#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#define BUFFER\_SIZE 3

int buffer[BUFFER\_SIZE];

int fill = 0;

int use = 0;

int count = 0;

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

pthread\_cond\_t empty = PTHREAD\_COND\_INITIALIZER;

pthread\_cond\_t full = PTHREAD\_COND\_INITIALIZER;

void put(int value) {

buffer[fill] = value;

fill = (fill + 1) % BUFFER\_SIZE;

count++;

}

int get() {

int tmp = buffer[use];

use = (use + 1) % BUFFER\_SIZE;

count--;

return tmp;

}

void \*producer(void \*arg) {

int i;

for (i = 0; i < 4; i++) {

pthread\_mutex\_lock(&mutex);

while (count == BUFFER\_SIZE) {

printf("Buffer is full, producer is waiting...\n");

pthread\_cond\_wait(&empty, &mutex);

}

printf("Producer produced %d\n", i);

put(i);

pthread\_cond\_signal(&full);

pthread\_mutex\_unlock(&mutex);

}

return NULL;

}

void \*consumer(void \*arg) {

int i, tmp = 0;

for (i = 0; i < 1; i++) {

pthread\_mutex\_lock(&mutex);

while (count == 0) {

printf("Buffer is empty, consumer is waiting...\n");

pthread\_cond\_wait(&full, &mutex);

}

tmp = get();

printf("Consumer consumed %d\n", tmp);

pthread\_cond\_signal(&empty);

pthread\_mutex\_unlock(&mutex);

}

return NULL;

}

int main(int argc, char \*argv[]) {

pthread\_t producer\_thread, consumer\_thread;

pthread\_create(&consumer\_thread, NULL, consumer, NULL);

pthread\_join(consumer\_thread, NULL);

pthread\_create(&producer\_thread, NULL, producer, NULL);

pthread\_join(producer\_thread, NULL);

pthread\_join(producer\_thread, NULL);

pthread\_join(producer\_thread, NULL);

pthread\_join(producer\_thread, NULL);

return 0;

}

OUTPUT

Buffer is empty, consumer is waiting...