Soluzioni software per la sincronizzazione

Dijkstra

Garantisce:

- Mutua esclusione
- Nessun deadlock

Problemi:

- Possibile starvation
- Read/write atomiche
- Condivisione di memoria per il parametro k

L'algoritmo di Dijkstra è una estensione dell'algoritmo di *Dekker* per la mutua esclusione, per 2 processi:

```
int me = 0, other = 1; // P0 (flip for P1)

while (true) {
   /*NCS*/
   flag[me] = true;
   while (flag[other]) {
      if (turn == other) {
         flag[me] = false;
         while (turn == other) /* busy wait */ ;
         flag[me] = true;
      }
   }
   /* CS */
   turn = other;
   flag[me] = false;
}
```

```
flag[me] = true;           I want to enter
while (flag[other]) {      If you want to enter
  if (turn==other) {       and if it's your turn
    flag[me] = false;      I don't want any more
    while (turn==other);   If it's your turn I'll wait
    flag[me] = true;       I want to enter
  }
}
/* CS */
turn = other;              You can enter next
flag[me] = false;          I don't want any more
```

ALGORITMO DI DIJKSTRA

```
int n = numero_di_processi;
bool interessati[n] = {false};//quale processo vuole usare la risorsa
bool passati[n] ={false}; // quale processo ha passato la fase 1
```

```
int k = il prossimo processo a cui sara permesso provare per la risorsa;

int i = il mio processo;

//inizio
intrested[i] = true; //ho interesse a entrare in CS
/*Aspetto il mio turno*/
while(k != i){
    passed[i] = false;
    if (intrested[k] = false){
        k = i
    }
}
/*adesso tocca a me*/
/*devo controllare pero che nessun altro processo, che non sono io abbia passato fase uno*/
passed[i] = true;
for (int j = 0; j < n; j++){
    if (j != i and passed[j] == true){
        goto "rifaccio parte 1"
    }
    else{
        adesso sono nella sezione critica;
        passed[i] = false; //adesson non sono più interessato a entrare
        intrested[i] = false;
    }
}
```

Backery

Parto dal concept di un negozio con un ticket e tante persone che aspettano il proprio turno.

- Non servono read e write come operazioni atomiche, posso leggere mentre altri scrivono.

```
// dichiarazione delle variabili globali comuni
bool sceglie[N] = {false}; // N costante numero di processi
int numero[N] = {0};

int i; // indice del thread in esecuzione
// ...
while (numero[i] == 0) {
    //prendo il numero
    sceglie[i] = true;
    numero[i] = 1 + max(numero[0], numero[1], ..., numero[N - 1]);//il più grande tra tutti i
numeri in possesso
    sceglie[i] = false; //ho preso il numero
    for j in 1 .. N except i {
        while (sceglie[j] == true); //aspetto la scelta di altri
        while (numero[j] != 0 && (numero[j],j) < (numero[i],i)); //controllo di avere il
numero più basso, dopo i vari zeri
}

    // <sezione critica>
    numero[i] = 0;
    // <sezione non critica>
}
```