Informe de la Práctica de Control

Prioridad Menor – Particiones Fijas y No Iguales – Mejor ajuste

Álvaro Manjón Vara

xmv1001@alu.ubu.es

Sistemas Operativos | Grado en Ingeniería Informática | Curso 2019 - 2020

Índice

1. Presentación del algoritmo

El algoritmo en el que se basa esta práctica está compuesto por dos partes, ya que es un algoritmo de **prioridad menor,** y además consta de **particiones fijas y no iguales, y mejor ajuste.** Un algoritmo de prioridad menor indica que, a la hora de entrar a ejecutarse los procesos, lo van a hacer en función de su prioridad, entrando antes aquellos procesos que posean los valores más bajos de prioridad. La memoria va a estar dividida en particiones que no varían, y cada partición va a tener un tamaño distinto al resto. El mejor ajuste indica que, a la hora de entrar un proceso a memoria, el proceso no se ubicará en la primera partición libre que encuentre, si no que analizará todas las particiones de la memoria y se alojará en aquella que pueda ocupar dejando el mínimo espacio libre posible. Al ser FIFO, el primer proceso que entre será el primero que salga.

2. Funciones del script

- ***crea_particiones:*** Función encargada de introducir por teclado cada una de las características de la memoria y particiones.
- ***crea_prioridad:*** Función encargada de pedir por teclado los valores máximo y minimo correspondientes a la prioridad.
- ***calcularTipoPrioridad:*** Función encargada de calcular qué tipo de prioridad se ha asignado, para transformar los valores dados en un rango de 0 a x.
- ***calculoSegunTipoPrioridad:*** Función encargada de aplicar las operaciones correspondientes en una prioridad dada, para transformarla en un valor compatible con el rango de 0 a x.
- ***ImprimeLineaProcesos:*** Función encargada de representar gráficamente en la banda de memoria el nombre del proceso asociado a la memoria ocupada en cada partición.
- ***ImprimeLineaProcesosBW:*** Función (en el documento del informe sin color) encargada de representar gráficamente en la banda de memoria el nombre del proceso asociado a la memoria ocupada en cada partición.
- ***ImprimeMemoria:*** Función encargada de representar gráficamente la memoria en la banda de memoria.
- ***ImprimeMemoriaBW:*** Función (en el documento del informe sin color) encargada de representar gráficamente la memoria en la banda de memoria.
- ***ImprimeLineaFinal:*** Función encargada de representar gráficamente en la banda de memoria el tamaño inicial y final asociado a cada partición, así como la posición de la unidad de memoria libre, después de situarse un proceso en dicha partición.
- ***ImprimeLineaFinalBW:*** Función encargada de representar gráficamente en la banda de memoria el tamaño inicial y final asociado a cada partición, así como la posición de la unidad de memoria libre, después de situarse un proceso en dicha partición.
- ***ImprimeProcesos:*** Función encargada de asociar procesos a la gráfica del tiempo.
- ***ImprimeProcesosBW:*** Función (en el documento del informe sin color) encargada de asociar procesos a la gráfica del tiempo.
- ***ImprimeGrafica:*** Función encargada de representar la gráfica que indica la ejecución de cada uno de los procesos respecto al tiempo.
- ***ImprimeGraficaBW:*** Función (en el documento del informe sin color) encargada de representar la gráfica que indica la ejecución de cada uno de los procesos respecto al tiempo.

- ***ImprimeLineaTemporal:*** Función encargada de asociar el tiempo a la gráfica del tiempo.
- ***ImprimeLineaTemporalBW:*** Función (en el documento del informe sin color) encargada de asociar el tiempo a la gráfica del tiempo.
- ***State:*** Función que imprime en pantalla y en el informe una tabla con cada una de las caracteristicas de los procesos introducidos.
- ***mapa:*** Función encargada de la implementación gráfica de las particiones.
- ***mapaBW:*** Función (en el documento del informe sin color) encargada de la implementación gráfica de las particiones.
- ***mapagrafica:*** Función encargada de la implementación gráfica de los intervalos de tiempo de ejecución de cada uno de los procesos.
- ***mapagraficaBW:*** Función (en el documento del informe sin color) encargada de la implementación gráfica de los intervalos de tiempo de ejecución de cada uno de los procesos.
- ***mapaprocesosdos:*** Función encargada de la implementación gráfica de la linea de procesos en la gráfica temporal.
- ***mapaprocesosdosBW:*** Función (en el documento del informe sin color) encargada de la implementación gráfica de la linea de procesos en la gráfica temporal.
- ***mapaprocesos:*** Función encargada de la implementación gráfica de la linea de procesos en la gráfica de las particiones.
- ***mapaprocesosBW:*** Función (en el documento del informe sin color) encargada de la implementación gráfica de la linea de procesos en la gráfica de las particiones.
- ***mapafinal:*** Función encargada de la implementación gráfica de la linea del tamaño que ocupa cada partición.
- ***mapamemorialibre:*** Función encargada de señalar dónde comienza la primera unidad de memoria libre en caso de estar ocupada por un proceso.
- ***mapatemporal:*** Función encargada de la implementación gráfica de la linea del tiempo en la gráfica temporal.
- ***mapatiempos:*** Función encargada de la correcta implementación de las unidades en la banda de tiempo.
- ***ocupamapatiempos:*** Función encargada de añadir los números que deben de aparecer en las unidades de la banda de tiempo.
- ***OcupaMemoria:*** Función encargada de ocupar con los procesos las particiones gráficamente sustituyendo el caracter que simboliza el tamaño de la partición (█) por el color asociado al proceso.
- ***OcupaProceso:*** Función encargada de situar encima de cada partición el nombre del proceso que está alojado en ella.
- ***OcupaProcesoBW:*** Función (en el documento del informe sin color) encargada de situar encima de cada partición el nombre del proceso que está alojado en ella.
- ***OcupaProcesoDos:*** Función encargada de situar encima de cada intervalo temporal el nombre del proceso correspondiente.
- ***OcupaProcesoDosBW:*** Función (en el documento del informe sin color) encargada de situar encima de cada intervalo temporal el nombre del proceso correspondiente.
- ***OcupaTiempo:*** Función encargada de rellenar cada intervalo temporal.
- ***OcupaLineaTemporal:*** Función encargada de ocupar debajo de cada intervalo temporal el tiempo en el que acaba de ejecutarse el proceso correspondiente.

