ESERCITAZIONE WEEK4 DAY2

EPICODE

Esercizio
Scheduling CPU

L'esercizio di oggi verte sui meccanismi di pianificazione dell'utilizzo della CPU (o processore).
In ottica di ottimizzazione della gestione dei processi, abbiamo visto come lo scheduler si sia evoluto nel tempo per passare da approccio mono-tasking ad approcci multi-tasking.

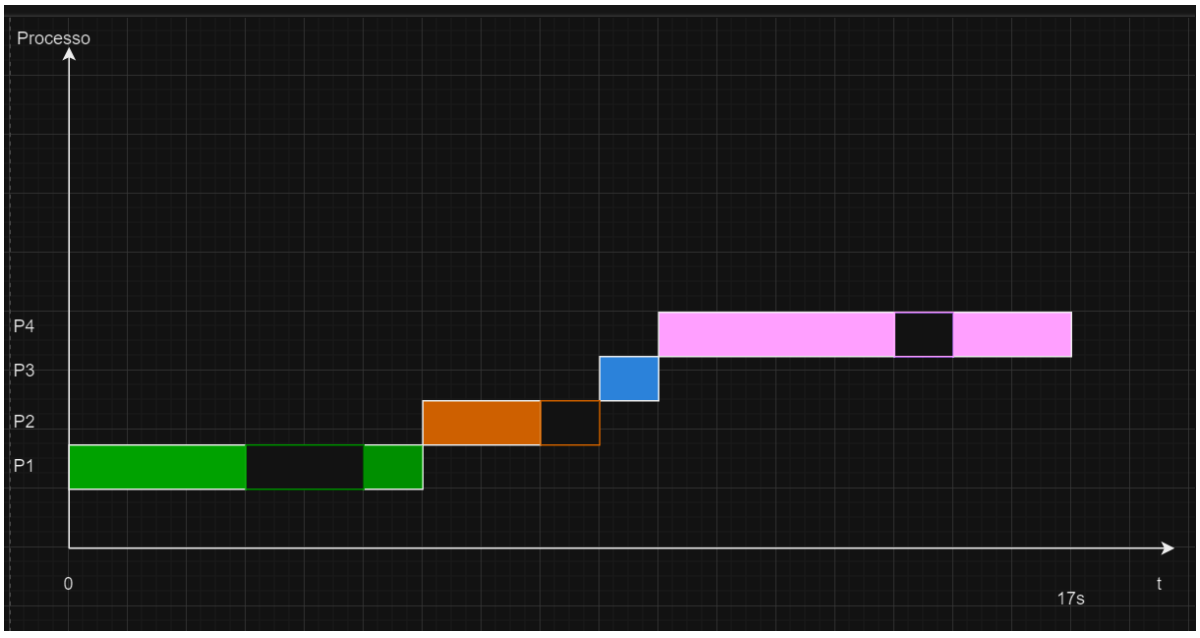
Traccia:

Si considerino 4 processi, che chiameremo P1, P2, P3, P4, con i tempi di esecuzione e di attesa input/output dati in tabella. I processi arrivano alle CPU in ordine P1, P2, P3, P4.
Individuare il modo più efficace per la gestione e l'esecuzione dei processi, **tra i metodi visti nella lezione teorica**.
Abbozzare un diagramma che abbia sulle ascisse il tempo passato da un instante «0» e sulle ordinate il nome del Processo.

