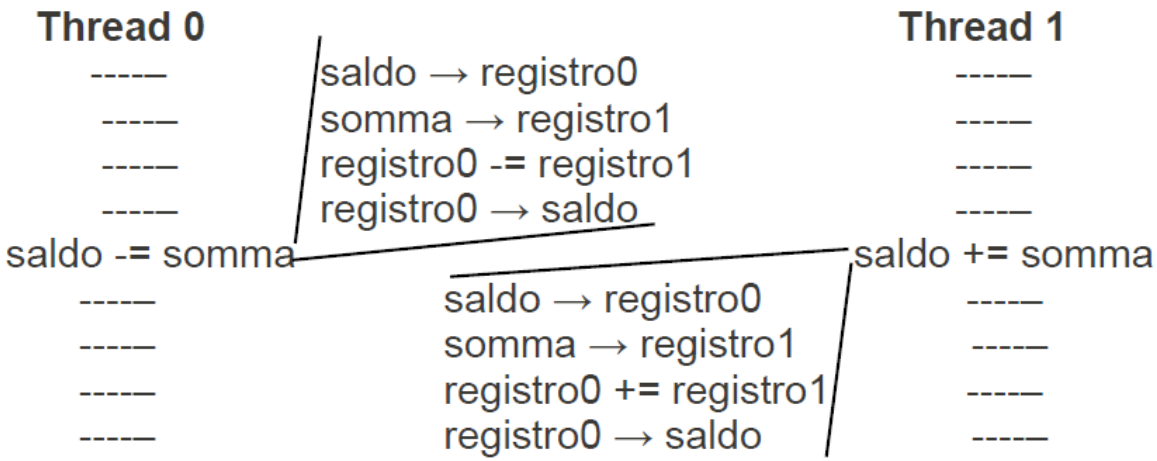
Introduzione alla Mutua esclusione

Il problema della Mutua Esclusione

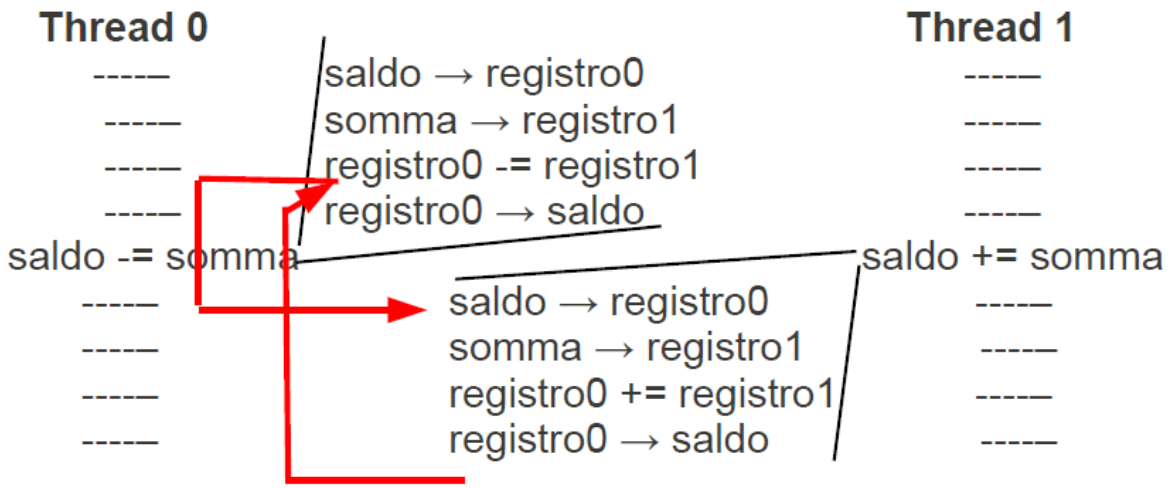
- Il problema della mutua esclusione nasce quando piú di un processo alla volta chiede l'accesso a variabili condivise.
- Le operazioni sulle variabili condivise sono la Sezione Critica di un processo.
- Il problema deve essere risolto in modo tale da ASSICURARE CHE LA SEZIONE CRITICA SIA ESEGUITA DA UN SOLO THREAD ALLA VOLTA.

