Esercizi di Preparazione all'Esame

Corso di Sistemi Operativi

Ingegneria e Scienze informatiche, Università di Bologna

Prof. Stefano Ferretti

Gli esercizi potrebbero essere soggetti a errori, problemi di comprensione o altro. Si prega di avvisare il docente, qualora si dovessero riscontrare problemi in questo senso. Grazie

Scheduling

Esercizio 1

Processi: P1, P2, P3
Lunghezza dei CPU-burst (ms):
- P1: 15+5
- P2: 7+9
- P3: 4+7

Attese di I/O brevi: 3 ms
Politiche di scheduling:
- Round Robin (quanto di tempo: 5 ms)
- Shortest Job First (SJF)
- First Come First Serve (FCFS)

Si discuta qual'è l'ordine di esecuzione del processi con le politiche di scheduling.
Si calcoli anche il tempo medio di attesa e il tempo medio di turnaround che si ottengono con le due politiche di scheduling.

Esercizio 2

Processi: P1, P2, P3
Lunghezza dei CPU-burst (ms):
- P1: 10+8
- P2: 6+6
- P3: 3+5

Attese di I/O brevi: 2 ms
Politiche di scheduling:
- Round Robin (quanto di tempo: 4 ms)
- Shortest Job First (SJF)

First Come First Serve (FCFS)

Si discuta qual'è l'ordine di esecuzione del processi con le politiche di scheduling.
Si calcoli anche il tempo medio di attesa e il tempo medio di turnaround che si ottengono con le due politiche di scheduling.

Esercizio 3

Processi: P1, P2, P3
Lunghezza dei CPU-burst (ms):
P1: 8+6
P2: 5+8
P3: 2+4
Attese di I/O brevi: 1 ms
Politiche di scheduling:
Round Robin (quanto di tempo: 3 ms)
Shortest Job First (SJF)
First Come First Serve (FCFS)

Si discuta qual'è l'ordine di esecuzione del processi con le politiche di scheduling.
Si calcoli anche il tempo medio di attesa e il tempo medio di turnaround che si ottengono con le due politiche di scheduling.

Esercizio 4

Processi: P1, P2, P3
Lunghezza dei CPU-burst (ms):
P1: 12+10
P2: 9+7
P3: 6+5
Attese di I/O: 5 ms
Politiche di scheduling:
Round Robin (quanto di tempo: 4 ms)
Shortest Job First (SJF)
First Come First Serve (FCFS)

Si discuta qual'è l'ordine di esecuzione del processi con le politiche di scheduling.
Si calcoli anche il tempo medio di attesa e il tempo medio di turnaround che si ottengono con le due politiche di scheduling.

Esercizio 5

Processi: P1, P2, P3
Lunghezza dei CPU-burst (ms):
P1: 20+5
P2: 10+8
P3: 5+3
Attese di I/O: 3 ms
Politiche di scheduling:
Round Robin (quanto di tempo: 6 ms)
Shortest Job First (SJF)
First Come First Serve (FCFS)

Si discuta qual'è l'ordine di esecuzione del processi con le politiche di scheduling.
Si calcoli anche il tempo medio di attesa e il tempo medio di turnaround che si ottengono con le due politiche di scheduling.

Esercizio 6

Processi: P1, P2, P3
Lunghezza dei CPU-burst (ms):
P1: 7+9
P2: 3+6
P3: 5+7
Attese di I/O brevi: 2 ms
Politiche di scheduling:
Round Robin (quanto di tempo: 4 ms)
Shortest Job First (SJF)
First Come First Serve (FCFS)

Si discuta qual'è l'ordine di esecuzione del processi con le politiche di scheduling.
Si calcoli anche il tempo medio di attesa e il tempo medio di turnaround che si ottengono con le due politiche di scheduling.

