Nombre: Juan Francisco Campos Aristizábal N° Alumno: 13622072

Tarea 1

1.

Una forma de implementarlo es teniendo una única cola de *Ready* y *Waiting*, tal y como lo tengo implementado ahora, de esta forma la CPU que termine antes la ejecución de un proceso pasará a ejecutar el primer proceso que estaba esperando en *Ready*.

Los procesos que salgan de la CPU seguiran el mismo orden de preferencias que tenían antes, es decir entraran sin problema a la cola *Waiting* o pasarán a *Ready* en caso que no hayan terminado su CPU *burst*. En el caso de que las dos CPU terminen de procesar al mismo tiempo, una de las dos tomara el primer proceso y la otra el segundo, esto es replicable en el caso que exitan mas núcleos.

Estadisticamente el *turnaround time* debería disminuir a la mitad ya que se van a poder procesar dos procesos simultáneamente. El *response time* de la misma forma también debería disminuir a la mitad debido a que se atenderán los procesos en la cola de *Ready* de forma más rápida. El *waiting time* tambien deberia disminuir pero producto que este es la suma entre el tiempo total de espera en la cola de *Ready* y de *Waiting*, probablemente este no disminuirá tanto ya que el tiempo que pasen los procesos en estado *Waiting* no va a variar pese a que existan dos o más CPU.

2) *O*(1) *scheduling*

Este *scheduling* permite elegir en tiempo constante cuál será el siguiente proceso a ejecutar(*O*(1) tiempo de ejecución) sin importar cuantos procesos existieran en la cola de ejecución este se demorará el mismo tiempo en elegir cuál tiene prioridad. Esto es debido a la forma de clasificación de los porcesos, para ello divide los procesos en dos tipos: * *Real Time* : * Los cuales tienen prioridades de 0(mayor prioridad) a 99. * Procesos Normales : * Tienen prioridades de 100 a 139(menor posible). * Pueden ser procesos que son *Interactive* o *Batch*.

Los tiempos para ejecutar de cada proceso son determinados de forma dinámica dada su prioridad, el cual es calculado dado una heuristica basada en su *sleep time*. Esto permite que el sistema sea interactivo al tener procesos con mayor prioridad como lo son los procesos en *Real Time* vs los procesos no tan importantes. Este *scheduling* fue implementado por primera vez en Linux 2.6.0, existieron varios cambios a lo largo de su uso de manera de mejorarlo hasta ser ocupado por última vez en Linux 2.6.22 en que fue reemplazado por Completely Fair Scheduler (CFS).