

Solaris Threads Example: barrier.c



The `barrier.c` program demonstrates an implementation of a barrier for Solaris threads. (See page 244 for a definition of barriers.)

Code Example A-1 Solaris Threads Example: `barrier.c`

```
#define _REENTRANT

/* Include Files */

#include<thread.h>
#include<errno.h>

/* Constants & Macros */

/* Data Declarations */

typedef struct {
    int      maxcnt;           /* maximum number of runners */
    struct _sb {
        cond_t wait_cv;      /* cv for waiters at barrier */
        mutex_t wait_lk;     /* mutex for waiters at barrier */
        int    runners;      /* number of running threads */
    } sb[2];
    struct _sb *sbp;          /* current sub-barrier */
} barrier_t;
```

Code Example A-1 Solaris Threads Example: barrier.c

```

/*
 * barrier_init - initialize a barrier variable.
 *
 */

int
barrier_init( barrier_t *bp, int count, int type, void *arg ) {
    int n;
    int i;

    if (count < 1)
        return(EINVAL);

    bp->maxcnt = count;
    bp->sbp = &bp->sb[0];

    for (i = 0; i < 2; ++i) {
#ifdef __cplusplus
        struct barrier_t::_sb *sbp = &( bp->sb[i] );
#else
        struct _sb *sbp = &( bp->sb[i] );
#endif
        sbp->runners = count;

        if (n = mutex_init(&sbp->wait_lk, type, arg))
            return(n);

        if (n = cond_init(&sbp->wait_cv, type, arg))
            return(n);
    }
    return(0);
}

/*
 * barrier_wait - wait at a barrier for everyone to arrive.
 *
 */

int
barrier_wait(register barrier_t *bp) {
#ifdef __cplusplus
    register struct barrier_t::_sb *sbp = bp->sbp;
#else
    register struct _sb *sbp = bp->sbp;

```

Code Example A-1 Solaris Threads Example: barrier.c

```
#endif

mutex_lock(&sbp->wait_lk);

if (sbp->runners == 1) {          /* last thread to reach barrier */
    if (bp->maxcnt != 1) {
        /* reset runner count and switch sub-barriers */
        sbp->runners = bp->maxcnt;
        bp->sbp = (bp->sbp == &bp->sb[0])
            ? &bp->sb[1] : &bp->sb[0];

        /* wake up the waiters */
        cond_broadcast(&sbp->wait_cv);
    }
} else {
    sbp->runners--;               /* one less runner */

    while (sbp->runners != bp->maxcnt)
        cond_wait( &sbp->wait_cv, &sbp->wait_lk);
}

mutex_unlock(&sbp->wait_lk);

return(0);
}

/*
 * barrier_destroy - destroy a barrier variable.
 */

int
barrier_destroy(barrier_t *bp) {
    int    n;
    int    i;

    for (i=0; i < 2; ++ i) {
        if (n = cond_destroy(&bp->sb[i].wait_cv))
            return( n );

        if (n = mutex_destroy( &bp->sb[i].wait_lk))
            return(n);
    }
}
```

Code Example A-1 Solaris Threads Example: barrier.c

```

    return(0);
}

#define NTHR      4
#define NCOMPUTATION 2
#define NITER     1000
#define NSQRT     1000

void *
compute(barrier_t *ba )
{
    int count = NCOMPUTATION;

    while (count--) {
        barrier_wait( ba );
        /* do parallel computation */
    }
}

main( int argc, char *argv[] ) {
    int
        i;
    int
        niter;
    int
        nthr;
    barrier_t
        ba;
    double
        et;
    thread_t
        *tid;

    switch ( argc ) {
        default:
            case 3 :
                niter  = atoi( argv[1] );
                nthr   = atoi( argv[2] );
                break;

            case 2 :
                niter  = atoi( argv[1] );
                nthr   = NTHR;
                break;

            case 1 :
                niter  = NITER;
                nthr   = NTHR;
                break;
    }

    barrier_init( &ba, nthr + 1, USYNC_THREAD, NULL );
}

```

Code Example A-1 Solaris Threads Example: barrier.c

```
tid = (thread_t *) calloc(nthr, sizeof(thread_t));

for (i = 0; i < nthr; ++i) {
    int    n;

    if (n = thr_create(NULL, 0, (void (*)( void *)) compute, &ba, NULL, &tid[i])) {
        errno = n;
        perror("thr_create");
        exit(1);
    }
}

for (i = 0; i < NCOMPUTATION; i++) {
    barrier_wait(&ba );
    /* do parallel algorithm */
}

for (i = 0; i < nthr; i++) {
    thr_join(tid[i], NULL, NULL);
}
}
```