Esercizio 7

Processi: P1, P2, P3
Lunghezza dei CPU-burst (ms):
P1: 9+10
P2: 5+8
P3: 4+6
Attese di I/O brevi: 3 ms
Politiche di scheduling:
Round Robin (quanto di tempo: 4 ms)
Shortest Job First (SJF)
First Come First Serve (FCFS)

Si discuta qual'è l'ordine di esecuzione del processi con le politiche di scheduling.
Si calcoli anche il tempo medio di attesa e il tempo medio di turnaround che si ottengono con le due politiche di scheduling.

Esercizio 8

Processi: P1, P2, P3
Lunghezza dei CPU-burst (ms):
P1: 10+6
P2: 8+9
P3: 6+4
Attese di I/O brevi: 1 ms
Politiche di scheduling:
Round Robin (quanto di tempo: 3 ms)
Shortest Job First (SJF)

First Come First Serve (FCFS)

Si discuta qual'è l'ordine di esecuzione del processi con le politiche di scheduling.
Si calcoli anche il tempo medio di attesa e il tempo medio di turnaround che si ottengono con le due politiche di scheduling.

Esercizio 9

Processi: P1, P2, P3
Lunghezza dei CPU-burst (ms):
P1: 14+5
P2: 6+6
P3: 8+7
Attese di I/O brevi: 2 ms
Politiche di scheduling:
Round Robin (quanto di tempo: 4 ms)
Shortest Job First (SJF)
First Come First Serve (FCFS)

Si discuta qual'è l'ordine di esecuzione del processi con le politiche di scheduling.
Si calcoli anche il tempo medio di attesa e il tempo medio di turnaround che si ottengono con le due politiche di scheduling.

Esercizio 10

Processi: P1, P2, P3
Lunghezza dei CPU-burst (ms):
P1: 11+6
P2: 7+8
P3: 3+5
Attese di I/O brevi: 2 ms
Politiche di scheduling:
Round Robin (quanto di tempo: 5 ms)
Shortest Job First (SJF)
First Come First Serve (FCFS)

Si discuta qual'è l'ordine di esecuzione del processi con le politiche di scheduling.
Si calcoli anche il tempo medio di attesa e il tempo medio di turnaround che si ottengono con le due politiche di scheduling.

Paginazione

Esercizio 11

Si consideri la seguente sequenza di pagine da caricare in memoria, con paginazione su richiesta: 2 7 9 5 2 7 6 3 8 5 9 3 4 6. Si illustri il comportamento dell'algoritmo LRU con 3 frame e si determini il numero di page fault.

Esercizio 12
Si consideri la seguente sequenza di pagine da caricare in memoria, con paginazione su richiesta: 4 3 6 8 3 7 9 5 1 4 6 7 8 2. Si illustri il comportamento dell'algoritmo LRU con 4 frame e si determini il numero di page fault.

Esercizio 13
Si consideri la seguente sequenza di pagine da caricare in memoria, con paginazione su richiesta: 3 8 6 4 2 9 7 5 1 8 6 3 2 5. Si illustri il comportamento dell'algoritmo LRU con 5 frame e si determini il numero di page fault.

Esercizio 14
Si consideri la seguente sequenza di pagine da caricare in memoria, con paginazione su richiesta: 5 9 2 6 1 7 4 3 2 8 5 1 4 6. Si illustri il comportamento dell'algoritmo LRU con 3 frame e si determini il numero di page fault.

Esercizio 15
Si consideri la seguente sequenza di pagine da caricare in memoria, con paginazione su richiesta: 6 3 1 8 7 5 4 2 9 3 6 8 7 5. Si illustri il comportamento dell'algoritmo LRU con 4 frame e si determini il numero di page fault.

Esercizio 16
Si consideri la seguente sequenza di pagine da caricare in memoria, con paginazione su richiesta: 8 2 5 7 4 1 3 6 9 8 2 5 4 7. Si illustri il comportamento dell'algoritmo LRU con 3 frame e si determini il numero di page fault.

