El Planificador de Procesos a través de un Simulador

M. Barrionuevo, A. Apolloni, F. Piccoli

Universidad Nacional de San Luis
Ejército de los Andes 950
5700 - San Luis - Argentina
e-mail: {mdbarrio, rubenga, mpiccoli}@unsl.edu.ar

Resumen El Sistema Operativo permite simplificar la gestión de recursos. La administración eficiente de los recursos implica una cuidadosa planificación del uso de cada uno. El procesador o CPU es un recurso clave, su corecta planificación constituye uno de los puntos centrales en el diseño de un buen Sistema Operativo.
Comprender cada uno de los conceptos relacionados a los Sistemas Operativos y la interrelación de todas sus componentes, no es simple. Su estudio, generalmente se basa en el análisis de conceptos teóricos con prácticas de escritorio, cambiar estas prácticas por buenas prácticas de laboratorio constituyó el disparador de este trabajo.
Si bien existen numerosos simuladores de planificación de la CPU, estos no constituían una herramienta didáctica íntegra. El desarrollo de SPPP tiene como objetivo proveer una herramienta software para simular la planificación de los procesos, permitiendo la definición de las características del sistema, la selección de la polítca de planificación y la comparación del desempeño. En este trabajo se presentan las consideraciones principales tanto del diseño como de la implementación de SPPP.

1. Introducción

Los Sistemas Operativos(SO) proporcionan un entorno para facilitar la tarea de los usuarios. Dependiendo del punto de vista, los sistemas operativos pueden ser considerados una *máquina extendida* (desde el hardware) o un administrador de recursos (desde el usuario)[14]. Cualquiera sea la visión del SO, es clara su complejidad.

Si bien los SOs pueden tener diferentes estructuras, por su complejidad, generalmente son divididos en partes, las cuales se encargan de los distintos tipos de componentes, los más comunes son: el procesador, la memoria principal y secundaria, el sistema de Entrada/Salida(E/S) y los archivos.

La complejidad de un SO no queda sólo en su diseño ni en su implementación, sino también en la comprensión de su funcionamiento y sus características. Comprender los conceptos relacionados a los SOs y sus interrelaciones no es simple para los alumnos de tercer año de la Licenciatura en Ciencias de la Computación.

La enseñanza, generalmente se plantea en forma teórica, lo cual si bien es ampliamente tratado en varios libros[2,3,13,14,15], no resulta simple de entender sin una práctica intensiva. Una práctica adecuada para realizar las actividades derivadas de la teoría se puede lograr a través de simuladores de las distintas partes del SO. La mayor cantidad de simuladores disponible correponden al Administrador del Procesador, más específicamente del Planificador de Procesos. Entre los simuladores utilizados en las prácticas de la materia podemos mencionar a Planp[8], sim[12], SoSim[6], Ovsos[9], RCOS[4]. Si bien su inclusión tuvo resultados satisfactorios, ninguno de ellos se adapta de manera completa a lo que se pretende trabajar en la materia. Es por ello que se plantea la necesidad de desarrollar una aplicación, la cual simule integramente el trabajo del Planificador de Procesos bajo distintas cargas de trabajo, permitiendo al usuario(alumno) definir las características de Sistema y evaluar su desempeño. A través del uso de esta herramienta, el alumno podrá visualizar la importancia del Planificador de Procesos y su forma de trabajar según la carga del Sistema, las características de los procesos, las políticas y mecanismos de Planificación seleccionados.

Este trabajo se organiza de la siguiente manera, en la sección 2 se explican las características básicas de un SO. En la siguiente sección se detallan las caracerísticas del Administrador de Procesos y las características del Planificador. Finalmente se especifican las propiedades del simulador *SPPP* a desarrollar, presentando algunas interfases y detallando las consideraciones más importantes para su implementación.

