Trabajo Final Sistemas Operativos:

Emulador de un SO

Moscheni Leandro ; Skalic Julian

8 de Julio 2013

Resumen

Es objetivo de este trabajo práctico, modelar e implementar un emulador de un SO, en el cual se apliquen los conceptos teóricos vistos en la materia.
Para realizar dicho trabajo, se hace uso del Lenguaje de Programación Python, y de técnicas de POO.
Con el fin de explicar el desarrollo de la aplicación, cada módulo que la compone tendrá su sección dedicada; en la que se explicaran el diseño, y se discutirán otras cuestiones que afecten al mismo.

Introducción:

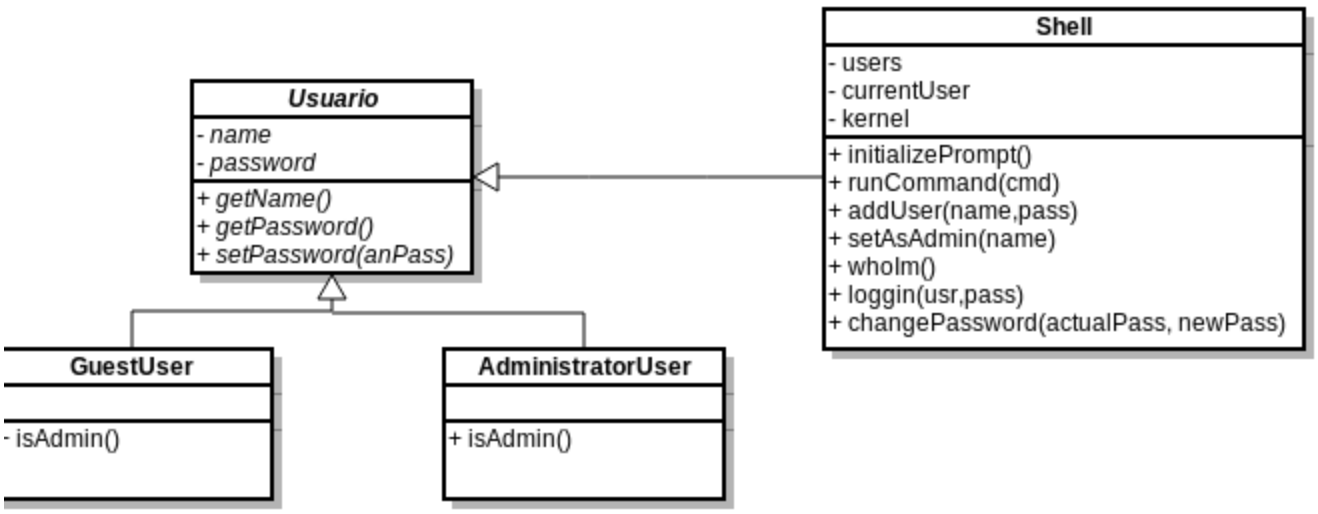
A lo largo de este informe, se irán desarrollando en cada sección los diferentes módulos que intervienen, en el emulador del SO y la explicación del diseño y funcionamiento del mismo.

El alcance de este trabajo está dado por los requerimientos impuestos en la materia, que en primera instancia, son la construcción de los siguientes módulos:

- Shell (y Usuarios)
- Kernel (Scheduler,Ready Queue,PCB)
- Interruption Handler
- Hardware (CPU,Memory,Hard Disk)

