

### ESERCIZIO Scheduling 1

In un sistema vengono generati 5 processi (A,B,C,D,E), con i tempi di arrivo e le durate (in millisecondi) sotto specificate:

Processo	Durata	Tempo di arrivo
A	25	0
B	60	1
C	5	2
D	15	3
E	10	4

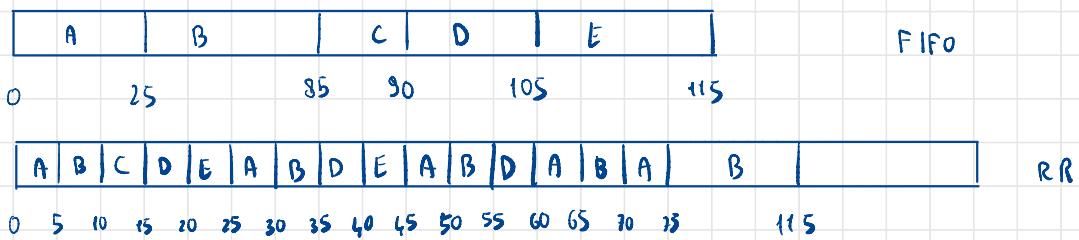
Tutti i processi avanzano senza mai sospendersi.

Calcolare il tempo di permanenza nel sistema di ogni processo (definito come differenza tra il tempo di completamento dell'esecuzione e il tempo di arrivo nel sistema) con le seguenti politiche di scheduling

1. Politica FIFO
2. Politica Round-Robin con quanto di tempo pari a 5 msec.

In entrambi i casi si ignori il tempo di commutazione di contesto.

**Soluzione**



Tempo di permanenza $\Rightarrow$	FIFO	ROUND ROBIN
A = 25		A = 75
B = 84		B = 114
C = 88		C = 13
D = 102		D = 57
E = 111		E = 41

## ESERCIZIO Scheduling 2

In un sistema vengono generati 6 processi (A,B,C,D,E,F), con i tempi di arrivo, le priorità e le durate (in millisecondi) sotto specificate:

PROCESSO	TEMPO DI ARRIVO	PRIORITÀ	DURATA
A	0	2	10
B	8	1	24
C	18	3	6
D	28	4	10
E	32	2	11
F	36	3	7

Lo scheduling del processore avviene con una politica a priorità (che assegna il processore al processo che ha il valore più elevato di priorità e a pari priorità, al processo arrivato per primo) e con prerilascio. Si suppone che, una volta in esecuzione, ogni processo avanzi senza mai sospendersi.

Riempire la seguente tabella, utilizzando una riga per ogni evento che provoca la riassegnazione del processore e specificando, in ogni riga, il processo in esecuzione e la composizione della coda pronti subito dopo il verificarsi dell'evento.

TEMPO	EVENTO	PROC. IN ESEC.	CODA DI READY	NOTE
0	arriva A	A	Ø	
8	arriva B	A	B	
10	Termina A	B	Ø	
13	arriva C	C	B	16 sec. residui a B
24	Termina C	B	Ø	
28	arriva D	D	B	12 sec. residui a B
32	arriva E	D	B → E	
36	arriva F	D	F → B → E	
38	Termina D	F	B → E	
45	Termina F	E	B	
56	Termina E	B	Ø	
68	Termina B	/	Ø	

### ESERCIZIO Scheduling 3

In un sistema vengono generati 5 processi (A,B,C,D,E), con i tempi di arrivo e le durate (in millisecondi) sotto specificate:

Processo	Durata	Tempo di arrivo
A	45	0
B	55	13
C	15	21
D	5	25
E	15	29

Si suppone che tutti i processi avanzino senza mai sospendersi.

Lo scheduler adotta la politica *Shortest Remaining Time First (SRTT)*, la quale seleziona per l'esecuzione il processo con minor tempo residuo in esecuzione e prevede il prerilascio.

