```
// POSIX CONDITION VARIABLES
// Illustration of pthread_cond_signal()
// NOTE:
// The manuals say that pthread_cond_signal() unblocks at least 1 thread
// In this system, pthread_cond_signal() had to be called as many times
// as the number of threads;
// otherwise only one of them will be unblocked
// FILE: cond_signal_nc.c
// (equal to cond_signal_nc.c but does not check results of system calls)
// gcc cond_signal_nc.c -lpthread -lrt -Wall -o cond_signal_nc
// JAS
//=======================================================================
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <pthread.h>
//=======================================================================
#define NTHREADS 5
//=======================================================================
int conditionMet = 0;
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
//=======================================================================
void *threadFunc(void *arg)
{
  int threadNum = *(int *)arg;
  pthread_mutex_lock(&mutex);
  while (!conditionMet)
  {
    printf("Thread %d blocked because condition is not met\n", threadNum);
    pthread_cond_wait(&cond, &mutex);
  }
  printf("Thread %d executing critical section for 5 seconds ...\n", threadNum);
  sleep(5);
  printf("Thread %d ended execution of critical section.\n", threadNum);
  pthread_mutex_unlock(&mutex);

  return NULL;
}
//=======================================================================
int main(int argc, char *argv[])
{
  int i, j;
  int threadnum[NTHREADS];
  pthread_t threadId[NTHREADS];

  printf("Main thread: creating %d threads\n", NTHREADS);
  for(i=0; i<NTHREADS; ++i)
  {
    threadnum[i]=i+1;
    pthread_create(&threadId[i], NULL, threadFunc, (void*) &threadnum[i]);
  }

  printf("Main thread:\n doing some work until condition is met ...\n");
  sleep(10);
  //The condition has occured ...! Don't ask me what condition or why ...
  //Set the flag and wake up any waiting threads
  pthread_mutex_lock(&mutex);
  conditionMet = 1;
  printf("Main thread:\n the condition was met;\n waking up all waiting threads,
using pthread_cond_signal()...\n");

  for (j=0; j<NTHREADS; j++) //comment this line (just this one!) and see what happens :-(
    pthread_cond_signal(&cond);

  pthread_mutex_unlock(&mutex);

  printf("Main thread: waiting for threads and cleanup\n");
  for (i=0; i<NTHREADS; ++i)
  {
    pthread_join(threadId[i], NULL);
  }

  pthread_cond_destroy(&cond);
  pthread_mutex_destroy(&mutex);

  printf("Main thread: exiting.\n");
  return 0;
}
```

```
/*
pinguim> ./cond_signal_nc
Main thread: creating 5 threads
Thread 3 blocked because condition is not met
Thread 1 blocked because condition is not met
Thread 4 blocked because condition is not met
Main thread:
 doing some work until condition is met ...
Thread 2 blocked because condition is not met
Thread 5 blocked because condition is not met
Main thread:
 the condition was met;
 waking up all waiting threads, using pthread_cond_signal()...
Main thread: waiting for threads and cleanup
Thread 3 executing critical section for 5 seconds ...
Thread 3 ended execution of critical section.
Thread 5 executing critical section for 5 seconds ...
Thread 5 ended execution of critical section.
Thread 4 executing critical section for 5 seconds ...
Thread 4 ended execution of critical section.
Thread 1 executing critical section for 5 seconds ...
Thread 1 ended execution of critical section.
Thread 2 executing critical section for 5 seconds ...
Thread 2 ended execution of critical section.
Main thread: completed.
pinguim>

AFTER COMMENTING THE LINE ABOVE REFERRED:

pinguim> ./cond_signal
Main thread: creating 5 threads
Thread 2 blocked because condition is not met
Thread 1 blocked because condition is not met
Thread 3 blocked because condition is not met
Main thread:
 doing some work until condition is met ...
Thread 4 blocked because condition is not met
Thread 5 blocked because condition is not met
Main thread:
 the condition was met;
 waking up all waiting threads, using pthread_cond_signal()...
Main thread: waiting for threads and cleanup
Thread 2 executing critical section for 5 seconds ...
Thread 2 ended execution of critical section.
