Tema 3. Sistemas operativos en tempo real

Introducción

Un sistema en tempo real é un **sistema informático que recibe estímulos dende dispositivos físicos externos** e que debe **reaccionar de maneira apropiada a eles nunha cantidade fixa de tempo** (considérase que ter a resposta correcta pero demasiado tarde é tan malo como non tela).
Por exemplo, un reproductor de CD's de audio.

Os sistemas de tempo real divídense en **dúas categorías**:

- **Tempo real duro**: ten *tempos absolutos* que se deben cumprir, xa que unha resposta tardía pode ter consecuencias fatais (ex: aplicacións de sistemas médicos)
- **Tempo real suave**: é *tolerable fallar* nun tempo límite algunha vez (ex: apps multimedia)

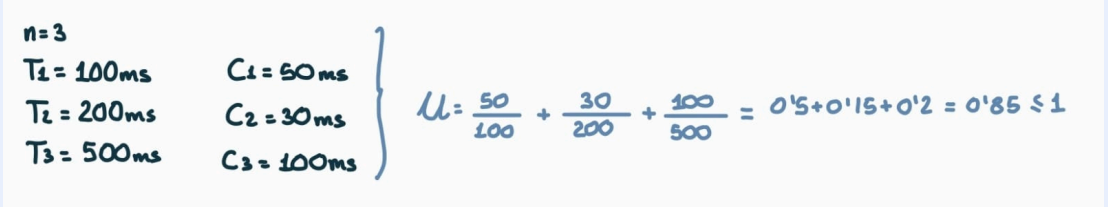
O **comportamento en tempo real** lógrase dividindo o programa en *procesos predecibles* con *tempos de vida moi curtos*, e que poden executase en moito menos de 1 segundo. Cando se **detecta un evento externo**, o planificador planifica os procesos respetando o cumprimento dos tempos límite. Estes *eventos* poden ser *periódicos* (ocorren a intervalos regulares) ou *aperiódcos* (ocorren de maneira impredecible).

Para saber **cándo é planificable un STR** debmos comprobar se compre co seguinte criterio.
Se hai n eventos periódicos, e o evento i ocorre co periodo T_i e require C_i segundos de tempo de CPU, entón só é planificable se *o factor de utilización U é menor ou igual a 1*:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} = U \leq 1$$

Exemplo

Se temos un STR con 3 eventos con periodos de 100, 200 e 500 ms respec. e que requiren 50, 30 e 100 ms de tempo de CPU por evento, entón este sistema é planificable, xa que $U \leq 1$



Algoritmos de planificación en tempo real

Os algoritmos de planificación en tempo real poden ser estáticos ou dinámicos. Os **estáticos** toman as súas decisións de planificación *antes* da execución, polo que só funcionan cando temos dispoñible información de antemán; mentres que os **dinámicos**, o fan durante o *tempo de execución*, polo que non teñen esa restricción.

Programación monotónica de frecuencia (RMS)

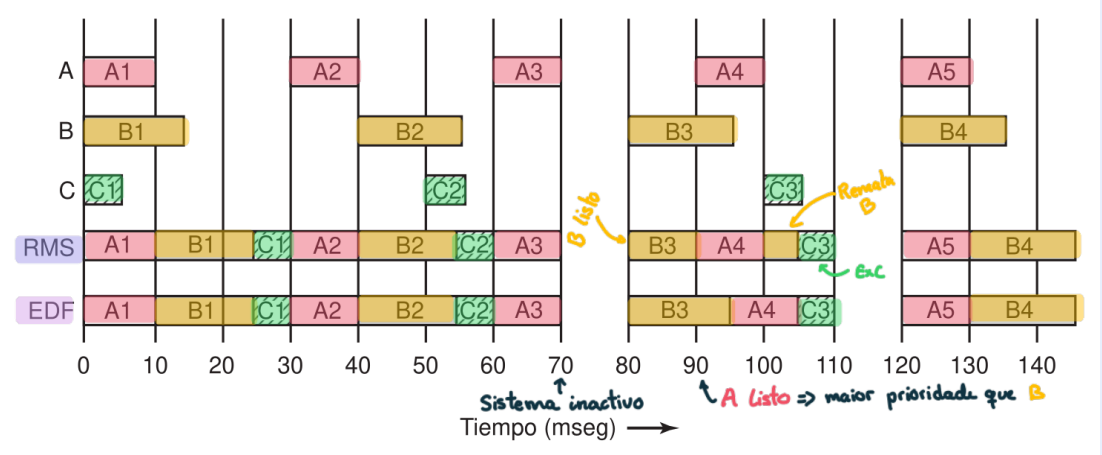
O algoritmo de programación de tempo real *estático* para los procesos periódicos é **RMS** (*Rate Monotonic Scheduling*). Para podelo empregar, o procesos deben cumprir as seguintes **condicións**:

1. Cada proceso periódico se debe *completar dentro do seu periodo*.
2. *Ningún proceso depende* de outro.
3. Cada proceso precisa a *mesma cantidade de tempo de CPU* en cada ráfaga.
4. *Ningún proceso aperiódico ten tempo de resposta*.
5. A *preferencia dos procesos* ocorre de forma *instantánea* e *sen sobrecarga*.

Así, o RMS **asigna a cada proceso unha prioridade fixa**, igual á frecuencia de ocorrencia do seu evento de activación, polo que *as prioridades son linais coa frecuencia* (por exemplo, un proceso que se execute cada 30 ms, execútase 33 veces cada segundo (1000ms/30ms), polo que obtén prioridade 33). Despois, en tempo de execución, o programador sempre **executa o proceso que estea listo e que teña a maior prioridade**.

Exemplo

Supoñendo procesos A, B e C con prioridades de 33, 25 e 20, respec., teñen a seguinte programación:



Primeiro escóllese o proceso con maior prioridade (A), e es executa ata que se completa. Despois execútanse B e C. En conxunto, tardaron 30ms en executarse os tres (tempo de frecuencia de A), polo que A volve a executarse nese momento. Seguen rotando así ata que o sistema se queda inactivo aos 70ms, xa que ningún proceso está listo. Aos 80ms, como B está listo, execútae, pero en t=90, como A está listo e ten maior prioridade, se substitúe por B e se

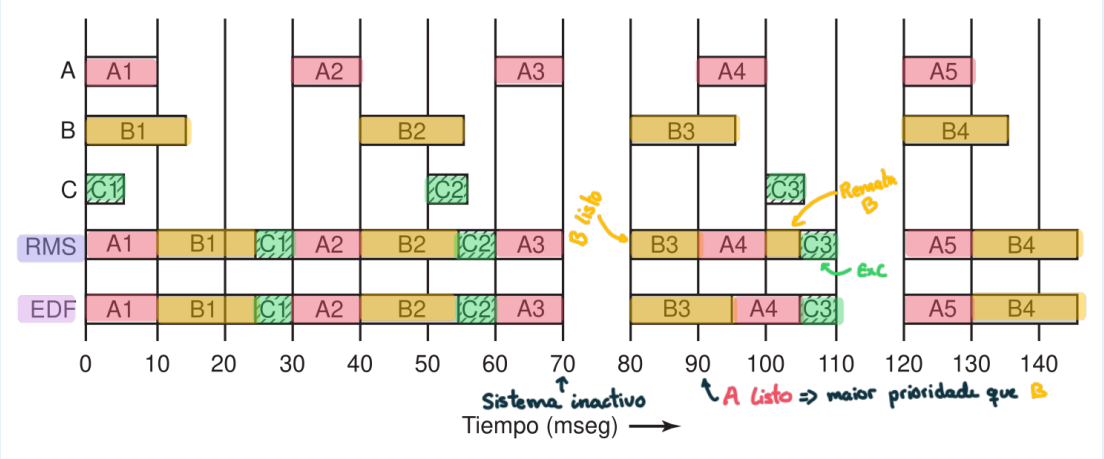
executa ata rematar (t=100). Despois o sistema continúa con B (maior prioridade) e, por último, executa C.

