**Centro Universitario de Ciencias Exactas e Ingeniería**



**Materia:** Sistemas Operativos

**Profesor:** Ramiro Lupercio Coronel.

**Alumnos:**

Angel Martin Ramirez Castorena – 216788066

Cristian Rafael Romero Chávez – 216787507

Jonathan Daniel Requenes Nuñez – 216787779

**Fecha:** 05 de noviembre 2019

**Practica:** Menor tiempo restante

**Menor tiempo restante (algoritmo).**

**Objetivo**

Desarrollar un programa donde se muestre el funcionamiento del algoritmo planificador de procesos “Menor tiempo restante”.

**Investigación**

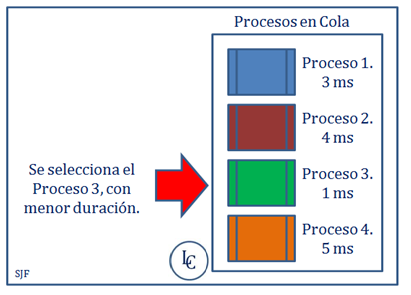
El planificador selecciona el proceso que tenga el tiempo restante de ejecución más corto y lo ejecuta. Este algoritmo es de tipo apropiativo, es decir, si un proceso más corto llega, el proceso que se está ejecutando actualmente se suspende y el nuevo proceso más corto se ejecuta.

Es la variedad expropiativa de SJF. Eso significa que el proceso con menor tiempo para acabar es el siguiente proceso en ejecutarse expropiando la CPU inmediatamente al proceso que este en ejecución en el instante correspondiente. El problema vendría en el caso que tengamos un proceso que requiera un tiempo de ejecución para finalizar igual que un proceso nuevo que entra. Existen dos soluciones, dar prioridad a los procesos nuevos sobre los procesos en ejecución o dar prioridad a los procesos en ejecución sobre los procesos nuevos.

Características

* Optimiza la media del tiempo de espera y de rendimiento.
* Da prioridad al proceso que le reste menos tiempo de CPU para terminar.
* Los procesos llegan a la lista de “ready” y solicitan un intervalo con la CPU. Si dicho intervalo es inferior al proceso en ejecución puede ser expulsado.
* El intervalo de CPU es difícil de predecir.
* Posibilidad de inanición debido a que los trabajos largos no se ejecutaran mientras haya trabajos cortos.

Es similar al primero el más corto, con la diferencia de que si un nuevo proceso pasa a listo se activa el *[dispatcher](https://lsi.vc.ehu.eus/pablogn/docencia/manuales/SO/TemasSOuJaen/PLANIFICACIONDEPROCESOS/1NivelesdePlanificacion.htm" \l "planificador)* para ver si es más corto que lo que queda por ejecutar del proceso en ejecución. Si es así el proceso en ejecución pasa a listo y su tiempo de estimación se decrementa con el tiempo que ha estado ejecutándose.



