Centro Universitario de Ciencias Exactas e Ingenierías.

Departamento de Ciencias Computacionales.

Materia:

Sistemas Operativos.

Profesora:

Becerra Velázquez Violeta del Rocío.

Alumno:

Maldonado Melendez Diego Alberto.

Código:

221977845.

Carrera:

Ingeniería en Computación.

Sección:

D04

Título de la investigación:

Algoritmos de Planificación.

Fecha:

17 de septiembre de 2023.

Índice

Tabla de Imágenes.

1. Formas para la Creación y Terminación de procesos.

La creación y eliminación de procesos es algo necesario para un sistema operativo. En sistemas sencillos se puede tener un registro y control de los procesos que se realizarán. No obstante, en sistemas de propósito general es más complejo, ya que son necesarias algunas maneras de creación y terminación de procesos, dependiendo de lo que demande la operación. Hay cuatro eventos que pueden crear un proceso; el arranque del sistema, la ejecución de una llamada al sistema para la creación de algún proceso, petición del usuario para crear un proceso, o el inicio de un trabajo por lotes.

Es común que cuando se enciende el sistema operativo, varios procesos son creados. Los procesos que interactúan con el usuario se consideran en primer plano, ya que realizan trabajos importantes para quien utiliza la máquina. Aquellos procesos que no se asocian con los usuarios se consideran en segundo plano, estos son importantes ya que, aunque no sean importantes para quien use el dispositivo, están en contacto con alguna función específica importante. Los sistemas se conforman de una gran cantidad de procesos en segundo plano.

No obstante, el arranque no es el único momento en donde se crean procesos, ya que esto limitaría mucho el uso de los computadores. Un proceso que está siendo ejecutado tiene la capacidad de llamar al sistema para crear uno o más procesos nuevos, para que sirvan de complemento en la tarea que se esté realizando. Igualmente, en los sistemas operativos más comunes, los usuarios pueden ejecutar un programa haciendo doble clic en el ícono de éste. Esta simple acción es capaz de iniciar un proceso.

Por último, los usuarios, en sistemas de procesamiento por lotes que se encuentran en las mainframes grandes, tienen la posibilidad de enviar trabajos de procesamiento por lotes al sistema. Cuando el sistema finaliza con un trabajo y libera los recursos necesarios, éste es capaz de crear un proceso y ejecutar el siguiente trabajo. Un común denominador en la creación de los procesos es que uno ya existente suele llamar al sistema de creación de proceso. Esta llamada es la que indica al sistema operativo que cree un nuevo proceso, además de indicar qué programa debe realizarlo.

El proceso de finalización es también sumamente necesario. Cada proceso creado, tarde o temprano, terminará. Algunas de las razones por las que puede finalizar un proceso son; salida normal, salida por error, error fatal o que sea eliminado por otro proceso. Usualmente los procesos terminan una vez concluye su labor, realizando una llamada al sistema que indica al sistema operativo que el proceso acabó. Ahora bien, no siempre es el caso. Hay ocasiones en las que el proceso descubre algún error, como que el usuario está intentando ejecutar un programa, pero dicho programa no existe.

Una vez un programa está en ejecución, también es posible que ocurra un error, siendo ésta otra posible razón para la finalización de un proceso. Este caso suele darse debido a algún error en el programa, como ejecutar alguna instrucción ilegal o hacer referencia a una sección de memoria prohibida o inexistente. Por último, un proceso podría ejecutar una llamada al

sistema que indique al sistema operativo que elimine otros procesos. El proceso eliminador debe poseer la autorización para cumplir esta hazaña. En algunos sistemas, cuando un proceso termina, todos los procesos creados a raíz de él también lo hacen, pero en sistemas como Windows o Unix, esto no ocurre así.

2. Diagrama de 5 estados.

Una cola es una estructura de datos del estilo First-in-First-out, donde el primer dato que ingresó es el primero que saldrá. Los procesos trabajan en una cola de procesos, donde a cada proceso se le asigna una cantidad de tiempo específica para ejecutarse y posteriormente volver a la cola, a menos de que sea bloqueado. No obstante, hay muchos procesos que pueden estar listos para ser ejecutados, mientras que otros pueden estar bloqueados esperando a alguna operación de ingreso o salida de datos para continuar. Es por ello por lo que el utilizar una única cola es ineficiente, ya que es complicado buscar el proceso que no está bloqueado y que ha estado durante una mayor cantidad de tiempo en la lista.

