Algoritmos de Planificación da CPU

Definicións

- **Expropiación (preemption):** Existen algoritmos con e sen expropiación da CPU. Isto significa que cando un proceso está a executarse na CPU é posible coutro proceso entre en execución na CPU e **saque (algoritmo expropiativo)** ó que se está executando ou **non o saque (algoritmo non expropiativo)** durante a súa execución.

- **Planificación apropiativa:** O Sistema Operativo pode quitar (expropiar) a CPU ao proceso/fío.

- **Planificación non apropiativa:** Non se pode retirar o proceso/fío da CPU, o proceso/fío libera a CPU voluntariamente ó bloquearse ou rematar.

- **Intervalos de tempo:** O proceso/fío pode executarse na CPU durante un determinado tempo en ciclos de CPU.

- **Duración:** Número de Ciclos de CPU que necesita o proceso/fío no estado execución na CPU.

- **Tempo de espera:** Tempo cun proceso/fío estivo esperando na cola de listos.

- **Tempo de execución:** Tempo cun proceso/fío estivo no estado execución na CPU.

- **Tempo de retorno:** Tempo cun proceso/fío tardou en completar a súa execución. É igual á suma do tempo de espera máis o tempo de execución.

- **Prioridades:** Os procesos/fíos poden posuír prioridades, as cales poden ser estáticas ou dinámicas.

Algoritmos de Planificación

Non Expropiativos (No Preemptive – no apropiativo)	Si Expropiativos (Preemptive – apropiativo-)
FCFS (First Come, First Serve) / **FIFO** (First Input, First Output) / **LILO** (Last In, Last Out)	**RR** (Round-Robin) (De roda) (Carrusel)
SJF (Shortest Job First)	**SRTF** (Shortest Remaining Time First)
Prioridades (estáticas e dinámicas)	**Prioridades** (estáticas e dinámicas)

¿?? **LCFS** (Last Come, First Serve) / **LIFO** (Last In, First Output) / **FILO** (First In, Last Out)

FCFS CPU

FCFS (First Come First Served)

Este algoritmo tamén chamado **FIFO (First Input First Output)** ou **PEPS (Primeiro en Entrar Primeiro en Sair)** ten en conta a quenda de chegada. Este algoritmo determina que cando entra un proceso éste acapara a CPU, non podendo entrar outro proceso, ata o final da súa execución. Unha vez rematada a súa execución entra o seguinte proceso por orde de chegada na cola, e non abandoará a CPU ata o final da súa execución, e continuarase así ata finalizar a cola de procesos a executar.

Imos ver un exemplo para explicar como traballa o algoritmo **FCFS**:

- Supoñemos a situación seguinte: 3 procesos chegan no mesmo instante, o tempo de chegada 0, e na orde P1, P2 e P3.

    - **Tempo de chegada:** 0
    - **Cola:** P1, P2, P3
    - **Duración Proceso:** P1-->12 ciclos de CPU, P2-->5 ciclos de CPU, P3-->7 ciclos de CPU.

  sendo,

  $te|_{Pi}$ O tempo de espera do Proceso Pi

  $tr|_{Pi}$ O tempo de retorno do Proceso Pi

  Imos calcula-lo tempo de espera(te), o tempo de retorno(tr) e o tempo medio de espera pra este algoritmo, así como o Diagrama de Gantt correspondente,

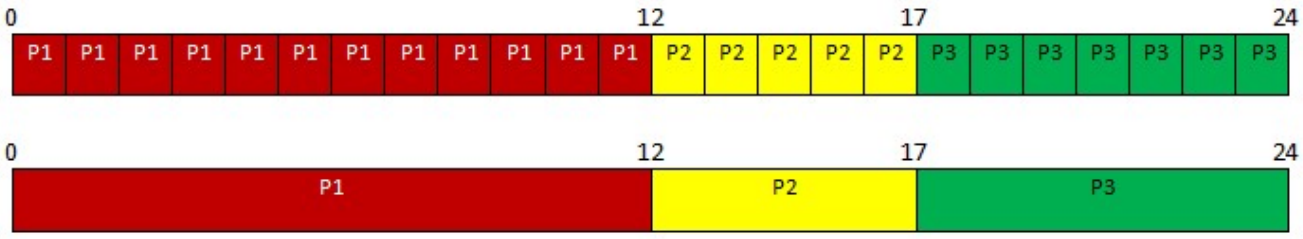
P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1											
P2	E	E	E	E	E	E	E	E	E	E	E	E	P2	P2	P2	P2	P2						
P3	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	P3	P3	P3	P3	P3	P3	P3
Ciclos CPU	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Tempo de chegada	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

↑
P1
P2
P3

$te|_{P1}=0$ $te|_{P2}=12$ $te|_{P3}=17$

$\overline{te} = [(te|_{P1} + te|_{P2} + te|_{P3})/3] = [(0+12+17)/3] = 9.7$

$tr|_{P1}=12$ $tr|_{P2}=17$ $tr|_{P3}=24$



Como podemos ver na imaxe o algoritmo FCFS segue este procedemento :

1. **Ciclo 1 da CPU-Tempo de Chegada 0:** O primeiro proceso en entrar na CPU é o proceso P1 pois na orde de chegada é o primeiro da cola de procesos. O algoritmo FCFS determina que ó entrar un proceso esté ocupará a CPU ata que o mesmo remate, así o proceso P1 acapara a CPU durante os primeiros 12 ciclos da mesma.
2. **Ciclo 13 da CPU-Tempo de Chegada 12:** A continuación entra na CPU o proceso P2 que é o seguinte na cola, co cal acapara a CPU 5 ciclos da mesma ata que o proceso remata.
3. **Ciclo 18 da CPU-Tempo de Chegada 17:** Prosegue o seguinte na cola que é o proceso P3 acaparando a CPU 7 ciclos da mesma.