Alcuni problemi comuni di Mutua Esclusione:

Operazioni con variabili condivise. Esempio:



Un problema comune di Mutua Esclusione



Se ci fossero i context switch indicati, il saldo sarebbe completamente sbagliato

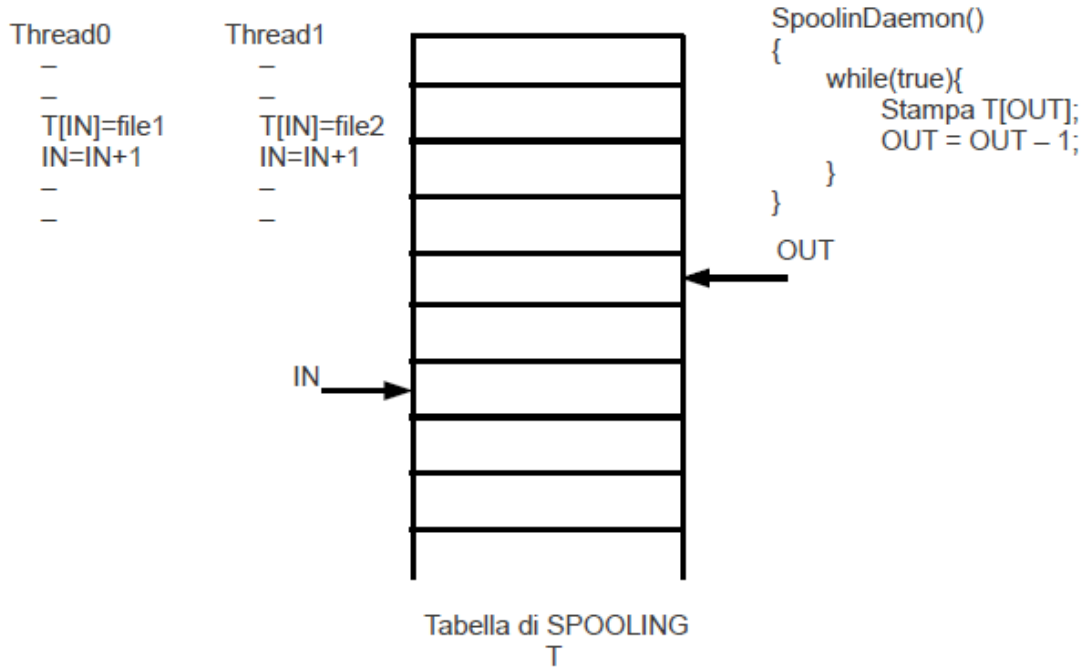
Un problema di Mutua Esclusione: gestione di uno stack

```
class stack{
    private int pila[100];
    private top=0;
    private x;

    Inserimento (int y){
         top:=top+1;
         pila[top]=y;
    }

    int Prelievo(){
          x = pila[top];
          top:=top-1;
          return(x);
    }
}
```

Se i due metodi fossero eseguiti in concorrenza: Variabili 'top' e 'pila' condivise. Una possibile schedulazione:
top = top+1; //metodo Inserimento
x = pila[top]; //metodo Prelievo: preleva un dato inesistente
top = top-1; //metodo Prelievo
pila[top] = y; //metodo Inserimento: sovrascrive l'elemento in cima allo stack