Es por ello por lo que surgió la concepción de dividir los procesos en dos, los que están listos para ser ejecutados y los que están bloqueados. Por consecuente, se creó un diagrama de estados que consta de los siguientes:

- En ejecución: Como su nombre lo indica, se trata del proceso que está siendo ejecutado.
- Listo: Es aquel proceso que se encuentre preparado para ser ejecutado cuando tenga la oportunidad.
- Bloqueado: Es un proceso que requiere que algún otro evento ocurra para poder ser ejecutado.
- Nuevo: Es un proceso que fue creado recientemente, aunque no haya sido admitido al conjunto de los procesos ejecutables por el sistema operativo. Usualmente, un nuevo proceso no ha sido cargado en la memoria principal.
- Terminado: Proceso que ha sido liberado del conjunto de procesos ejecutables por el sistema operativo.

Los estados de nuevo y salida son bastante útiles para la administración de procesos. Un programa puede requerir información acerca de la historia de un proceso, para posibles efectos de desempeño o análisis. Una vez el programa obtiene esta información, el sistema operativo no requiere mantenerla, por lo que es eliminada. Los procesos también cuentan con posibles transiciones, que son las siguientes:

- Null → Nuevo: Un nuevo proceso es creado para ejecutar algún programa.
- Nuevo → Listo: El sistema operativo es capaz de mover un proceso de nuevo a listo cuando está preparado. Los sistemas suelen asignar un límite basado en los procesos existentes o la cantidad de memorial virtual asignada a procesos existentes, con la intención de que no haya muchos procesos activos que puedan empeorar el desempeño.

- Listo → En ejecución: El sistema operativo selecciona alguno de los procesos que están en la lista de Listos, cuando es el momento apropiado. Este trabajo es realizado por el planificador.
- En ejecución → Terminado: El sistema operativo finaliza el proceso actual, en caso de que éste haya finalizado o haya sido abortado.
- En ejecución → Listo: Se da principalmente en caso de que el proceso en ejecución alcanzó el tiempo máximo que tiene para ejecución ininterrumpida. Existen algunas otras posibles causas para que se de esta transición, pero no todas están implementadas en la mayoría de los sistemas operativos. Un caso importante es cuando el sistema operativo asigna diferentes jerarquías a los procesos.
- En ejecución → Bloqueado: Un proceso permuta a bloqueado si requiere de algo por lo que debe esperar. Un requerimiento suele darse como una llamada de servicio del sistema, como cuando un programa en ejecución llama a un procedimiento que es parte del sistema operativo. Puede requerir algún recurso como un archivo o sección de una memoria virtual, y estos pueden no estar disponibles. Cuando hay comunicación de procesos, un proceso puede ser bloqueado mientras espera a que otro provee alguna información o mensaje.
- Bloqueado → Listo: Un proceso Bloqueado puede pasar a Listo cuando el evento que estaba esperando ya ocurrió.
- Listo → Terminado: En algunos sistemas, una tarea padre puede finalizar un proceso hijo en cualquier momento, al igual que si un proceso padre termina, los procesos hijos pueden terminar.
- Bloqueado → Terminado: Se comporta igual que la transición de Listo a Salida.

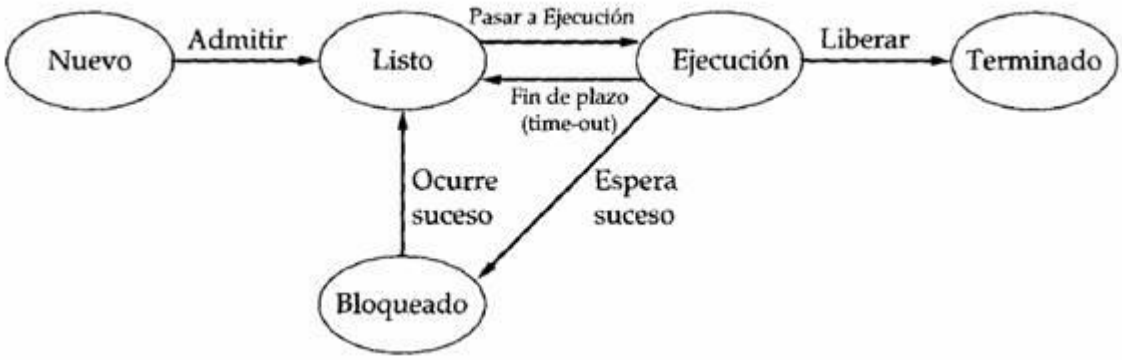

Figura 1. Diagrama de 5 estados.

