Ejemplos usando variables de condición

Este ejemplo fue tomado de los apuntes de Remzi (original):

MAIN PROGRAM

```
int main(int argc, char *argv[]) {
  max = atoi(argv[1]);
  loops = atoi(argv[2]);
  consumers = atoi(argv[3]);
  buffer = (int *) Malloc(max * sizeof(int));
  pthread_t pid, cid[CMAX];
  Pthread_create(&pid, NULL, producer, NULL);
  for (int i = 0; i < consumers; i++)
    Pthread_create(&cid[i], NULL, consumer, NULL);
    Pthread_join(pid, NULL);
  for (i = 0; i < consumers; i++)
    Pthread_join(cid[i], NULL);
}
```

QUEUE GET/PUT

```
void do_fill(int value) {
  buffer[fillptr] = value;
  fillptr = (fillptr + 1) % max;
  numfull++;
}

int do_get() {
  int tmp = buffer[useptr];
  useptr = (useptr + 1) % max;
  numfull--;
  return tmp;
}
```

Solution v1 (Single CV)

```
void *producer(void *arg) {
  for (int i = 0; i < loops; i++) {
    Mutex_lock(&m); // p1
    while (numfull == max) // p2
      Cond_wait(&cond, &m); // p3
    do_fill(i); // p4
    Cond_signal(&cond); // p5
    Mutex_unlock(&m); // p6
  }
}

void *consumer(void *arg) {
  while (1) {
    Mutex_lock(&m); // c1
    while (numfull == 0) // c2
      Cond_wait(&cond, &m); // c3
    int tmp = do_get(); // c4
    Cond_signal(&cond); // c5
   Mutex_unlock(&m); // c6
```

```
    printf("%d\n", tmp);
  }
}
```

Solution v2 (2 CVs, "if")

```
void *producer(void *arg) {
  for (int i = 0; i < loops; i++) {
    Mutex_lock(&m); // p1
    if (numfull == max) // p2
      Cond_wait(&empty, &m); // p3
    do_fill(i); // p4
    Cond_signal(&fill); // p5
    Mutex_unlock(&m); // p6
  }
}

void *consumer(void *arg) {
  while (1) {
    Mutex_lock(&m); // c1
    if (numfull == 0) // c2
      Cond_wait(&fill, &m); // c3
    int tmp = do_get(); // c4
    Cond_signal(&empty); // c5
    Mutex_unlock(&m); // c6
    printf("%d\n", tmp);
  }
}
```

Solution v3 (2 CVs, "while")

```
void *producer(void *arg) {
  for (int i = 0; i < loops; i++) {
    Mutex_lock(&m); // p1
    while (numfull == max) // p2
      Cond_wait(&empty, &m); // p3
    do_fill(i); // p4
    Cond_signal(&fill); // p5
    Mutex_unlock(&m); // p6
  }
}

void *consumer(void *arg) {
  while (1) {
    Mutex_lock(&m); // c1
    while (numfull == 0) // c2
      Cond_wait(&fill, &m); // c3
    int tmp = do_get(); // c4
    Cond_signal(&empty); // c5
    Mutex_unlock(&m); // c6
    printf("%d\n", tmp);
  }
}
```

Solution v4 (2 CVs, "while", unlock)

```
void *producer(void *arg) {
  for (int i = 0; i < loops; i++) {
    Mutex_lock(&m); // p1
    while (numfull == max) // p2
      Cond_wait(&empty, &m); // p3
    Mutex_unlock(&m); // p3a
    do_fill(i); // p4
    Mutex_lock(&m); // p4a
    Cond_signal(&fill); // p5
    Mutex_unlock(&m); // p6
  }
}

void *consumer(void *arg) {
  while (1) {
    Mutex_lock(&m); // c1
    while (numfull == 0) // c2
      Cond_wait(&fill, &m); // c3
    Mutex_unlock(&m); // c3a
    int tmp = do_get(); // c4
    Mutex_lock(&m); // c4a
    Cond_signal(&empty); // c5
    Mutex_unlock(&m); // c6
  }
}
```

