```
void parallelDo(int n, void *f(void *), void *arg[]){
  int ii;
  Barrier b = new Barrier(n);
  for (ii = 0; ii < n; ii++){
    sthread_create(&t[i], stub, f, arg[i], b);
  }
  b->waitToDone();
}

void *stub(void *f(void*), void *arg, Barrier b){
  (*f)(arg);
  b->done();
}

class Barrier{
  private
    int max;
    int done;
    smutex_t mutex;
    scond_t allDone;
  public:
    Barrier(int max);
    void waitToDone();
    void done();
};

Barrier::Barrier(int max_){
  max = max_;
  done = 0;
  scond_init(&allDone);
  smutex_init(&mutex);
}

void Barrier::done(){
  smutex_acquire(&mutex);
  done++;
  if(done == max){
    scond_signal(&allDone);
  }
  smutex_release(&mutex);
}

void Barrier::waitToDone(){
  smutex_acquire(&mutex);
  while(done < max){
    scond_wait(&allDone, &mutex);
  }
  smutex_release(&mutex);
}
```