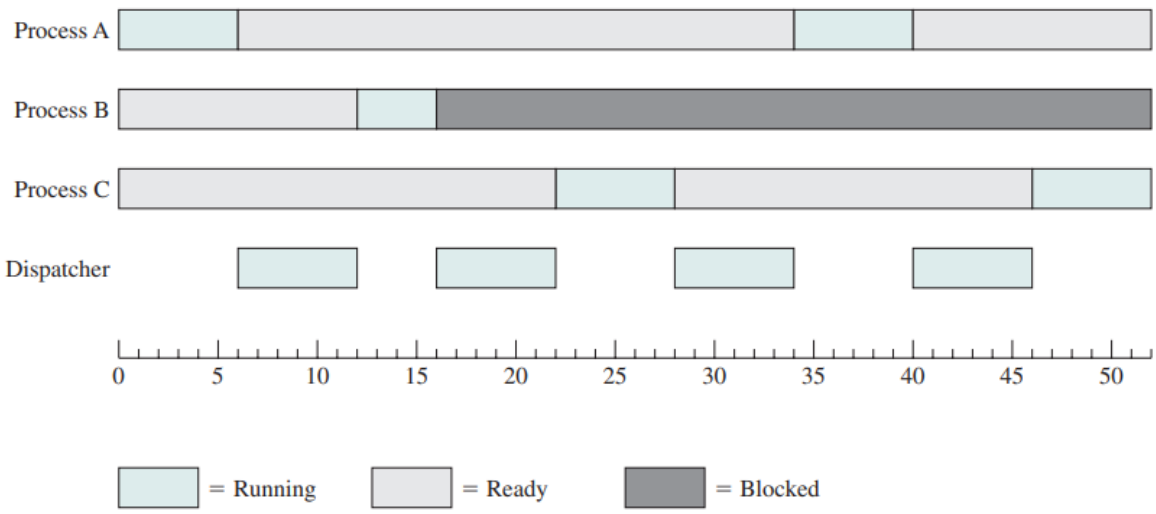

Figura 2. Ejemplo de estados de procesos.

3. Algoritmos de Planificación.

Es común que, cuando una computadora posee características de multiprogramación, existan varios procesos o hilos que demandan de recursos de procesador al mismo momento, caso que acontece cuando más de un proceso están en "listo". El problema llega cuando solo hay un CPU capaz de realizar los procesos, se requiere decidir cuál será el que se ejecutará a continuación. Los sistemas operativos tienen un apartado llamado planificador de procesos, y el método que éste emplea para decidir es conocido como algoritmo de planificación.

Los algoritmos de planificación son métodos empleados para encontrar determinar el orden con el que se ejecutarán ciertas tareas, además de distribuir los recursos con la finalidad de cumplir algún objetivo. El uso de recursos es una de las metas que tienen este tipo de objetivos, ya que buscan optimizarlos. Otros objetivos es el maximizar el rendimiento de los dispositivos y disminuir tiempos de espera.

3.1 Políticas de planificación.

Los algoritmos de planificación basan sus políticas en la manera en la que manejan las interrupciones de reloj. La primera política es la *No apropiativa*. Los algoritmos no apropiativos seleccionan algún proceso para ejecutarlos, para posteriormente dejar que se ejecute hasta que el mismo proceso se bloquee, principalmente por alguna espera a algún otro proceso, o a operaciones de entrada y salida. Además de bloquearse, el proceso también puede liberar recursos de CPU de manera voluntaria. Esta política no toma decisiones de planificación durante

interrupciones de reloj. Una vez finalizada la interrupción de reloj, la ejecución del proceso se reanuda.

Un algoritmo de planificación con política *Apropiativa* tiene la capacidad de seleccionar algún proceso y determinar que se ejecute por un máximo de tiempo fijo. Una vez finalice este tiempo fijo, aunque siga en ejecución, el planificador de procesos lo suspende y selecciona algún otro proceso disponible para ejecutarlo. Para esta política se requiere una interrupción de reloj en el final del tiempo máximo, a modo que la CPU retorne el control al planificador. Esta política tiene la ventaja de evitar que algún proceso monopolice los recursos del procesador, teniendo así un mejor tiempo de respuesta.

