基本ソフトウェア特論第 12 週課題
学籍番号 1cjnm001 氏名井上　駿佑
課題
哲学者の食事問題のプログラムを作成せよ。実行結果を示しなさい

```
1   #include<stdio.h>
2   #include<pthread.h>
3   #include<semaphore.h>
4   #include <unistd.h>
5   //バッファの数
6   #define N 5
7   //状態
8   #define THINKING 0//考えている状態
9   #define HUNGRY 1//空腹状態
10  #define EATING 2//食事中
11
12  //N 個の状態保存配列
13  int state[N]={THINKING,THINKING,THINKING,THINKING,THINKING};
14  //semaphore
15  sem_t mutex;
16  sem_t s[N];
17  //THINKING の処理
18  void think(){
19      sleep(1);
20  }
21
22  void test(int phnum){
23      int right,left;
24      right=(phnum+N-1)%N;
25      left=(phnum+1)%N;
26      //哲学者番号 phnum が食べようとしていて彼の左右の哲学者が食事していない
    時
27      if (state[phnum] == HUNGRY && state[right] != EATING &&
    state[left] != EATING) {// 食べられる！！W！
28          state[phnum] = EATING;//状態を EATING にする
```

```
29          printf("¥nPhilosopher %d takes fork %d and %d¥n",phnum,
    right, left);
30          sem_post(&s[phnum]);//phnum 番のフォークを置く
31      }
32  }
33  //食事中
34  void eat(){
35      sleep(1);
36  }
37  //フォークを取る
38  void take_fork(int phnum){
39      sem_wait(&mutex);
40      state[phnum]=HUNGRY;//空腹状態のする
41      test(phnum);//フォークを取りに行く
42      sem_post(&mutex);
43      sem_wait(&s[phnum]);//phbum のフォークを取れるまで待機
44  }
45  void put_fork(int phnum){
46      int right,left;
47      sem_wait(&mutex);
48      right=(phnum+N-1)%N;
49      left=(phnum+1)%N;
50      state[phnum] = THINKING;//phnum の哲学者は考える状態にする
51      // 左右の哲学者に声をかける
52      //（両隣の哲学者の状態を変更）
53      test(right);
54      test(left);
55      sem_post(&mutex);
56  }
57  void *philosopher(int i){
58      int loop=0;
59
60      while (loop<N) {
61          think( ); // sleep(1); で 1 秒思案中
62          take_fork(i);//フォークをとる or ブロック
63          eat();  //sleep(1); で 1 秒で食事
```

```
64          put_fork(i);//フォークを置き、// 両隣の哲学者に声をかける
65          loop++;
66      }
67  }
68
69  void main(void){
70
71      int i;
72      pthread_t thread_id[5];//スレッド 2 個分の管理領域
73      //関数ポインタ宣言
74
75      void *exe[])(int)
    ={philosopher,philosopher,philosopher,philosopher,philosopher};

76      //バイナリセマフォ
77      sem_init(&mutex, 0, 1);
78      for(i=0;i<5;i++){
79          sem_init(&s[i],0,1);
80      }
81      //data の初期化

82      //関数 philosopher を開始
83      pthread_create(&thread_id[0],NULL,exe[0],0);
84      pthread_create(&thread_id[1],NULL,exe[1],1);
85      pthread_create(&thread_id[2],NULL,exe[2],2);
86      pthread_create(&thread_id[3],NULL,exe[3],3);
87      pthread_create(&thread_id[4],NULL,exe[4],4);
88       //スレッド終了を待つ
89      for(i=0;i<5;i++){
90          pthread_join(thread_id[i],NULL);
91      }
92  }
```

実行結果

```
Philosopher 0 takes fork 4 and 1

Philosopher 3 takes fork 2 and 4

Philosopher 2 takes fork 1 and 3

Philosopher 0 takes fork 4 and 1

Philosopher 3 takes fork 2 and 4

Philosopher 1 takes fork 0 and 2

Philosopher 4 takes fork 3 and 0

Philosopher 3 takes fork 2 and 4

Philosopher 0 takes fork 4 and 1

Philosopher 2 takes fork 1 and 3

Philosopher 4 takes fork 3 and 0

Philosopher 1 takes fork 0 and 2

Philosopher 3 takes fork 2 and 4

Philosopher 0 takes fork 4 and 1

Philosopher 2 takes fork 1 and 3

Philosopher 4 takes fork 3 and 0

Philosopher 3 takes fork 2 and 4
```

Philosopher 1 takes fork 0 and 2

Philosopher 2 takes fork 1 and 3

Philosopher 4 takes fork 3 and 0

Philosopher 1 takes fork 0 and 2