```
monitor dining_controller;
cond ForkReady[5];        /* condition variable for synchronization */
boolean fork[5] = {true};       /* availability status of each fork */

void get_forks(int pid)         /* pid is the philosopher id number */
{
   int left = pid;
   int right = (++pid) % 5;
   /*grant the left fork*/
   if (!fork[left])
      cwait(ForkReady[left]);         /* queue on condition variable */
   fork[left] = false;
   /*grant the right fork*/
   if (!fork[right])
      cwait(ForkReady[right]);        /* queue on condition variable */
   fork[right] = false:
}
void release_forks(int pid)
{
   int left = pid;
   int right = (++pid) % 5;
   /*release the left fork*/
   if (empty(ForkReady[left])     /*no one is waiting for this fork */
      fork[left] = true;
   else                      /* awaken a process waiting on this fork */
      csignal(ForkReady[left]);
   /*release the right fork*/
   if (empty(ForkReady[right])    /*no one is waiting for this fork */
      fork[right] = true;
   else                      /* awaken a process waiting on this fork */
      csignal(ForkReady[right]);
}
```

```
void philosopher[k=0 to 4]          /* the five philosopher clients */
{
   while (true) {
      <think>;
      get_forks(k);         /* client requests two forks via monitor */
      <eat spaghetti>;
      release_forks(k);     /* client releases forks via the monitor */
   }
}
```

Figure 6.14 A Solution to the Dining Philosophers Problem Using a Monitor