Esercizio 17
Si consideri la seguente sequenza di pagine da caricare in memoria, con paginazione su richiesta: 7 1 4 6 3 5 9 2 8 7 1 4 6 3. Si illustri il comportamento dell'algoritmo LRU con 5 frame e si determini il numero di page fault.

Esercizio 18
Si consideri la seguente sequenza di pagine da caricare in memoria, con paginazione su richiesta: 1 5 9 3 2 6 8 7 4 1 5 9 3 2. Si illustri il comportamento dell'algoritmo LRU con 4 frame e si determini il numero di page fault.

Esercizio 19
Si consideri la seguente sequenza di pagine da caricare in memoria, con paginazione su richiesta: 9 7 2 4 6 3 1 8 5 9 7 2 4 6. Si illustri il comportamento dell'algoritmo LRU con 3 frame e si determini il numero di page fault.

Esercizio 20

Si consideri la seguente sequenza di pagine da caricare in memoria, con paginazione su richiesta: 3 6 8 1 4 9 2 5 7 3 6 8 1 4. Si illustri il comportamento dell'algoritmo LRU con 4 frame e si determini il numero di page fault.

Gestione Risorse

Esercizio 21

Consideriamo un sistema con tre tipi di risorse: CPU, memoria, e dischi. Ci sono cinque processi, P1, P2, P3, P4, e P5, che possono richiedere e rilasciare risorse durante l'esecuzione. La tabella seguente mostra le risorse attualmente allocate, le massime richieste, e le risorse disponibili nel sistema.

	CPU Allocata	Memoria Allocata	Disco Allocato	CPU Massima	Memoria Massima	Disco Massimo
P1	1	2	2	3	3	2
P2	2	0	1	3	2	2
P3	3	1	1	3	2	2
P4	0	3	2	2	3	2
P5	1	1	2	3	2	3

Risorse disponibili nel sistema:

- CPU: 1
- Memoria: 1
- Disco: 0

Domande:

1. Determinare se il sistema si trova in uno stato sicuro, e in caso, mostrare una sequenza sicura per l'esecuzione dei processi.
2. Supponiamo che P2 richieda 1 CPU e 1 memoria. Determinare se la richiesta può essere soddisfatta immediatamente senza portare il sistema in uno stato insicuro.

Soluzione:

1. Determinare se il sistema è in uno stato sicuro

Prima, calcoliamo il bisogno di ogni processo:

- **P1**: CPU: 3-1=2, Memoria: 3-2=1, Disco: 2-2=0
- **P2**: CPU: 3-2=1, Memoria: 2-0=2, Disco: 2-1=1
- **P3**: CPU: 3-3=0, Memoria: 2-1=1, Disco: 2-1=1
- **P4**: CPU: 2-0=2, Memoria: 3-3=0, Disco: 2-2=0
- **P5**: CPU: 3-1=2, Memoria: 2-1=1, Disco: 3-2=1

Risorse disponibili nel sistema:

- CPU: 1
- Memoria: 1
- Disco: 0

Per verificare lo stato di sicurezza, dobbiamo trovare una sequenza sicura. La sequenza sicura deve permettere che ogni processo possa completare, uno alla volta, rilasciando risorse per il prossimo processo.

Sequenza di verifica:

- P3: CPU: 1+3=4, Memoria: 1+1=2, Disco: 0+1=1 (rilascia risorse)
- P4: CPU: 4+0=4, Memoria: 2+3=5, Disco: 1+2=3 (rilascia risorse)
- P1: CPU: 4-1=3+3=6, Memoria: 5-2=3+3=6, Disco: 3-2=1+2=3 (rilascia risorse)
- P5: CPU: 6-1=5+3=8, Memoria: 6-1=5+2=7, Disco: 3-2=1+3=4 (rilascia risorse)
- P2: CPU: 8-2=6+3=9, Memoria: 7-0=7+2=9, Disco: 4-1=3+2=5 (rilascia risorse)

La sequenza sicura è: P3, P4, P1, P5, P2. Quindi, il sistema è in uno stato sicuro.