Processo	Tempo di esecuzione	Tempo di attesa	Tempo di esecuzione dopo attesa
P1	3 secondi	2 secondi	1 secondo
P2	2 secondi	1 secondo	-
P3	1 secondi	-	-
P4	4 secondi	1 secondo	2 secondi

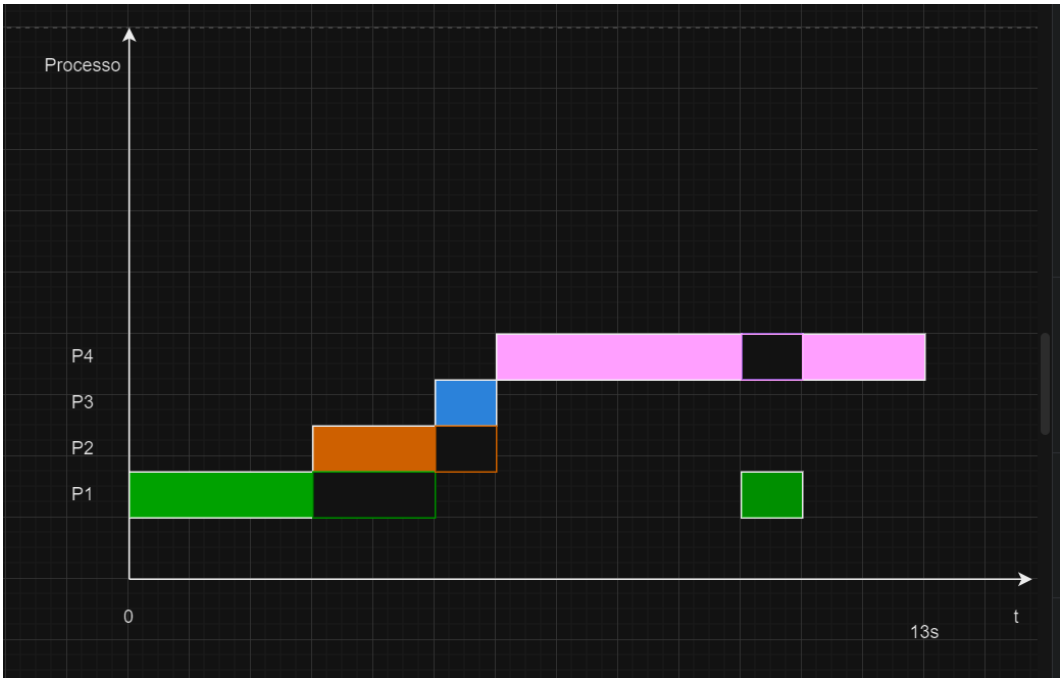
3

Se si scegliesse un sistema mono-tasking, non in grado dunque di eseguire su una CPU più processi parallelamente,dovendo rispettare l'ordine di arrivo dei processi alla CPU P1-P2-P3-P4, si otterrebbe una situazione del tipo:



dove ogni processo ha un suo tempo di esecuzione, un tempo di attesa (rettangoli vuoti) e un tempo di eventuale esecuzione post attesa. In questa modalità si deve attendere che ogni processo termini per eseguire il successivo e quindi ci vogliono 17 s in totale per terminare l'esecuzione dei 4 processi. È evidente uno spreco di tempo e quindi un inefficiente utilizzo della CPU.

