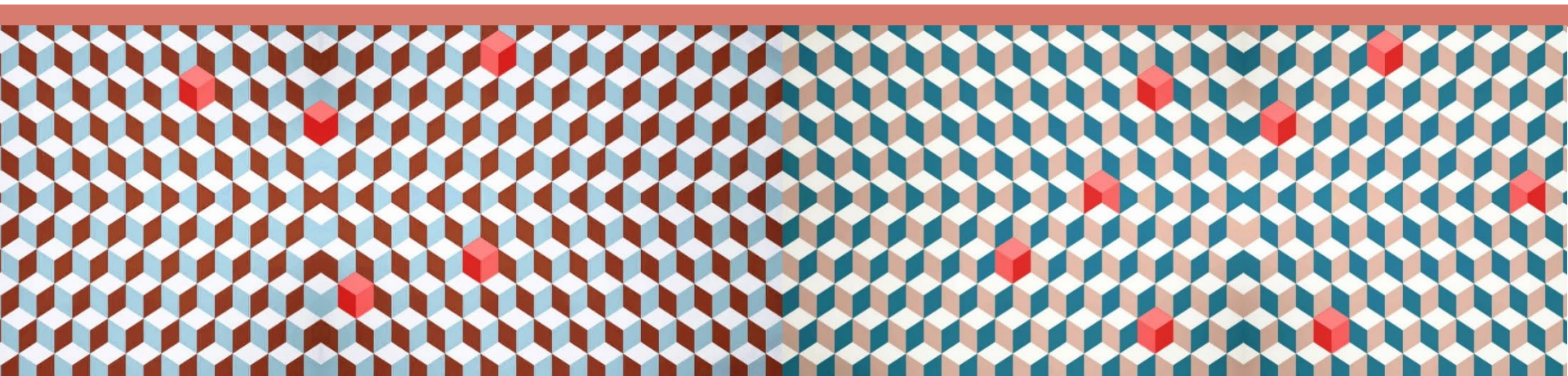


Scheduling CPU:

Esercitazione e concetti

Tutor: Giovanni Hauber



L Terminologia

- **TEMPO DI ARRIVO**

Istante in cui un utente **invia** un processo

- **TEMPO DI AMMISSIONE**

Istante in cui un processo viene **considerato** per lo scheduling

- **TEMPO DI SERVIZIO / CPU BURST**

Tempo totale richiesto da un processo per **completare** la sua richiesta

- **TEMPO DI ATTESA**

Lasso di **tempo** che un processo aspetta, dal momento del suo arrivo, **senza fare niente**

- **TEMPO DI COMPLETAMENTO**

Istante di tempo in cui un processo **termina**

- **TURNAROUND**

Tempo trascorso dalla **sottomissione** da un utente di un processo **fino** al momento del suo **completamento**.

- **CAMBIO DI CONTESTO**

Quantità di tempo necessaria ad effettuare la prelazione

- **THROUGHPUT**

Numero di **processi completati** e serviti da un SO in una singola **unità di tempo**

L Formule

- **TEMPO DI ATTESA**

- **Normale:** $(T_{\text{completamento}} - T_{\text{arrivo}} - T_{\text{servizio}})$
- **Medio:** $\sum (T_{\text{completamento}} - T_{\text{arrivo}} - T_{\text{servizio}}) / N_{\text{processi}}$

- **TURNAROUND**

- **Normale:** $T_{\text{completamento}} - T_{\text{arrivo}}$
- **Medio:** $\sum (T_{\text{completamento}} - T_{\text{arrivo}}) / N_{\text{processi}}$
- **Normalizzato:** $T_{\text{completamento}} - T_{\text{arrivo}} / \text{Burst}$
- **Normalizzato Medio:** $\sum (\text{Normalizzato}) / N_{\text{proc}}$

- **THROUGHPUT**

Calcolato come:

- $N_{\text{processi}} / \text{Tempo Totale}$

01 CONCETTI L Politiche

- **NON PRELAZIONABILI**

- Politiche di algoritmi che **non rendono possibile interrompere** le esecuzioni di un processo, aspettando la fine delle loro esecuzioni.
 - FCSF

- **PRELAZIONABILI**

- Politiche di algoritmi che **rendono possibile scambiare** un processo attualmente in esecuzione con un altro processo anche se la sua esecuzione non risulti finita.
 - STG

- **PRIORITA'**

- Politica che **favorisce i processi con priorità più alta**. Applicabile ad entrambe le politiche menzionate prima, e può essere **prelazionabile o non**.