^C <----- NOTE: THE PROGRAM NEVER ENDS ...
pinguim>

*/
```

```c
// POSIX CONDITION VARIABLES
// Illustration of pthread_cond_broadcast()
// NOTE: pthread_cond_broadcast() unblocks all waiting threads
// FILE: cond_broadc_nc.c
// (equal to cond_broadc.c but without checking results of system calls)
// gcc cond_broadc_nc.c -lpthread -lrt -Wall -o cond_broadc_nc
// JAS
//========================================================================
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <pthread.h>
//========================================================================
#define NTHREADS 5
//========================================================================
int conditionMet = 0;
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
//========================================================================
void *threadFunc(void *arg)
{
  int threadNum = *(int *)arg;
  pthread_mutex_lock(&mutex);
  while (!conditionMet)
  {
    printf("Thread %d blocked because condition is not met\n", threadNum);
    pthread_cond_wait(&cond, &mutex);
  }
  printf("Thread %d executing critical section for 5 seconds ...\n",
    threadNum);
  sleep(5);
  pthread_mutex_unlock(&mutex);
  return NULL;
}
//========================================================================
int main(int argc, char *argv[])
{
  int i;
  int threadnum[NTHREADS];
  pthread_t threadId[NTHREADS];
  printf("Main thread: creating %d threads\n", NTHREADS);
  for(i=0; i<NTHREADS; ++i)
  {
    threadnum[i]=i+1;
    pthread_create(&threadId[i], NULL, threadFunc, (void*) &threadnum[i]);
  }
  printf("Main thread: doing some work until condition is met ...\n");
  sleep(10);
  //The condition has occured ...! Don't ask me what condition or why ...
  //Set the flag and wake up any waiting threads
  pthread_mutex_lock(&mutex);
  conditionMet = 1;
  printf("Main thread: the condition was met;\n waking up all waiting threads, 
using pthread_cond_broadcast()...\n");
  pthread_cond_broadcast(&cond);
  pthread_mutex_unlock(&mutex);
  printf("Main thread: waiting for threads and cleanup\n");
  for (i=0; i<NTHREADS; ++i)
  {
    pthread_join(threadId[i], NULL);
  }
  pthread_cond_destroy(&cond);
  pthread_mutex_destroy(&mutex);
  printf("Main thread: completed.\n");
  return 0;
}

/*
pinguim> ./cond_broadc_nc
Main thread: creating 5 threads
Thread 2 blocked because condition is not met
Thread 1 blocked because condition is not met
Thread 3 blocked because condition is not met
Thread 4 blocked because condition is not met
Main thread: doing some work until condition is met ...
Thread 5 blocked because condition is not met
Main thread: the condition was met;
 waking up all waiting threads, using pthread_cond_broadcast()...
Main thread: waiting for threads and cleanup
Thread 2 executing critical section for 5 seconds ...
Thread 1 executing critical section for 5 seconds ...
Thread 3 executing critical section for 5 seconds ...
Thread 4 executing critical section for 5 seconds ...
Thread 5 executing critical section for 5 seconds ...
Main thread: completed.
pinguim>
*/
```

```
// POSIX CONDITION VARIABLES
// Illustration of pthread_cond_signal()
// NOTE:
// pthread_cond_signal() must be called as many times
// as the number of threads;
// otherwise only one of them will be unblocked
// FILE: cond_signal.c
// gcc cond_signal.c -lpthread -lrt -Wall -o cond_signal
// JAS
//==========================================================================
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <pthread.h>
//==========================================================================
#define NTHREADS 5
//==========================================================================
int conditionMet = 0;
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

int numEndedThreads = 0;
pthread_mutex_t mutex_nET = PTHREAD_MUTEX_INITIALIZER;
//==========================================================================
// Function to check the return code
// and exit the program if the function call failed
void checkResult(char *string, int err)
{
  if (err != 0)
  {
    printf("Error %d on %s\n", err, string);
    exit(EXIT_FAILURE);
  }
  return;
}
//==========================================================================
void *threadFunc(void *arg)
{
  int res;
  int threadNum = *(int *)arg;
  res = pthread_mutex_lock(&mutex);
  checkResult("pthread_mutex_lock()\n", res);
  while (!conditionMet)
  {
    printf("Thread %d blocked because condition is not met\n", threadNum);
    res = pthread_cond_wait(&cond, &mutex);
    checkResult("pthread_cond_wait()\n", res);
  }
  printf("Thread %d executing critical section for 5 seconds ...\n", threadNum);
  sleep(5);
  printf("Thread %d ended execution of critical section.\n", threadNum);
  res = pthread_mutex_unlock(&mutex);
  checkResult("pthread_mutex_lock()\n", res);

  pthread_mutex_lock(&mutex_nET);
  numEndedThreads++;
  pthread_mutex_unlock(&mutex_nET);
  return NULL;
}
//======================================================================
int main(int argc, char *argv[])
{
  int res=0;
  int i, j;
  int threadnum[NTHREADS];
  pthread_t threadId[NTHREADS];

  printf("Main thread: creating %d threads\n", NTHREADS);
  for(i=0; i<NTHREADS; ++i)
  {
    threadnum[i]=i+1;
    res = pthread_create(&threadId[i], NULL, threadFunc, (void*) &threadnum[i]);
    checkResult("pthread_create()\n", res);
  }

  printf("Main thread:\n doing some work until condition is met ...\n");
  sleep(10);
  //The condition has occured ...! Don't ask me what condition or why ...
