Brea

```
Script started, file is typescript
bjk47@maroon22:~/Desktop/p3$ ls
calcPI2    genHosts.pl  Makefile
pthreadReduction.h
calcPI2.c  hosts       pthreadBarrier.h  typescript
bjk47@maroon22:~/Desktop/p3$ cat calcPI.c
pthreadReduction.h pthreadbarrier.h
cat: calcPI.c: No such file or directory
/* pthreadReduction.h implements the reduce and
barrier patterns
 *
 * pthreadReductionSum by Brea Koenes, fall 2021
 * pthreadBarrier and barrierCleanup by Joel Adams,
Calvin College, Fall 2013.
 */

#include <pthread.h>    // various pthread functions
#include <stdio.h>
#include "pthreadBarrier.h"
#include <stdlib.h>

long double * reductionSumArray;

// pthreadReductionSum implements reduction a
barrier patterns
void pthreadReductionSum(long double localSum,
unsigned long numThreads, unsigned long id, volatile
long double * pi) {
    if (id == 0){
        reductionSumArray = malloc(sizeof(long double)
* numThreads);
    }
    pthreadBarrier(numThreads);
    reductionSumArray[id] = localSum;

    for (int i=2; i < numThreads * 2; i *= 2) {
         pthreadBarrier((numThreads * 2) / i);
         if (id % i == 0 && (id + i/2 < numThreads)) {
             reductionSumArray[id] +=
reductionSumArray[id + i/2];
         }
         else break;
    }
    if (id == 0) {
        *pi = reductionSumArray[0];
        free(reductionSumArray);
    }
    barrierCleanup();
}

cat: pthreadbarrier.h: No such file or directory
bjk47@maroon22:~/Desktop/p3$ cat pthreadBarrier.h
/* pthreadBarrier.h implements the Barrier pattern
using pthreads.
 *
 * Joel Adams, Calvin College, Fall 2013.
 */

#include <pthread.h>    // various pthread functions

// Shared Variables used to implement the barrier
   pthread_mutex_t barrierMutex =
PTHREAD_MUTEX_INITIALIZER;
   pthread_cond_t allThreadsPresent =
PTHREAD_COND_INITIALIZER;
   double barrierThreadCount = 0;

/* the Barrier pattern for pthreads
 * params: numThreads, the number of threads being
synchronized
 * postcondition: all of those threads have reached
this call
 *                and are now ready to proceed.
 */
void pthreadBarrier(unsigned long numThreads) {
   pthread_mutex_lock( &barrierMutex );
   barrierThreadCount++;
   if (barrierThreadCount == numThreads) {
      barrierThreadCount = 0;
      pthread_cond_broadcast( &allThreadsPresent );
   } else {
      while ( pthread_cond_wait( &allThreadsPresent,
&barrierMutex) != 0 );
   }
   pthread_mutex_unlock( &barrierMutex );
}

void barrierCleanup() {
   pthread_mutex_destroy(&barrierMutex);
   pthread_cond_destroy(&allThreadsPresent);
}
bjk47@maroon22:~/Desktop/p3$ make
make: 'calcPI2' is up to date.
bjk47@maroon22:~/Desktop/p3$ ./calcPI2
1000000000 4
Estimation of pi is
3.14159265358979359174587675518 4 in 1.465034
secs
(actual pi value is
3.141592653589793238462643383279...)
bjk47@maroon22:~/Desktop/p3$ exit
Script done, file is typescript
bjk47@maroon22:~/Desktop/p3$ cat typescript
Script started on 2021-10-29 13:59:13-04:00
[TERM="xterm-256color" TTY="/dev/pts/0"
COLUMNS="80" LINES="24"]
bjk47@maroon22:~/Desktop/p3$ ls
```

Brea

```
calcPI2    genHosts.pl  Makefile
pthreadReduction.h
calcPI2.c  hosts        pthreadBarrier.h  typescript
bjk47@maroon22:~/Desktop/p3$ cat calcPI.c
pthreadReduction.h pthreadbarrier.h
cat: calcPI.c: No such file or directory
/* pthreadReduction.h implements the reduce and
barrier patterns
 *
 * pthreadReductionSum by Brea Koenes, fall 2021
 * pthreadBarrier and barrierCleanup by Joel Adams,
Calvin College, Fall 2013.
 */

#include <pthread.h>    // various pthread functions
#include <stdio.h>
#include "pthreadBarrier.h"
#include <stdlib.h>

long double * reductionSumArray;

// pthreadReductionSum impliments reduction a
barrier patterns
void pthreadReductionSum(long double localSum,
unsigned long numThreads, unsigned long id, volatile
long double * pi) {
    if (id == 0){
        reductionSumArray = malloc(sizeof(long double)
* numThreads);
    }
    pthreadBarrier(numThreads);
    reductionSumArray[id] = localSum;

    for (int i=2; i < numThreads * 2; i *= 2) {
         pthreadBarrier((numThreads * 2) / i);
         if (id % i == 0 && (id + i/2 < numThreads)) {
             reductionSumArray[id] +=
reductionSumArray[id + i/2];
         }
         else break;
    }
    if (id == 0) {
        *pi = reductionSumArray[0];
        free(reductionSumArray);
    }
    barrierCleanup();
}

cat: pthreadbarrier.h: No such file or directory
bjk47@maroon22:~/Desktop/p3$ cat pthreadBarrier.h
/* pthreadBarrier.h implements the Barrier pattern
using pthreads.
 *
 * Joel Adams, Calvin College, Fall 2013.
 */

#include <pthread.h>    // various pthread functions

// Shared Variables used to implement the barrier
   pthread_mutex_t barrierMutex =
PTHREAD_MUTEX_INITIALIZER;
   pthread_cond_t allThreadsPresent =
PTHREAD_COND_INITIALIZER;
   double barrierThreadCount = 0;

/* the Barrier pattern for pthreads
 * params: numThreads, the number of threads being
synchronized
 * postcondition: all of those threads have reached
this call
 *                 and are now ready to proceed.
 */
void pthreadBarrier(unsigned long numThreads) {
   pthread_mutex_lock( &barrierMutex );
   barrierThreadCount++;
   if (barrierThreadCount == numThreads) {
      barrierThreadCount = 0;
      pthread_cond_broadcast( &allThreadsPresent );
   } else {
      while ( pthread_cond_wait( &allThreadsPresent,
&barrierMutex) != 0 );
   }
   pthread_mutex_unlock( &barrierMutex );
}

void barrierCleanup() {
   pthread_mutex_destroy(&barrierMutex);
   pthread_cond_destroy(&allThreadsPresent);
}
bjk47@maroon22:~/Desktop/p3$ make
make: 'calcPI2' is up to date.
bjk47@maroon22:~/Desktop/p3$ ./calcPI2
1000000000 4
Estimation of pi is
3.141592653589793591745876755184 in 1.465034
secs
(actual pi value is
3.141592653589793238462643383279...)
bjk47@maroon22:~/Desktop/p3$ exit

Script done on 2021-10-29 14:00:19-04:00
[COMMAND_EXIT_CODE="0"]
```