- ***DesocupaMemoria:*** Función que desocupa la memoria sustituyendo gráficamente el nombre del proceso por el caracter que simboliza la partición (▉).
- ***DesocupaProceso:*** Función que desaloja el nombre del proceso que se encuentra en una partición determinada.
- ***DesocupaProcesoBW:*** Función (en el documento del informe sin color) que desaloja el nombre del proceso que se encuentra en una partición determinada.
- ***CalculaDiferenciaMinima:*** Función que calcula la diferencia mínima entre la memoria que ocupa un proceso y todas las particiones, para ver en cuál se ajusta mejor.
- ***AsignaMemoriaMejorAjuste:*** Función que introduce los procesos en la partición que mejor se ajuste, para ello, mediante un bucle, realizamos las comprobaciones pertinentes.
- ***estadosiniciales:*** Función encargada de inicializar cada uno de los estados asociados a cada proceso.
- ***GestionDeMemoria:*** Función encargada de todo lo relacionado con la ejecución del algoritmo.
- ***Ordenar:*** Función encargada de ordenar los procesos.
- ***es_entero:*** Nos permite saber si el parámetro pasado es entero positivo.
- ***ComprobarPalabras:*** Función que comprueba que un nombre no tenga más de dos palabras separadas por espacios.
- ***ComprobarPrioridad:*** Función que comprueba que la prioridad introducida a cada proceso no esté fuera del rango de prioridades.

3. Modificaciones y correcciones realizadas en el script

- Se ha indentado todo el código bien, para una lectura más sencilla.
- En el programa se han eliminado todos los colores que sobraban, se han borrado líneas, asteriscos y guiones de separación, se han quitado tabuladores que eran innecesarios y se ha puesto un proceso de cada color (ya que antes, al ordenar los procesos, este color cambiaba, y ahora se mantiene fijo), quedando así todo mucho más limpio y sencillo de leer, especialmente para terminales que puedan llegar a tener un color de fondo que haga que el texto de determinados colores no se llegue a leer bien.
- Los nombres de los procesos ahora se generan automáticamente, en vez de tener que introducirlos manualmente.
- Se ha corregido el orden de los procesos a la hora de ponerlos en cola, ya que no estaba bien implementado, quedando ordenados por tiempo de llegada, después por tiempo de ejecución, y, por último, por tiempo de introducción.
- La memoria completa ahora se calcula a partir de las particiones, y, por lo tanto, no hay que indicarla de antemano, como se hacía anteriormente.
- En las tablas se han ajustado los nombres a la izquierda, y los números a la derecha.
- Se ha cambiado por completo el orden de la pantalla de ejecución, estando ahora compuesta por una cabecera, un listado, unos tiempos medios de espera y ejecución, y posteriormente las bandas de memoria y ejecución.
- Se ha cambiado por completo el aspecto de las bandas de memoria y tiempo, ahora cada unidad queda representada por tres cuadrados, para así poder llegar a representar gráficamente cada unidad. Los procesos estarán ajustados a la izquierda,

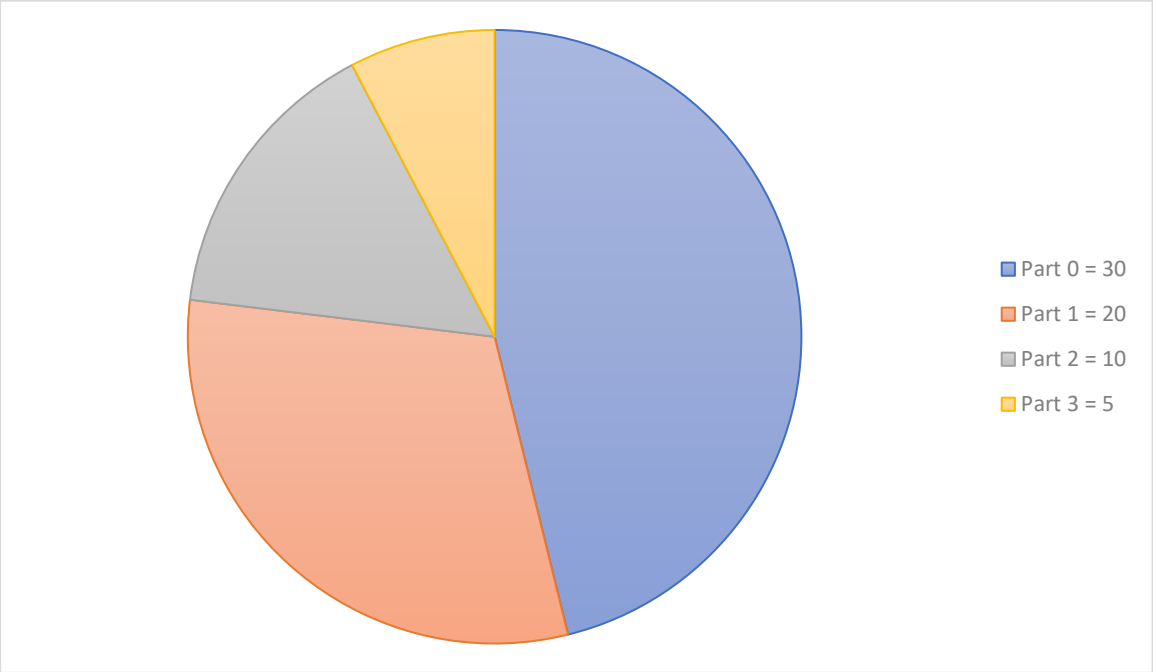
mientras que las unidades numéricas lo estarán a la derecha. En caso de la banda de memoria, las particiones estarán separadas por una barra vertical, y, aparte de mostrar la primera unidad de cada partición, también se ha añadido que se muestre la primera unidad libre después de cada proceso que ocupa una partición. En caso de la banda de tiempo, ahora sólo muestra la gráfica de las unidades que ya se han ejecutado, además de la que se encuentra ejecutándose en ese momento (es decir, va creciendo con el tiempo), y sólo se muestran las unidades correspondientes a la primera unidad de un evento, es decir, el comienzo de la ejecución de un proceso, o una zona libre de ejecución. La gráfica va a ser pintada por el color de los procesos que se han ejecutado en esas determinadas unidades. Los números y los nombres de los procesos se han ajustado para que vayan a la par de las unidades.

