Algoritmos de planificación de CPU

DigitalHouse >
Coding School

Certified Developer
The Ultimate Tech Degre

Índice

¿Por qué la necesidad de usarlos?

“Estos métodos nacieron por la necesidad de poder ordenar los procesos para ganar eficiencia, son los encargados de ordenar y dirigir los procesos para asegurar que ninguno de ellos monopolice el uso de la CPU.”

1 | FIFO

Primero en llegar, primero en salir (FIFO)

Este algoritmo es muy sencillo y simple, pero también el que menos rendimiento ofrece. Básicamente en este algoritmo el primer proceso que llega se ejecuta y una vez terminado, se ejecuta el siguiente.

Procesos	Llegada	Tiempo uso CPU
P1	0	11
P2	2	3
P3	3	3
P4	4	3

Primero en llegar, primero en salir (FIFO)

Este algoritmo es muy sencillo y simple, pero también el que menos rendimiento ofrece. Básicamente en este algoritmo el primer proceso que llega se ejecuta y una vez terminado, se ejecuta el siguiente.

Procesos	Llegada	Tiempo uso CPU
P1	0	11
P2	2	3
P3	3	3
P4	4	3

Primero en llegar, primero en salir (FIFO)

Este algoritmo es muy sencillo y simple, pero también el que menos rendimiento ofrece. Básicamente en este algoritmo el primer proceso que llega se ejecuta y una vez terminado, se ejecuta el siguiente.

Procesos	Llegada	Tiempo uso CPU
P1	0	11
P2	2	3
P3	3	3
P4	4	3

Primero en llegar, primero en salir (FIFO)

Este algoritmo es muy sencillo y simple, pero también el que menos rendimiento ofrece. Básicamente en este algoritmo el primer proceso que llega se ejecuta y una vez terminado, se ejecuta el siguiente.

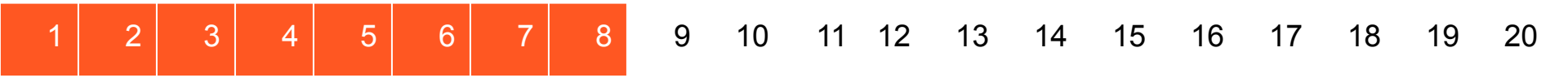
Procesos	Llegada	Tiempo uso CPU
P1	0	11
P2	2	3
P3	3	3
P4	4	3

2 | SJF

El siguiente proceso, el más corto (SJF)

Se prioriza los procesos más cortos primero, independientemente de su llegada; y en caso de que los procesos sean iguales, utilizara el método FIFO.

Procesos	Llegada	Tiempo uso CPU
P1	0	8
P2	2	5
P3	4	2
P4	5	5



El siguiente proceso, el más corto (SJF)

Se prioriza los procesos más cortos primero, independientemente de su llegada; y en caso de que los procesos sean iguales, utilizara el método FIFO.

Procesos	Llegada	Tiempo uso CPU
P1	0	8
P2	2	5
P3	4	2
P4	5	5

El siguiente proceso, el más corto (SJF)

Se prioriza los procesos más cortos primero independientemente de su llegada y en caso de que los procesos sean iguales utilizara el método FIFO.

Procesos	Llegada	Tiempo uso CPU
P1	0	8
P2	2	5
P3	4	2
P4	5	5

El siguiente proceso, el más corto (SJF)

Se prioriza los procesos más cortos primero, independientemente de su llegada; y en caso de que los procesos sean iguales, utilizara el método FIFO.

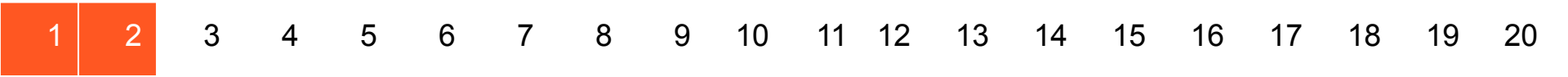
Procesos	Llegada	Tiempo uso CPU
P1	0	8
P2	2	5
P3	4	2
P4	5	5

3 | SRTF

Próximo proceso, el de tiempo restante más corto (SRTF)

Añadiendo la expulsión de procesos al algoritmo SJF obtenemos SRTF, capaz de expulsar un proceso largo en ejecución para ejecutar otros más cortos.