2. SO: Características y Funciones

Es el SO el responsable de crear el vínculo entre los recursos(procesador, memoria, dispositivos, información), el usuario y las aplicaciones[2,3,13]. Son funciones del SO:

- Administrar el procesador: Distribuir el procesador entre los distintos procesos a fin de obtener una buena ejecución en el sistema. El SO es el responsable de: Crear y destruir los procesos, Pararlos y reanudarlos, y Ofrecer mecanismos para su comunicación y sincronización.
- Administrar la memoria: Gestionar el espacio de memoria asignado a cada proceso y a cada usuario. Cuando la memoria física es insuficiente, el SO será responsable de administrar la "*memoria secundaria*" o "*virtual*". El SO es el responsable de:
  - Para la Memoria Principal: Conocer qué partes de la memoria están utilizadas y por quién, Decidir qué procesos se cargarán en memoria cuando haya espacio disponible, y Asignar y reclamar espacio de memoria cuando sea necesario.
  - Para la Memoria Secundaria: Planificar el almacenamiento secundario, Gestionar el espacio libre, Asignar el almacenamiento a nuevos procesos, Mover bloques entre memoria principal y memoria secundaria, y Minimizar los movimientos de bloques entre memoria principal y secundaria.

- Gestionar las E/S: Unificar y controlar el acceso de los procesos a los recursos. El SO debe gestionar el almacenamiento temporal de E/S y atender las interrupciones de los dispositivos.
- Administrar la Seguridad: Garantizar el uso de recursos sólo por aquellos procesos que tengan las autorizaciones para hacerlo. El SO se encarga de: Distinguir entre uso autorizado y no autorizado, Especificar los controles de seguridad a realizar, y Forzar el uso de mecanismos de protección.
- Gestionar los Archivos: Manejar la lectura y escritura en el sistema de archivos y las autorizaciones de acceso a los archivos. El SO es responsable de: Construir y eliminar archivos y directorios, Ofrecer funciones para manipular archivos y directorios, Establecer la correspondencia entre archivos y unidades de almacenamiento, y Realizar copias de seguridad.

Además de su estructura, los SOs pueden tener diferentes características, se puede hablar de SOs con multiprogramación o sin ella, de tiempo compartido, monousuario o multiusuario, etc.. En un sistema multiprogramado, múltiples procesos están presentes en memoria principal al mismo tiempo, esto surgió con la idea de maximizar el uso de la CPU, manteniendo siempre algún proceso en ejecución. La propiedad de *Tiempo Compartido* fue concebida con la idea de conmutar o cambiar continuamente la CPU entre procesos de forma tal que todos los usuarios puedan interactuar con sus procesos mientras están ejecutando.

3. Administrador de Procesos

Uno de los componentes básicos de un SO es el Administrador de Procesos. Como se mencionó anteriormente es el responsable de administrar los procesos y sus actividades en el SO. Entre las tareas a realizar está la asignación del procesador a los distintos procesos en el sistema, conocida como Planificación de Procesos y a la que nos abocaremos en este trabajo. En esta sección se detallan las características básicas del Planificador de Procesos y las consideraciones a tener en cuenta a la hora de diseñarlo.

3.1. Procesos y Ciclo de Vida en un SO

Un proceso es una entidad activa y un programa es una entidad pasiva. Se define a un proceso como un programa en ejecución. Un programa puede generar varios procesos durante su ejecución.

Los procesos pueden tener diferente naturaleza dependiendo de las operaciones de E/S que realizan, si un proceso hace uso intensivo de E/S se lo denomina *I/O bound* o si hace uso intensivo de CPU, se lo conoce como *CPU bound*, una mezcla de ambos también es válida, por ejemplo un proceso *mix(K)* significa: El proceso tiene un $K\,\%$ de operaciones orientadas a CPU, el resto, $(100 - K)\,\%$, son operaciones de E/S.