- En las bandas de memoria y tiempo, se ha implementado un correcto salto de línea, calculando por ello el ancho de la terminal, para así saber en qué momento tiene que realizar el salto de línea, quedando así bien estructurados todos los valores de unidades y procesos.
- En la ejecución, ahora se muestra siempre T=0, ya que antes sólo se mostraba en caso de que sucediera algún evento.
- Se han reescrito varias funciones encargadas con la gestión de memoria, como CalculaDiferenciaMinima, AsignaMemoriaMejorAjuste o GestionDeMemoria, puesto que la asignación de memoria no se realizaba correctamente en algunos casos, ya que había procesos que no entraban en espera cuando era su turno, y algunos procesos entraban en memoria cuando los que estaban delante de ellos en la cola todavía estaban en espera, haciendo que los resultados no fueran correctos.
- Se han eliminado los archivos temporales, así como un archivo llamado datosIntroducidos.txt, que devolvía los datos usados en la última ejecución (algo duplicado, ya que es algo que ya lo hace con datosEntradaPred.txt)
- Se ha cambiado la estructura de datosEntradaPred.txt, haciendo que la entrada de datos por fichero sea mucho más sencilla de utilizar, en caso de tener que modificar algún dato utilizando el propio archivo.
- Se han estructurado correctamente tanto los informes de color como el de blanco y negro, ya que había datos que no aparecían en estos, como las preguntas de introducción de datos, y se han eliminado colores que aparecían en el archivo de blanco y negro, creando para ello variantes de las funciones de las gráficas (las funciones que tienen como nombre xxxxxBW)
- Se ha implementado que no se puedan introducir particiones iguales, algo que anteriormente si se podía hacer.
- En el listado de la ejecución, cuando un proceso se encuentra fuera del sistema, los valores de tiempo de espera, de retorno y tiempo restante de ejecución se muestran con guion, ya que estos procesos no tienen valor en ese estado. Cuando se encuentran en espera, si que tendrán tiempo de espera y de retorno, pero en el tiempo restante de ejecución se seguirá mostrando un guion, al igual que pasa cuando el estado del proceso es finalizado.
- Se han corregido los valores del tiempo de retorno, que antes no eran correctos.
- Se ha corregido el que sólo se muestren las ventanas en las que pasa algo, y no todas, ya que antes esto estaba implementado, pero funcionaba mal.

- Se ha añadido una nueva columna al listado, que indica el número de partición en la que se encuentra un proceso, en caso de estar en memoria o en ejecución (si no lo está se muestra un guion)
- Se han implementado correctamente las prioridades, para que, en caso de dar un rango en el que la prioridad menor tenga un valor mayor que la prioridad mayor, la práctica funcione de tal manera que las prioridades se inviertan, es decir, el programa pase a ejecutar los procesos con mayor valor de prioridad antes. Esto lo hace mediante una operación que se aplica a todas las prioridades, que transforma el rango dado a un rango del 0 a x, siendo 0 la prioridad mínima y x la máxima.

4. Desarrollo de ejemplo a mano

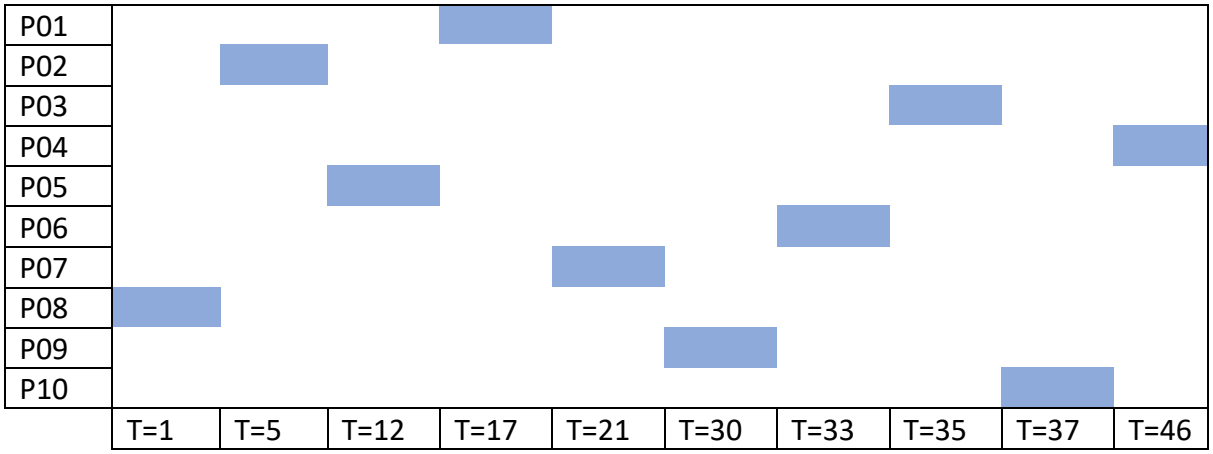
Vamos a realizar un ejercicio para entender mejor el funcionamiento del algoritmo, y así poder trasladarlo posteriormente al script. Vamos a partir de una memoria con particiones fijas y no iguales, que va a poseer las siguientes características:



En cuanto a los procesos, esta va a ser la lista:

Nombre	Tiempo de llegada	Tiempo de ejecución	Memoria	Prioridad
P01	7	4	9	1
P02	2	7	5	-5
P03	10	2	25	6
P04	12	9	12	8
P05	3	5	2	0
P06	4	2	30	9
P07	5	9	10	3
P08	1	4	8	6
P09	7	3	1	7
P10	17	9	3	6