Programación do menor tempo de resposta primeiro (EDF)

O EDF é un algoritmo *dinámico* que non require nin que os procesos sexan periódicos (ao igual que o RMS), nin que o tempo de execución en cada ráfaga sexa o mesmo.
Cada vez que un **proceso ncesita tempo de CPU**, anuncia a súa presencia e seu **tempo de resposta**. O programador ten unha *lista de procesos executables ordenados polo seu tempo de resposta*. Así, o algoritmo *executa o primer proceso da lista* (o que ten o tempo de resposta máis cercano), e cada vez que un novo proceso está listo, o sistema comproba se ocorre o seu *tempo de resposta antes que o do proceso que se está executando*, e, se é así, o novo proceso reemplaza ao actual.

Exemplo

Supoñendo o exemplo de antes:

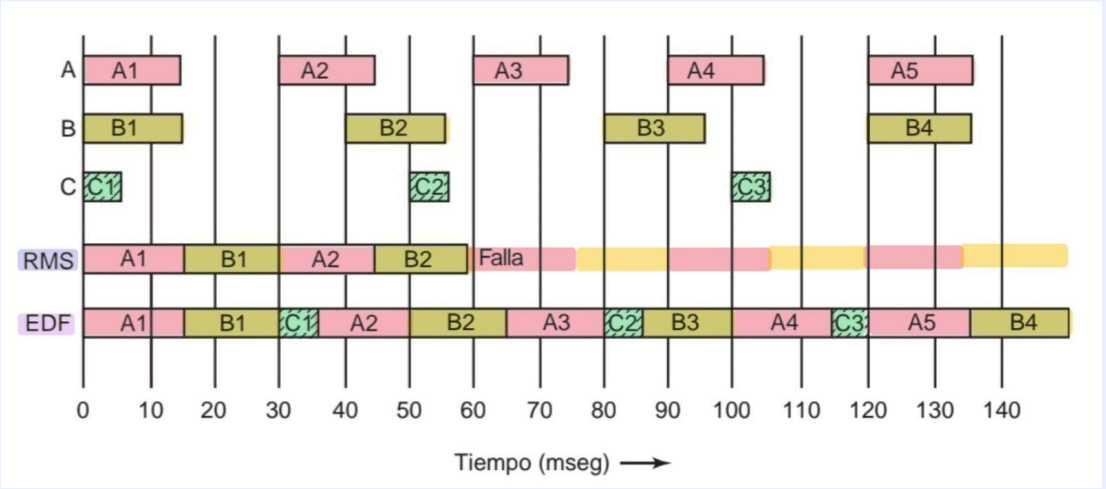


EDF ten o mesmo comprotamento que RMS ata chegar a t=90. Nese momento, tanto A como B teñen o mesmo tempo de espera (120ms), polo que EDF segue executando B en vez de cambiar a A.

Aínda así, é importante denotar que **RMS e EDF non sempre dan os mesmos resultados**.

Exemplo

Os periodos de A, B e C son iguais que no exemplo anterior, pero agora A necesita 15ms de CPU por ráfaga en vez de 10:



Neste caso RMS falla, xa que só executaría A e B, sen darlle tempo de CPU a C; mentres que EDF logra unha planificación correcta, xa que, por exemplo, en t=30, executa C1 xa que a deadline de C1=50, mentres que a de A2=60, polo que a máis próxima e a de C1.

Por qué fallou RMS? Porque ese algoritmo só funciona cando se compre que:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/m} - 1) \equiv n = 3, usoCPU \leq 0.780$$

Dado que no exemplo anterior o uso de CPU era de 0.975, *RMS tiña moitas probabilidades de fallar*, mentres que pola contra, **EDF sempre funciona para calquera conxunto programable de procesos**.