  //Set the flag and wake up any waiting threads
  res = pthread_mutex_lock(&mutex);
  checkResult("pthread_mutex_lock()\n", res);
  conditionMet = 1;
  printf("Main thread:\n the condition was met;\n waking up all waiting threads,
using pthread_cond_signal()...\n");
  for (j=0; j<NTHREADS; j++) //comment this line (just this one!) and see what
happens :-(
  {
    res = pthread_cond_signal(&cond);
    checkResult("pthread_cond_signal()\n", res);
  }
  res = pthread_mutex_unlock(&mutex);
  checkResult("pthread_mutex_unlock()\n", res);

  printf("Main thread: waiting for threads and cleanup\n");
  for (i=0; i<NTHREADS; ++i)
  {
    res = pthread_join(threadId[i], NULL);
    checkResult("pthread_join()\n", res);
  }

  res = pthread_cond_destroy(&cond);
  checkResult("pthread_cond_destroy()\n", res);
  res = pthread_mutex_destroy(&mutex);
  checkResult("pthread_mutex_destroy()\n", res);

  printf("Main thread: completed.\n");
  return 0;
}
```

```c
// POSIX CONDITION VARIABLES
// Illustration of pthread_cond_broadcast()
// NOTE: pthread_cond_broadcast() unblocks all waiting threads
// FILE: cond_broadc.c
// gcc cond_broadc.c -lpthread -lrt -Wall -o cond_broadc
// JAS
//======================================================================
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <pthread.h>
//======================================================================
#define NTHREADS 5
//======================================================================
int conditionMet = 0;
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
//======================================================================
// Function to check the return code
// and exit the program if the function call failed
void checkResult(char *string, int err)
{
  if (err != 0)
  {
    printf("Error %d on %s\n", err, string);
    exit(EXIT_FAILURE);
  }
  return;
}
//======================================================================
void *threadFunc(void *arg)
{
  int res;
  int threadNum = *(int *)arg;
  res = pthread_mutex_lock(&mutex);
  checkResult("pthread_mutex_lock()\n", res);
  while (!conditionMet)
  {
    printf("Thread %d blocked because condition is not met\n", threadNum);
    res = pthread_cond_wait(&cond, &mutex);
    checkResult("pthread_cond_wait()\n", res);
  }
  printf("Thread %d executing critical section for 5 seconds ...\n",
    threadNum);
  sleep(5);
  res = pthread_mutex_unlock(&mutex);
  checkResult("pthread_mutex_lock()\n", res);
  return NULL;
}
//======================================================================
int main(int argc, char *argv[])
{
  int res=0;
  int i;
  int threadnum[NTHREADS];
  pthread_t threadId[NTHREADS];
  printf("Main thread: creating %d threads\n", NTHREADS);
  for(i=0; i<NTHREADS; ++i)
  {
    threadnum[i]=i+1;
    res = pthread_create(&threadId[i], NULL, threadFunc, (void*) &threadnum[i]);
    checkResult("pthread_create()\n", res);
  }
  printf("Main thread: doing some work until condition is met ...\n");
  sleep(10);
  //The condition has occured ...! Don't ask me what condition or why ...
  //Set the flag and wake up any waiting threads
  res = pthread_mutex_lock(&mutex);
  checkResult("pthread_mutex_lock()\n", res);
  conditionMet = 1;
  printf("Main thread: the condition was met;\n waking up all waiting threads,
using pthread_cond_broadcast()...\n");
  res = pthread_cond_broadcast(&cond);
  checkResult("pthread_cond_broadcast()\n", res);
  res = pthread_mutex_unlock(&mutex);
  checkResult("pthread_mutex_unlock()\n", res);
  printf("Main thread: waiting for threads and cleanup\n");
  for (i=0; i<NTHREADS; ++i)
  {
    res = pthread_join(threadId[i], NULL);
    checkResult("pthread_join()\n", res);
  }
  res = pthread_cond_destroy(&cond);
  checkResult("pthread_cond_destroy()\n", res);
  res = pthread_mutex_destroy(&mutex);
  checkResult("pthread_mutex_destroy()\n", res);
  printf("Main thread: completed.\n");
  return 0;
}
```