Nota: è possibile applicare la prelazione anche ad algoritmi NON-PREEMPTIVE!

Scheduling

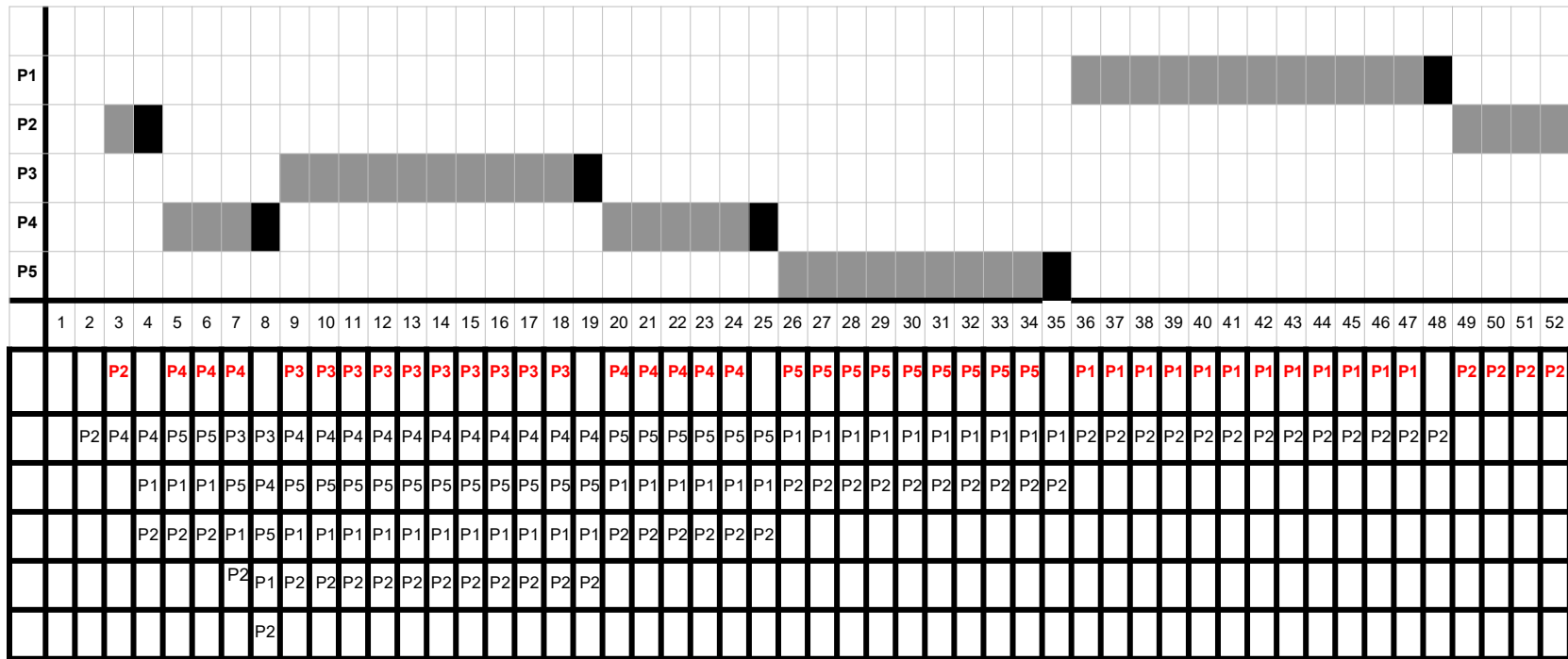
Esercizio 1

- Si considerino i seguenti processi, attivi in un sistema multiprogrammato:

Processo	Tempo di Arrivo	CPU Burst	Priorità
P1	4ms	12ms	2
P2	2ms	5ms	1
P3	7ms	10ms	5
P4	3ms	8ms	4
P5	5ms	9ms	3

- Supponendo che il cambio di contesto sia 1ms, si mostri l'ordine di esecuzione dei processi e quanto vale il tempo di attesa medio, il tempo di turnaround medio ed il tempo di turnaround normalizzato medio per ciascuno dei seguenti algoritmi di scheduling della CPU:
 - Priorità con prelazione (la priorità massima è 5)

- Diagramma di esecuzione: (Priorità con prelazione)



Processo	Tempo di Arrivo	CPU Burst	Priorità
P1	4ms	12ms	2
P2	2ms	5ms	1
P3	7ms	10ms	5
P4	3ms	8ms	4
P5	5ms	9ms	3

- Tempi di esecuzione: (Priorità con prelazione)

- Tempo di Attesa:

$$P1 = 47 - 4 - 12 = 31 \text{ ms}$$

$$P2 = 52 - 2 - 5 = 45 \text{ ms}$$

$$P3 = 18 - 7 - 10 = 1 \text{ ms}$$

$$P4 = 24 - 3 - 8 = 13 \text{ ms}$$

$$P5 = 34 - 5 - 9 = 20 \text{ ms}$$

$$\text{Tempo di attesa medio} = 22 \text{ ms}$$

- Turnaround

$$P1 = 47 - 4 = 43 \text{ ms}$$

$$P2 = 52 - 2 = 50 \text{ ms}$$

$$P3 = 18 - 7 = 11 \text{ ms}$$

$$P4 = 24 - 3 = 21 \text{ ms}$$

$$P5 = 34 - 5 = 29 \text{ ms}$$

$$\text{Turnaround medio} = 30,8 \text{ ms}$$

- Tempi di esecuzione(Priorità con prelazione)