3.2 Tabla comparativa algoritmos de clasificación.

Algoritmo	Descripción	Política	Ventaja	Desventaja
RR o Round Robin	Cada proceso recibe un intervalo de tiempo, llamado quántum, durante el cual se ejecuta. En caso de que no haya finalizado la ejecución al terminar el quántum, la CPU pasa a otro proceso. Una vez un trabajo utiliza su quántum, pasa al final de la lista. Un aspecto importante es la duración del quántum, Si el quántum es corto, hay muchos cambios entre procesos, reduciendo la eficiencia, en cambio si es largo, las respuestas interactivas cortas pueden ofrecer resultados erróneos.	Apropiativa	• Simpleza. • Es equitativo. • Optimiza la utilización de la CPU.	• Suele tener tiempos de retorno altos. • El rendimiento suele ser variable. • Provee baja productividad en ocasiones.
FCFS o First-Come, First-Served	La Unidad Central de Procesamiento asigna el orden en de los procesos tomando en cuenta cuándo la solicitaron. Existe una única cola de procesos listos. La primera tarea que ingresa inicia su ejecución y tiene	No apropiativa	• Es fácil de comprender y programar. • Es equitativo. • Optimiza la utilización de la CPU.	• Posee tiempos de respuesta altos para procesos largos. • Tiene un rendimiento variable.

	todo el tiempo que necesite para ejecutarse. Los siguientes trabajos que estén en "listo", se colocan al final de la cola. Si el proceso en ejecución se bloquea, el que esté en la cima de la cola se ejecutará, mientras que, cuando el proceso bloqueado pase a listo, éste se irá al final de la cola.			
SRT o Shortest Remaining Time	El planificador de procesos suele seleccionar aquel proceso cuyo tiempo restante de ejecución sea menor. Al momento de ingresar una nueva tarea, se compara su tiempo total con el restante del proceso en ejecución. En caso de que el nuevo proceso tenga un tiempo total menor, se suspende el que está en ejecución y se inicia el nuevo proceso.	Apropiativa	• Trabajos cortos nuevos tienen un buen servicio. • Se aumenta el rendimiento. • Minimiza tiempos de espera	• Puede causar inanición en caso de que se agreguen nuevos procesos cortos constantemente. • Es complejo los cálculos de tiempo restante.
SJF o Shortest Job First	En ocasiones en las que varios procesos tienen igual prioridad, estando todos en estado "listo", se selecciona el trabajo más corto.	No apropiativa	• Reduce tiempos de espera. • Provee eficiencia en el procesamiento.	• Es óptimo cuando todos los trabajos están en "listo" al mismo tiempo. • Puede producir inanición si llegan muchos procesos cortos.

Prioridades	A cada proceso le corresponde una prioridad, el proceso en estado de "listo" con una prioridad más alta es el que se ejecutará. Es importante evitar que los procesos de alta prioridad se ejecuten todo el tiempo, por lo que el planificador de procesos puede reducir prioridades tomando en cuenta pulsos de reloj. A las prioridades se le pueden asignar procesos estática o dinámicamente. De manera estática, un usuario asigna la prioridad de manera voluntaria dependiendo de la importancia que les dé a los procesos. Por otro lado, la asignación dinámica puede cambiar las propiedades en respuesta a eventos, como puede ser procesos limitados a entradas y salidas.	Apropiativa /No Apropiativa	• Ayuda a que procesos urgentes puedan ser ejecutados primero. • Útil cuando se requiere respuesta rápida. • Permite la personalización para que se adapte a las necesidades del usuario.	• Los procesos de poca prioridad pueden sufrir inanición. • Pueden asignarse prioridades incorrectas, lo que puede provocar retrasaos y monopolización de recursos. • Es complejo el asignar prioridades. • Puede ser difícil la implementación y la demanda de recursos.
Colas múltiples	Es necesario un algoritmo que pueda distinguir entre distintos tipos de procesos, siendo los dos tipos principales los interactivos y los lotes. Ambos tipos de procesos se comportan de manera distinta, requiriendo así diferentes tiempos de respuestas y de necesidades. Este algoritmo divide la cola de procesos en estado	Apropiativa /No Apropiativa	• Permite distinguir los diferentes tipos de trabajos que existen, pudiendo tener prioridades. • Mejora la eficiencia del sistema.	• Nuevamente puede costar la asignación de prioridades. • Puede ser difícil de implementar.