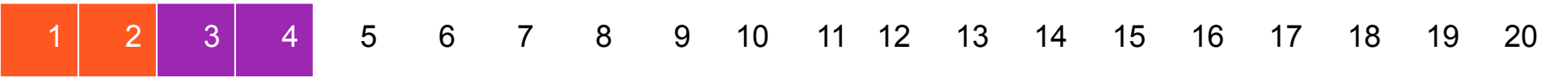
Procesos	Llegada	Tiempo uso CPU
P1	0	8
P2	2	5
P3	4	2
P4	5	5



Próximo proceso, el de tiempo restante más corto (SRTF)

Añadiendo la expulsión de procesos al algoritmo SJF obtenemos SRTF, capaz de expulsar un proceso largo en ejecución para ejecutar otros más cortos.

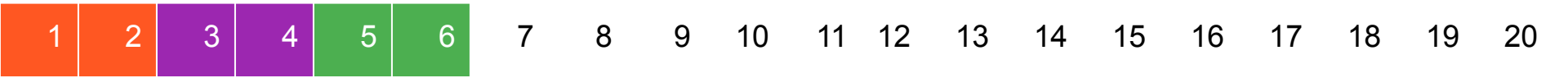
Procesos	Llegada	Tiempo uso CPU
P1	0	8
P2	2	5
P3	4	2
P4	5	5



Próximo proceso, el de tiempo restante más corto (SRTF)

Añadiendo la expulsión de procesos al algoritmo SJF obtenemos SRTF, capaz de expulsar un proceso largo en ejecución para ejecutar otros más cortos.

Procesos	Llegada	Tiempo uso CPU
P1	0	8
P2	2	5
P3	4	2
P4	5	5



Próximo proceso, el de tiempo restante más corto (SRTF)

Añadiendo la expulsión de procesos al algoritmo SJF obtenemos SRTF, capaz de expulsar un proceso largo en ejecución para ejecutar otros más cortos.

Procesos	Llegada	Tiempo uso CPU
P1	0	8
P2	2	5
P3	4	2
P4	5	5

Próximo proceso, el de tiempo restante más corto (SRTF)

Añadiendo la expulsión de procesos al algoritmo SJF obtenemos SRTF, capaz de expulsar un proceso largo en ejecución para ejecutar otros más cortos.

Procesos	Llegada	Tiempo uso CPU
P1	0	8
P2	2	5
P3	4	2
P4	5	5

Próximo proceso, el de tiempo restante más corto (SRTF)

Añadiendo la expulsión de procesos al algoritmo SJF obtenemos SRTF, capaz de expulsar un proceso largo en ejecución para ejecutar otros más cortos.

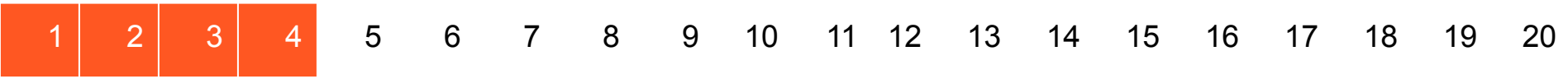
Procesos	Llegada	Tiempo uso CPU
P1	0	8
P2	2	5
P3	4	2
P4	5	5

4 | Round Robin

Round-Robin (RR)

Este algoritmo es circular, volviendo siempre al primer proceso una vez terminado con el último, para controlar este método a cada proceso se le asigna un intervalo de tiempo llamado quantum. A continuación, un ejemplo con quantum= 4.

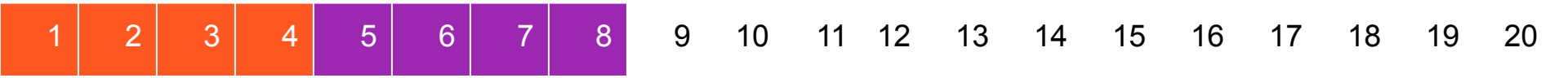
Procesos	Llegada	Tiempo uso CPU
P1	0	9
P2	1	5
P3	2	3
P4	3	3



Round-Robin (RR)

Este algoritmo es circular, volviendo siempre al primer proceso una vez terminado con el último, para controlar este método a cada proceso se le asigna un intervalo de tiempo llamado quantum. A continuación, un ejemplo con quantum= 4.