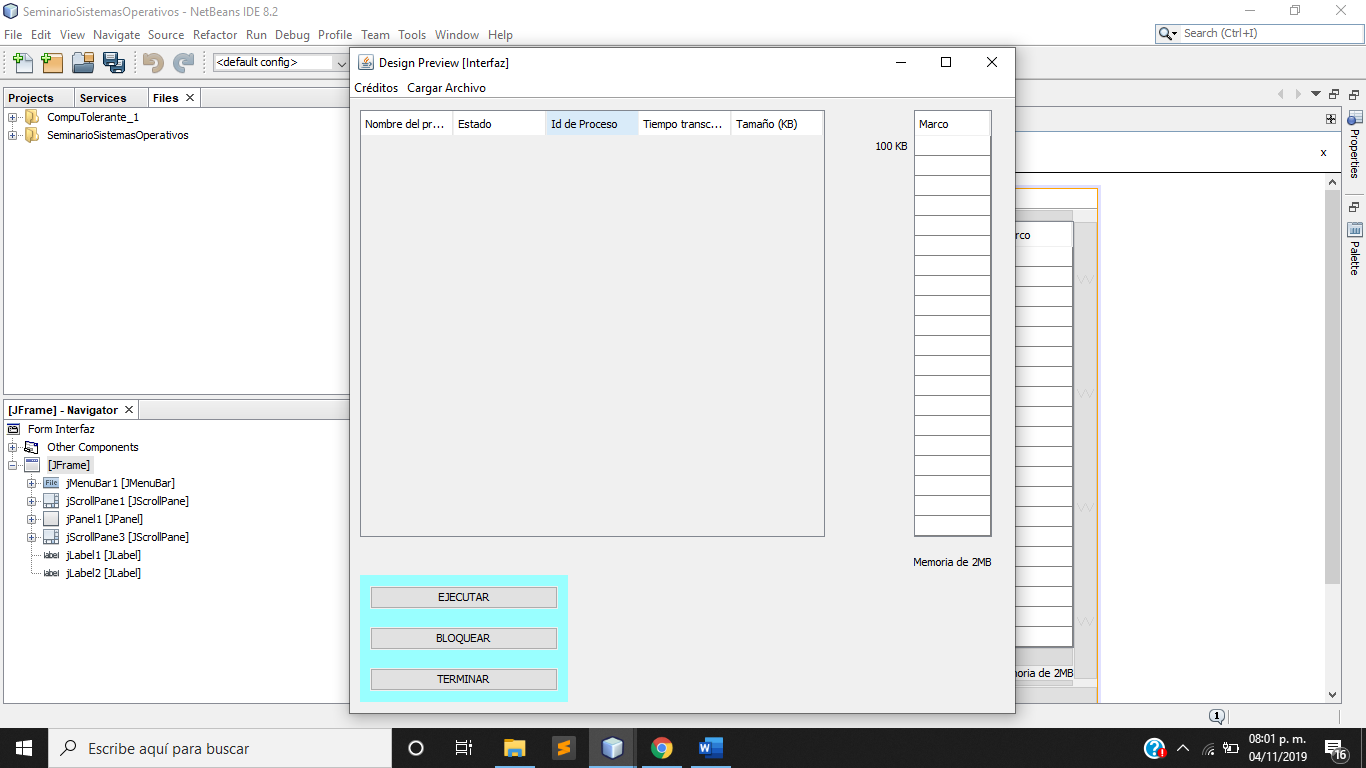
La política del menor tiempo restante (SRT, Shortest Remaining Time) es una versión apropiativa del SPN, en la que el planificador siempre elige al proceso que le queda menos tiempo esperado de ejecución. Cuando se añade un nuevo proceso a la cola de Listos, puede quedarle un tiempo esperado de ejecución menor que al proceso que está ejecutándose en ese momento.

Por consiguiente, el planificador puede apropiarse del procesador siempre que un proceso nuevo esté listo. Como en el SPN, el planificador debe disponer de una estimación del tiempo de proceso para poder llevar a cabo la función de selección, existiendo el riesgo de inanición para procesos largos.