	"listo", en nuevas colas. Cada nuevo proceso es asignado a una cola, tomando en cuenta alguna propiedad como tipo o prioridad. Cada cola puede poseer algún algoritmo distinto. Existe otro algoritmo llamado colas múltiples realimentadas, el cual permite que los procesos permuten entre colas.			
Planificación garantizada	Se realizan promesas reales a los usuarios acerca del rendimiento, para posteriormente cumplirlas. Para cumplir estas promesas, el sistema debe tener en cuenta la potencia que la CPU ha destinado a cada proceso desde que fueron creados.	Apropiativa	• Es justo. • Mejora la eficiencia del sistema. • Optimiza los recursos del CPU.	• Es complejo el determinar la cantidad de recursos que cada proceso pueda necesitar para cumplir sus promesas.
Planificación por Partes Equitativas	En algunos procesos se toma en cuenta el usuario que lo creó. Cada usuario posee una cierta fracción de la CPU, mientras que el planificador de procesos selecciona alguno que cumpla con este modelo. Cada usuario recibe una parte equitativa de los recursos disponible.	Apropiativa	• Evita que los recursos sean acaparados. • Hay equidad para el acceso a los recursos, importante para entornos compartidos.	• Complicado la asignación de cuotas equitativas. • No presta atención a necesidades críticas.

Tabla 1. Algoritmos de planificación.

3.3 Elementos de la tabla de procesos.

Un BCP es una estructura de datos que almacena la información sobre procesos. Es un registro especial que emplean los sistemas operativos. Las siglas BCP hacen referencia a Bloque

de Control del Procesos, o en inglés, Process Control Block (PCB). Un BCP tiene como objetivo localizar la información sobre el proceso por parte del sistema operativo, además de mantener registrados los datos del proceso en caso de tener que suspender temporalmente la ejecución, sabiendo que puede reanudarla después. Un BCP almacena información como:

- Identificador: Es único, y se le asocia al proceso, distinguiéndolo de otros.
- Estado: En caso de que esté siendo ejecutado, se encuentra en estado de ejecución.
- Prioridad: En comparación a otros procesos cada proceso tiene una prioridad.
- Contador del programa: La dirección de la siguiente instrucción a ser ejecutada.
- Apuntadores de memoria: Apuntadores que guían al código del programa y a los datos asociados al proceso, además de cualquier bloque de memoria compartido con otros trabajos.
- Datos de contexto: Información que está presente en registros en el procesador mientras se ejecuta el proceso.
- Información de estatus de entradas y salidas: Incluyendo dispositivos de entradas y salidas o peticiones de entradas y salidas.
- Información de contabilidad: Tiempo de CPU utilizado.

Esta información se suele encontrar en la memoria principal en disco, accediendo a ella en los momentos necesarios, como para consultar algún dato. Los datos correspondientes a estados del proceso se encuentran en memoria principal. Cada vez que un proceso es creado, también se crea un PCB para él. Cuando un proceso finaliza, el PCB asociado se destruye. El PCB es la estructura clave para que el sistema operativo soporte distintos procesos y sea capaz de realizar multiprocesamiento.

4. Preguntas.

4.1 ¿En qué consisten los algoritmos de planificación No apropiativos?

Un algoritmo con política no apropiativa no puede ser interrumpido. El proceso tiene la capacidad de continuar en ejecución hasta que finalice su labor o permute a estado bloqueado, principalmente porque espera alguna respuesta o a dispositivos de entrada y salida. La CPU no puede ser interrumpido a mitad de ejecución de este tipo de algoritmos. Suelen ser algoritmos simples, además de justos, aunque también tienen algunas desventajas, como que puede provocar inanición dependiendo del caso.

4.2 ¿Cómo se obtienen los tiempos solicitados en la actividad 6?

Existen diversos tiempos que describen el comportamiento de un proceso. El primero es el tiempo de llegada, que es el momento en el cuál un proceso pasa de un estado “nuevo”, a un estado de “listo. El tiempo de servicio o ejecución hace referencia a todo el tiempo que el proceso realizó algún trabajo o estuvo en ejecución, este se puede obtener con la fórmula Tiempo de servicio = Tiempo de retorno – Tiempo de espera, se explicarán estos dos conceptos a continuación.

El tiempo de espera es la suma de todos los tiempos que no sean en estado de “ejecución”, es decir, todo el tiempo que el proceso no trabajó. Este se puede calcular como Tiempo de espera = Tiempo de retorno – Tiempo de servicio.

El tiempo de retorno es todo el tiempo que estuvo el proceso dentro del sistema, desde el momento en que ingresó hasta cuando finalizó. Se puede obtener como Tiempo de retorno = Tiempo finalización – Tiempo de llegada, o también Tiempo de retorno = Tiempo de espera + Tiempo de servicio.