Shell

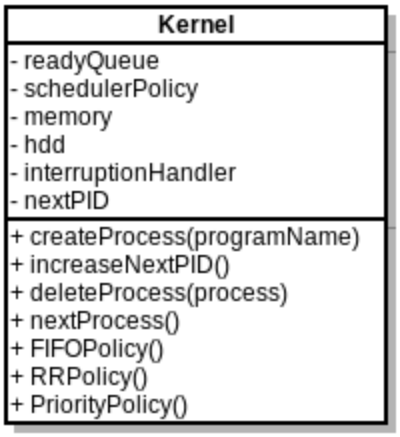
El shell es la interfaz que permite comunicar a un usuario con el sistema operativo. Para hacer esto, el shell provee al usuario de una serie de comandos predefinidos, entre los que se encuentran los comandos para loguear,crear, o volver administrador a un usuario, entro otros tantos.
Además de esto, si nosotros ingresamos al shell un comando, que en principio no conoce, pero es un programa que en disco, el kernel se encarga de ejecutar al mismo, y realizar las acciones pertinentes.
Para hablar del Shell, también es necesario hablar de los usuarios, que son el elemento básico con el que interactúan y el kernel(aunque por ahora solo lo nombraremos).
Se puede decir que el shell es el mediador entre los usuarios y el kernel(el SO en sí), ya que el usuario conoce al shell, y el shell lo conoce; por otro lado el shell conoce al kernel, pero el kernel no conoce a priori a ninguno de ellos.
Esta relación sería la siguiente:



A la hora de decidir el diseño de los usuarios, se barajó la idea, de crear un único tipo de usuario,que tuviese una variable de instancia booleana, indicando si era administrador o no, si bien la solucion era mas facil, y cumplia con su función, en el caso de que los usuarios ganasen comportamiento, como por ejemplo permisos de ejecución, etc, se iba a quedar corto, por lo que se optó, por crear una jerarquía de usuarios, y que un usuario invitado, fuera diferente a un usuario Administrador.

Kernel

El Kernel es el núcleo del sistema operativo, es la esencia del mismo. Es el encargado de interactuar con el hardware, de crear procesos, y gestionarlos, así como de otras operaciones.
El Kernel en este emulador, es representado como una clase que agrupa, varias otras, osea, es una clase que hace colaborar a otras clases, para su propio beneficio, en sí el comportamiento intrínseco del kernel, no es demasiado grande, si no que se limita a crear y eliminar procesos, y a decidir cual es el siguiente proceso que se debe ejecutar; también se encarga de comunicarse con el hardware, para cargar datos en memoria, o ir a buscar datos al disco.
Este componente fue diseñado de la siguiente manera:



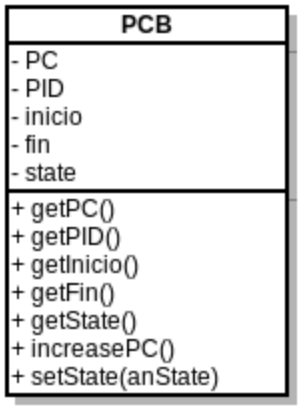
Entre las características que se le atribuyen al kernel, se cuentan una cola de listos, en la que están alojados todos los proceso (PCB's) que están listos para ejecutarse, una política de planificación a corto plazo, que es básicamente la forma de elegir a qué proceso saca de la cola, y el manejador de interrupciones, que es la forma que tiene de avisar a los otros componentes los cambios, o cosas que vaya necesitando, como lo puede ser el cambio de contexto, o el cambio a modo kernel.
Entre las capacidades que tiene el kernel, se encuentran la de crear un proceso, eliminarlo, y darle a la cpu, bajo una política especificada alguno de los procesos listos. Además de esto, el kernel puede seleccionar bajo qué política de selección/planificación actuará.

Una vez aclarado el fin, y funcionamiento del kernel, pasaremos a los diferentes componentes con los que interactúa:

PCB

PCB o Process Control Block, es la entidad, que tiene todos los datos asociados a un proceso,

que son básicamente el estado de los flags en la ejecución del programa (pc,ipc), la posición de memoria en donde comienza el programa, la posición en donde finaliza, el estado del proceso, la prioridad del mismo, si es que tiene, y el número identificatorio de proceso o PID.