En un sistema de computadora, el procesador (CPU) ejecuta, en un período de tiempo, una gran cantidad de procesos. Durante su ejecución, un procesos

necesita recursos: tiempo del procesador, memoria, archivos, dispositivos de E/S. La asignación de estos recursos puede darse al inicio de la ejecución o durante la misma. Si el sistema es multiprogramado, varios procesos están en ejecución al mismo tiempo y la CPU se alterna entre ellos.

La figura 1 muestra el ciclo de vida de un proceso en el SO. Cuando un proceso entra al sistema (**T1**) es colocado en una cola de trabajos. Una vez que el proceso está listo para ejecutar, es colocado en la cola de procesos listos (*ready*). Cuando al proceso se le asigna la CPU (**T2**), ejecuta por un tiempo y eventualmente el proceso terminará (**T4**), será interrumpido (**T3**) o esperará por la ocurrencia de algún evento (**T5**). Cumplido el evento, el proceso pasa nuevamente a estar listo para ejecutar (**T6**). Cuando existe más de un proceso listo para ejecutar, el sistema debe decidir a cuál le asignará la CPU. La planificación de procesos busca: Garantizarle a cada proceso una proporción justa de tiempo de CPU, Maximizar el número de tareas procesadas y el uso del procesador, minimizando su tiempo ocioso y el tiempo de respuesta de los procesos.

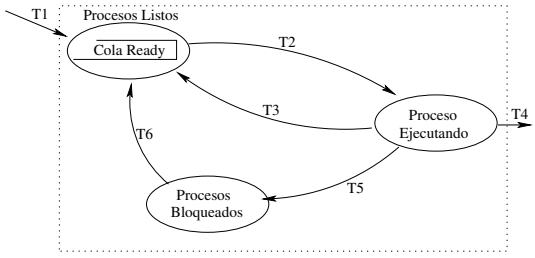

Figura 1: Diagrama de Estados de un Proceso

La decisión la toma el *Planificador* en función de un mecanismo de selección, *Algoritmo de Planificación*, quien implementa una política. Además existen diferentes niveles de planificación, los cuales son explicados en la siguiente sección

3.2. Niveles de planificación - Tipos de Planificadores

La planificación se hace en cuatro instantes de tiempo, una de ellas no la hace el SO, sino el usuario, quien establece como iniciará sus procesos[2]. Los tres niveles de planificación que nos interesan y corresponden al SO se detallan a continuación.

Planificación a Largo Plazo El planificador a largo plazo, *Scheduler*, se encarga de organizar la ejecución ordenada y eficiente de los trabajos con un adecuado planeamiento de recursos considerando la modalidad de procesamiento. Este planificador se ejecuta con poca frecuencia, sólo cuando se necesita crear un nuevo proceso en el sistema (**T1** de figura 1), cuando termina un proceso (**T4** de la misma figura) o ingresa un usuario en el sistema. Alguna de sus tareas son: Mantener un registro de estado de todos los trabajos en el sistema, Asignar recursos (memoria, dispositivos, procesador, etc.) a cada trabajo, Pedir/Recuperar los recursos, Aceptar nuevos trabajos, entre otros. Además, es el responsable de controlar el nivel de multiprogramación del sistema y el balance de carga del mismo.

Planificación a Mediano Plazo Es el que decide sacar de memoria principal y llevar a disco (swap-out) a aquellos procesos inactivos o a los activos cuyo estado sea bloqueado temporal o momentáneamente para luego, cuando desaparezca la causa del bloqueo, traerlos nuevamente a memoria (swap-in) y continuar su ejecución. Este planificador puede utilizarse cuando existe memoria disponible, el número de procesos en el sistema es bajo o hay muchos procesos bloqueados.

Planificación a Corto Plazo También llamado *low scheduler*, es el responsable de decidir quién, cuándo, cómo y por cuánto tiempo recibe el procesador un proceso listo (en la *cola ready*) para ejecutar.

El planificador a corto plazo es invocado cada vez que un suceso (interno o externo) modifica el estado global del sistema: finalización del tiempo asignado de CPU, realización de una operación de E/S, entre otros.