Un problema di Mutua Esclusione: Spooling System

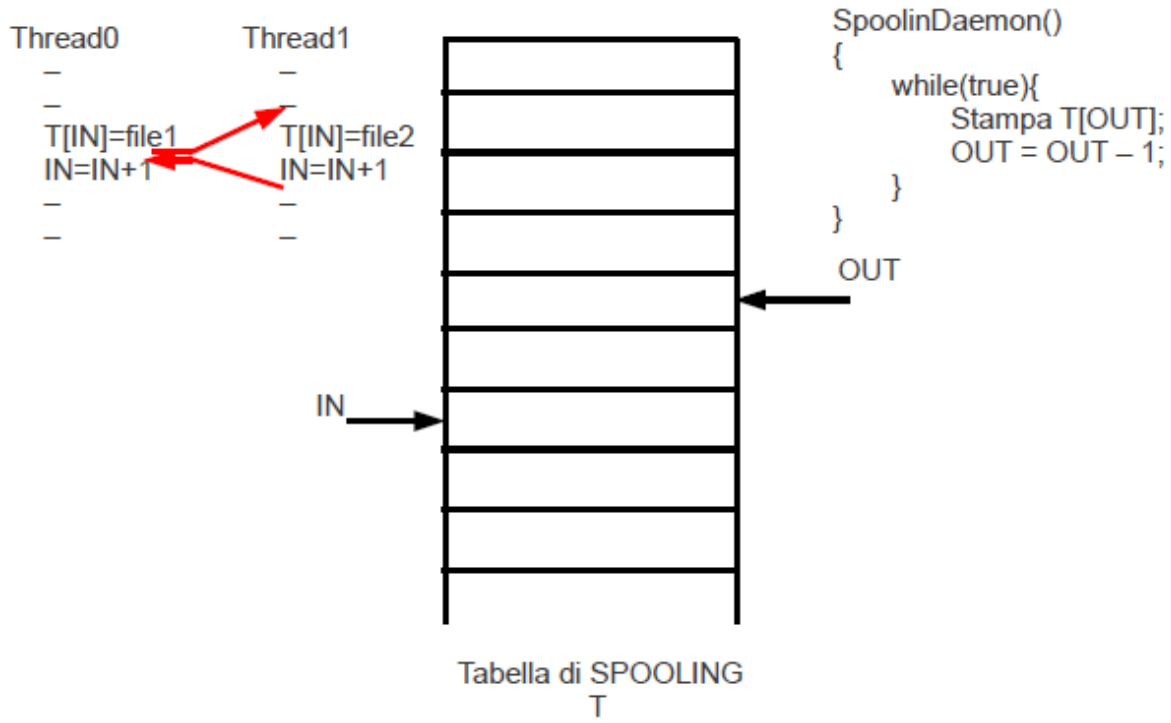
SPOOL = Simultaneous Peripheral Operations On-Line



Lo Spooling Daemon cerca continuamente file da stampare

I Thread 0 e 1 producono file d'uscita e per stamparli scrivono il loro nome nella spooling table.

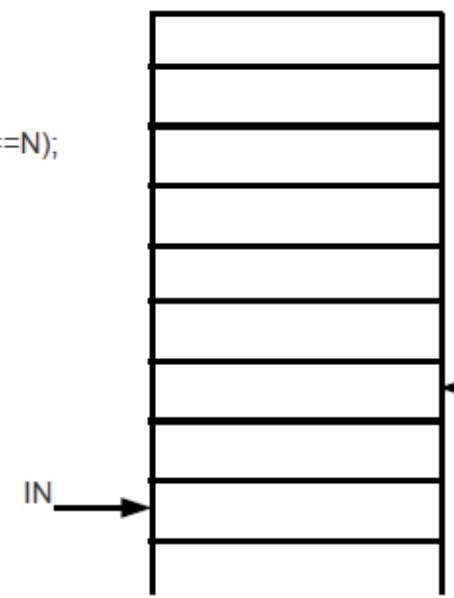
Un problema di Mutua Esclusione: Spooling System



Se ci fossero i context switch indicati verrebbe sovrascritto il nome del file e l'indice IN verrebbe incrementato di due.

Un problema di Mutua Esclusione: Buffer circolare

```
void Produttore(){
    item p;
    while(true){
        p=produci();
        while(counter==N);
        buffer[IN]=p;
        IN=(IN+1)%N;
        Counter++;
    }
}
```