2. Verificare la richiesta di P2

P2 richiede 1 CPU e 1 memoria. Risorse disponibili:

- CPU: 1-1=0
- Memoria: 1-1=0
- Disco: 0

Dopo la richiesta, le risorse allocate a P2:

- CPU: 2+1=3
- Memoria: 0+1=1
- Disco: 1

Verifichiamo ancora lo stato di sicurezza:

- P3: CPU: 0+3=3, Memoria: 0+1=1, Disco: 0+1=1 (rilascia risorse)
- P4: CPU: 3+0=3, Memoria: 1+3=4, Disco: 1+2=3 (rilascia risorse)

- P1: CPU: 3-1=2+3=5, Memoria: 4-2=2+3=5, Disco: 3-2=1+2=3 (rilascia risorse)
- P5: CPU: 5-1=4+3=7, Memoria: 5-1=4+2=6, Disco: 3-2=1+3=4 (rilascia risorse)
- P2: CPU: 7-3=4+3=7, Memoria: 6-1=5+2=7, Disco: 4-1=3+2=5 (rilascia risorse)

La sequenza sicura è ancora: P3, P4, P1, P5, P2. Quindi, la richiesta di P2 può essere soddisfatta senza portare il sistema in uno stato insicuro

Esercizio 22

Consideriamo un sistema con tre risorse denominate A, B e C. Ci sono quattro processi, P1, P2, P3, e P4. La tabella seguente mostra le risorse attualmente allocate, le massime richieste e le risorse disponibili nel sistema.

	A Alloc	B Alloc	C Alloc	A Max	B Max	C Max
P1	1	2	1	3	3	2
P2	2	1	0	3	2	2
P3	0	3	1	2	3	1
P4	1	0	2	3	1	3

Risorse disponibili: A: 1, B: 1, C: 2.

1. Verificare se lo stato è sicuro.
2. Determinare una sequenza sicura per l'esecuzione dei processi.

Esercizio 23

Consideriamo un sistema con tre risorse denominate CPU, Memoria e Disco. Ci sono cinque processi, P1, P2, P3, P4, e P5. La tabella seguente mostra le risorse attualmente allocate, le massime richieste e le risorse disponibili nel sistema.

	CPU All	Mem All	Disco All	CPU Max	Mem Max	Disco Max
P1	2	3	1	3	5	3
P2	1	2	1	4	2	2
P3	3	1	2	4	3	2
P4	1	2	1	2	3	1
P5	2	1	2	5	3	2

Risorse disponibili: CPU: 1, Memoria: 1, Disco: 0.

1. Verificare se lo stato è sicuro.
2. Determinare una sequenza sicura per l'esecuzione dei processi.

Esercizio 24
Consideriamo un sistema con quattro risorse denominate X, Y, Z e W. Ci sono cinque processi, P1, P2, P3, P4, e P5. La tabella seguente mostra le risorse attualmente allocate, le massime richieste e le risorse disponibili nel sistema.

	X Alloc	Y Alloc	Z Alloc	W Alloc	X Max	Y Max	Z Max	W Max
P1	2	1	1	2	3	2	2	3
P2	1	2	0	1	2	3	1	2
P3	1	1	1	0	2	2	2	1
P4	0	3	2	1	1	3	3	2
P5	2	1	1	2	3	2	2	3

Risorse disponibili: X: 2, Y: 1, Z: 1, W: 0.

1. Verificare se lo stato è sicuro.
2. Determinare una sequenza sicura per l'esecuzione dei processi.

Esercizio 25
Consideriamo un sistema con tre risorse denominate R1, R2 e R3. Ci sono quattro processi, P1, P2, P3, e P4. La tabella seguente mostra le risorse attualmente allocate, le massime richieste e le risorse disponibili nel sistema.