Para mantener el rendimiento del SO, este planificador debe ser rápido y no generar carga extra al procesador.

4. Simulador del Planificador de Procesos: *SPPP*

Por todo lo expuesto, se puede observar que el Planificador de Procesos tiene una función importante en los SOs, su buen desempeño permitirá entre otras, el máximo aprovechamiento de la CPU, y en consecuencia un rendimiento óptimo. Esta propiedad es la que nos interesa que el alumno comprenda a través de prácticas de laboratorio. Es por ello que proponemos un **S**imulador del **P**lanificador de **P**rocesos, *SPPP*.

El diseño e implementación de *SPPP* no es una tarea trivial. *SPPP* implica varias tareas, su desarrollo puede ser hecho en diferentes etapas, las cuales en primer instancia coinciden con los diferentes niveles de planificación. En una primer etapa, proponemos desarrollar el planificador de bajo nivel y realizar la simulación de la asignación de la CPU a los procesos activos en el SO.

Para trabajar, SPPP necesita establecer el estado del sistema:

- Cuántos procesos y de qué tipo ejecutarán concurrentemente en el SO,
- Cuál política de planificación utilizará.
- Cuánto tiempo se asignará la CPU a un proceso, ráfaga de CPU.

Una vez estalecido el ambiente del SO, los procesos inician su ejecución en el momento de creación asignado. A partir de su ingreso al sistema, el proceso compite por el procesador según el Planificador seleccionado. Para concluir su ejecución, cada proceso debe consumir el total de ráfagas de CPU asignada al momento de su creación, como así también permanecer bloqueado por E/S el tiempo indicado.

4.1. Políticas de Planificación

Existen numerosas políticas de asignación de la CPU a los procesos listo para ejecutar, esperando en la cola *ready*[2,3,13,14,15]. Las políticas pueden ser

clasificadas según el esquema de asignación de la CPU a los distintos procesos, por ello se pueden tener dos categorías generales de algoritmos:

- *Nonpreemptive scheduling* (*Apropiativo*) Una vez asignada la CPU a un proceso, éste continúa ejecutando hasta su finalización.
- *Preemptive scheduling* (*No Apropiativo*) El SO puede interrumpir la ejecución del proceso actual y desplazarlo a la cola ready.

SPPP permitirá seleccionar entre las políticas más conocidas, ellas son:

- *Primero en llegar, Primero en ser Servido* (FCFS): Algoritmo Apropiativo. Es el más sencillo, se basa en la filosofía: primer proceso en solicitar la CPU es el primero en recibirla.
- *Round Robin*(RR): Algoritmo No Apropiativo. La CPU se asigna un período de tiempo(Quantum) a cada proceso en la cola ready. Si el proceso no finaliza en dicho intervalo de tiempo vuelve al final de la cola ready como proceso listo para ejecutrar.
- *Primero el proceso más corto* (SPF Shortest process first): Algoritmo Apropiativo. Selecciona al proceso con menor tiempo de ejecución esperado. Una versión No Apropiativa es la política de *Menor tiempo restante* (SRT Shortest remaining time first), la cual elige el proceso con menor tiempo esperado de ejecución restante.
- *Planificación con múltiples colas*: La cola ready es dividida en varias colas, los procesos son asignados a alguna de ellas. Cada cola tiene su propio algoritmo de planificación y hay una planificación entre las colas. Existen diferentes variantes de esta planificacón, entre ellas están: *Multiples niveles de colas fijas* (Cada proceso es asignado siempre a la misma cola, según alguna propiedad), *Múltiples niveles de colas Dinámicas* (Idem al anterior pero los procesos pueden cambiar de cola durante su ejecución) y *Múltiples niveles de colas con realimentación*(Al llegar un proceso es ubicado en la cola de mayor prioridad, cada vez que ejecuta y agota el Quantum asignado, el proceso es colocado en la cola de prioridad siguiente menor, así sucesivamente hasta llegar a la última cola en donde permanece hasta finalizar. Las colas de los niveles superior se comportan según la política FCFS, no así la última la que aplica el algoritmo RR)