El tiempo de respuesta es cuando el proceso permuta de “listo” a “ejecución”. Es importante recalcar que se toma en cuenta la primera vez en la que hizo el cambio entre estos estados, ya que puede realizarlo múltiples ocasiones, pero para tiempo de respuesta únicamente se toma en cuenta la primera vez.

4.3 ¿Qué significa BCP?

Las siglas BCP hacen referencia a Bloque de Control de procesos. Es una estructura de datos diseñada a la medida para almacenar la información primordial que puede contener un proceso, como lo es un identificador, su estado, el contador del programa, registros de CPU, información de la iteración, información del manejo de memoria, información de contabilidad, información de estado de entradas y salidas, y algunas credenciales.

Toda esta información es de suma importancia para la ejecución de procesos, sobre todo para mantener registros del comportamiento y duración de los trabajos en el sistema, esto en caso de que se suspenda temporalmente la ejecución del proceso o se coloquen nuevamente en la lista de “listo”. De este modo, planificador de tareas puede estar consiente en todo momento de la información que acarrea cada proceso.

5. Conclusiones.

Con el desarrollo de los sistemas operativos todo es tan sencillo. Nosotros como usuario nos preocupamos únicamente de abrir el programa que vamos a utilizar, utilizarlo y posteriormente cerrar el programa y apagar el dispositivo. Pero hay un mundo tan complicado detrás de esto que da miedo. Jamás hubiese pensado que era tan complejo decirle a una máquina a cuál proceso debe prestarle atención primero, es un conocimiento que debió tomarle mucho tiempo a los creadores de los sistemas operativos.

Podría parecer que es sencillo crear los procesos que uno desee, no obstante, no cualquier evento puede realizarlo, de hecho. Es impresionante el hecho de que, en sistemas operativos complejos como los actuales, nosotros creamos procesos a diestra y siniestra, lo que conlleva a constante comunicación interna entre distintas partes de la máquina. En lo personal, me intriga mucho la creación de procesos mediante llamadas al sistema de otro proceso, porque se me hace una analogía a una empresa u organización entre personas, que, para realizar un trabajo, un empleado debe pedir permiso a su superior y esperar autorización. También es de creencia común que destruir es más sencillo que crear, pero no es el caso, ya que para finalizar un proceso también se requiere que se cumplan una serie de condiciones.

El diagrama de 5 estados me causó confusión, ya que había fuentes que colocaban únicamente 3 estados y sus transiciones, otras solamente 4. En general, había un poco de diferencias entre las fuentes. No obstante, es bastante intuitivo el entender el camino que debe recorrer un proceso desde que es creado hasta que termina.

Debo admitir que también vi muchas similitudes entre algunas estructuras de datos y los algoritmos de planificación, en especial entre FCFS y FIFO. De hecho, en mi programa 1, vi claramente la similitud entre mi sistema de lotes y una cola. No obstante, es algo un poco más complejo. De hecho, noté que en este tema implican ciertos aspectos de subjetividad, cosa que no se suele ver en ingenierías. Esto principalmente a posibles asignaciones de prioridad, además de incluir conceptos de justicia.

Es aquí donde ingresan políticas como la no apropiativa, que permite que cada proceso se ejecute, sin que sea interrumpido, o la apropiativa, en la que puede haber interrupciones. Cada política, junto con sus algoritmos, tienen sus ventajas y desventajas, algunos son más justos y equitativos, pero eso provoca que no sean tan eficientes. Otros apuestan por aprovechar al máximo recursos, pero pueden terminar en inanición. Queda en manos del usuario seleccionar el algoritmo que mejor se adapte a sus necesidades.

Por último, también noté una analogía clara entre un Bloque de Control de Procesos y una posible “clase” en programación orientada a objetos. Desde el programa 1 también vi claramente que un proceso puede ser tomado como objeto, con sus atributos, como identificador, contador de programa, etc. Siento que toda la información adquirida en esta investigación facilitará el trabajo para siguientes programas, lo cual es bastante motivador.

6. Referencias.

Flynn, I. M., & McHoes, A. M. (2001). *Sistemas operativos*. Cengage Learning Latin America.

Stallings, W. (2015). *Operating systems: Internals and Design Principles*. Pearson Prentice Hall.

Tanenbaum, A. S. (2008). *Sistemas Operativos Modernos*. Pearson Prentice Hall.