1. 소스코드

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <pthread.h>  void \*ssu\_thread1(void \*arg);  void \*ssu\_thread2(void \*arg);  //뮤텍스 변수 초기화  pthread\_mutex\_t mutex1 = PTHREAD\_MUTEX\_INITIALIZER;  pthread\_mutex\_t mutex2 = PTHREAD\_MUTEX\_INITIALIZER;  //조건 변수 초기화  pthread\_cond\_t cond1 = PTHREAD\_COND\_INITIALIZER;  pthread\_cond\_t cond2 = PTHREAD\_COND\_INITIALIZER;  int count = 0;  int input = 0;  int t1 = 0, t2 = 0;  int main(void) {  pthread\_t tid1, tid2;  int status;  //스레드 생성  if(pthread\_create(&tid1, NULL, ssu\_thread1, NULL) != 0) {  fprintf(stderr, "pthread\_create error\n");  exit(1);  }  if(pthread\_create(&tid2, NULL, ssu\_thread2, NULL) != 0) {  fprintf(stderr, "pthread\_create error\n");  exit(1);  }  while(1) {  printf("2개 이상의 개수 입력 : ");  scanf("%d", &input);  if(input >= 2) {  //cond1을 기다리고 있는 스레드를 재시작  pthread\_cond\_signal(&cond1);  break;  }  }  //스레드가 종료할 때까지 대기  pthread\_join(tid1, (void \*)&status);  pthread\_join(tid2, (void \*)&status);  printf("complete\n");  exit(0);  }  void \*ssu\_thread1(void \*arg) {  while(1) {  //lock  pthread\_mutex\_lock(&mutex1);  if(input < 2)//cond1을 받을 때까지 대기  pthread\_cond\_wait(&cond1, &mutex1);  if(input == count) {  //cond2를 기다리고 있는 스레드를 재시작  pthread\_cond\_signal(&cond2);  break;  }  if(count == 0) {  t2++;  count++;  printf("Thread 1 : %d\n", t1);  }  else if(count % 2 == 0) {  t1 += t2;  count++;  printf("Thread 1 : %d\n", t1);  }  //cond2를 기다리고 있는 스레드를 재시작  pthread\_cond\_signal(&cond2);  //cond1을 받을 때까지 대기  pthread\_cond\_wait(&cond1, &mutex1);  //unlock  pthread\_mutex\_unlock(&mutex1);  }  return NULL;  }  void \*ssu\_thread2(void \*arg) {  while(1) {  //lock  pthread\_mutex\_lock(&mutex2);  if(input < 2)  //cond2를 받을 때까지 대기  pthread\_cond\_wait(&cond2, &mutex2);  if(input == count) {  //cond1을 기다리고 있는 스레드를 재시작  pthread\_cond\_signal(&cond1);  break;  }  if(count == 1) {  count++;  printf("Thread 2 : %d\n", t2);  }  else if(count % 2 == 1) {  t2 += t1;  count++;  printf("Thread 2 : %d\n", t2);  }  //cond1을 기다리고 있는 스레드를 재시작  pthread\_cond\_signal(&cond1);  //cond2를 받을 때까지 대기  pthread\_cond\_wait(&cond2, &mutex2);  //unlock  pthread\_mutex\_unlock(&mutex2);  }  return NULL;  } |

2. 실행결과