El alumno podrá seleccionar uno de los algoritmos de planificación, inicializando los parámetros requeridos por el algoritmo seleccionado, por ejemplo en el caso de RR el tamaño del Quantum. Una vez definida y luego de la simulación, podrá evaluar el comportamiento del SO y compararlo con otros algoritmos que se estén considerando. En la siguiente sección se detallan los criterios de evaluación a considerar por el alumno a la hora de evaluar los algoritmos de planificación.

4.2. Criterios para la Evaluación

El objetivo de planificar los procesos es: Asignar tiempo de CPU para optimizar ciertos aspectos del sistema. Para analizar el desempeño de un SO según un

algoritmo de planificación, necesitamos algún criterio de comparación y recolección de datos estadísticos[3]. Los criterios se pueden dividir en orientados a los usuarios y orientados al sistema dependiendo de:

1. Si interesa analizar el comportamiento del sistema tal como lo perciben los usuarios, los parámetros a analizar son:
   - Tiempo de Respuesta: Mide el tiempo entre la emisión de una solicitud y la obtención de la primer respuesta. Tiempo que toma en contestar.
   - Turnaround Time: Mide cuánto tiempo demora en ejecutar un proceso, incluyendo el tiempo que el proceso se demora en ingresar al sistema y los que permanece ocioso.
   - Tiempo de Espera (Waiting Time). Mide el tiempo que el proceso está en la cola ready.
2. Si interesa el uso efectivo y eficiente del procesador, los parámetros a tener en cuenta son:
   - Eficiencia en el uso de los recursos, por ejemplo ocupación de la CPU.
   - Porcentaje de Utilización de la CPU por procesos de usuario.
   - Throughput o Productividad: Número de procesos completados por unidad de tiempo.

A través de la selección de una política, el alumno prodrá analizar la optimización de todos los criterios aunque es difícil satisfacerlo a todos al mismo tiempo, algunos son contradictorios entre si, por ejemplo tiempo de respuesta y throughput.

4.3. Modelo de Simulación

La simulación de sistemas implica la construcción de modelos. Su objetivo es averiguar qué pasa en el sistema bajo determinadas hipótesis, es decir, mostrar qué sucederá en un sistema real ante determinados cambios y ciertas condiciones [5,10].

La simulación del planificador involucra dos faces, la primera es la construcción del modelo y la segunda, el ensayo de diversas alternativas a fin de elegir y adoptar la mejor en el sistema real, procurando ser la óptima.

La face de construcción del modelo implica recrear un sistema de computación. A través de estructuras de datos se representan los principales componentes del sistema[1]. En la estructura del modelo se definen:

- Entidades permanentes para:
  - El reloj del Sistema.
  - El procesador.
  - La cola de procesos *ready*.
  - La cola de procesos bloqueados o en espera de una E/S.
- Entidades provisorias: Una para cada proceso en el sistema.
- Los eventos que provocan los cambios de estado según Figura 1:
  - Solicitud de E/S(T5).
  - Agotamiento del Quantum(T3 y T2).

- Finalización del evento de E/S (T6).
- Ingreso o finalización de un proceso (T1 y T4).

Conforme transcurre la simulación, se recolectan los datos estadísticos para evaluar el desempeño del sistema.

La simulación permitirá al alumno ensayar diversas alternativas para sistemas con características controladas, con el fin de seleccionar la mejor planificación para un sistema real. La experiencia permitirá tomar la decisión óptima o lo suficientemente aproximada.