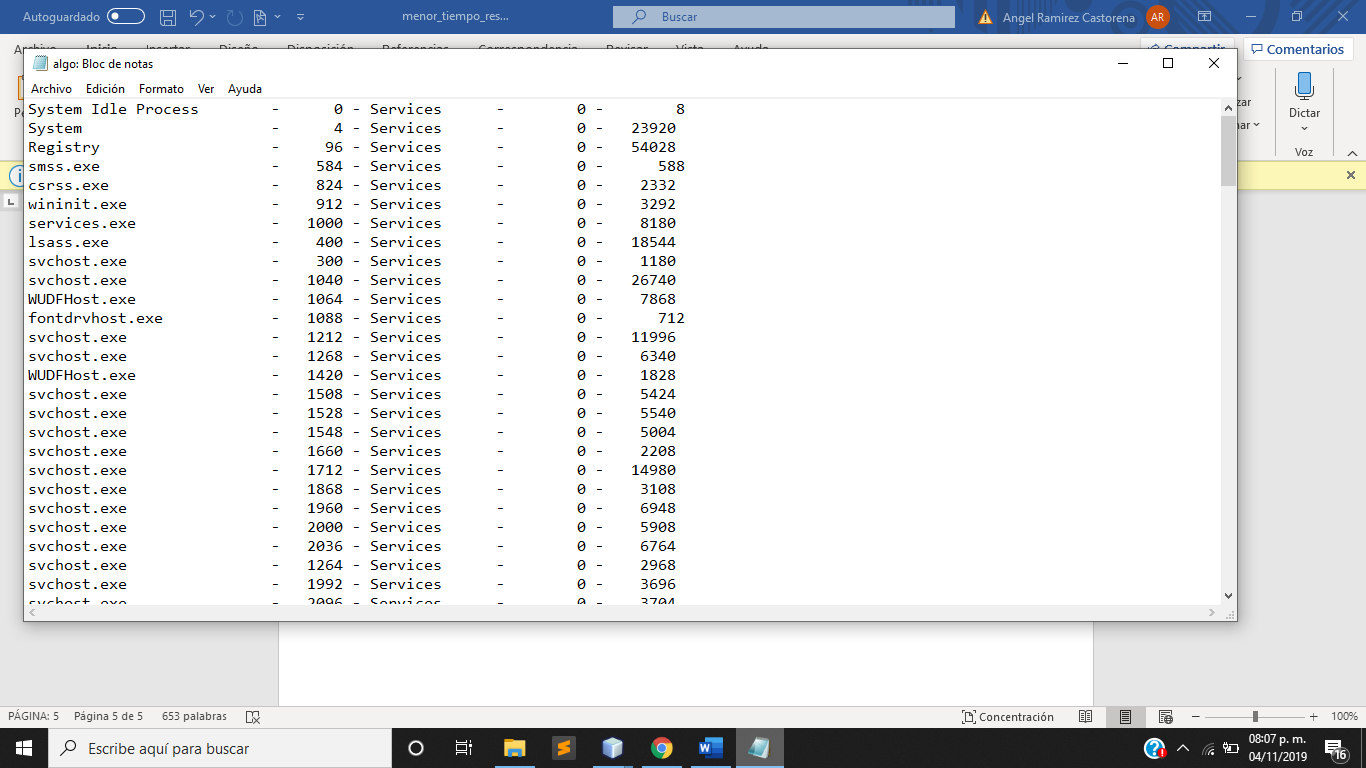
El SRT no presenta el sesgo favorable a los procesos largos del PCFS. Al contrario que el turno rotatorio, no se generan interrupciones adicionales y, así, el coste se ve reducido. Por contra, se deben los tiempos de servicio transcurridos, lo que contribuye a la sobrecarga. El SRT también debe producir unos tiempos de retorno mejores que los del SPN, puesto que los trabajos cortos reciben una atención inmediata y preferente a los trabajos largos.

**Desarrollo**

Para el desarrollo de esta practica se reutilizó lo anterior visto en la materia Seminario de Sistemas Operativos, el cual carga de manera inicial una lista de procesos desde un archivo de texto, en dicho archivo de texto esta toda la información relevante a los procesos como el nombre, tamaño en KB, prioridad y tiempo aproximado de ejecución. Y toda esta información se miestra de manera gráfica en la ventana principal del programa. Cabe señalar que cada proceso es una clase que hereda de la clase Thread y hay un gestor de procesos que se encarga propiamente de administrar estos procesos.



Como se puede observar la ventana principal contiene unos controles que manejan los estados de los procesos: ejecutar, bloquear, terminar. Y también cuenta con una barra de memoria la cual se va llenando conforme se van cargando los procesos a memoria.



Este es el txt de donde se carga toda la información respecto a los procesos.

Una vez que se cargan los procesos a una lista de Procesos con este algoritmo:

public Interfaz() {

initComponents();

ruta = "C:\\Users\\Angel\\Documents\\nuevo.txt";

modelo = (DefaultTableModel) jTable.getModel();

modeloMemoria = (DefaultTableModel) jTbMemoria.getModel();

proceso\_principal = new GestorProcesos(modelo, jTable, jTbMemoria);

//MiRender mr = new MiRender();

//jTbMemoria.setDefaultRenderer(Object.class, mr);

//proceso\_principal.start();

leeArchivo();

}

Se procede a ordenar dicha lista del menor tiempo de ejecución al de mayor tiempo de ejecución e ir ejecutando la mayor cantidad posible que quepa dentro de la memoria RAM asignada. Cabe señalar que se utiliza el algoritmo de ordenamiento de burbuja por simplicidad.

for(int i = 0; i < arreglo.length - 1; i++)

{

for(int j = 0; j < arreglo.length - 1; j++)

{

if (arreglo[j].tiempo > arreglo[j + 1].tiempo)

{

Tiempo tmp = arreglo[j+1];

arreglo[j+1] = arreglo[j];

arreglo[j] = tmp;