Imos ver que pasaría se agora,

- Supoñemos a situación seguinte: 3 procesos chegan no mesmo instante, o tempo de chegada 0, e na orde P2, P3 e P1.

- **Tempo de chegada:** 0
- **Cola:** P2, P3, P1
- **Duración Proceso:** P1-->12 ciclos de CPU, P2-->5 ciclos de CPU, P3-->7 ciclos de CPU.

sendo,

$te|_{Pi}$ O tempo de espera do Proceso Pi **$tr|_{Pi}$** O tempo de retorno do Proceso Pi

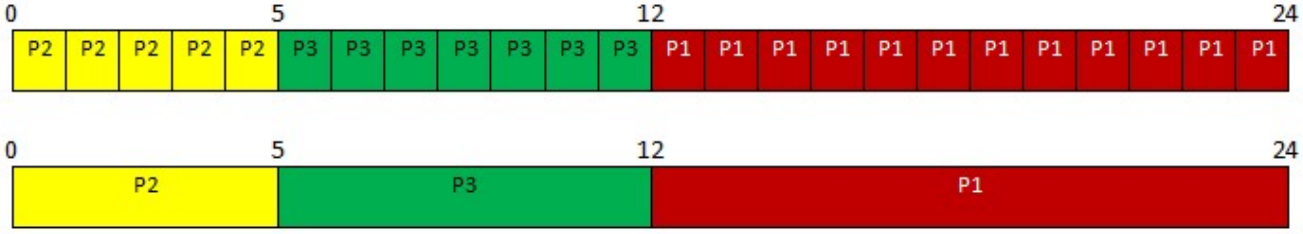
P1	E	E	E	E	E	E	E	E	E	E	E	E	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1	P1
P2	P2	P2	P2	P2	P2																			
P3	E	E	E	E	E	P3	P3	P3	P3	P3	P3	P3												
Ciclos CPU	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Tempo de chegada	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

↑
P2
P3
P1

$te|_{P1}=12$
$te|_{P2}=0$
$te|_{P3}=5$

$\overline{te} = [(te|_{P1} + te|_{P2} + te|_{P3})/3] = [(12+0+5)/3] = 5.7$

$tr|_{P1}=24$
$tr|_{P2}=5$
$tr|_{P3}=12$



Como podemos ver na imaxe o algoritmo FCFS segue este procedemento :

1. **Ciclo 1 da CPU-Tempo de Chegada 0:** O primeiro proceso en entrar na CPU é o proceso P2 pois na orde de chegada é o primeiro da cola de procesos. O algoritmo FCFS determina que ao entrar un proceso esté ocupará a CPU ata que o mesmo remate, así o proceso P2 acapara a CPU durante os primeiros 5 ciclos da mesma.
2. **Ciclo 6 da CPU-Tempo de Chegada 5:** A continuación entra na CPU o proceso P3 que é o seguinte na cola, co cal acapara a CPU 7 ciclos da mesma ata que o proceso remata.
3. **Ciclo 13 da CPU-Tempo de Chegada 12:** Prosegue o seguinte na cola que é o proceso P1 acaparando a CPU 12 ciclos da mesma.

RR

RR (Round-Robin; *De roda; Carrusel*)

Este algoritmo respecta a quenda de chegada, mais soamente deixa un quanto (q) de tempo o uso da CPU para cada proceso.

Imos ver un exemplo pra explicar como traballa o algoritmo **RR**:

- Supoñemos a situación seguinte:

  - **Tempo de chegada:** P1-->0, P2-->5, P3-->4, P4-->2
  - **Cola:** P1, P4, P3, P2
  - **Duración Proceso:** P1-->4 ciclos de CPU, P2-->7 ciclos de CPU, P3-->4 ciclos de CPU, P4-->1 ciclo de CPU.
  - **Quanto:** q=2

  sendo,

  $te|_{Pi}$ O tempo de espera do Proceso Pi

  $tr|_{Pi}$ O tempo de retorno do Proceso Pi

  Imos calcula-lo tempo de espera(te), o tempo medio de espera, o tempo de espera máximo e o tempo de retorno(tr), así como o Diagrama de Gantt correspondente,

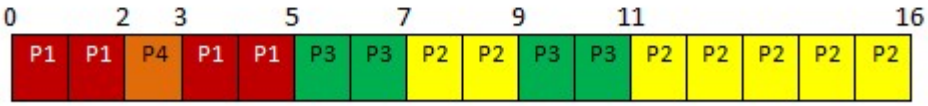
P1	P1	P1	E	P1	P1											
P2						E	E	P2	P2	E	E	P2	P2	P2	P2	P2
P3					E	P3	P3	E	E	P3	P3					
P4			P4													
Ciclos CPU	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Tempo de chegada	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	↑ P1		↑ P4		↑ P3	↑ P2										