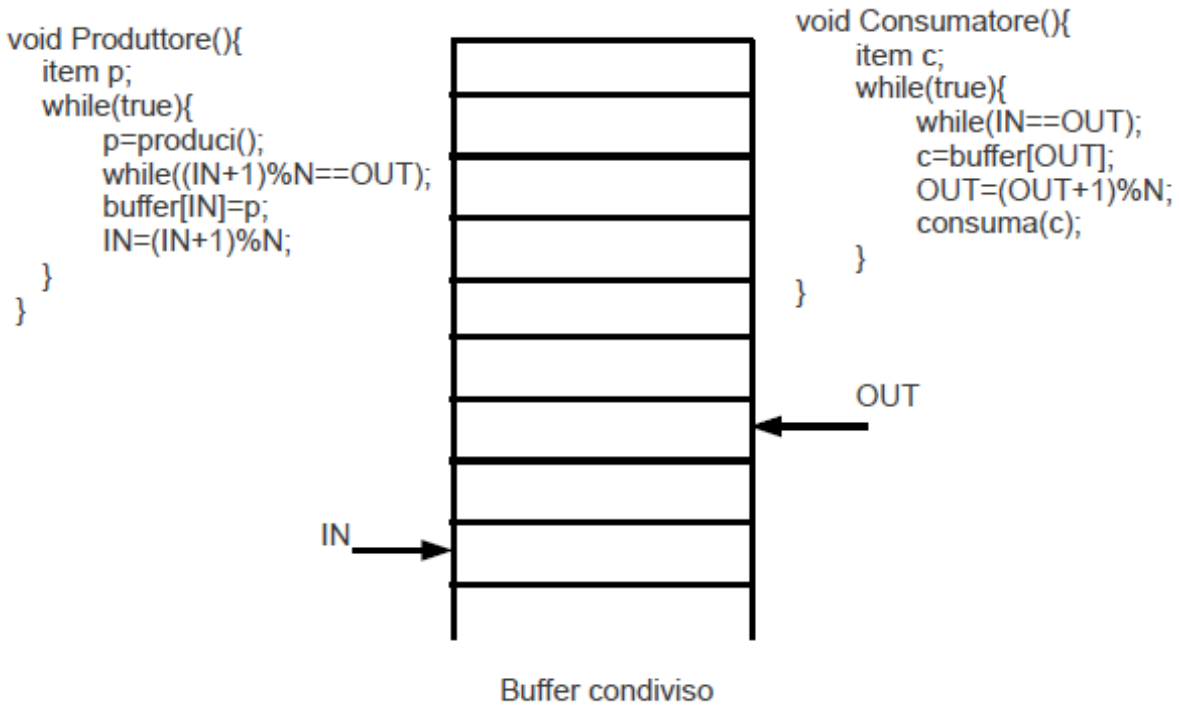


Buffer condiviso

```
Void Consumatore(){
    item c;
    while(true){
        while(counter==0);
        c=buffer[OUT];
        OUT=(OUT+1)%N;
        Counter--;
        consuma(c);
    }
}
```

Le variabili condivise sono il buffer e 'counter'. Context switch durante l'incremento di 'counter' produce problemi di mutua esclusione.

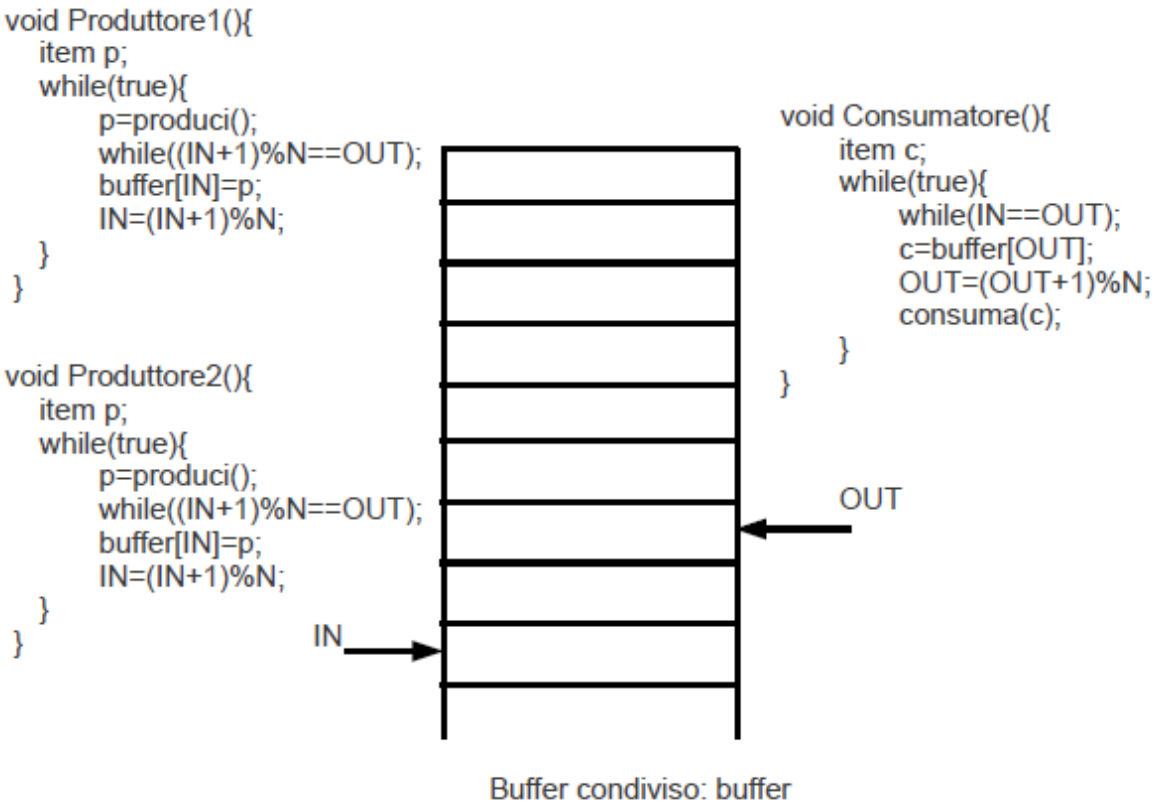
Un problema di Mutua Esclusione: Buffer circolare