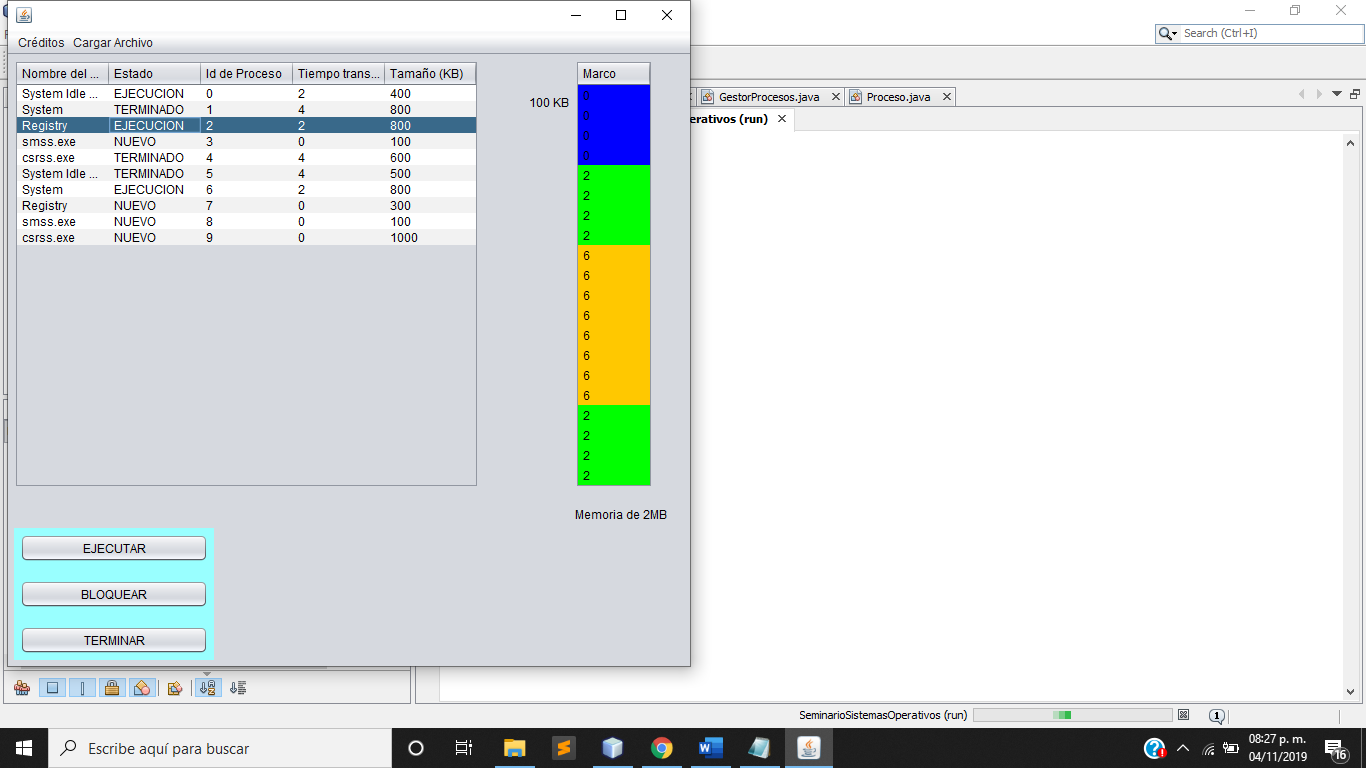
}

}

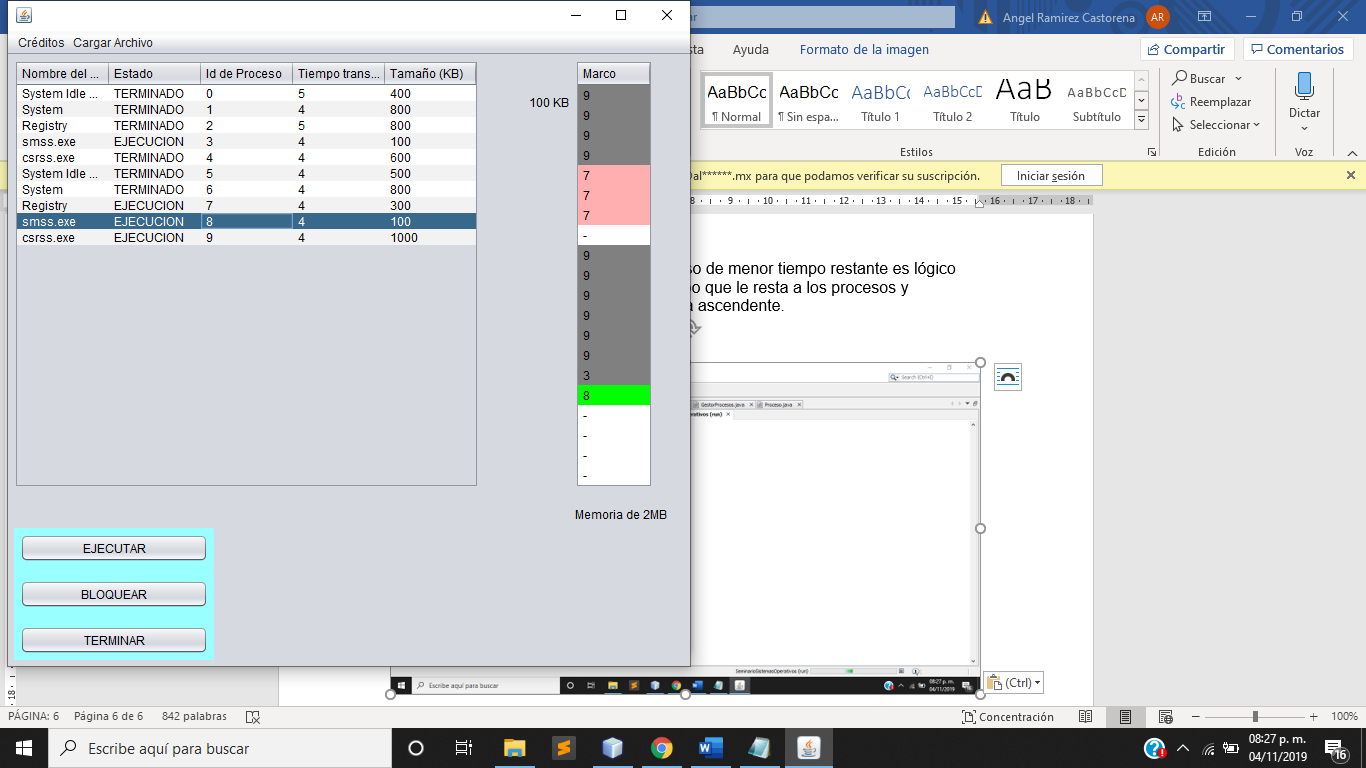
}

Cómo lo que interesa es obtener el proceso de menor tiempo restante es lógico hacer la comparación en función del tiempo que le resta a los procesos y ordenarlos del menor al mayor, de manera ascendente.

**Resultados**



Como se puede observar los procesos con menor duración fueron los que se atendieron primero y después de estos siguen los siguientes mayores de tiempo, no se ejecutan todos a la vez debido a que no “caben en la memoria” y esto provocaría un desbordamiento dando origen a la caída del sistema operativo.



Finalmente se ejecutan los procesos que tardarían más tiempo, para al final tener un mejor rendimiento en la ejecución y gestión de procesos.

**Conclusión**

Podemos concluir de manera general que el algoritmo del menor tiempo restante tiene una gran ventaja debido a que se encarga de procesar antes a los que demorarán menos en terminar con su ciclo de vida de ejecución. Si bien puede ser una ventaja en que el proceso de menor tiempo se apropie de la cpu, en caso de que existan una cantidad casi infinita de procesos pequeños puede darse el caso de que los procesos mas grandes nunca sean ejecutados. Para solventar este problema se podría optar por combinar este algoritmo como, por ejemplo, el rotativo, en donde una vez ordenado los procesos tengan un cuanto fijo para ser ejecutados despachando primero a los procesos con menor tiempo, pero también despachando a los procesos más tardados.

Y para finalizar en este practica agregamos lo que seria la gestión de memoria en los sistemas operativos, en donde la memoria se divide en bloques de tamaños fijos pequeños llamados marcos los cuales se utilizan para albergar a las paginas de los procesos, que también son secciones de tamaños fijos de los procesos que por lo general tienen el mismo tamaño que los marcos de la memoria. En este ejemplo que se desarrollo las paginas de los procesos son cargados a la memoria y solo se valida que no se desborde la memoria.

**Referencias**

Gonzalez, J. A. (2012, 27 agosto). El Menor Tiempo Restante A Continuación. Recuperado 5 noviembre, 2019, de http://algoritmosplanificacion.blogspot.com/2012/08/el-menor-tiempo-restante-continuacion.html