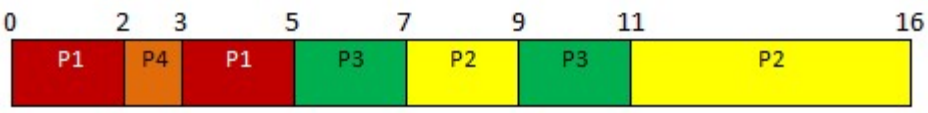
$te|_{P1}=1$ $tr|_{P1}=5$

$te|_{P2}=4$ $\overline{te} = [(te|_{P1} + te|_{P2} + te|_{P3} + te|_{P4})/4]=[(1+4+3+0)/4]=2$ $tr|_{P2}=11$

$te|_{P3}=3$ $te|_{max}=2$ (Procesos P2 e P3) $tr|_{P3}=7$

$te|_{P4}=0$ $tr|_{P4}=1$





Como podemos ver na imaxe o primeiro en entrar na CPU é o proceso P1 pois na orde de chegada é o primeiro da cola de procesos. O algoritmo RR determina que ao entrar un proceso esté ocupará a CPU durante un quanto de tempo **q**, no exemplo **q=2**, deixando liberada a CPU pro seguinte proceso ca acaparé segundo o algoritmo RR, así:

1. **Ciclo 1 da CPU-Tempo de Chegada 0:** Entra o proceso P1 na CPU e acapara 2 ciclos da mesma, quedando para o remate do mesmo outros 2 ciclos -pois a duración deste proceso son 4 ciclos de CPU-.
2. **Ciclo de CPU 3-Tempo de Chegada 2:** A continuación entra o proceso P4 que acapararía 2 ciclos de CPU, xa que **q=2**, mais soamente acapara 1 ciclo de CPU -pois a duración deste proceso é 1 ciclo de CPU-.
3. **Ciclo de CPU 4-Tempo de Chegada 3:** Nesta situación aínda non temos ningún outro proceso en cola agás o P1 co cal entrará o P1 outros dous ciclos de CPU, xa que **q=2**, rematando así o proceso -pois a duración deste proceso son 4 ciclos de CPU-.
4. **Ciclo de CPU 6-Tempo de Chegada 5:** Agora entra na CPU o proceso P3, xa que os procesos P3 e P2 están en cola mais o proceso P3 leva máis tempo na mesma -cando debe entrar un proceso na CPU e varios dos que están en cola teñen as mesmas posibilidades ou probabilidades resolvemos este conflicto mediante o algoritmo FCFS-. Así entra o proceso P3 durante 2 ciclos de CPU -quedandolle outros 2 ciclos para o seu remate-.
5. **Ciclo de CPU 8-Tempo de Chegada 7:** Logo entra o proceso P2 durante 2 ciclos de CPU -quedándolle 5 ciclos de duración-.
6. **Ciclo de CPU 10-Tempo de Chegada 9:** Logo entra de novo P3 2 ciclos de CPU e remata a súa execución.
7. **Ciclo de CPU 12-Tempo de Chegada 11:** Entra P2 2 ciclos de CPU -quedándolle 3 ciclos de duración-.
8. **Ciclo de CPU 14-Tempo de Chegada 13:** Como agora non existe ningún outro proceso en cola segue entrando na CPU o proceso P2 2 ciclos de CPU –quedándolle 1 ciclo de duración-.
9. **Ciclo de CPU 16-Tempo de Chegada 15:** Como tampouco existe agora ningún outro proceso en cola segue entrando na CPU o proceso P2 ata o remate da súa execución.

SJF

SJF (Shortest Job First)

Este algoritmo chamado **SJF (Shortest Job First)** ten en conta a quenda de chegada sendo non expropiativo, así determina co proceso a entrar na CPU de todolos posibles será aquel que teña menos duración de execución na mesma, isto é, entrará o proceso con menor ciclos de CPU a executar.

Imos ver un exemplo para explicar como traballa o algoritmo **SJF**:

- Supoñemos a situación seguinte:

  - **Tempo de chegada:** P1-->0, P2-->5, P3-->4, P4-->2
  - **Cola:** P1, P4, P3, P2
  - **Duración Proceso:** P1-->4 ciclos de CPU, P2-->7 ciclos de CPU, P3-->4 ciclos de CPU. P4-->1 ciclo de CPU.

  sendo,

  $te|_{Pi}$ O tempo de espera do Proceso Pi

  $tr|_{Pi}$ O tempo de retorno do Proceso Pi

  Imos calcular o tempo medio de espera para este algoritmo, así como o Diagrama de Gantt correspondente,

P1	P1	P1	P1	P1												
P2						E	E	E	E	P2	P2	P2	P2	P2	P2	P2
P3					E	P3	P3	P3	P3							
P4			E	E	P4											
Ciclos CPU	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Tempo de chegada	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	↑		↑		↑	↑										
	P1		P4		P3	P2										

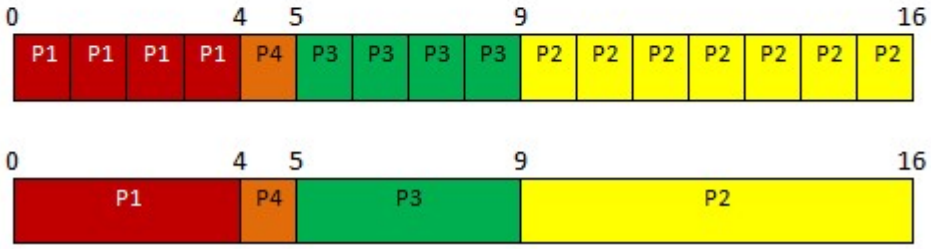
$te|_{P1}=0$

$te|_{P2}=4$

$te|_{P3}=1$

$te|_{P4}=2$

$$\overline{te} = [(te|_{P1} + te|_{P2} + te|_{P3} + te|_{P4})/4] = [(0+4+1+2)/4] = 1.75$$