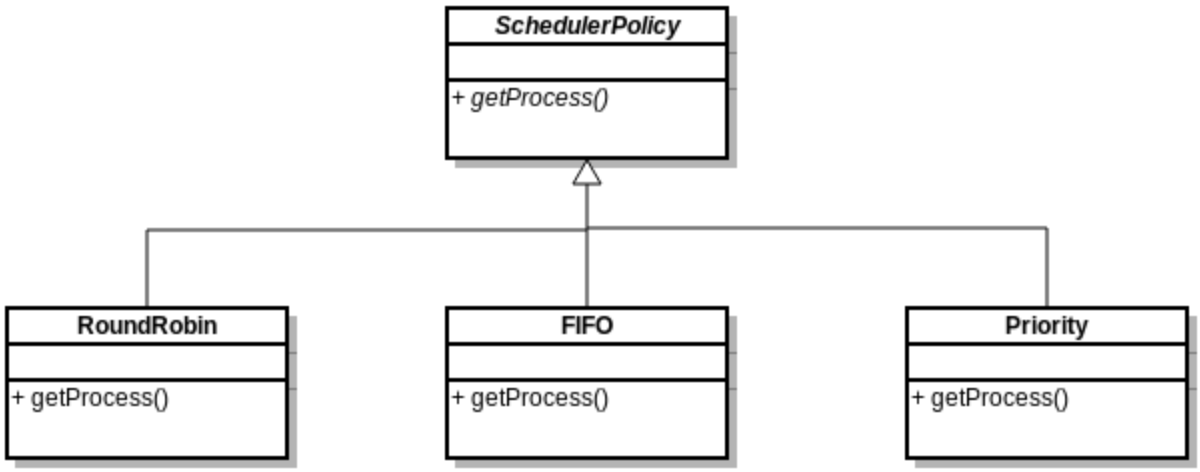


Ready Queue

La cola de listos, se decidió implementar simplemente como una cola “común”.

Scheduler Policy’s

Las políticas de planificación, se implementaron como un Patrón Strategy, donde el contexto asociado a la estrategia es el Kernel, y cada Estrategia/Política se implementó cambiando la forma en la que buscaban al “primer elemento de la cola”.



Interruption Handler

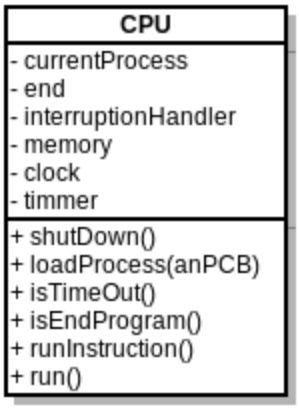
Es el manejador de interrupciones, y es la entidad encargada, de comunicar las interrupciones entre los componentes que interactúan, en este caso el Kernel y la CPU.

Teniendo conocimiento de las partes que se comunican en cada interrupción lanzada, el manejador, procede en pos ambas, y se encarga por ejemplo de efectuar los cambios de contexto, o de avisar la terminación de un proceso,etc.
Por esto mismo, también hace cosas como interrumpir la ejecución porque se ha pasado al modo Kernel, por lo que también desempeña tareas de concurrencia, más específicamente de sincronización, hace de semáforo.

Hardware

CPU

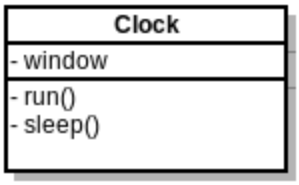
La CPU, es el hardware encargada de ejecutar instrucciones, más precisamente de realizar las operaciones aritméticas-lógicas. En este Tp, se decidió darle la siguiente forma, y de otorgarle determinada funcionalidad como ejecutar instrucciones, etc.