Calcolare il tempo di permanenza nel sistema di ogni processo (definito come differenza tra il tempo di completamento dell'esecuzione e il tempo di arrivo nel sistema)

Tempo	Evento	Processo in esecuzione	Tempo residuo per processo in esecuzione	Coda ready (con tempo residuo)
0	arriva A	A	45	
13	arriva B	A	32	B(55)
21	arriva C	C	15	A(24) → B(55)
25	arriva D	D	5	C(11) → A(24) → B(55)
29	arriva E	D	1	E(15) → C(11) → A(24) → B(55)
30	finisce D	C	11	E(15) → A(24) → B(55)
41	finisce C	E	15	A(24) → B(55)
56	finisce E	A	24	B(55)
80	finisce A	B	55	
135	finisce B			

$$T_c(D) = 30 - 25 = 5$$

$$T_c(C) = 41 - 21 = 20$$

$$T_c(E) = 56 - 28 = 27$$

$$T_c(A) = 80 - 0 = 80$$

$$T_c(B) = 135 - 13 = 122$$

## ESERCIZIO Scheduling 4

Un sistema gestisce il processore con politica RoundRobin con quanto di tempo di **5 msec**. Quando un processo va in esecuzione gli viene assegnato un intero quanto di tempo di **5 ms**, indipendentemente dal tempo consumato nel precedente turno di esecuzione.

Nel sistema sono presenti 4 processi (A,B,C,D) e un semaforo di mutua esclusione *mux*. Al tempo t passa in esecuzione il processo A, la coda pronti contiene i processi B->C->D e il semaforo *mux* ha valore 0 e coda vuota.

Si chiede quale è il processo in esecuzione e la composizione della coda pronti *nell'intervallo di tempo da t a t+20*, nel corso del quale si verificano i seguenti eventi:

1. il processo in esecuzione si sospende al tempo t+8 sul semaforo *mux*;
2. il processo in esecuzione si sospende al tempo t+14 sul semaforo *mux*;
3. il processo in esecuzione esegue una *signal(mux)* al tempo t+16

T	Evento	In esecuzione	Coda ready	Sospesi in MUX
0		A	B → C → D	∅
5	prilar. de A	B	C → D → A	
8	B in blocca	C	D → A	B
13	prilar. de C	D	A → C	B
14	B in blocca	A	C	B → D
16	signal (MUX)	A	C → B	D
19	prilar. de A	C	B → A	D
20	c in esecuz.	C	B → A	∅

## ESERCIZIO Scheduling 5

Un sistema gestisce il processore combinando le politiche a priorità e RoundRobin con la tecnica delle code multiple (una coda FIFO per ogni valore di priorità; i processi pronti di uguale priorità sono inseriti in una stessa coda; il processore viene assegnato al processo che occupa la prima posizione nella coda non vuota di massima priorità; ai processi pronti di uguale priorità si applica la politica Round Robin).

Il quanto di tempo è di **10 msec**.

La politica prevede il prerilascio, che avviene immediatamente dopo l'evento che lo provoca, senza attendere l'esaurimento del quanto di tempo corrente. Quando un processo va in esecuzione gli viene assegnato un intero quanto di tempo, indipendentemente dal tempo consumato nel precedente turno di esecuzione.

Al tempo T, nel sistema sono presenti i seguenti processi:

- Processo A, con priorità 2, che al tempo T è in stato di attesa sul semaforo Sem1;
- Processo B, con priorità 3, che al tempo T è in stato di attesa sul semaforo Sem1;
- Processo C, con priorità 1, che al tempo T è in stato di pronto;
- Processo D, con priorità 2, che al tempo T passa in stato di esecuzione;
- Processo E, con priorità 1, che al tempo T è in stato di pronto;

Al tempo T la coda corrispondente alla priorità 1 contiene: C->E (C è in testa), e tutte le altre code sono vuote. La coda del semaforo Sem1 contiene invece A->B (A è in testa).

Si chiede quale è il processo in esecuzione e la composizione delle 3 code al tempo T+30 se si verifica la seguente sequenza di eventi:

1. al tempo T+8 il processo in esecuzione esegue una *signal* sul semaforo Sem1;
2. al tempo T+12 il processo in esecuzione esegue una *wait* sul semaforo Sem1;
3. al tempo T+14 termina il processo in esecuzione;
4. al tempo T+26 il processo in esecuzione esegue una *wait* sul semaforo Sem1.

Tempo	lineare	Esecuz.	Coda 1	Coda 2	Coda 3	Semaforo, codice
0	D in exec.	D	C->E			L0, A->B >
8	Signal Sem1	D	C->E	A		L0, BS
10	TS	A	C->E	D		
12	Wait Sem1	D	C->E			L0, B->A >
14	Termina D	C	E			
26	TS	E	C			L0, B->A->E)
30	Wait Sem1	C				
	C in exec.	C				