In general, how can we perform conditional synchronization
with semaphores instead of condition variables?

Consider the following general threaded (pseudo-)code:

```
   pthread_mutex_t mtx // must protect all condition variables
   pthread_cond_t cvar // one for each condition


   // wait:
   mutex_lock(&mtx);
   while (!condition)
      cond_wait(&cvar,&mtx);
   ...
   mutex_unlock(&mtx);


   // broadcast:
   mutex_lock(&mtx);
   ...
   cond_broadcast(&cvar);
   mutex_unlock(&mtx);
```

We can replace this code with a semaphore version as follows (keys might be IPC_PRIVATE if done before all forks):
(assuming data in a shared memory segment!)

```
   // Binary semaphore with initial value 1:
   int mtx_semid = psemget(semkey, 1, 0600 | IPC_CREAT | IPC_EXCL);
   psem_up(mtx_semid, 0); // Binary semaphore with initial value 1
```

```
// Integer semaphore with initial value 0:
int cnd_semid = psemget(key, 1, 0600 | IPC_CREAT | IPC_EXCL);

// Shared integer variable (must be in a shared memory segment):
waiting_processes = 0;

// wait:
psem_down(mtx_semid, 0); // lock
while (!condition)
{
   waiting_processes++;
   psem_up(mtx_semid, 0); // unlock
   psem_down(cnd_semid, 0); // wait
   psem_down(mtx_semid, 0); // lock
}
...
psem_up(mtx_semid, 0); // unlock


// broadcast:
psem_down(mtx_semid, 0); // lock
...
while(waiting_processes > 0)
{
   psem_up(cnd_semid, 0); // signal
   waiting_processes--;
}
psem_up(mtx_semid, 0); // unlock
```