

```

/*****
* FILE: join.c
* DESCRIPTION:
*   This example demonstrates how to "wait" for thread completions by using
*   the Pthread join routine.  Threads are explicitly created in a joinable
*   state for portability reasons. Use of the pthread_exit status argument is
*   also shown. Compare to detached.c
* AUTHOR: 8/98 Blaise Barney
* LAST REVISED: 01/30/09
*****/
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define NUM_THREADS      4

void *BusyWork(void *t)
{
    int i;
    long tid;
    double result=0.0;
    tid = (long)t;
    printf("Thread %ld starting...\n",tid);
    for (i=0; i<1000000; i++)
    {
        result = result + sin(i) * tan(i);
    }
    printf("Thread %ld done. Result = %e\n",tid, result);
    pthread_exit((void*) t);
}

int main (int argc, char *argv[])
{
    pthread_t thread[NUM_THREADS];
    pthread_attr_t attr;
    int rc;
    long t;
    void *status;

    /* Initialize and set thread detached attribute */
    pthread_attr_init(&attr);
    pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);

    for(t=0; t<NUM_THREADS; t++) {
        printf("Main: creating thread %ld\n", t);
        rc = pthread_create(&thread[t], &attr, BusyWork, (void *)t);
        if (rc) {
            printf("ERROR; return code from pthread_create() is %d\n", rc);
            exit(-1);
        }
    }

    /* Free attribute and wait for the other threads */
    pthread_attr_destroy(&attr);
    for(t=0; t<NUM_THREADS; t++) {
        rc = pthread_join(thread[t], &status);
        if (rc) {
            printf("ERROR; return code from pthread_join() is %d\n", rc);
            exit(-1);
        }
        printf("Main: completed join with thread %ld having a status of %ld\n",t,(long)status);
    }

    printf("Main: program completed. Exiting.\n");
    pthread_exit(NULL);
}

```