4.4. Aspectos de Implementación

SPPP será un simulador de código libre. El simulador cuenta con una interfase, la cual está compuesta por tres partes: Configuración, Visualización de la Simulación y Estadisticas. Cada una de ellas tiene las siguientes características:

- *Interfase de Configuración*: La figura 2 muestra un prototipo de la Interfase. A través de ella se podrán inicializar los parámetros necesarios para establecer las características del sistema y de la política a simular. Por ejemplo, si la politica elegida es RR es necesario establecer el tamaño del Quantum. Dentro de los datos de configuración del sistema se encontrará la cantidad de procesos a crear, el tipo de proceso (si es I/O Bound o CPU Bound), la cantidad de ráfagas de CPU de cada proceso, el tiempo de E/S por proceso y el tiempo de simulación.

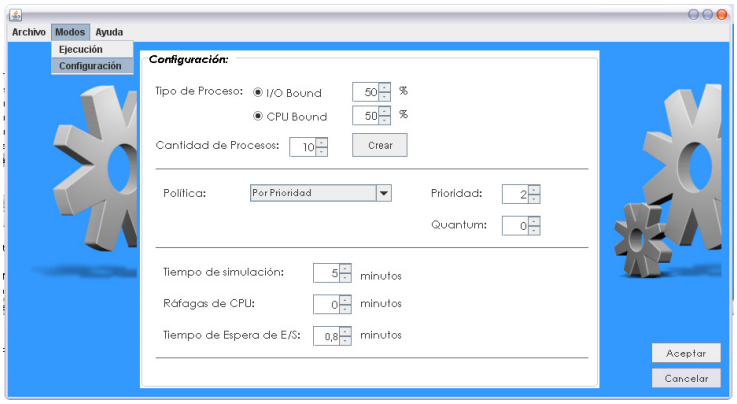

Figura 2. Interfase de Configuración

- *Interfase de Visualización de la Simulación*: Recreará el estado del sistema durante el transcurso de la simulación. La figura 3 muestra una instantánea de la simulación. Como puede observarse, se podrá ver cómo los procesos

cambian de estado pasando de la cola ready al procesador y de allí nuevamente a la cola ready(por agotar el Quantum), a la cola de procesos bloqueados(por E/S) o salir del sistema (por finalización). Si el proceso regresa a la cola ready, se ubicará en la posición determinada por la política, ahora si está bloqueado por E/S, permanecerá en ese estado hasta finalizar el tiempo de bloqueo indicado y retornar a la cola ready. Esta interfase permitirá seguir el ciclo de vida de los procesos en el sistema.

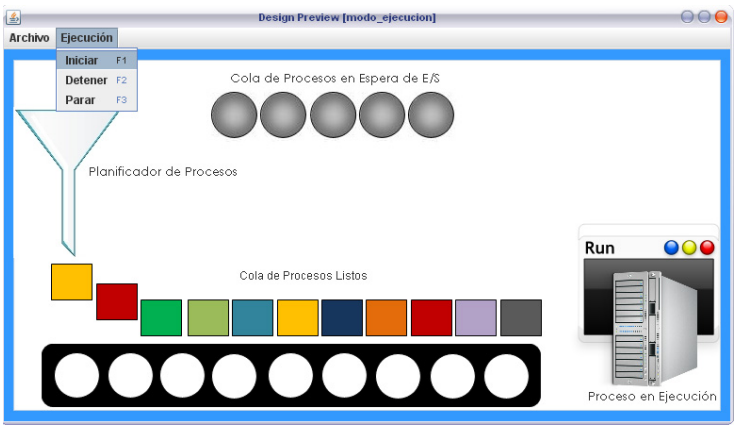

Figura 3. Interfase de Ejecución

- *Interfase de Estadisticas*: Permitirá extraer los resultados de la simulación. En ella se mostrará toda la información relevante de la ejecución de los procesos con la politica elegida y las condiciones establecidas en la configuración del sistema. Además permitirá realizar comparaciones de distintas políticas y graficar la información. Los resultados de las simulaciones se mostrarán siguiendo los criterios de evaluación enunciados anteriormente o sólo aquellos solicitados por el usuario (alumno).