Questa versione (1 produttore e 1 consumatore funziona correttamente)

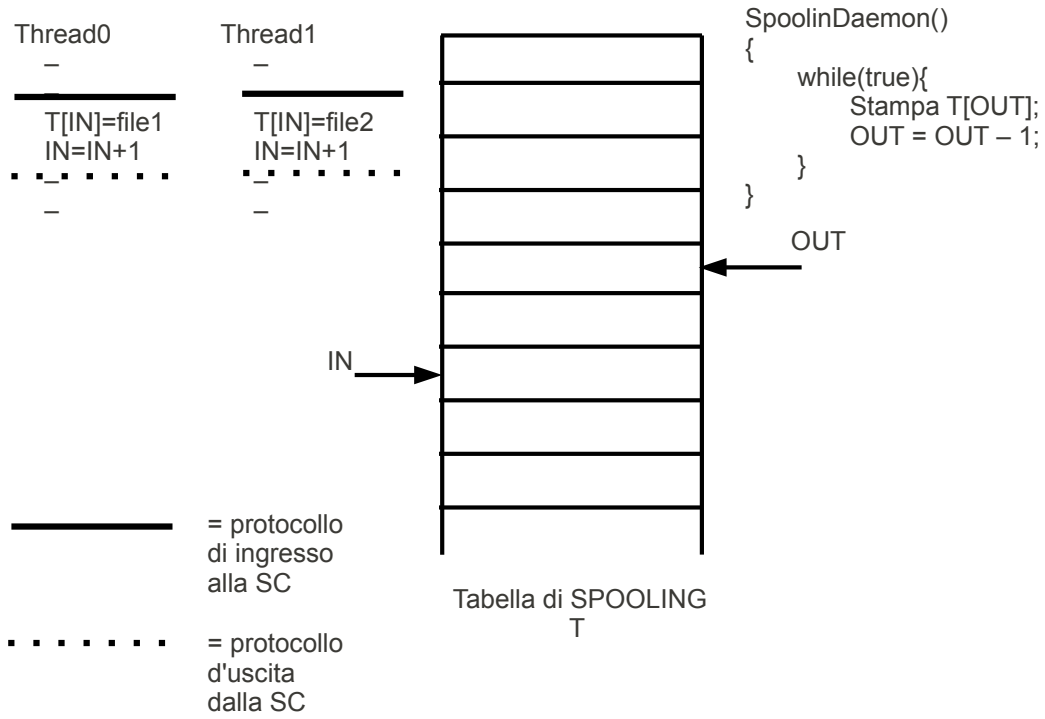
Un problema di Mutua Esclusione: Buffer circolare

Il problema nasce quando ci sono piú Produttori o piú Consumatori



In questo caso ci sono problemi per la condivisione di 'IN'. Esempio: context switch dopo la while

Soluzione: inserimento di un arbitro prima della Sezione Critica per controllare che la Sezione critica sia eseguita da un solo processo.
Esempio: Spool



Esempio di una Non Soluzione ai problemi di Mutua Esclusione

```
Thread0() {
    while(1){
        while( turno == 0 ) ;
        Sezione_Critica0();
        turno = 0;
        Sezione_NON_Critica0();
     }
}
```

```
Thread1(){
      while(1){
        while( turno == 1 ) ;
        Sezione_Critica1();
        turno = 1;
        Sezione_NON_Critica1()
       }
 }
```

Questa soluzione:

- soddisfa la mutua esclusine
- evita lo stallo tra i processi
- evita la starvation

Attenzione: questa non e' una soluzione perché un blocco o un rallentamento in una sezione NON critica blocca o rallenta l'altro processo.

Questa soluzione é valida solo se é certo che la sezione NON critica non contiene blocchi