Una vez tengamos claros los datos, comenzamos la ejecución:



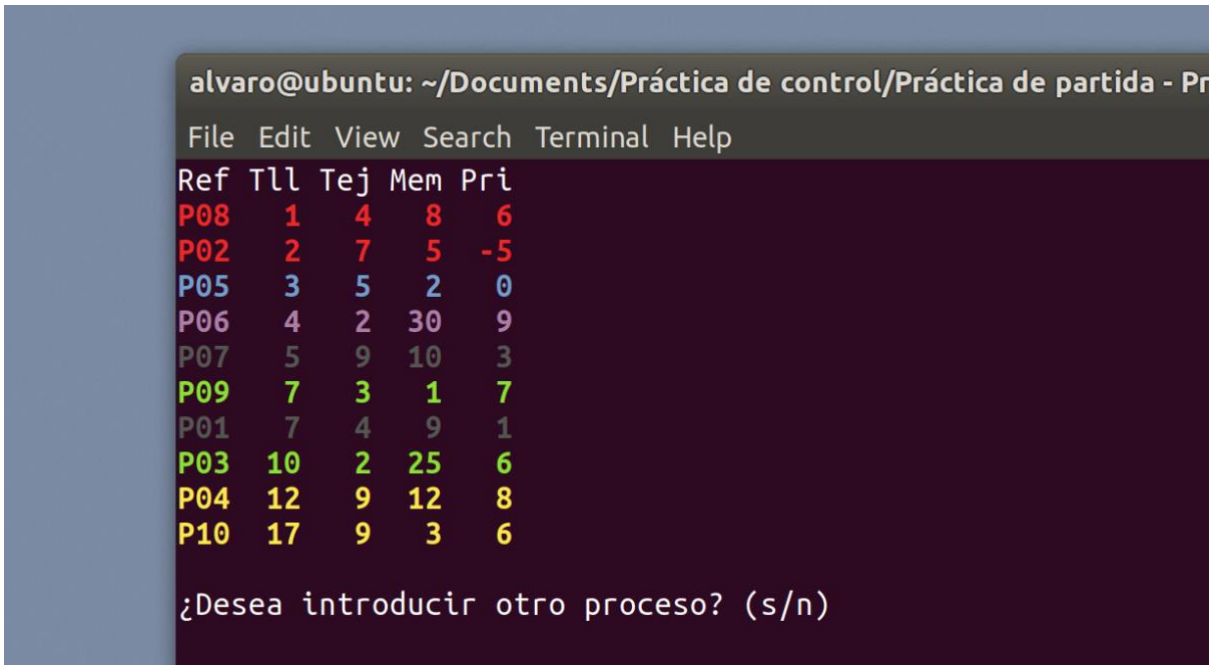
En T=1, P08 va a colocarse en la partición 2, y, como es el único proceso en memoria, comienza su ejecución.
En T=2, P02 va a situarse en memoria, en la partición 3.
En T=3, P05 va a situarse en memoria, en la partición 1.
En T=4, P06 va a situarse en memoria, en la partición 0.
En T=5, P07 va a situarse en memoria, en la partición 2, que acaba de quedar libre, ya que P08 termina su ejecución, y comienza la ejecución de P02, ya que es el que posee menor prioridad de los procesos que se encuentran en memoria.
En T=7, P09 y P01 entran en el sistema, pero al no haber ninguna partición libre, se quedan en espera.
En T=10, P03 entra en el sistema, pero al no haber ninguna partición libre, se queda en espera.
En T=12, P09 va a situarse en memoria, en la partición 3, que acaba de quedar libre, ya que P02 termina su ejecución, y comienza la ejecución de P05, ya que es el que posee menor prioridad de los procesos que se encuentran en memoria.
En T=17, P01 va a situarse en memoria, en la partición 1, que acaba de quedar libre, ya que P05 termina su ejecución, y comienza la ejecución de este mismo proceso, ya que es el que posee menor prioridad de los procesos que se encuentran en memoria.
En T=21, P01 termina su ejecución, y comienza la ejecución de P07, ya que es el que posee menor prioridad de los procesos que se encuentran en memoria.
En T=30, P07 termina su ejecución, y comienza la ejecución de P09, ya que es el que posee menor prioridad de los procesos que se encuentran en memoria.
En T=33, P09 termina su ejecución, y comienza la ejecución de P06, ya que es el que posee menor prioridad de los procesos que se encuentran en memoria (puesto que es el único).
En T=35, P03, P04 y P10 van a situarse en memoria, en las particiones 0, 1 y 3 respectivamente (puesto que P06 estaba ocupando la partición 0, y como P03 sólo cabe en la partición 0 y era el que primero se encontraba en la cola, el resto no podían situarse en memoria hasta que lo hiciese este) ya que P06 termina su ejecución, y comienza la ejecución de P03, ya que es el que posee menor prioridad de los procesos que se encuentran en memoria.

En T=37, P03 termina su ejecución, y comienza la ejecución de P10, ya que es el que posee menor prioridad de los procesos que se encuentran en memoria.
En T=46, P10 termina su ejecución, y comienza la ejecución de P04, ya que es el que posee menor prioridad de los procesos que se encuentran en memoria.
En T=55, P04 termina su ejecución, y termina el programa.