Clock:

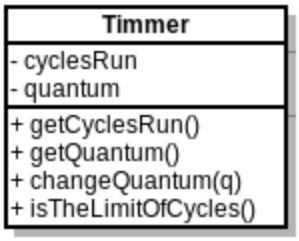
Representa a lo que físicamente es un oscilador, que básicamente genera una señal cuadrada, de dos estados “Alto” y “Bajo”, que representan la ventana de tiempo en la que la cpu puede trabajar.
Por cada tiempo **t** estado alto, hay un tiempo **t** de estado bajo, generando una señal asi:

ck ‾|_|‾|_|‾|_|‾|_|‾|_|‾|_|‾|_|‾|_|‾|_|‾|_|‾|_|‾|_|‾|_



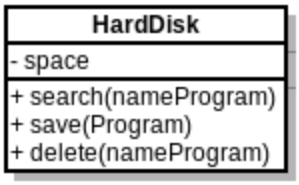
Timmer:

El timer por su parte, es la entidad que se encarga de contar la cantidad de ciclos que corre un programa cada vez que se le asigna la CPU, y además se encarga de proveer la interfaz, para verificar, de ser necesario, que cumple con el timeout, especificado.



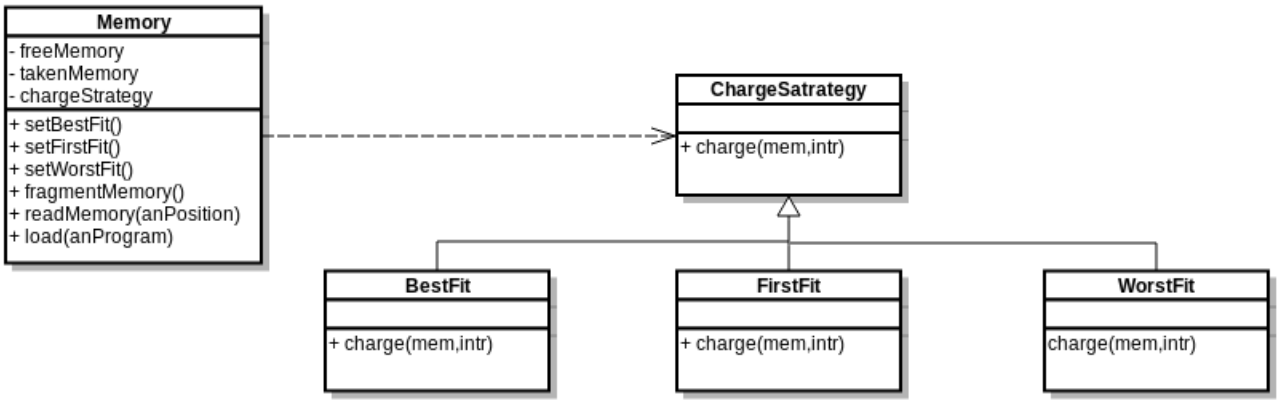
Hard Disk

El disco duro, se ha modelado de una forma simple, sin ningún sistema de archivos, ni nada complejo, simplemente funciona como un diccionario, asociando: *key → value,* donde la key, es el atributo name, de un programa, y el value es dicho programa.



Memory

La memoria, fue pensada a nivel lógico, y se implementó los diferentes manejos de memoria vistos, como son Primer Ajuste, Mejor Ajuste, Peor Ajuste y Paginación.
La estructura de la memoria esta dada de la siguiente forma:



Conclusiones

A lo largo del desarrollo del tp, nos hemos encontrado con diversos problemas tanto de diseño, como de puesta a punto del simulador, entre los Problemas más destacados se encuentran:

- Diseño e implementación de la CPU
- Diseño e implementación de Paginación
- Diseño e implementación del Interruption Handler, mas que nada por el manejo de concurrencia.

Pese a las dificultades que fueron surgiendo, creemos que se pudo llegar a buen puerto, y que si bien hay cosas que a la hora de implementar no se tuvieron en cuenta, como la optimización del modelo, etc, son cosas que al modelo conceptual del simulador no afectan.
Para cerrar, cabe destacar que la complejidad también está dada, en el hecho de los requerimientos, que no son claros hasta muy adentrado el cuatrimestre, y por ello, hubo muchas cosas que modificar, una vez visto todos los contenidos, porque como en todo sistema, sus diferentes componentes interactúan entre sí, es decir no son aislados.