&dp0);
10
11    // calculate the rtcdate after n seconds
12    dp0.second += n;
13    // ... （長いので省略）
14
15    int l = 0;
16    // sleep until the current rtcdate dp is ahead of dp0
17    acquire(&tickslock);
18    while (1) {
19      if (myproc()->killed) {
20        release(&tickslock);
21        return -1;
22      }
23
24      l++;
25      l = l % 40;
26      if (l == 0) { // only call costime every 40 ticks to save resource
27        cmostime(&dp);
28        if (dp.second >= dp0.second && dp.minute >= dp0.minute &&
29            dp.hour >= dp0.hour && dp.day >= dp0.day && dp.month >= dp0.month &&
30            dp.year >= dp0.year) {
31          break;
32        }
33      }
34      sleep(&ticks, &tickslock);
35    }
36    release(&tickslock);
37
38    return 0;
39  }
```

図 1: sleep_sec() システムコールのコード一部

2　実装についての考察

sleep_sec のシステムコールが必要とする資源は、uint 型の ticks と spinlock 構造体の tickslock である。それらを使用している間は、tickslock を aquire することによって、排他制御をしている。従って、sleep_sec のシステムコールが OS の他の動作に影響を与えることは基本的にない。ただし、計算資源を使うので、それに関連する影響は生じる可能性がある。

計算資源の利用ができるだけ少なくなるようにいくつか工夫した。

まず、引数秒後の rtcdate の計算をする部分では、sleep_sec() の引数が小さい場合に、無駄な計算をすることがないように、秒の計算から行なっている。具体的には、最初に rtcdate の second に引数を足す。その結果 second が 60 を超えていれば、minute に 60 で割った商を加え、second は余りとする。これと同じように、年まで計算することにより、秒より単位が大きい時間は必要がある時にだけ計算されるようになっている。

次に、引数秒後の時間になるまでの while 文については、cmostime の実行に計算資源が多く必要であることを考慮し、40ticsk(約一秒) ごとにのみ、cmostime を呼び出し及びループの抜け出しの条件確認をするようにしている。

3　実装までの経緯

スリープロックで引数秒間プロセスをスリープさせる sleep_sec システムコールを実装するというのが今回の課題だった。よって、スリープロックが xv6 のどこかに実装がないかを探すところから始め、sleep システムコールを見つけた。すでに実装されている sleep システムコールの引数を tick 数ではなく、秒数に変更するという方針で実装を試みた。従って、sleep_sec システムコールの基本的な構造は、sleep システムコールと同じになり、rtcdate 構造体と cmostime の扱いが主な注目点となった。特に、引数秒後の rtcdate 構造体の計算をする際に、うるう年等の処理のコードが最も長い。