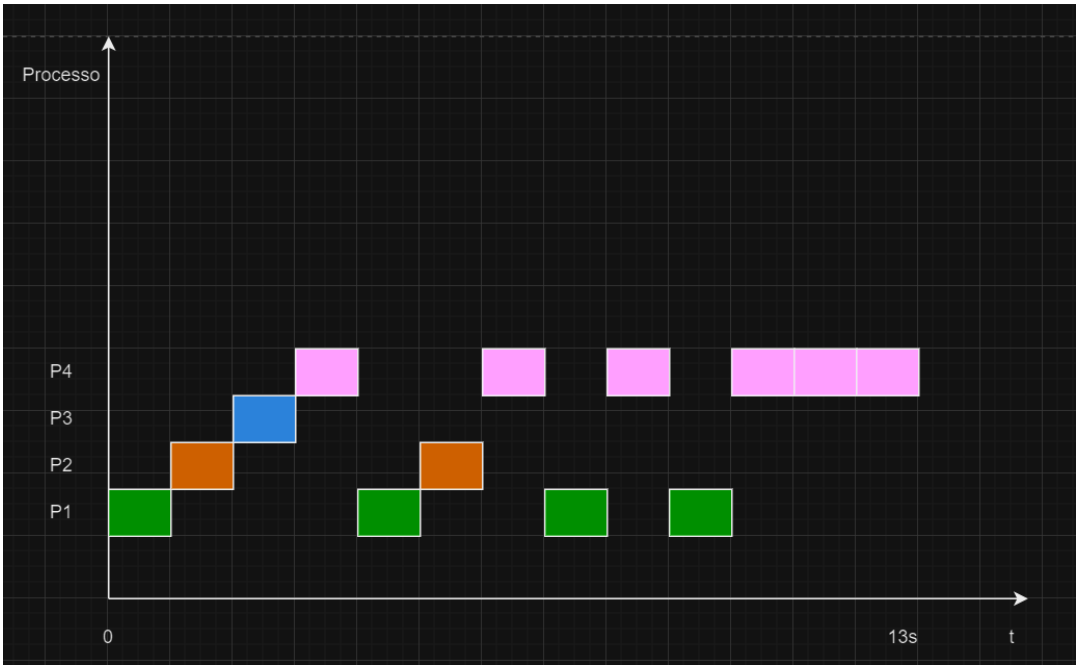
Con un approccio multi -tasking invece, con la pianificazione della prelazione, si sfruttano i tempi di attesa di un processo per eseguire un altro processo in modo da non sprecare risorse della CPU. Un esempio potrebbe essere il seguente, ipotizzando un necessario ordine di arrivo alla CPU P1-P2-P3-P4:



C'è un evidente recupero in termini temporali: la CPU esegue tutti e 4 i processi in 13 s anziché 17 s.

Durante il tempo di attesa di P1, può essere eseguito completamente P2 e durante il suo tempo di attesa di 1 s si esegue completamente P3. Durante il tempo di attesa di P4 si conclude l'esecuzione di P1.

Se invece si adopera l'approccio time sharing:



si potrebbe pensare di lavorare con un lasso di tempo standard detto “quanto” in modo che ogni processo venga eseguito in maniera ciclica per questo tempo e con una CPU di velocità elevata, si ha l’idea che i 4 processi siano eseguiti parallelamente.

Ipotizziamo che il quanto sia in questo caso 1 s:

- P1 inizia con un tempo di esecuzione di 3 s e poi avrà un tempo post attesa di 1 s. Dato che il quantum è di 1 secondo, la CPU eseguirà inizialmente P1 per 1 secondo, lasciando 2 secondi di tempo di esecuzione rimanente e 1 s di tempo post attesa. Quindi al momento il tempo di esecuzione di P1 è 1 s;
- P2 inizia con un tempo di esecuzione di 2 s. La CPU eseguirà P2 per 1 secondo, lasciando a 1 s il tempo di esecuzione rimanente per terminare P2. Quindi il tempo di esecuzione di P2 al momento è 1 secondo;
- P3 inizia con un tempo di esecuzione di 1 secondo. P3 verrà eseguito per l'intero tempo di esecuzione;
- P4 inizia con un tempo di esecuzione di 4 s e poi avrà un tempo di attesa di 2 s. Verrà eseguito dalla CPU per 1 s, lasciando 3 s di tempo di esecuzione rimanente e 2 s di post attesa per terminare la sua esecuzione; anche qui attualmente il tempo di esecuzione di P4 è 1 secondo;

Si ripete il ciclo fino a terminare tutti i processi. Al termine si ottiene:

Tempo totale di esecuzione di P1: 4 s

Tempo totale di esecuzione di P2: 2 s

Tempo totale di esecuzione di P3: 1 s

Tempo totale di esecuzione di P4: 6 s

Quindi, la CPU impiegherà 13 secondi per eseguire i 4 processi con l’approccio time sharing, a differenza dei 17 s dei sistemi mono-tasking.