**Semaphores**

The pthreads library itself does not provide a semaphore; however, a separate POSIX standard does define them. The necessary declarations to use these semaphores are contained in semaphore.h.

**NOTE:** Do not confuse these with SystemV semaphores which are in sys/sem.h.

#include <semaphore.h>

#include <pthread.h>

#include <stdio.h>

#define THREADS 20

sem\_t OKToBuyMilk;

int milkAvailable;

void\* buyer(void \*arg)

{

// P()

sem\_wait(&OKToBuyMilk);

if(!milkAvailable)

{

// Buy some milk

++milkAvailable;

}

// V()

sem\_post(&OKToBuyMilk);

return NULL;

}

int main(int argc, char \*\*argv)

{

pthread\_t threads[THREADS];

milkAvailable = 0;

// Initialize the semaphore with a value of 1.

// Note the second argument: passing zero denotes

// that the semaphore is shared between threads (and

// not processes).

if(sem\_init(&OKToBuyMilk, 0, 1))

{

printf("Could not initialize a semaphore\n");

return -1;

}

for(int i = 0; i < THREADS; ++i)

{

if(pthread\_create(&threads[i], NULL, &buyer, NULL))

{

printf("Could not create thread %d\n", i);

return -1;

}

}

for(int i = 0; i < THREADS; ++i)

{

if(pthread\_join(threads[i], NULL))

{

printf("Could not join thread %d\n", i);

return -1;

}

}

sem\_destroy(&OKToBuyMilk);

// Make sure we don't have too much milk.

printf("Total milk: %d\n", milkAvailable);

return 0;

}

The semaphore API has several functions of note:

* **sem\_init**: Initialize a new semaphore. Note, the second argument denotes *how* the semaphore will be shared. Passing zero denotes that it will be shared among **threads** rather than processes. The final argument is the initial value of the semaphore.
* **sem\_destroy**: Deallocate an existing semaphore.
* **sem\_wait**: This is the P() operation.
* **sem\_post**: This is the V() operation.