5. Desarrollo de ejemplo a través del script

Vamos a ejecutar el script utilizando los mismos datos que hemos utilizado en el ejemplo anterior, para comprobar el correcto funcionamiento de este.
El script funciona de tal manera que, nada más iniciarlo, nos permite realizar la introducción de datos tanto a mano, como mediante fichero (utilizará los datos usados en la última ejecución a mano).
En caso de hacerlo a mano (como vamos a hacer en este caso), nos pedirá el tamaño de las particiones que queremos utilizar (debemos recordar que son particiones fijas y no iguales, por lo tanto, no deben de repetirse los tamaños). A continuación, se nos pedirá tanto la prioridad mínima como la prioridad máxima, es decir, el rango de valores entre los que se puede encontrar la prioridad de los procesos. El algoritmo está programado de manera que, en caso de introducir una prioridad mínima cuyo número sea mayor al de la prioridad máxima (como, por ejemplo, 20 de prioridad mínima y -10 de prioridad máxima), a la hora de la ejecución, se van a ejecutar antes los procesos cuyo número de prioridad sea mayor, es decir, se van a invertir las prioridades. En nuestro caso no va a ser así, puesto que vamos a introducir como datos una prioridad mínima de -10 y una prioridad máxima de 20.
Después de ser esta asignada, llegaremos a la pantalla en la que se produce la inserción de procesos. Se nos pedirán los siguientes datos:

- **Tiempo de llegada:** Es el tiempo en el cual el proceso entrará en memoria (en el caso, claro está, de que la partición que indique el mejor ajuste esté libre, en caso negativo se quedará en espera).
- **Tiempo de ejecución:** Una vez entre el proceso a ejecutarse, es el tiempo que va a durar su ejecución.
- **Memoria:** Es la memoria que va a ocupar el proceso.
- **Prioridad:** Es la prioridad del proceso.



Según vayamos introduciendo los procesos, podemos observar cómo se nos muestra una tabla en la que aparecen los procesos que ya hemos introducido, además del que estemos introduciendo en el momento. También podemos ver que, según introducimos los procesos, el script se encarga de ir ordenando los procesos automáticamente según el siguiente criterio:

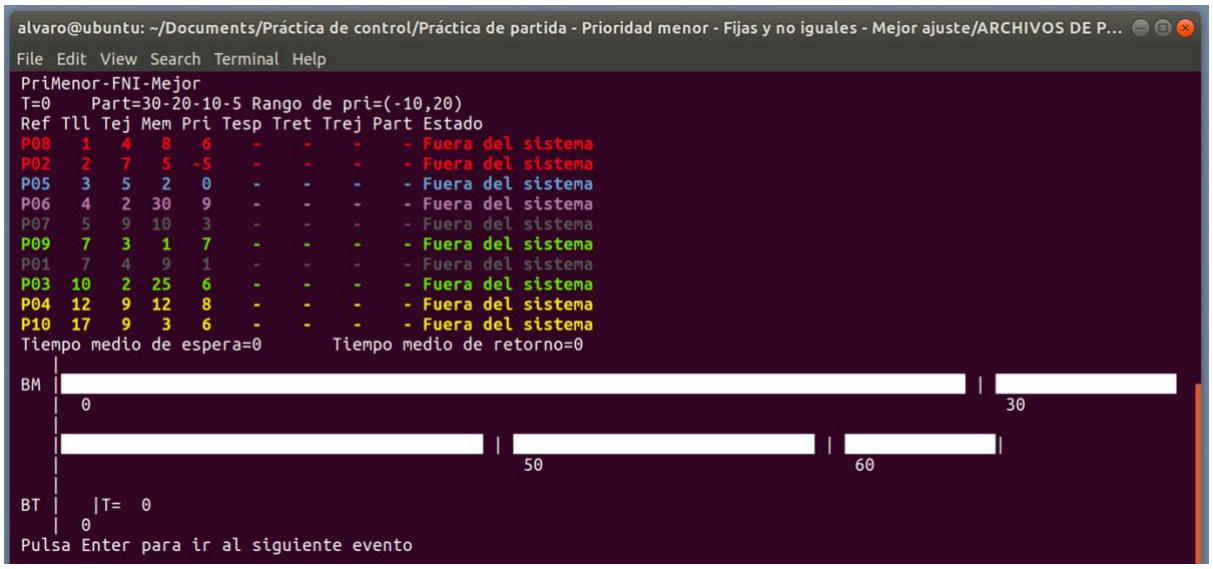
- Se comenzará ordenando los procesos por tiempo de llegada, encontrándose arriba los que tengan valores más pequeños.
- En caso de haber dos procesos con mismo tiempo de llegada, se situará arriba el que tenga menor tiempo de ejecución.
- Si se da el caso de que existan dos procesos con mismo tiempo de llegada y de ejecución, se situará arriba el que tenga mayor tiempo de introducción, es decir, el que haya sido introducido primero.

Una vez se hayan introducido todos los procesos, pasaremos, tras una corta cuenta atrás, a la ventana de ejecución, la cual está compuesta por una breve descripción del algoritmo, tiempo en el que nos encontramos y particiones de la memoria, un listado con todos los procesos y su estado de ejecución, los tiempos medios tanto de espera como de retorno, y una banda de memoria y otra de tiempo, que representan las particiones de la memoria, así como la ejecución del algoritmo a través del tiempo.
El listado de los procesos contiene los siguientes apartados (sin contar los mencionados anteriormente):

- **Tiempo de espera:** Es el tiempo que ha tenido que esperar el proceso para poder entrar en la partición deseada y comenzar a ejecutarse.
- **Tiempo de retorno:** Es el tiempo que ha pasado desde que se ha introducido el proceso en el sistema, hasta que ha finalizado su ejecución.
- **Tiempo restante de ejecución:** Es el tiempo que le queda al proceso para terminar su ejecución, en caso de que lo esté.
- **Partición:** Es la partición en la que se encuentra el proceso, en caso de estar en memoria o en ejecución (las particiones van desde 0 hasta n-1).