Además de ser de código libre, buscamos desarrollar una herramienta portable a distintas plataformas de Hardware y Software. Si bien existen diferentes lenguajes de programación adecuado para la implementación de SPPP, la decisión se basó en características tales como licencia, portabilidad, velocidad de desarrollo, independencia de la plataforma de ejecución, etc. El lenguaje que cumple con las características deseadas es Python[7,11,16].

¿Por qué usar Python? Python es un lenguaje de programación orientado a objeto que puede ser usado para muchos tipos de desarrollo de software. Es *open source*, ofrece un fuerte soporte para la integración con otros lenguajes y herramientas, posee amplias bibliotecas y es fácil de aprender. Además es un lenguaje de programación interpretado, tiene un modo interactivo de programación, es multiplataforma e independiente del paradigma, entre otras.

5. Conclusiones

En este trabajo se presentaron las caraterísticas y condiciones básicas que tendrá SPPP, el Simulador del Planificador de Procesos, portable, de software libre y multiplataforma.

SPPP permitirá presentar, estudiar y comprender el planificador de procesos en un sistema con condiciones reguladas. Constituirá un buen recurso didáctico para los docentes y una herramienta útil para los alumnos, quienes podrán recrear sistemas con características controladas, visualizar y analizar el comportamiento del sistema según el planificador seleccionado, realizar comparaciones, elaborar conclusiones y, fundamentalmente, entender el funcionamiento de los diferentes algoritmos de planificación de los procesos desde una práctica amigable.

Referencias

1. J. Banks, J. Carson, B.L. Nelson, D. Nicol. Discrete-Event System Simulation, 4/E. Prentice Hall (2004). ISBN: 0131446797.
2. H.M. Deitel, P.J. Deitel, D.R. Choffnes. Operating Systems, 3/E. Prentice Hall (2003). ISBN: 0131828274.
3. P.B. Galvin, G. Gagne, A. Silberschatz. Fundamentos de Sistemas Operativos, 7/E. McGraw-Hill (2006). ISBN: 8448146417
4. D. Jones, A. Newman. A constructivist-based tools for operating systems education, Proceedings of EdMedia'2002, Denver, Colorado, Jun. 2002.
5. A.M. Law. Simulation Modeling and Analysis, 4/E. McGraw-Hill (2007). ISBN: 0073294411.
6. L.P. Maia, A.C. Pacheco. A Simulator Supporting Lectures on Operating Systems. 33rd ASEE/IEEE Frontiers In Education Conference 2003 Boulder, Colorado.
7. A. Martelli. Python. Guía de referencia. Anaya Multimedia/O'Reilly. ISBN: 9788441523173.
8. H.M. Mérida. Planp: Herramienta para la simulación de políticas de planificación de procesos. Simulador de planificación de CPU
9. OVSOS: Other Visual Simulation of an O.S. http://sourceforge.net/projects/ovsos/
10. J.A. Payne. Introduction to Simulation Programming Techniques and Methods of Analysis. McGraw-Hill College (1982). ISBN: 9780070489455.
11. B. Rempt. Gui Programming With Python: Using the Qt Toolkit. Opendocs Llc; Bk&CD-Rom (2002). ISBN: 0970033044.
12. Software para la Simulación de un Planificador de Procesos. Escuela Universitaria Politécnica de Teruel, Universidad de Zaragoza. España. http://eupt2.unizar.es/asignaturas/itig/sistemas_operativos_1/PFC.htm
13. W. Stallings. Operating Systems: Internals and Design Principles, 4/E. Prentice Hall (2001). ISBN: 0130319996.
14. A.S. Tanenbaum. Modern Operating Systems, 3/E. Prentice Hall (2007). ISBN: 0136006639.
15. A.S. Tanenbaum, A.S Woodhull. Operating Systems Design and Implementation, 3/E. Prentice Hall (2006). ISBN: 0131429388.
16. T. Ziadé. Expert Python Programming. Packt Publishing (2008). ISBN: 9781847194947.