第 10 回　pthread

授　業 名	基本ソフトウェア特論		
演　習 日	2021 年 12 月 06 日（月）	提出日	2021 年 12 月　06 日（月）

専攻 名	情報通信学専攻
学籍番号	1CJNM001
氏　　名	井上　駿佑

教員記入欄

提出確認	完了

特記事項

課題 1
POSIXAPI「pThaead」のプログラムを実行せよ
プログラム

行	1cjnm001_inoue_week10_print.c
1	#include<stdio.h>
2	#include<pthread.h>
3	void *FuncA(char *arg);
4	void *FuncB(void);
5	void *FuncA(char *arg){
6	printf("%s",arg);//ポインタ arg から始まる文字列表示
7	}
8	void *FuncB(void){
9	printf("井上　駿佑¥n");
10	}
11	int main(void){
12	//スレッドの個数カウント
13	int i;
14	pthread_t thread_id[2];//スレッド 2 個分の管理領域
15	char num[10]="1CJNM001¥n¥0";
16	//関数 FuncA を開始
17	pthread_create(&thread_id[0],NULL,FuncA,&num);//num のアドレスを渡す
18	//関数 FuncB を開始
19	pthread_create(&thread_id[1],NULL,FuncB,NULL);
20	
21	for(i=0;i<2;i++){
22	//スレッド終了を待つ
23	pthread_join(thread_id[i],NULL);
24	}
25	}

実行結果

```
inoue@DESKTOP-GAHI01I:/mnt/f/2021_autumn/基本ソフトウェア特論$ gcc 1cjnm001_inoue_week10_print.c -pthread
1cjnm001_inoue_week10_print.c: In function 'main':
1cjnm001_inoue_week10_print.c:17:36: warning: passing argument 3 of 'pthread_create' from incompatible pointer type [-
Wincompatible-pointer-types]
  pthread_create(&thread_id[0],NULL,FuncA,&num);//numのアドレスを渡す

In file included from 1cjnm001_inoue_week10_print.c:2:0:
/usr/include/pthread.h:234:12: note: expected 'void * (*)(void *)' but argument is of type 'void * (*)(char *)'
 extern int pthread_create (pthread_t *__restrict __newthread,

1cjnm001_inoue_week10_print.c:19:36: warning: passing argument 3 of 'pthread_create' from incompatible pointer type [-
Wincompatible-pointer-types]
  pthread_create(&thread_id[1],NULL,FuncB,NULL);

In file included from 1cjnm001_inoue_week10_print.c:2:0:
/usr/include/pthread.h:234:12: note: expected 'void * (*)(void *)' but argument is of type 'void * (*)(void)'
 extern int pthread_create (pthread_t *__restrict __newthread,

inoue@DESKTOP-GAHI01I:/mnt/f/2021_autumn/基本ソフトウェア特論$ ./a.out
1CJNM001
井上　駿佑
inoue@DESKTOP-GAHI01I:/mnt/f/2021_autumn/基本ソフトウェア特論$
```

考察

Pthread は pthread_create 関数の第 4 引数が*void 型になっていた。これは起動する関数にポインタのアドレスを渡せるようだった。void 型は c 言語ではすべての型を受け付

けることもできる。よって FuncA が引数*char を取るようにした。＊void 型を＊char 型に変数型変換する事が手間だと感じたので今回は FuncA で*char 型で関数定義している。今回は文字列型を使ったためポインタの値(*char)を使ったが普通に int 型でも使える。またスレッドが 2 つ（FuncA と FuncB）ともに正常に動いている

課題 2 以下のプログラムの実行結果をまとめよ

行	1cjnm001_inoue_week10.c
1	#include<stdio.h>
2	#include<pthread.h>
3	#define N 5000
4	//共有変数 x
5	int x;
6	void *FuncA(void){
7	int i;//カウンタ変数 i の宣言
8	for(i=0;i<N;i++){//N 回繰り返し(i が 0~1000 未満まで実行)
9	x++;//カウントアップ（x を 1 加算する）
10	}
11	}
12	void *FuncB(void){
13	int i;//カウンタ変数 i の宣言
14	for(i=0;i<N;i++){//N 回繰り返し(i が 0~1000 未満まで実行)
15	x++;//カウントアップ（x を 1 加算する）
16	}
17	}
18	void main(void){
19	int i;
20	pthread_t thread_id[2];//スレッド 2 個分の管理領域
21	//関数 FuncA を開始
22	pthread_create(&thread_id[0],NULL,FuncA,NULL);
23	//関数 FuncB を開始
24	pthread_create(&thread_id[1],NULL,FuncB,NULL);
25	//スレッド終了を待つ
26	for(i=0;i<2;i++){
27	pthread_join(thread_id[i],NULL);
28	}
29	//カウントの表示
30	printf("¥nx=%d¥n",x);
31	}

。

ループ回数 N	正解 x	成功回数
1000	2000	10
5000	10000	8

考察

このプログラムは FuncA と FuncB が終了するまで動いている。そして競合状態になることがある。競合状態になる共有変数 x の値が小さくなることがある。

課題 3 mutex を使って競合状態を回避せよ

行	1cjnm001_inoue_week10_mutex.c
1	#include<stdio.h>
2	#include<pthread.h>
3	#include<semaphore.h>
4	#define N 1000
5	//共有変数 x
6	int x;
7	sem_t mutex; //binary semaphore
8	void *FuncA(void){
9	int i;//カウンタ変数 i の宣言
10	for(i=0;i<N;i++){//N 回繰り返し(i が 0~1000 未満まで実行)
11	//セマフォ取得
12	sem_wait(&mutex);
13	x++;//カウントアップ（x を 1 加算する）
14	//セマフォ戻す
15	sem_post(&mutex);
16	}
17	
18	}
19	void *FuncB(void){
20	int i;//カウンタ変数 i の宣言
21	for(i=0;i<N;i++){//N 回繰り返し(i が 0~1000 未満まで実行)
22	sem_wait(&mutex);
23	x++;//カウントアップ（x を 1 加算する）
24	sem_post(&mutex);
25	}
26	}
27	void main(void){
28	int i;
29	pthread_t thread_id[2];//スレッド 2 個分の管理領域
30	sem_init(&mutex, 0, 1);
31	//1 にセットしておく
32	//関数 FuncA を開始
33	pthread_create(&thread_id[0],NULL,FuncA,NULL);
34	//関数 FuncB を開始
35	pthread_create(&thread_id[1],NULL,FuncB,NULL);
36	//スレッド終了を待つ
37	for(i=0;i<2;i++){
38	pthread_join(thread_id[i],NULL);
39	}

```
40          //カウントの表示 2
41          printf("¥nx=%d¥n",x);
42  }
```

考察

Pthread を扱ったスレッドコンピューティングは競合状態をしないように工夫すべきと思った。また依存関係のない処理をすべきで大きなシステムでスレッドコンピューティングをすべきと思った。