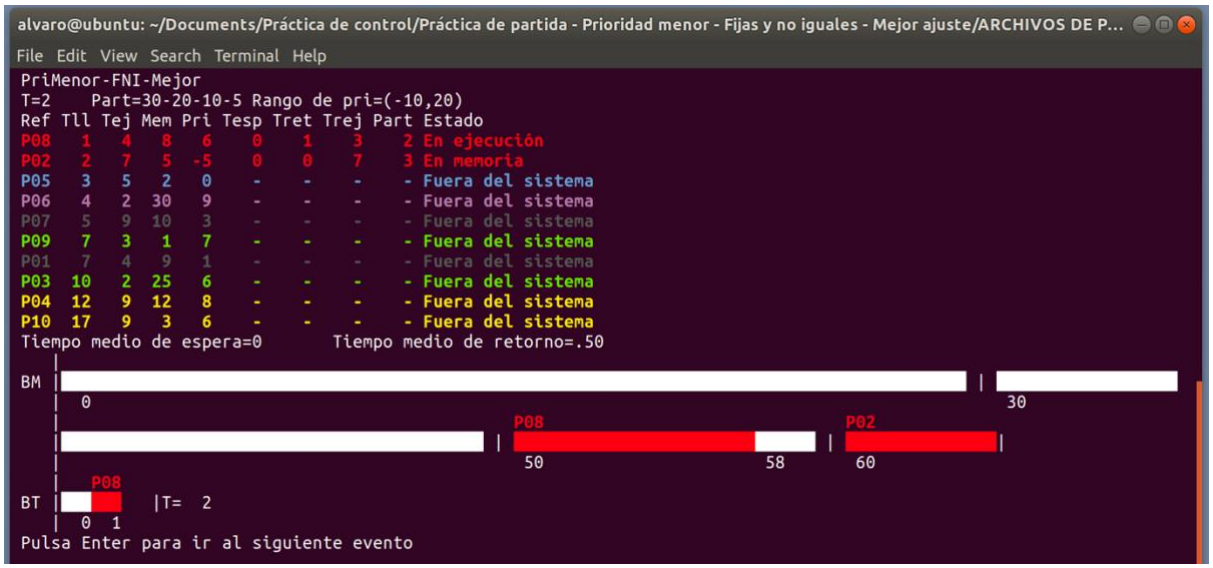
- **Estado:** Es el estado en el que se encuentra el proceso, ya sea fuera del sistema, en espera, en memoria, en ejecución o terminado.

Podremos pulsar Enter para pasar al siguiente tiempo en el que ocurra algo, es decir, que algún proceso cambie de estado.

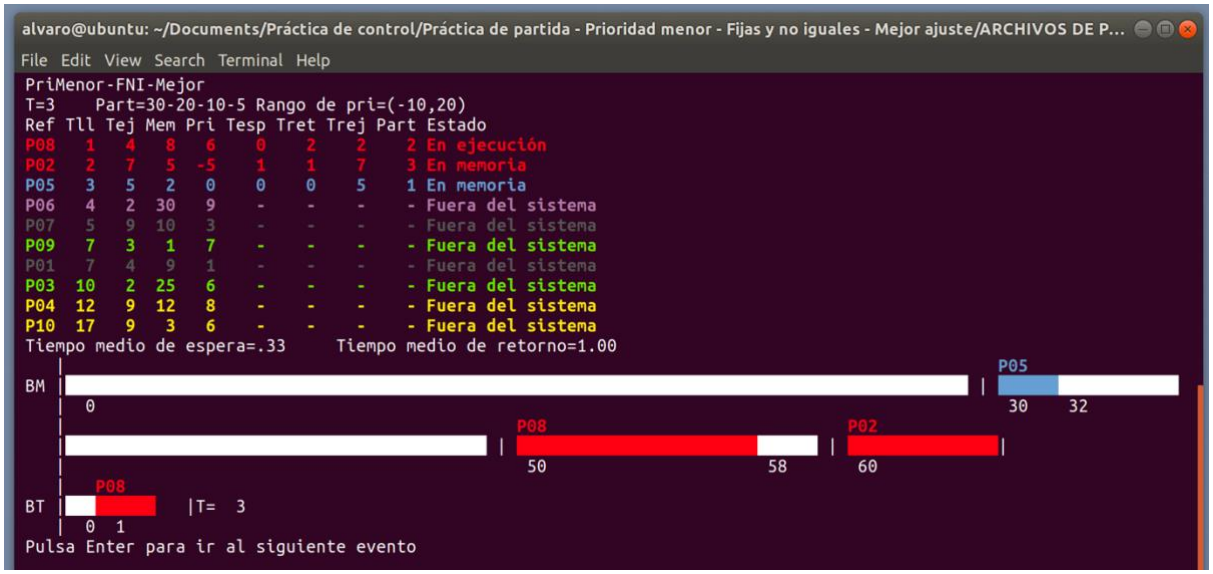


En T=0, todos los procesos se encuentran fuera del sistema, ya que el primer proceso en la cola no entra en sistema hasta T=1.

En T=1, P08 se sitúa en memoria, y comienza su ejecución.

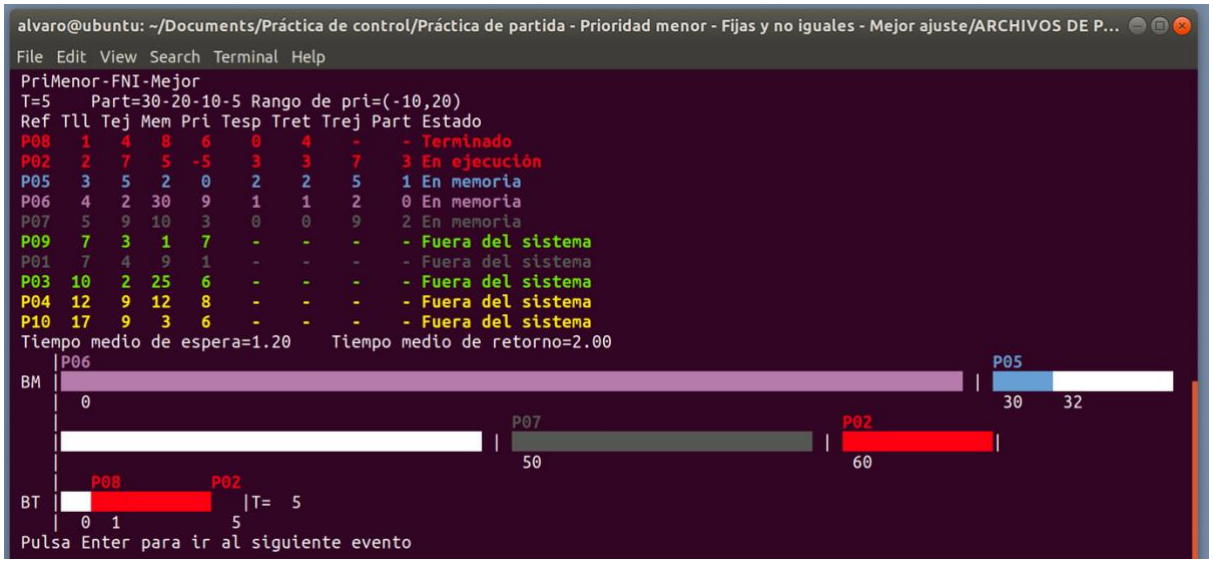


En T=2, P02 se sitúa en memoria.