- Turnaround normalizzato

$$P1 = (47 - 4)/12 = 43/12 = 3,5 \text{ ms}$$

$$P2 = (52 - 2)/5 = 50/5 = 10 \text{ ms}$$

$$P3 = (18 - 7)/10 = 11/10 = 1,1 \text{ ms}$$

$$P4 = (24 - 3)/8 = 21/8 = 2,6 \text{ ms}$$

$$P5 = (34 - 5)/9 = 29/9 = 3,2 \text{ ms}$$

$$\text{Turnaround normalizzato medio} = 4,08$$

- Throughput

$$X = \frac{5}{52000} * 1000 = 0,09$$

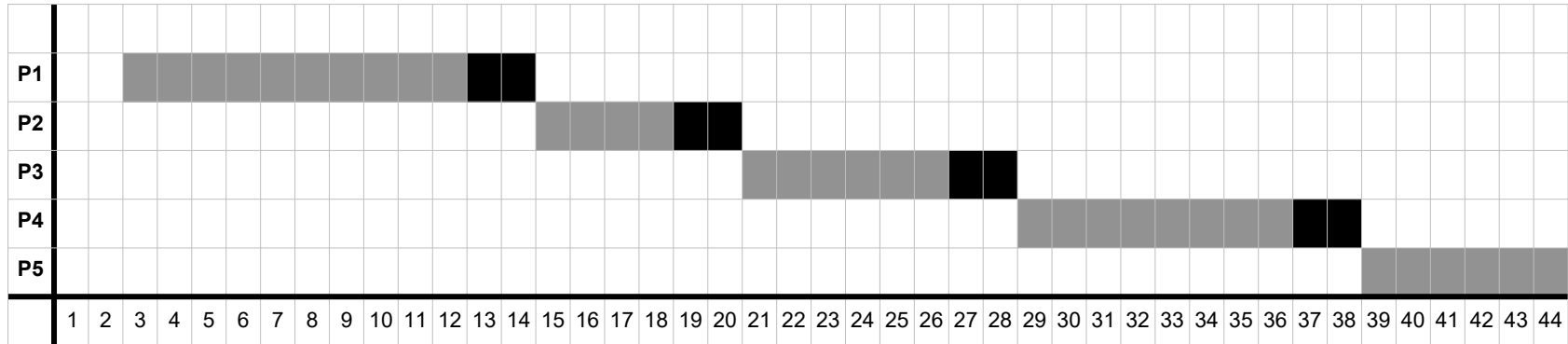
Esercizio 2

- Si considerino i seguenti processi, attivi in un sistema multiprogrammato:

Processo	Tempo di Arrivo	CPU Burst
P1	2ms	10ms
P2	4ms	4ms
P3	10ms	6ms
P4	14ms	8ms
P5	18ms	6ms

- Assumendo che il context switch richieda 2 ms, fornire il diagramma di esecuzione, il tempo medio di attesa e completamento per lo scheduling:
 - FCFS
 - STG

- Diagramma di esecuzione: (FCFS)



Processo	Tempo di Arrivo	CPU Burst
P1	2ms	10ms
P2	4ms	4ms
P3	10ms	6ms
P4	14ms	8ms
P5	18ms	6ms

- Tempi di esecuzione: (FCFS)

- Tempo di Attesa:

$$P1 = 12 - 10 - 2 = 0 \text{ ms}$$

$$P2 = 18 - 4 - 4 = 10 \text{ ms}$$

$$P3 = 26 - 6 - 10 = 10 \text{ ms}$$

$$P4 = 36 - 8 - 14 = 14 \text{ ms}$$

$$P5 = 44 - 6 - 18 = 20 \text{ ms}$$

$$\text{Tempo di attesa medio} = 10,8\text{ms}$$

- Turnaround:

$$P1 = 12 - 2 = 10 \text{ ms}$$

$$P2 = 18 - 4 = 14 \text{ ms}$$

$$P3 = 26 - 10 = 16 \text{ ms}$$

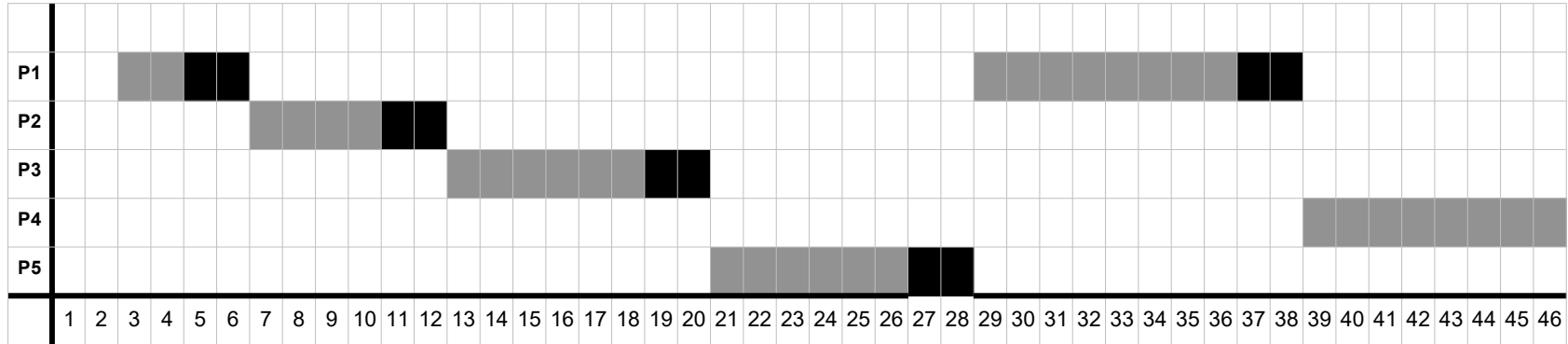
$$P4 = 36 - 14 = 22 \text{ ms}$$

$$P5 = 44 - 18 = 26 \text{ ms}$$

$$\text{Turnaround medio} = ..$$

- Tempi di esecuzione: (FCFS)
- Turnaround Normalizzato:
Turnaround Normalizzato Medio:
- Throughput:

- Diagramma di esecuzione: (STG)



Processo	Tempo di Arrivo	CPU Burst
P1	2ms	10ms
P2	4ms	4ms
P3	10ms	6ms
P4	14ms	8ms
P5	18ms	6ms

- Tempi di esecuzione: (SJF)

- Tempo di Attesa:

$$P1 = 36 - 10 - 2 = 22 \text{ ms}$$

$$P2 = 10 - 4 - 4 = 2 \text{ ms}$$

$$P3 = 18 - 6 - 10 = 2 \text{ ms}$$

$$P4 = 46 - 8 - 14 = 24 \text{ ms}$$

$$P5 = 26 - 6 - 18 = 2 \text{ ms}$$

$$\text{Tempo di attesa medio} = 10,4 \text{ ms}$$

- Turnaround:

$$P1 = 36 - 2 = 34 \text{ ms}$$

$$P2 = 10 - 4 = 6 \text{ ms}$$

$$P3 = 18 - 10 = 8 \text{ ms}$$

$$P4 = 46 - 14 = 32 \text{ ms}$$

$$P5 = 26 - 18 = 8 \text{ ms}$$

$$\text{Turnaround medio} = ..$$

- Tempi di esecuzione: (SJF)
- Turnaround Normalizzato:
Turnaround Normalizzato Medio:
- Throughput:

Esercizio 3

- Si considerino i seguenti processi, attivi in un sistema multiprogrammato:

Processo	Tempo di Arrivo	CPU Burst	Priorità
P1	4ms	6ms	3
P2	9ms	14ms	2
P3	6ms	12ms	1
P4	6ms	8ms	5
P5	5ms	16ms	4

- Supponendo che il cambio di contesto sia 3ms, si mostri l'ordine di esecuzione dei processi e quanto vale il tempo di attesa medio, il tempo di turnaround medio ed il tempo di turnaround normalizzato medio per ciascuno dei seguenti algoritmi di scheduling della CPU:
 - STG
 - FCFS
 - Priorita (senza prelazione)

03 ESERCITAZIONE

L'Esercizio

Si consideri il seguente problema: In un ambulatorio medico lavorano due medici ed un infermiere. Inizialmente i due medici sono in attesa dell'arrivo dei pazienti. Un paziente entra nell'ambulatorio e si reca dal primo medico libero che esegue la visita medica. Al termine della visita, il medico redige un referto che inserisce in un portadocumenti con M posizioni. L'infermiere preleva un referto alla volta e lo inserisce nel database dell'ambulatorio. Se entrambi i medici sono impegnati, il paziente si accomoda nella sala d'aspetto che dispone di N sedie. Se le N sedie sono tutte occupate, il paziente lascia l'ambulatorio.

L Esercizio (non wall of text)

Si consideri il seguente problema:

- In un ambulatorio medico lavorano due medici ed un infermiere.
- Inizialmente i due medici sono in attesa dell'arrivo dei pazienti.
- Un paziente entra nell'ambulatorio e si reca dal primo medico libero che esegue la visita medica.
- Al termine della visita, il medico redige un referto che inserisce in un portadocumenti con M posizioni.
- L'infermiere preleva un referto alla volta e lo inserisce nel database dell'ambulatorio.
- Se entrambi i medici sono impegnati, il paziente si accomoda nella sala d'aspetto che dispone di N sedie.
- Se le N sedie sono tutte occupate, il paziente lascia l'ambulatorio.