Archivos

Los siguientes archivos muestran la implementación en código de los ejemplos cuyo pseudocódigo se expuso anteriormente:

mythreads.h

```
#ifndef __MYTHREADS_h__
#define __MYTHREADS_h__

#include <pthread.h>
#include <assert.h>
#include <sched.h>

#ifdef __linux__
#include <semaphore.h>
#endif

#define Pthread_create(thread, attr, start_routine, arg) assert(pthread_create(thread,
attr, start_routine, arg) == 0);
#define Pthread_join(thread, value_ptr)                  assert(pthread_join(thread,
value_ptr) == 0);

#define Pthread_mutex_lock(m)                            assert(pthread_mutex_lock(m) ==
0);
#define Pthread_mutex_unlock(m)                          assert(pthread_mutex_unlock(m)
== 0);
#define Pthread_cond_signal(cond)
assert(pthread_cond_signal(cond) == 0);
#define Pthread_cond_wait(cond, mutex)                   assert(pthread_cond_wait(cond,
mutex) == 0);

#define Mutex_init(m)                                    assert(pthread_mutex_init(m,
NULL) == 0);
#define Mutex_lock(m)                                    assert(pthread_mutex_lock(m) ==
0);
#define Mutex_unlock(m)                                         assert(pthread_mutex_unlock(m)
== 0);
#define Cond_init(cond)                                         assert(pthread_cond_init(cond,
NULL) == 0);
#define Cond_signal(cond)
assert(pthread_cond_signal(cond) == 0);
#define Cond_wait(cond, mutex)                                  assert(pthread_cond_wait(cond,
mutex) == 0);

#ifdef __linux__
#define Sem_init(sem, value)                                    assert(sem_init(sem, 0, value)
== 0);
#define Sem_wait(sem)                                           assert(sem_wait(sem) == 0);
#define Sem_post(sem)                                           assert(sem_post(sem) == 0);
#endif // __linux__

#endif // __MYTHREADS_h__
```

main-pc.c

```
#include <stdio.h>
#include <unistd.h>
#include <assert.h>
#include <pthread.h>
#include <semaphore.h>

#include "mythreads.h"

int max;
int loops;
int *buffer; // have to allocate space

int useptr  = 0;
int fillptr = 0;
int numfull = 0;

pthread_cond_t empty  = PTHREAD_COND_INITIALIZER;
pthread_cond_t fill   = PTHREAD_COND_INITIALIZER;
pthread_mutex_t m     = PTHREAD_MUTEX_INITIALIZER;

#define CMAX (10)
int consumers = 1;

int verbose = 1;


void do_fill(int value)
{
    buffer[fillptr] = value;
    fillptr = (fillptr + 1) % max;
    numfull++;
}

int do_get()
{
    int tmp = buffer[useptr];
    useptr = (useptr + 1) % max;
    numfull--;
    return tmp;
}
```

```c
void *
producer(void *arg)
{
    int i;
    for (i = 0; i < loops; i++) {
       Mutex_lock(&m); // p1
       while (numfull == max) //p2
           Cond_wait(&empty, &m); // p3
       do_fill(i); // p4
       Cond_signal(&fill); //p5
       Mutex_unlock(&m); // p6
    }

    // end case
    for (i = 0; i < consumers; i++) {
       Mutex_lock(&m);
       while (numfull == max)
           Cond_wait(&empty, &m);
       do_fill(-1);
       Cond_signal(&fill);
       Mutex_unlock(&m);
    }

    return NULL;
}

void *
consumer(void *arg)
{
    int tmp = 0;
    while (tmp != -1) { // end case
       Mutex_lock(&m); //c1
       while (numfull == 0) //c2
           Cond_wait(&fill, &m); //c3
       tmp = do_get(); //c4
       Cond_signal(&empty); //c5
       Mutex_unlock(&m); //c6
       if (verbose) printf("%d\n", tmp);
    }
    return NULL;
}

int main(int argc, char *argv[])
{
    if (argc != 4) {
       fprintf(stderr, "usage: %s <buffersize> <loops> <consumers>\n", argv[0]);
       exit(1);
    }
    max   = atoi(argv[1]);
    loops = atoi(argv[2]);
    consumers = atoi(argv[3]);
    assert(consumers <= CMAX);

    buffer = (int *) Malloc(max * sizeof(int));
    int i;
    for (i = 0; i < max; i++) {
       buffer[i] = 0;
    }

    pthread_t pid, cid[CMAX];
    Pthread_create(&pid, NULL, producer, NULL);
    for (i = 0; i < consumers; i++) {
       Pthread_create(&cid[i], NULL, consumer, NULL);
    }
    Pthread_join(pid, NULL);
    for (i = 0; i < consumers; i++) {
        Pthread_join(cid[i], NULL);
    }
    return 0;
}
```

Códigos completos

En la siguiente tabla se muestra el código completo asociado a cada caso de estudio:

Caso	Descripción	Archivos asociados
1	☐ Solution v1 (Single CV)	mythreads.h, main_v1.c
2	☐ Solution v2 (2 CVs, "if")	mythreads.h, main_v2.c
3	☐ Solution v3 (2 CVs, "while")	mythreads.h, main_v3.c
4	☐ Solution v4 (2 CVs, "while", unlock)	mythreads.h, main_v4.c

Todos estos códigos se encuentra agrupados en el directorio codigos_CV.zip