	R1 Alloc	R2 Alloc	R3 Alloc	R1 Max	R2 Max	R3 Max
P1	2	1	0	4	3	1
P2	1	3	2	3	4	3
P3	1	2	3	4	3	4
P4	2	1	1	3	3	3

Risorse disponibili: R1: 1, R2: 1, R3: 2.

1. Verificare se lo stato è sicuro.
2. Determinare una sequenza sicura per l'esecuzione dei processi.

Esercizio 26
Consideriamo un sistema con tre risorse denominate P1, P2 e P3. Ci sono cinque processi, P1, P2, P3, P4, e P5. La tabella seguente mostra le risorse attualmente allocate, le massime richieste e le risorse disponibili nel sistema.

	P1 Alloc	P2 Alloc	P3 Alloc	P1 Max	P2 Max	P3 Max
P1	1	2	1	2	3	2
P2	2	0	0	4	1	2
P3	1	1	2	2	2	2
P4	0	3	3	1	3	3
P5	3	1	0	3	2	2

Risorse disponibili: P1: 1, P2: 2, P3: 1.

1. Verificare se lo stato è sicuro.
2. Determinare una sequenza sicura per l'esecuzione dei processi.

Esercizio 27

Consideriamo un sistema con tre risorse denominate CPU, Memoria e Disco. Ci sono quattro processi, P1, P2, P3, e P4. La tabella seguente mostra le risorse attualmente allocate, le massime richieste e le risorse disponibili nel sistema.

	CPU Alloc	Memoria Alloc	Disco Alloc	CPU Max	Memoria Max	Disco Max
P1	1	2	1	3	4	2
P2	2	0	0	3	3	1
P3	1	3	2	2	3	3
P4	3	1	1	4	2	2

Risorse disponibili: CPU: 1, Memoria: 1, Disco: 2.

1. Verificare se lo stato è sicuro.
2. Determinare una sequenza sicura per l'esecuzione dei processi.

Concorrenza

Esercizio 28

Completare le seguenti porzioni di codice, utilizzando Semafori e variabili condivise, in modo che a video compaia stampata ripetutamente la stringa “Hello World!”.

```
Process A {
  while (true) {
```

```
    print("Hel");
  }
}

Process B {
  while (true) {
    print("lo ");
  }
}

Process C {
  while (true) {
    print("World!\n");
  }
}
```

Esercizio 29

Completare le seguenti porzioni di codice, utilizzando Semafori e variabili condivise, in modo che i processi stampino "ABECDF" in sequenza.

```
Process A {
  while (true) {
    print("A");
    print("E");
  }
}

Process B {
  while (true) {
    print("B");
    print("F");
  }
}

Process C {
  while (true) {
    print("C");
  }
}

Process D {
```

```
  while (true) {
    print("D");
  }
}
```

Esercizio 30

Utilizzare semafori per sincronizzare due processi che calcolano la somma di due numeri, garantendo che il processo B inizi solo dopo che il processo A ha calcolato il primo numero.

Riscrivere poi il problema per risolverlo coi Monitor.

Riscrivere poi il problema per risolverlo usando le tecniche di Message Passing.

```
Process A {
  while (true) {
    int x = calculate();
  }
}


Process B {
  while (true) {
    int y = calculate();
    int sum = x + y;
    print(sum);
  }
}
```

Esercizio 31

Implementare un sistema con 3 processi che utilizzano message passing per calcolare una media di valori. Ogni processo invia il proprio dato agli altri due processi e riceve i dati dagli altri.

```
Process A {
  while (true) {
    x = generateNumber();
    int average = (x + y + z) / 3;
    print(average);
  }
}

Process B {
  while (true) {
```

```
    y = generateNumber();
    int average = (x + y + z) / 3;
    print(average);
  }
}

Process C {
  while (true) {
    z = generateNumber();
    int average = (x + y + z)
  }
}
```

Come si risolverebbe il problema se si avessero a disposizione i Semafori?
Come si risolverebbe il problema se si avessero a disposizione i Monitor?