Como podemos ver na imaxe o primeiro en entrar na CPU é o proceso P1 pois na orde de chegada é o primeiro da cola de procesos. O algoritmo SJF determina que ó entrar un proceso esté ocupará a CPU ata co mesmo remate, xa que o algoritmo é non expropiativo, así:

1. **Ciclo de CPU 1-Tempo de Chegada 0:** Entra o proceso P1 na CPU e acapara os ciclos da mesma ata o remate da súa execución, co cal acapara a CPU 4 ciclos da mesma.
2. **Ciclo de CPU 3-Tempo de Chegada 2:** Entra o proceso P4, mais como o algoritmo **SJF** é non expropiativo segue executándose o proceso P1 ata que remate a súa execución.
3. **Ciclo de CPU 5-Tempo de Chegada 4:** A continuación entra o proceso P4, xa que o algoritmo SJF determina que o proceso a entrar na CPU sexa aquel que ocupe menos ciclos da mesma. Así o proceso P4 somentes ocuparía a CPU 1 ciclo a diferenza do P3 que ocuparía a CPU 4 ciclos da mesma. Entón entra P4 que acapararía 1 ciclo de CPU.
4. **Ciclo de CPU 6-Tempo de Chegada 5:** Nesta situación temos 2 procesos en cola, o proceso P3 e o proceso P2, co cal o algoritmo SJF determina que o seguinte en entrar en cola é o que menos duración ocupe na CPU, entón entrará P3 ata que remate.
5. **Ciclo de CPU 10-Tempo de Chegada 9:** Entra P2 ata que remate.

SRTF

SRTF (Shortest Remaining Time First)

Este algoritmo chamado **SRTF (Shortest Remaining Time First)** ten en conta a quenda de chegada sendo expropiativo, é dicir, ven sendo o algoritmo **SJF pero expropiativo**, así determina que o proceso a entrar na CPU de todolos posibles será aquel que teña menos duración de execución na mesma, isto é, entrará o proceso con menor ciclos de CPU a executar, tendo en conta que se quere entrar un proceso con menor duración na CPU co proceso que se está executando expulsará ao proceso que se está executando e entrará na CPU pra executarse -pois o algoritmo é expropiativo-.

Imos ver un exemplo para explicar como traballa o algoritmo *SRTF'*:

- Supoñemos a situación seguinte:

  - **Tempo de chegada:** P1-->0, P2-->5, P3-->4, P4-->2
  - **Cola:** P1, P4, P3, P2
  - **Duración Proceso:** P1-->4 ciclos de CPU, P2-->7 ciclos de CPU, P3-->4 ciclos de CPU. P4-->1 ciclo de CPU.

  sendo,

  $te|_{Pi}$ O tempo de espera do Proceso Pi

  $tr|_{Pi}$ O tempo de retorno do Proceso Pi

  Imos calcula-lo tempo medio de espera pra este algoritmo, así como o Diagrama de Gantt correspondente,

P1	P1	P1	E	P1	P1											
P2						E	E	E	E	P2	P2	P2	P2	P2	P2	P2
P3					E	P3	P3	P3	P3							
P4			P4													
Ciclos CPU	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Tempo de chegada	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	↑ P1		↑ P4		↑ P3	↑ P2										

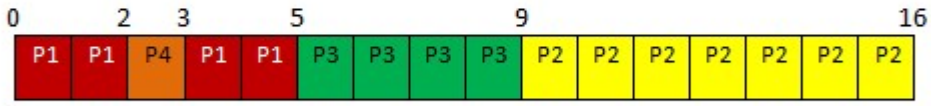
$te|_{P1}=1$

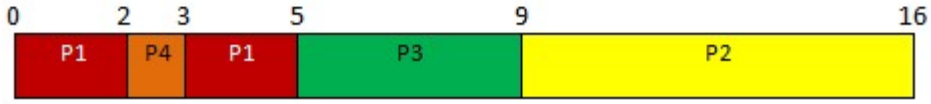
$te|_{P2}=4$

$te|_{P3}=1$

$te|_{P4}=0$

$$\overline{te} = [(te|_{P1} + te|_{P2} + te|_{P3} + te|_{P4})/4] = [(1+4+1+0)/4] = 1.5$$





Como podemos ver na imaxe o primeiro en entrar na CPU é o proceso P1 pois na orde de chegada é o primeiro da cola de procesos. O algoritmo **SRTF** determina que ó entrar un proceso esté ocupará a CPU ata coutro proceso teña máis preferencia de uso, neste caso, ata que queira entrar na CPU outro proceso con menor número de ciclos de CPU pra a súa execución, xa co algoritmo é expropiativo, así:

1. **Ciclo de CPU 1-Tempo de Chegada 0:** Entra o proceso P1 na CPU e acapara os ciclos da mesma ata a entrada do proceso P4 -paso seguinte no cal estudiarase o que acontecerá-, co cal -de momento- acapara a CPU 2 ciclos da mesma.
2. **Ciclo de CPU 3-Tempo de Chegada 2:** A continuación entra o proceso P4, xa co algoritmo **SRTF** determina co proceso a entrar na CPU sexa aquel que ocupe menos ciclos da mesma. Así o proceso P4 soamente ocuparía a CPU 1 ciclo a diferenza do P1 que ocuparía, aínda, a CPU 2 ciclos máis da mesma. Entón entra P4 que acapararía 1 ciclo de CPU expulsando o proceso P1 da súa execución.
3. **Ciclo de CPU 4-Tempo de Chegada 3:** Nesta situación temos soamente 1 proceso en cola, o proceso P1, co cal execútase P1 durante 1 ciclo de execución -o ciclo de CPU 4-, xa que no ciclo de CPU 5 quere entrar P3, co cal debemos mirar segundo o algoritmo **SRTF** quen ocupa a CPU na súa execución.
4. **Ciclo de CPU 5-Tempo de Chegada 4:** Agora vemos que o proceso P3 ocuparía 4 ciclos de CPU e o proceso P1 ocuparía 1 ciclo de CPU, logo segue executándose o proceso P1 durante un ciclo da mesma, finalizando así a súa execución.
5. **Ciclo de CPU 6-Tempo de Chegada 5:** O algoritmo **SRTF** determina co seguinte en entrar na CPU é o que menos duración ocupe na mesma, entón entrará P3 ata que remate.
6. **Ciclo de CPU 10-Tempo de Chegada 9:** Entra P2 ata que remate.

Prioridades Non Expropiativo

Prioridades Non Expropiativo (Prioridades No Preemptive)

Este algoritmo chamado **Prioridades Non Expropiativo** ten en conta a quenda de chegada sendo non expropiativo, así determina co proceso a entrar na CPU de todolos posibles será aquel que teña maior prioridade, isto é, entrará o proceso con maior prioridade independentemente dos ciclos de CPU a executar.

Imos ver un exemplo para explicar como traballa o algoritmo **Prioridades Non Expropiativo**:

- Supoñemos a situación seguinte:

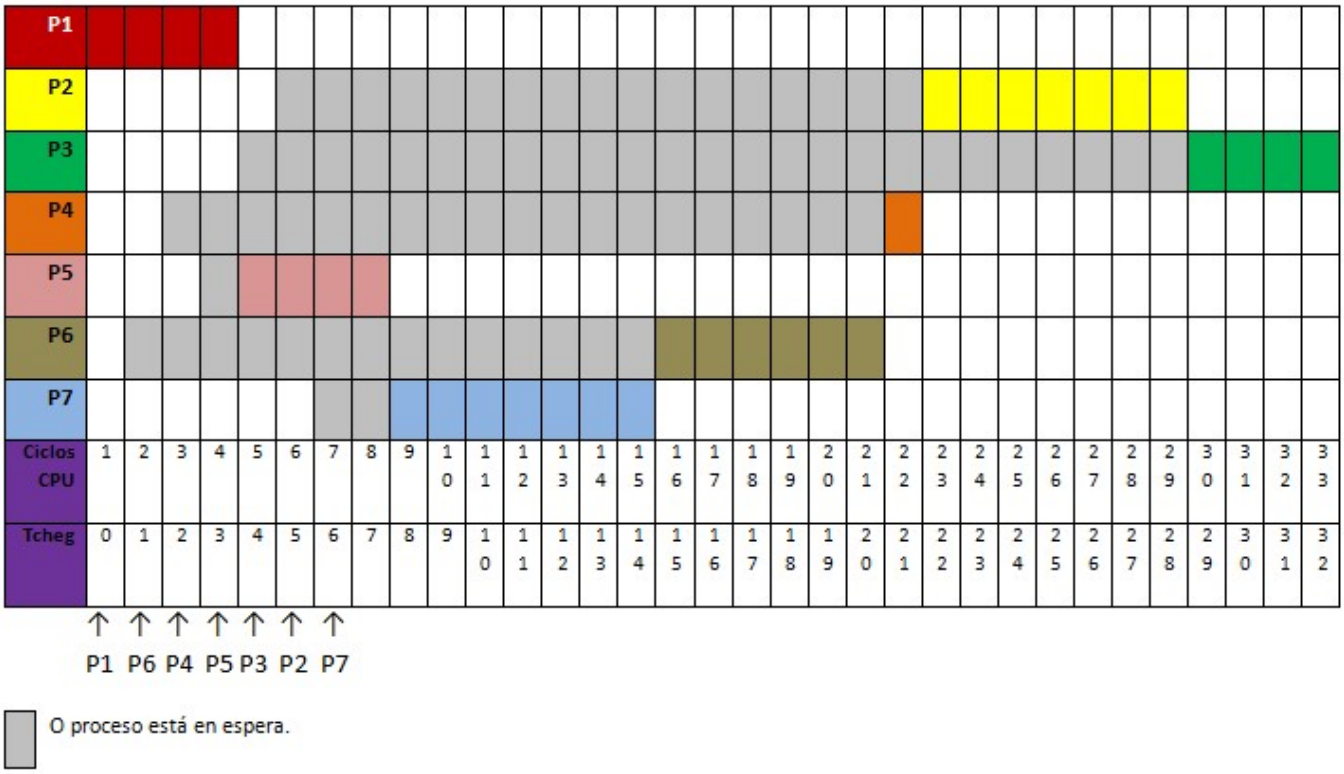
  - **Tempo de chegada:** P1-->0, P2-->5, P3-->4, P4-->2, P5-->3, P6-->1, P7-->6
  - **Cola:** P1, P2, P3, P4, P5, P6, P7
  - **Duración Proceso:** P1-->4 ciclos de CPU, P2-->7 ciclos de CPU, P3-->4 ciclos de CPU. P4-->1 ciclo de CPU, P5-->4 ciclos de CPU, P6-->6 ciclos de CPU, P7-->7 ciclos de CPU.
  - **Prioridade (A maior valor o proceso terá maior prioridade):** P1-->15, P2-->10, P3-->5, P4-->20, P5-->30, P6-->25, P7-->40