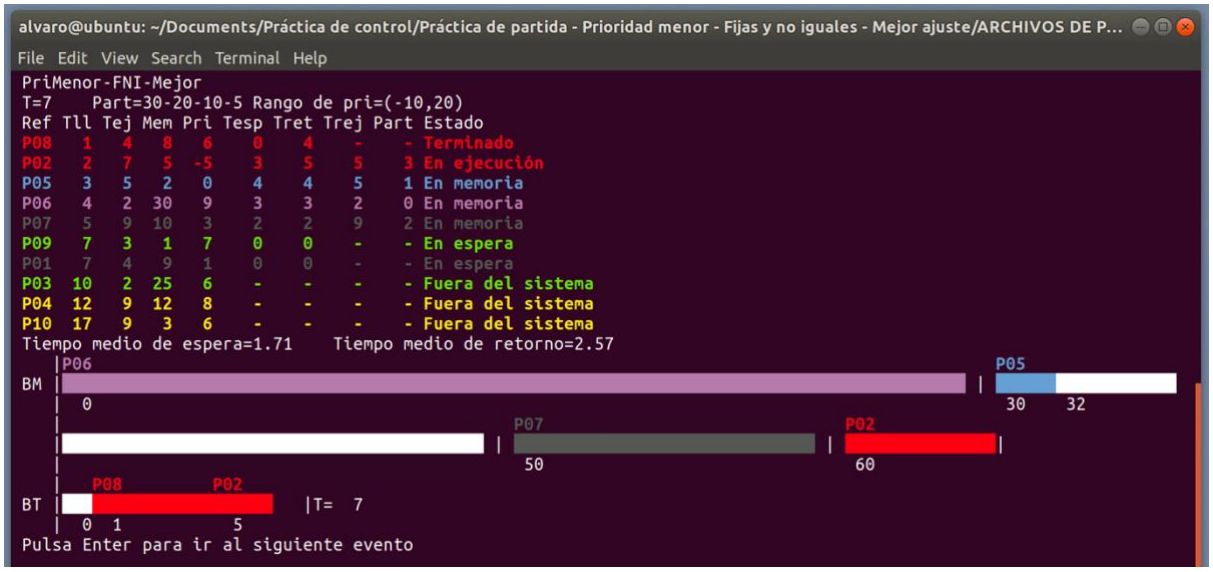


En T=3, P05 se sitúa en memoria.

En T=4, P06 se sitúa en memoria.

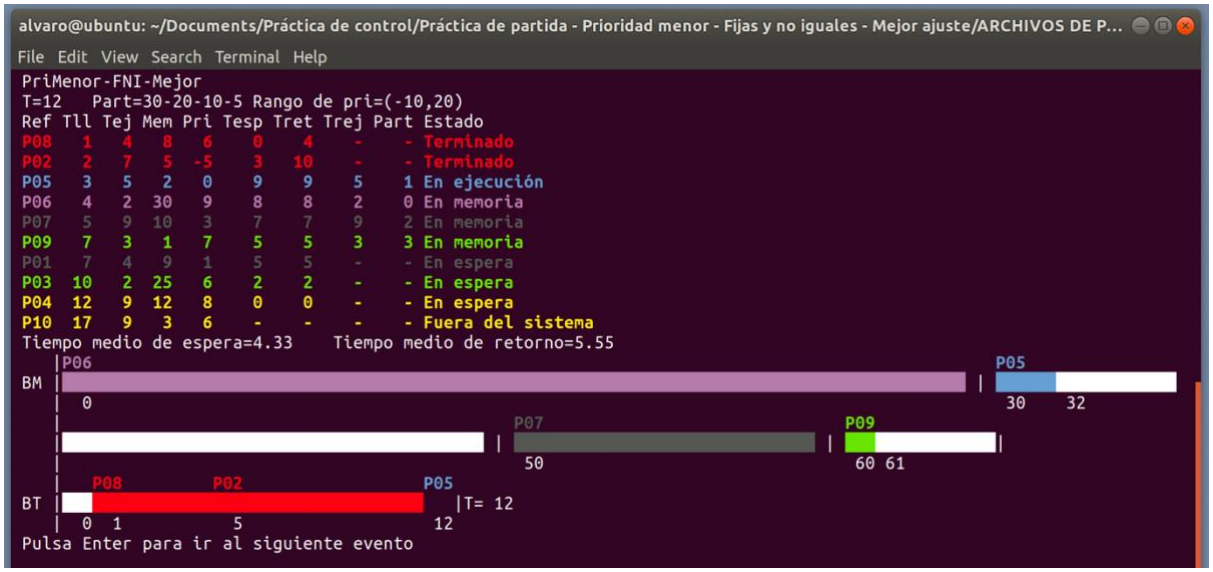


En T=5, P07 se sitúa en memoria, P08 termina su ejecución, y comienza a ejecutarse P02.



En T=7, P09 y P01 entran en el sistema, pero se quedan en espera al no haber ninguna partición libre.

En T=10, P03 entra en el sistema, pero se queda en espera.



En T=12, P09 se sitúa en memoria, P02 termina su ejecución, y comienza a ejecutarse P05.