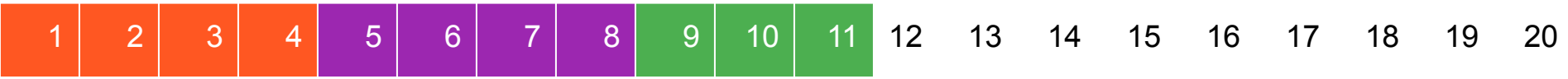
Procesos	Llegada	Tiempo uso CPU
P1	0	9
P2	1	5
P3	2	3
P4	3	3



Round-Robin (RR)

Este algoritmo es circular, volviendo siempre al primer proceso una vez terminado con el último, para controlar este método a cada proceso se le asigna un intervalo de tiempo llamado quantum. A continuación, un ejemplo con quantum= 4.

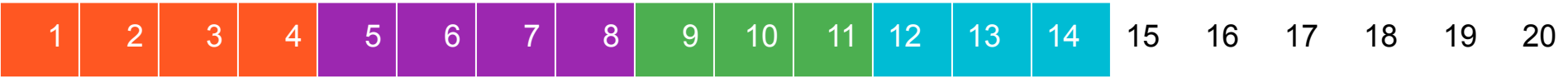
Procesos	Llegada	Tiempo uso CPU
P1	0	9
P2	1	5
P3	2	3
P4	3	3



Round-Robin (RR)

Este algoritmo es circular, volviendo siempre al primer proceso una vez terminado con el último, para controlar este método a cada proceso se le asigna un intervalo de tiempo llamado quantum. A continuación, un ejemplo con quantum= 4.

Procesos	Llegada	Tiempo uso CPU
P1	0	9
P2	1	5
P3	2	3
P4	3	3



Round-Robin (RR)

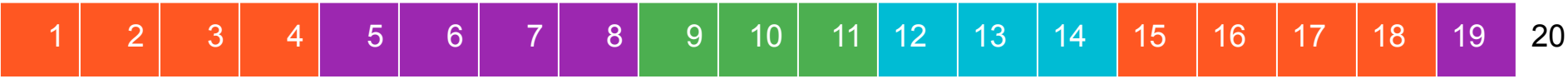
Este algoritmo es circular, volviendo siempre al primer proceso una vez terminado con el último, para controlar este método a cada proceso se le asigna un intervalo de tiempo llamado quantum. A continuación, un ejemplo con quantum= 4.

Procesos	Llegada	Tiempo uso CPU
P1	0	9
P2	1	5
P3	2	3
P4	3	3

Round-Robin (RR)

Este algoritmo es circular, volviendo siempre al primer proceso una vez terminado con el último, para controlar este método a cada proceso se le asigna un intervalo de tiempo llamado quantum. A continuación, un ejemplo con quantum= 4.

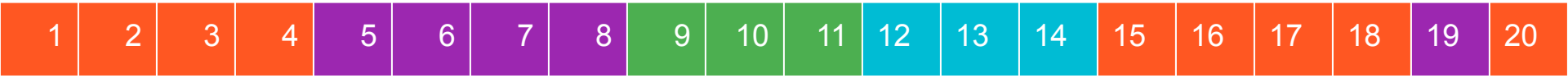
Procesos	Llegada	Tiempo uso CPU
P1	0	9
P2	1	5
P3	2	3
P4	3	3



Round-Robin (RR)

Este algoritmo es circular, volviendo siempre al primer proceso una vez terminado con el último, para controlar este método a cada proceso se le asigna un intervalo de tiempo llamado quantum. A continuación, un ejemplo con quantum= 4.

Procesos	Llegada	Tiempo uso CPU
P1	0	9
P2	1	5
P3	2	3
P4	3	3




DigitalHouse>
Coding School