  sendo,

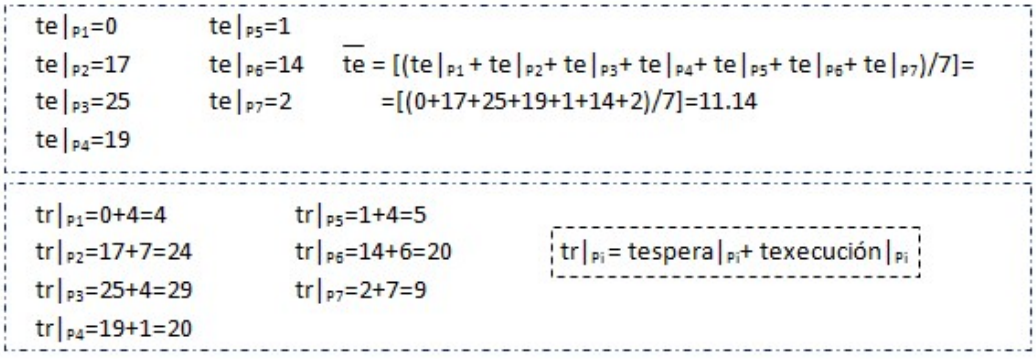
  $\mathbf{te|_{Pi}}$ O tempo de espera do Proceso Pi

  $\mathbf{tr|_{Pi}}$ O tempo de retorno do Proceso Pi

  Imos calcula-lo tempo de retorno e o tempo medio de espera para este algoritmo, así como o Diagrama de Gantt correspondente,



A continuación, a seguinte imaxe amosa os tempos de retorno e o tempo medio de espera, así como tamén o Diagrama de Gantt.



Como podemos ver na imaxe o primeiro en entrar na CPU é o proceso P1 pois na orde de chegada é o primeiro da cola de procesos. O algoritmo **Prioridades Non Expropiativo** determina co entrar un proceso esté ocupará a CPU ata que remate a súa execución, pois o algoritmo é non expropiativo, así que aínda que outro proceso que posúa máis prioridade queira entrar na CPU, non expulsará ó proceso en execución mentres este estea executándose, xa co algoritmo é non expropiativo, así:

1. **Ciclo de CPU 1-Tempo de Chegada 0:** Entra o proceso P1 na CPU e acapara os ciclos da mesma ata a finalización da súa execución, xa que aínda que durante a súa execución chegan procesos con maior prioridade que P1 -que posúe prioridade 15-, como por exemplo P4 -que posúe prioridade 20-, éste non pode expulsar a P1 do estado de execución pois o algoritmo é non expropiativo.
2. **Ciclo de CPU 5-Tempo de Chegada 4:** Agora temos en cola os procesos P6, P4, P5 e P3, co cal vemos as prioridades destes procesos en cola, posuíndo maior prioridade o proceso P5 -con prioridade 30-, co cal entra o proceso P5, xa que o algoritmo **Prioridades Non Expropiativo** determina que o proceso a entrar na CPU sexa aquel que posúa maior prioridade. Así o proceso P5 posúe prioridade 30, prioridade maior ca do proceso P6 -con prioridade 25-, maior ca do proceso P4 -con prioridade 20-, e maior ca do proceso P3 -con prioridade 5-. Entón entra P5 ata o remate da súa execución xa que o algoritmo é non expropiativo.
3. **Ciclo de CPU 9-Tempo de Chegada 8:** Nesta situación temos 5 procesos en cola, os procesos P6, P4, P3, P2 e P7, co cal execútase P7 pois é o proceso con maior prioridade -posúe prioridade 40- ata o remate da súa execución pois o algoritmo é non expropiativo.
4. **Ciclo de CPU 16-Tempo de Chegada 15:** Agora quedan en cola os procesos P6, P4, P3 e P2 executándose o proceso con maior prioridade, neste caso é o proceso P6 -con prioridade 25- ata finalizar a súa execución, pois o algoritmo é non expropiativo.
5. **Ciclo de CPU 22-Tempo de Chegada 21:** Na cola quedan os procesos P4, P3 e P2, sendo o proceso con maior prioridade o proceso P4 -con prioridade 20-. Este proceso executarase ata rematar a súa execución pois o algoritmo é non expropiativo.
6. **Ciclo de CPU 23-Tempo de Chegada 22:** A continuación temos 2 procesos na cola: P2 e P3, entrando a executarse o proceso P2 que posúe maior prioridade -P2 posúe prioridade 10- ata que remate a súa execución pois o algoritmo é non expropiativo.
7. **Ciclo de CPU 30-Tempo de Chegada 29:** Soamente queda na cola o proceso P3 -con prioridade 5-, co cal execútase ata finalizar a súa execución pois o algoritmo é non expropiativo.

