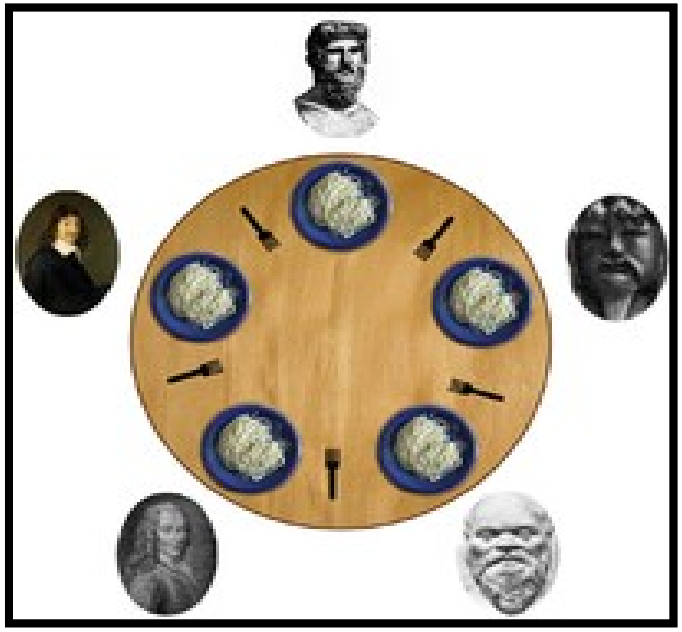
Filósofos Comelones

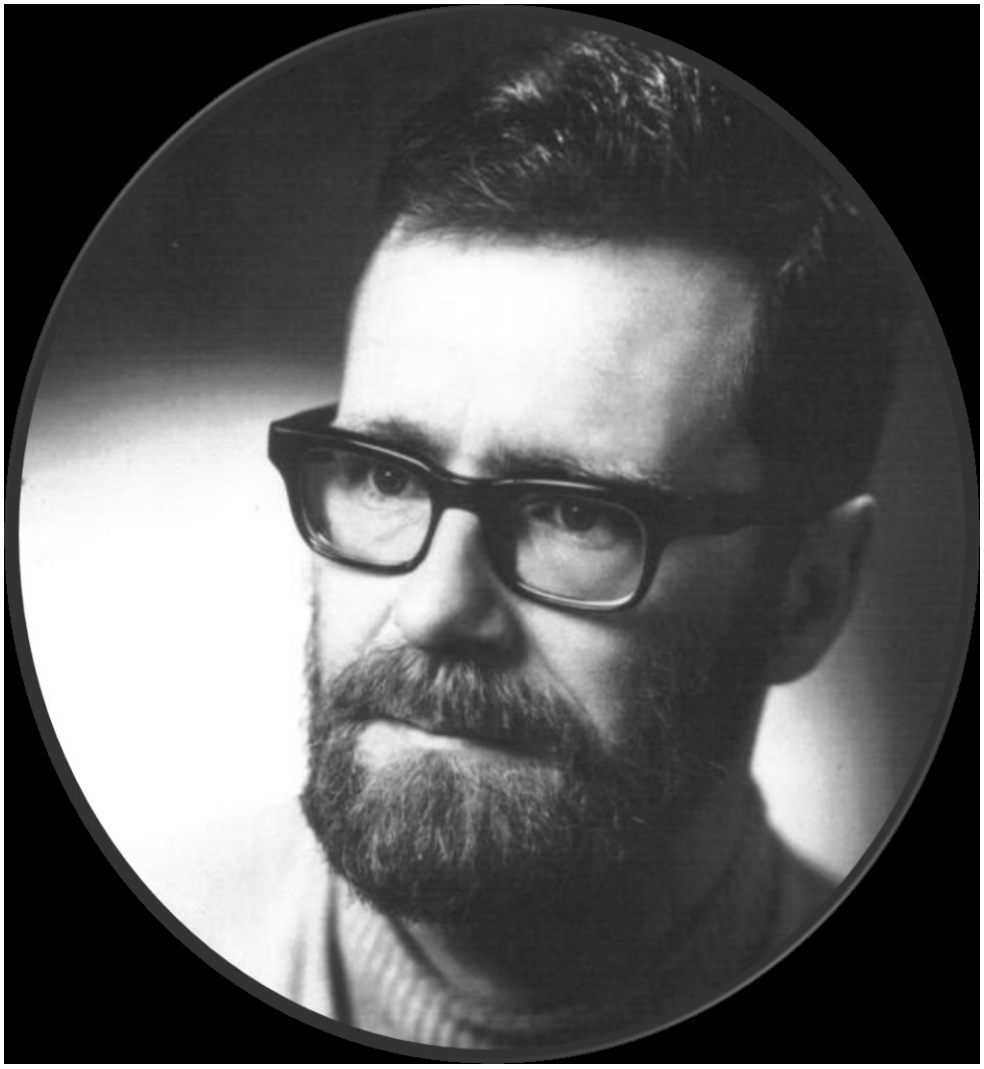
Sistemas Operativos



Uri Yael Leal Cortes

Orígenes

En 1965, Dijkstra propuso y resolvió un problema de sincronización al que llamó el problema de los filósofos comelones.

