Sistemas Operativos II

Trabajo práctico N° 1

Actividad a desarrollar

El trabajo práctico consiste en la codificación de los algoritmos de colocación de memoria según lo estudiado en las clases teóricas, para que los mismos sean capaces de emular el comportamiento de la colocación de bloques en la memoria principal.

Los algoritmos a codificar son:

- Primer Ajuste
- Mejor Ajuste
- Peor Ajuste

Luego de la codificación, se deben realizar las pruebas mediante la carga de datos a la memoria utilizando la función “colocar” de la misma, pasándole una estructura Bloque y validar que los datos sean cargados correctamente.

Posteriormente se deben eliminar alguno de estos bloques de memoria y colocar nuevos bloques, los cuales deben utilizar los espacios libres (en caso de que existan) o ser colocados al final de la lista de bloques en caso de que ninguno de los disponibles tenga la capacidad deseada o no existan bloques libres.

Por último se deben programar dos funciones en una subclase de Memoria para realizar optimizaciones en la memoria:

- Combinación de bloques adyacentes
- Compactación de bloques

Estas optimizaciones pueden ser aplicadas en forma manual, pero deben ser aplicadas en forma automática por los algoritmos cuando se dé el caso que un proceso quiere utilizar memoria y ninguno de los bloques disponibles posea la cantidad de memoria necesaria, pero si hay suficiente memoria libre (sumando es espacio de todos los bloques libres).

Cabe aclarar que para que los algoritmos sean considerados correctos, deben basarse en el contenido de la estructura “datos” de la memoria, y no utilizar el contenido de “valores” para determinar bloques libres u ocupados.

Estructura base

La cátedra provee una serie de archivos con el código fuente base para que los alumnos puedan codificar y ejecutar su algoritmos, esta arquitectura consiste de:

- **catedra.py**: este módulo contiene las definiciones para las estructuras de datos utilizadas, el cual **no** debe ser modificado por el alumno.
- **alumno.py**: este módulo contiene la estructura básica para que el alumno pueda codificar los algoritmos solicitados, es en este archivo donde el alumno realizará su trabajo.
- **genera_pkl.py**: es un programa para generar datos que luego consumirá el código de los algoritmos al ser ejecutado por alumno.

Estructuras de datos

Bloque

Representa un bloque de memoria. A esta estructura se le cargan los valores de inicio y fin (posiciones de memoria) y se lo usa como estructura de la propia memoria para saber los bloques que posee asignados. Al estar como bloque de memoria (con valores en inicio y fin), si posee un valor de ID, significa que el bloque está siendo utilizado por ese proceso, si no posee un valor de ID, quiere decir que el bloque está disponible para ser usado. En caso de tener asociado un proceso contiene la información del mismo y la cantidad de memoria utilizada.

Memoria

Posee un lugar finito de datos dado por su “longitud”, el cual será representado como una cadena de caracteres, siendo el espacio en blanco “ “ la representación de la memoria libre, o un caracter con el ID del proceso indicando que dicho espacio está ocupado.
Cuando la memoria está ocupada con un valor de ID de proceso, dicho valor es meramente figurativo y no garantiza que sea el ID del proceso que realmente está utilizando el espacio de memoria. Se decidió utilizar ese valor para que la visualización del contenido de la memoria sea más simple, pero no se puede utilizar dicho valor ni el valor en blanco “ “ para los controles de libre/ocupado de los algoritmos.

Algoritmo

Posee una función “colocar” que se utiliza para aplicar la lógica de un algoritmo de colocación y es invocada por la “Memoria” para saber donde colocar los datos de un proceso cuando éste lo solicite.

Forma de trabajo y evaluación

- El trabajo debe realizarse de manera grupal, en grupos de no más de 3 personas.
- Toda la codificación debe estar documentada.
- El código debe superar los tests entregados por la cátedra para tomarse en cuenta como válidos para ser corregidos.

Fecha y forma de presentación

- El trabajo se debe enviar a la cátedra vía correo electrónico.
- El archivo adjunto debe contener el archivo alumno.py con el código fuente solicitado y un documento con las ejecuciones realizadas indicando los datos de entrada y el resultado de las operaciones a medida que se ejecutaron (colocar, liberar, optimizar, etc.).
- En el cuerpo del correo electrónico se debe indicar los apellidos de los integrantes del grupo que lo realiza.
- **El Trabajo se puede entregar hasta el 30/04/2015 a las 18:00hs.**