Prioridades Si Expropiativo

Prioridades Si Expropiativo (Prioridades Preemptive)

Este algoritmo chamado **Prioridades Si Expropiativo** ten en conta a quenda de chegada sendo o algoritmo expropiativo, é dicir, ven sendo o algoritmo **Prioridades Non Expropiativo pero expropiativo**, así determina co proceso a entrar na CPU de todolos posibles será aquel que teña maior prioridade de execución, isto é, entrará o proceso con maior prioridade de execución na CPU independentemente dos ciclos de CPU a executar, tendo en conta que se quere entrar un proceso con maior prioridade na CPU co proceso que se está executando expulsará ó proceso que se está executando e entrará na CPU pra executarse -pois o algoritmo é expropiativo-.

Imos ver un exemplo pra explicar como traballa o algoritmo *Prioridades Si Expropiativo*:

- Supoñemos a situación seguinte:

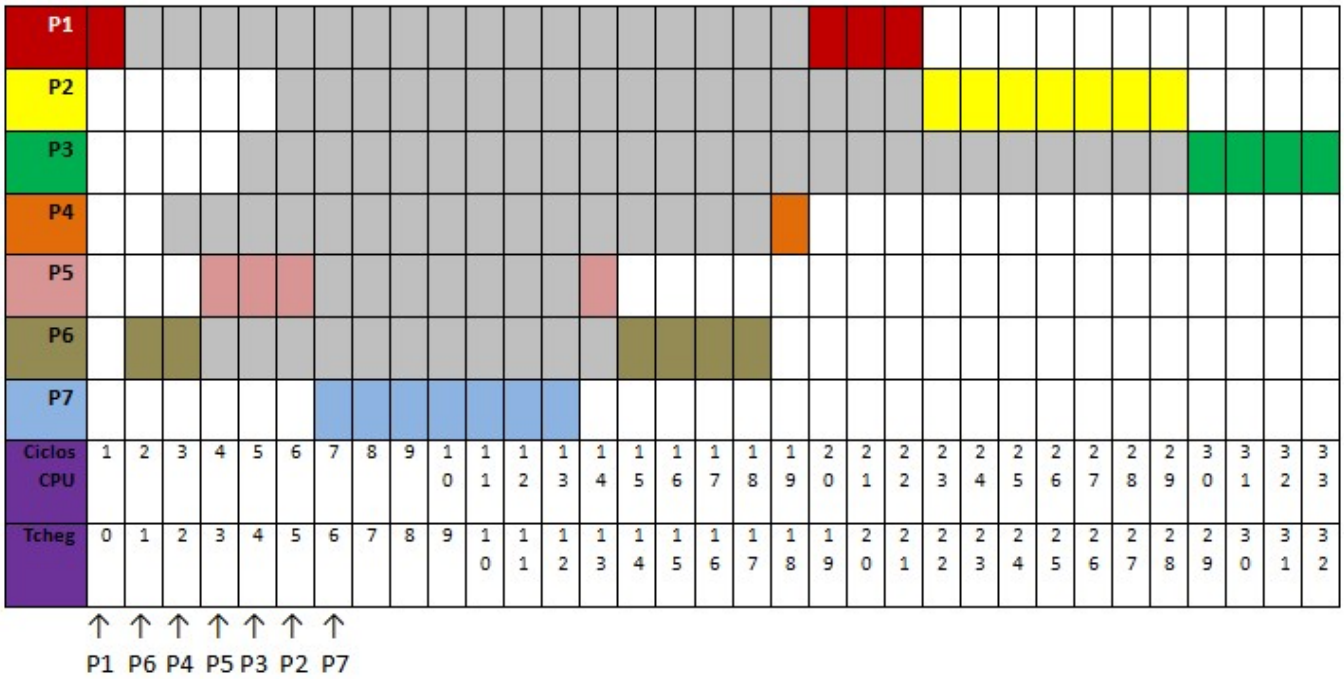
  - **Tempo de chegada:** P1-->0, P2-->5, P3-->4, P4-->2, P5-->3, P6-->1, P7-->6
  - **Cola:** P1, P2, P3, P4, P5, P6, P7
  - **Duración Proceso:** P1-->4 ciclos de CPU, P2-->7 ciclos de CPU, P3-->4 ciclos de CPU. P4-->1 ciclo de CPU, P5-->4 ciclos de CPU, P6-->6 ciclos de CPU, P7-->7 ciclos de CPU.
  - **Prioridade (A maior valor o proceso terá maior prioridade):** P1-->15, P2-->10, P3-->5, P4-->20, P5-->30, P6-->25, P7-->40

  sendo,

  $\mathbf{te|_{Pi}}$ O tempo de espera do Proceso Pi

  $\mathbf{tr|_{Pi}}$ O tempo de retorno do Proceso Pi

  Imos calcular o tempo de retorno e o tempo medio de espera para este algoritmo, así como o Diagrama de Gantt correspondente,



O proceso está en espera.

A continuación, a seguinte imaxe amosa os tempos de retorno e o tempo medio de espera, así como tamén o Diagrama de Gantt.