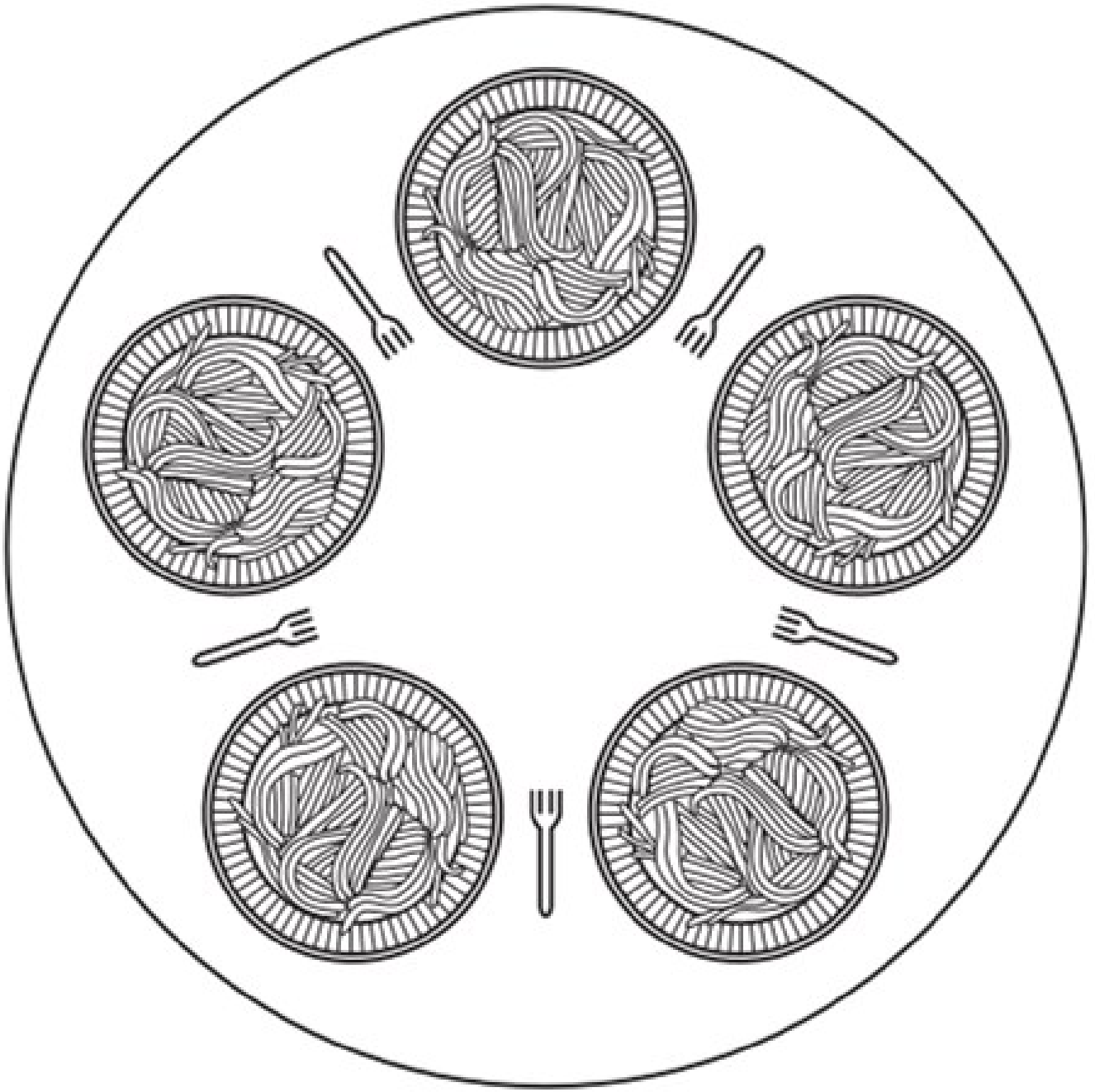


El Problema

- Cinco filósofos están sentados alrededor de una mesa circular.

- Cada filósofo tiene un plato de espagueti. El espagueti es tan resbaloso, que un filósofo necesita dos tenedores para comerlo.

- Entre cada par de platos hay un tenedor.



Solución

- Cuando un filósofo tiene hambre, trata de adquirir sus tenedores izquierdo y derecho, uno a la vez, en cualquier orden.

- Si tiene éxito al adquirir dos tenedores, come por un momento, después deja los tenedores y continúa pensando.

Algoritmo

- El programa utiliza un arreglo de semáforos, uno por cada filósofo, de manera que los filósofos hambrientos puedan bloquearse si los tenedores que necesitan están ocupados.

Algoritmo

```
#define N                   5                   /* número de filósofos */
#define IZQUIERDO           (i+N-1)%N           /* número del vecino izquierdo de i */
#define DERECHO             (i+1)%N             /* número del vecino derecho de i */
#define PENSANDO            0                   /* el filósofo está pensando */
#define HAMBRIENTO          1                   /* el filósofo trata de obtener los tenedores */
#define COMIENDO            2                   /* el filósofo está comiendo */
typedef int semaforo;                           /* los semáforos son un tipo especial de int */
int estado[N];                                  /* arreglo que lleva registro del estado de todos*/
semaforo mutex = 1;                             /* exclusión mutua para las regiones críticas */
semaforo s[N];                                  /* un semáforo por filósofo */

void filosofo(int i)                            /* i: número de filósofo, de 0 a N−1 */
{