Possibile soluzione: Main

```
main() {

    for i=0 to K:
        crea_paziente();

    for i=0 to 2:
        crea_medico();

    crea_infermiere();

    // Aspetta che finiscano tutti ed esci

}
```

```
// Semafori contatore: il primo rappresenta i contenitori
// disponibili, il secondo invece i ripiani per lo specifico
// contenitore (array di semafori contatore)
```

```
// Inizialmente 2 medici
sem cont: medici = 2
```

```
// Inizialmente non c'è nessun paziente
sem cont: aspetta_paziente = 0
```

```
// Inizialmente l'infermiere non ha nessun referto
sem cont: aspetta_referto = 0
```

```
// Inizialmente il medico può produrre
Sem cont: produci_referto = M
```

```
// Inizialmente la visita non è finita
Sem bin: visita_finita = 0
```

```
// Inizialmente la sala d'attesa è vuota (N posti disp)
sem cont: sala_attesa = N
```

```
// Inizialmente non c'è nessun referto pronto
sem cont: referto_pronto = 0
```

```
// Sezione critica: due semafori, uno per barbiere uno per
// prod/cons
mutex: cs = 1
```

```
// Portadocumenti di M posizioni e indici vari
array int: portadocumenti[M] = {0}
int indice_doc = 0
int in_attesa = 0
int SEDIE = N
```

L Possibile soluzione: Medico

Funzione per il processo medico

```
medico() {
    repeat
        wait(aspetta_paziente);
        fai_visita();
        signal(visita_finita);

        referto = crea_referto();

        wait(produci_referto)
        lock(cs);
        portadocumenti[indice_doc++] = referto
        signal(sala_attesa)
        if in_attesa > 0 {in_attesa-=1};
        unlock(cs)

        signal(referto_pronto)

    forever
}
```

```
sem cont: medici = 2
sem cont: aspetta_paziente = 0
sem cont: aspetta_referto = 0
sem cont: sala_attesa = N
sem cont: referto_pronto = 0
Sem cont: produci_referto = M
Sem bin: visita_finita = 0
mutex: cs = 1
array int: portadocumenti[M] = {0}
int indice_doc = 0
int in_attesa = N
int SEDIE = N
```

L Possibile soluzione: Paziente

Funzione per il processo Paziente

```
paziente() {
```

```
    lock(cs)
```

```
    if(sem_getval(medici) != 0) {
```

```
        // La wait è sempre a buon fine poiche medici != 0
```

```
        wait(medici)
```

```
        unlock(cs)
```

```
        signal(aspetta_paziente)
```

```
        wait(visita_finita)
```

```
    } else {
```

```
        if (in_attesa < SEDIE) {
```

```
            in_attesa += 1
```

```
            unlock(cs)
```

```
            wait(sala_attesa)
```

```
            // Quando il paziente esce dalla sala d'attesa ritorna all'inizio e
```

```
            // fa di nuovo il controllo su medico
```

```
        } else {
```

```
            unlock(cs)
```

```
            exit(1)
```

```
        }
```

```
    }
```

```
sem cont: medici = 2
```

```
sem cont: aspetta_paziente = 0
```

```
sem cont: aspetta_referto = 0
```

```
sem cont: sala_attesa = N
```

```
sem cont: referto_pronto = 0
```

```
Sem cont: produci_referto = M
```

```
Sem bin: visita_finita = 0
```

```
mutex: cs = 1
```

```
array int: portadocumenti[M] = {0}
```

```
int indice_doc = 0
```

```
int in_attesa = N
```

```
int SEDIE = N
```

L Possibile soluzione: Infermiere

Funzione per il processo Paziente

```
infermiere() {
    repeat
        wait(referto_pronto)
        lock(cs)
        referto = portadocumenti[indice_doc--]
        signal(produci_referto)
        unlock(cs)
        inserisci_db(referto)

    forever
}
```

```
sem cont: medici = 2
sem cont: aspetta_paziente = 0
sem cont: aspetta_referto = 0
sem cont: sala_attesa = N
sem cont: referto_pronto = 0
Sem cont: produci_referto = M
Sem bin: visita_finita = 0
mutex: cs = 1
array int: portadocumenti[M] = {0}
int indice_doc = 0
int in_attesa = N
int SEDIE = N
```