$te|_{P1}=18$ $te|_{P5}=7$

$te|_{P2}=17$ $te|_{P6}=11$ $\overline{te} = [(te|_{P1} + te|_{P2} + te|_{P3} + te|_{P4} + te|_{P5} + te|_{P6} + te|_{P7})/7] =$

$te|_{P3}=25$ $te|_{P7}=0$ $=[(18+17+25+16+7+11+0)/7]=13.43$

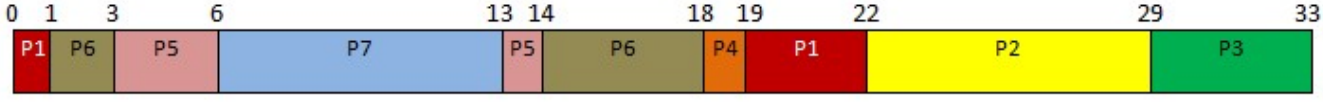
$te|_{P4}=16$

$tr|_{P1}=18+4=22$ $tr|_{P5}=7+4=11$

$tr|_{P2}=17+7=24$ $tr|_{P6}=11+6=17$ $tr|_{Pi} = tespera|_{Pi} + texecución|_{Pi}$

$tr|_{P3}=25+4=29$ $tr|_{P7}=0+7=7$

$tr|_{P4}=16+1=17$



Como podemos ver na imaxe o primeiro en entrar na CPU é o proceso P1 pois na orde de chegada é o primeiro da cola de procesos. O algoritmo **Prioridades Si Expropiativo** determina co entrar un proceso esté ocupará a CPU ata que chegue outro proceso con maior prioridade de uso, pois o algoritmo é expropiativo, así :

1. **Ciclo de CPU 1-Tempo de Chegada 0:** Entra o proceso P1 na CPU e acapara os ciclos da mesma ata a entrada do proceso P6 -paso seguinte no cal estudiarase o que acontecerá-, co cal -de momento- acapara a CPU 1 ciclo da mesma.
2. **Ciclo de CPU 2-Tempo de Chegada 1:** A continuación entra o proceso P6, xa que o algoritmo **Prioridades Si Expropiativo** determina que o proceso a entrar na CPU sexa aquel que posúa maior prioridade de uso da CPU. Así o proceso P6 posúe prioridade 25 mentres que o proceso P1 posúe prioridade 15, entón entra P6 que acapararía 1 ciclo de CPU expulsando o proceso P1 da súa execución.
3. **Ciclo de CPU 3-Tempo de Chegada 2:** Nesta situación temos en cola os procesos P1, P6 e P4 co cal vemos que proceso ten maior prioridade, que ven sendo o proceso P6 -con prioridade 25-. Así segue executándose P6 1 ciclo de CPU.
4. **Ciclo de CPU 4-Tempo de Chegada 3:** Entra o proceso P5 na CPU, xa que agora en cola temos os procesos P1, P6, P4 e P5, dos cales o que posúe maior prioridade é o proceso P5 -con prioridade 30-. Entón execútase o proceso P5 1 ciclo de CPU expulsando o proceso P6 da súa execución.
5. **Ciclo de CPU 5-Tempo de Chegada 4:** Agora temos en cola os procesos P1, P6, P4, P5 e P3, co cal vemos as prioridades destes procesos en cola, posuíndo maior prioridade o proceso P5 -con prioridade 30-, co cal entra o proceso P5, xa que o algoritmo **Prioridades Non Expropiativo** determina que o proceso a entrar na CPU sexa aquel que posúa maior prioridade. Así o proceso P5 posúe prioridade 30, prioridade maior ca do proceso P1 -con prioridade 15-, maior ca do proceso P6 -con prioridade 25-, maior ca do proceso P4 -con prioridade 20-, e maior ca do proceso P3 -con prioridade 5-. Entón segue executándose P5 durante 1 ciclo de CPU.
6. **Ciclo de CPU 6-Tempo de Chegada 5:** Nesta situación seguimos tendo os mesmos procesos en cola que no paso anterior máis un novo proceso: o proceso P2 con prioridade 10, co cal segue executándose P5 durante 1 ciclo de CPU e remata a súa execución.
7. **Ciclo de CPU 7-Tempo de Chegada 6:** Agora quedan en cola os procesos P1, P6, P4, P3 e P2, máis un novo proceso que quere entrar en execución: o proceso P7 con prioridade 40. De todos estes procesos o que posúe maior prioridade é o proceso P7, co cal execútase o proceso P7 ata finalizar a súa execución pois non chega ningún outro proceso á cola da CPU.
8. **Ciclo de CPU 14-Tempo de Chegada 13:** Na cola quedan os procesos P1, P6, P4, P5, P3 e P2, sendo o proceso con maior prioridade o proceso P5 -con prioridade 30-. Este proceso executarase ata rematar a súa execución pois non chega ningún outro proceso á cola da CPU.
9. **Ciclo de CPU 15-Tempo de Chegada 14:** A continuación temos os en cola os procesos: P1, P6, P4, P3 e P2, entrando a executarse o proceso P6 que posúe maior prioridade -P6 posúe prioridade 25- ata que remate a súa execución pois non chega ningún outro proceso á cola da CPU.
10. **Ciclo de CPU 19-Tempo de Chegada 18:** Quedan en cola os procesos P1, P4, P3 e P2, co cal execútase o proceso P4 -con prioridade 20-, ata que remate a súa execución pois non chega ningún outro proceso á cola da CPU.
11. **Ciclo de CPU 20-Tempo de Chegada 19** Procesos en cola: P1, P3 e P2. Entón execútase o proceso P1 -con prioridade 15-, ata que remate a súa execución pois non chega ningún outro proceso á cola da CPU.
12. **Ciclo de CPU 23-Tempo de Chegada 22** Procesos en cola: P3 e P2. Entón execútase o proceso P2 -con prioridade 10-, ata que remate a súa execución pois non chega ningún outro proceso á cola da CPU.
13. **Ciclo de CPU 30-Tempo de Chegada 29** Soamente queda en cola o proceso P3. Entón execútase o proceso P3 -con prioridade 5-, ata que remate a súa execución pois non chega ningún outro proceso á cola da CPU.