    while(TRUE){                                /* se repite en forma indefinida */
        pensar();                               /* el filósofo está pensando */
        tomar_tenedores(i);                     /* adquiere dos tenedores o se bloquea */
        comer();                                /* come espagueti */
        poner_tenedores(i);                     /* pone de vuelta ambos tenedores en la mesa */
    }
}
```

```
void tomar_tenedores(int i)                    /* i: número de filósofo, de 0 a N−1 */
{
     down(&mutex);                             /* entra a la región crítica */
     estado[i] = HAMBRIENTO;                   /* registra el hecho de que el filósofo i está hambriento */
     probar(i);                                /* trata de adquirir 2 tenedores */
     up(&mutex);                               /* sale de la región crítica */
     down(&s[i]);                              /* se bloquea si no se adquirieron los tenedores */
}

void poner_tenedores(i)                        /* i: número de filósofo, de 0 a N–1 */
{
     down(&mutex);                             /* entra a la región crítica */
     estado[i] = PENSANDO;                     /* el filósofo terminó de comer */
     probar(IZQUIERDO);                        /* verifica si el vecino izquierdo puede comer ahora */
     probar(DERECHO);                          /* verifica si el vecino derecho puede comer ahora */
     up(&mutex);                               /* sale de la región crítica */
}

void probar(i)                 /* i: número de filósofo, de 0 a N−1 */
{
     if (estado[i] == HAMBRIENTO && estado[IZQUIERDO] != COMIENDO && estado[DERECHO] != COMIENDO) {
          estado[i] = COMIENDO;
          up(&s[i]);
     }
}
```