```c
#define N 5 /* numero de filosofos */
#define LEFT (i+N-1)%N /* numero do vizinho a esquerda de i */
#define RIGHT (i+1)%N /* numero do vizinho a direita de i */
#define THINKING 0 /* o filosofo esta pensando */
#define HUNGRY 1 /* o filosofo esta tentando pegar garfos */
#define EATING 2 /* o filosofo esta comendo */

typedef int semaphore; /* semaforos sao um tipo especial de int */
int state[N]; /* arranjo para controlar o estado de cada um */
semaphore mutex = 1; /* exclusao mutua para as regioes criticas */
semaphore s[N]; /* um semaforo por filosofo */

void philosopher(int i) /* i: o numero do filosofo, de 0 a N-1 */
{
    while (TRUE) { /* repete para sempre */
    think( ); /* o filosofo esta pensando */
    take_forks(i); /* pega dois garfos ou bloqueia */
    eat( ); /* hummm, espaguete! */
    put_forks(i); /* devolve os dois garfos a mesa */
    }
}

void take_forks(int i) /* i: o numero do filosofo, de 0 a N-1 */
{
    down(&mutex); /* entra na região critica */
    state[i] = HUNGRY; /* registra que o filosofo esta faminto */
    test(i); /* tenta pegar dois garfos */
    up(&mutex); /* sai da região critica */
    down(&s[i]); /* bloqueia se os garfos não foram pegos */
}

void put_forks(i) /* i: o numero do filosofo, de 0 a N-1 */
{
    down(&mutex); /* entra na regiao critica */
    state[i] = THINKING; /* o filosofo acabou de comer */
    test(LEFT); /* ve se o vizinho da esquerda pode comer agora */
    test(RIGHT); /* ve se o vizinho da direita pode comer agora */
    up(&mutex); /* sai da regiao critica */
}

void test(i)/* i: o numero do filosofo, de 0 a N-1 */
{
    if (state[i] == HUNGRY && state[LEFT] != EATING && state[RIGHT] !=
EATING) {
        state[i] = EATING;
```

```
        up(&s[i]);
    }
}

int main(){
   int i;
   pthread_t thread_id[N]; //identificadores das
                           //threads
   sem_init(&mutex,0,1);
   for(i=0;i<N;i++)
      sem_init(&S[i],0,0);
   for(i=0;i<N;i++)
   {
      pthread_create(&thread_id[i], NULL, filosofo,
&nfilosofo[i]);
      //criar as threads
      printf("Filosofo %d esta a pensar.\n",i+1);
   }
   for(i=0;i<N;i++)
   pthread_join(thread_id[i],NULL); //para
                                    //fazer a junção das threads
    return 0;
}
```

Como funciona: existem N filósofos sentados em uma mesa com N garfos. A rotina deles é: pensar, pegar os garfos, comer e soltar os garfos. É gerado uma thread para cada filósofo e elas seguem a rotina indicada até que o programa seja encerrado. A ação de pensar apenas coloca a thread em espera. Quando ela acaba a próxima rotina é pegar os garfos, para isso o algoritmo espera que o semáforo esteja liberado, pois irá entrar em uma região crítica, para mudar o estado do filósofo para ‘com fome’ e começar a verificação da possibilidade de pegar os dois garfos. Caso os dois garfos do lado dele estão liberados ele os pega e começa a comer e libera o semáforo, caso contrário ele espera até que o filósofo ao lado dele solte um dos garfos. Após terminar de comer ele deve soltar os garfos ,para isso ele espera o semáforo estar liberado, quando ele estiver, solta os garfos e o filósofo volta a pensar e então o algoritmo faz a verificação para os outros dois filósofos do lado dele para verificar se eles estão na espera por um garfo.