How can we perform conditional synchronization with semaphores in a similar way as condition variables?

Since semaphores are (special!) counting (shared) variables, the "trick" is to count the number of waiting active entities.

MUTEX & CONDITION VARIABLES

```
// INITIALIZATION:

// mutex:
pthread_mutex_t mtx;
mutex_init(&mtx, NULL);

// condition variable:
pthread_cond_t cvar;
cond_init(&cvar, NULL);


// WAIT:

mutex_lock(&mtx);
while (!condition)
{

   cond_wait(&cvar,&mtx);

}
...
mutex_unlock(&mtx);


// BROADCAST:

mutex_lock(&mtx);
...

cond_broadcast(&cvar);

mutex_unlock(&mtx);


// SIGNAL:

mutex_lock(&mtx);
...

cond_signal(&cvar);

mutex_unlock(&mtx);
```

SHARED MEMORY & SEMAPHORES

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

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

// Shared integer variable (in shared memory!):
int shmid = pshmget(shmkey, sizeof(int), 0600 | IPC_CREAT | IPC_EXCL);
int* cvar_waiters = pshmat(shmid, NULL, 0); // before fork(s)
*cvar_waiters = 0;
...
pshmdt(cvar_waiters); // at the end, after all wait[pid]


// wait (assumes cvar_waiters attached):
psem_down(mtx_id, 0); // lock
while (!condition)
{
   (*cvar_waiters)++;
   psem_up(mtx_id, 0); // unlock
   psem_down(cvar_id, 0); // wait
   psem_down(mtx_id, 0); // lock
}
...
psem_up(mtx_id, 0); // unlock


// broadcast (assumes cvar_waiters attached):
psem_down(mtx_id, 0); // lock
...
while((*cvar_waiters) > 0)
{
   psem_up(cvar_id, 0); // signal one
   (*cvar_waiters)--;
}
psem_up(mtx_id, 0); // unlock


// signal (assumes cvar_waiters attached):
psem_down(mtx_id, 0); // lock
...
if((*cvar_waiters) > 0)
{
   psem_up(cvar_id, 0); // signal one
   (*cvar_waiters)--;
}
psem_up(mtx_id, 0); // unlock
```