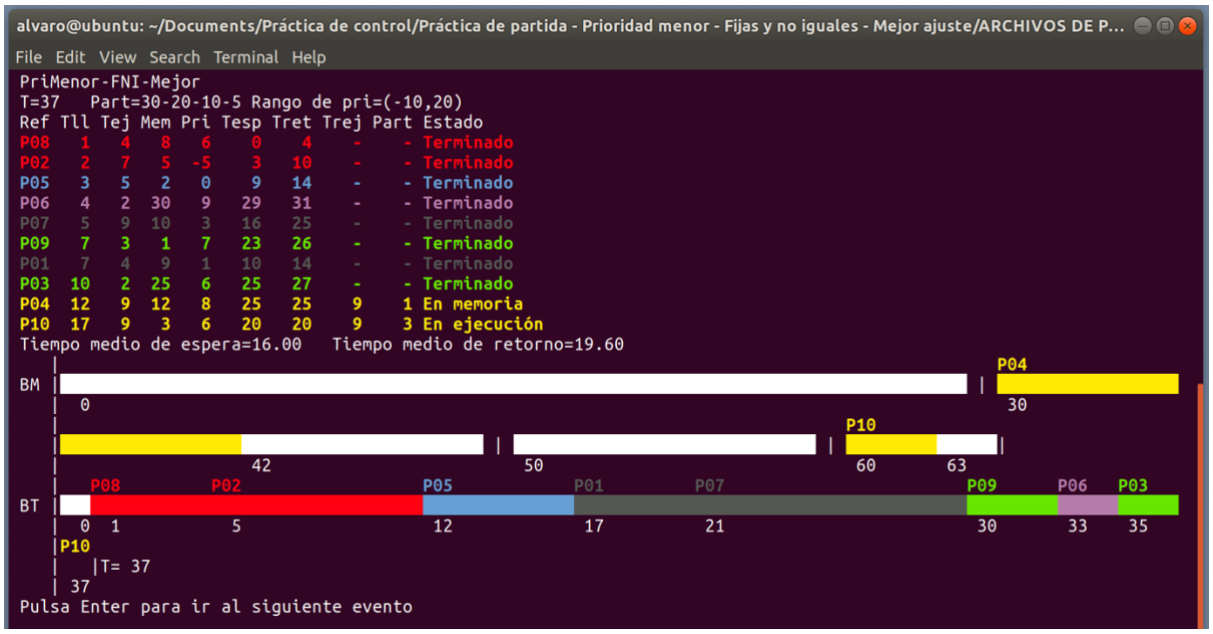
En T=17, P01 se sitúa en memoria, P05 termina su ejecución, y comienza a ejecutarse P01.

En T=21, P01 termina de ejecutarse, y empieza a ejecutarse P07.

En T=30, P07 termina de ejecutarse, y comienza a ejecutarse P09.

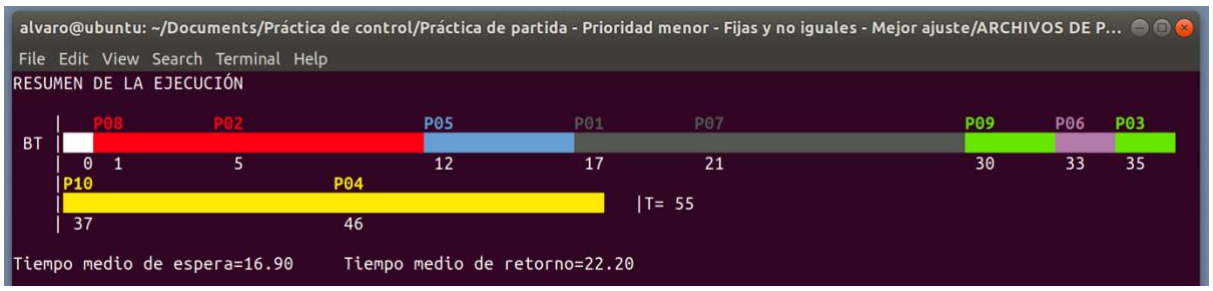
En T=33, P09 termina de ejecutarse, y comienza a ejecutarse P06.

En T=35, P03, P04 y P10 se sitúan en memoria, P06 termina de ejecutarse, y comienza a ejecutarse P03.



En T=37, P03 termina de ejecutarse, y comienza a ejecutarse P10.

En T=46, P10 termina de ejecutarse, y comienza a ejecutarse P04.



En T=55, P04 termina de ejecutarse y termina el programa.

Al llegar al final de la ejecución del programa, se nos mostrará una banda de tiempo con un resumen de toda la ejecución. Se nos dará la opción de visualizar un informe con una vista detallada de toda la ejecución del script, así como de abrir un archivo .txt con la misma (con la diferencia de que este es sin color).

Por último, en caso de escoger introducir los datos mediante fichero, el fichero que utilizará es el llamado **datosEntradaPred.txt** (que ahora almacenará todos los datos de la ejecución que acabamos de realizar). Este fichero consta de la siguiente estructura:

```
30;20;10;5;
-10;20;
7;4;9;1;
2;7;5;-5;
10;2;25;6;
12;9;12;8;
3;5;2;0;
4;2;30;9;
5;9;10;3;
1;4;8;6;
7;3;1;7;
17;9;3;6;
~
```

En la primera línea, tenemos los valores de memoria de cada una de las particiones, separados por un ;
En la segunda línea, tenemos los valores de prioridad mínima y máxima, separados por un ;
En las siguientes líneas tenemos los procesos a ser ejecutados, cada uno en una línea distinta, con los valores de tiempo de llegada, tiempo de ejecución, memoria y prioridad separados por ;

6. Conclusiones

La verdad es que ha sido un trabajo bastante largo y complicado, pero que, sin embargo, creo que nos puede servir mucho a la hora de mejorar nuestras habilidades en el campo de la programación, especialmente a la hora de interpretar código, ya que partimos de algo que no hemos escrito nosotros y que debemos de analizar.
Creo que, si bien este algoritmo optimiza bastante bien la memoria a la hora de poder ejecutar un solo proceso, en caso de poder ejecutar más sería bastante conveniente el hecho de poder almacenar más de un proceso en memoria, ya que así la aprovecharía al máximo.

7. Bibliografía

- *TLDP – Advanced Bash-Scripting Guide*
- *ComputerHope – Bash builtins help*
- *LinuxHint*
- *LikeGeeks – 31+ ejemplos para el comando sed de Linux en la manipulación de texto*
- *LinuxConfig – How to debug Bash scripts*
- *StackOverflow*