UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

LABORATORIO Nº3
SISTEMAS OPERATIVOS VESPERTINO

Simulación de algoritmos de posicionamiento de datos en memoria principal en particionamiento dinámico

Profesor: Fernando Rannou **Fecha de entrega:** Domingo 5 de Julio de 2015 (23:59 hrs)

Ayudante: Cristian Jeldes J. cristian.jeldes @usach.cl

OBJETIVOS

Este laboratorio tiene como objetivo que los estudiantes simulen los algoritmos de posicionamiento de memoria Best-Fit, First-Fit y Next-Fit en el lenguaje de programación C.

ENUNCIADO

Explicación de los algoritmos

Best-Fit
Este algoritmo al recibir una petición de memoria, usa el bloque de memoria más pequeño entre los bloques de memoria disponibles que sirven para atender la petición.

First-Fit
Este algoritmo al recibir una petición de memoria, busca desde el comienzo el primer bloque de memoria que sirva.

Next-Fit
Este algoritmo al recibir una petición de memoria, busca desde la última posición el primer bloque de memoria que sirva.

Desfragmentación
Este algoritmo elimina la fragmentación externa, juntando todos los bloques de memoria que estén en uso, organizándolos uno tras otro desde el comienzo de la memoria.

La simulación

Contexto:

Como usted decida (con la estructura de datos que prefiera) deberá simular un espacio de memoria de N MegaBytes, es decir N*1024 KiloBytes. 1 KiloByte es igual a 1000 Bytes. En este espacio se deben poder realizar operaciones de reserva de memoria y liberación de memoria por procesos. Se deberá mantener un archivo de texto de historial con los cambios que se realicen (reservas y liberaciones de memoria), se deberá registrar junto a este cambio, el momento en el que fue realizado (momento en base a el tiempo en que partió el programa, es decir, cuando parte el programa el tiempo es 0 ms); cada registro de cambio debe seguir el siguiente formato:

- <Marca de tiempo>;<Nombre del proceso>;<Tipo de petición>;<Cantidad de KiloBytes o Dirección de memoria>;

Entrada:

1. La primera parte de la entrada consistirá en un archivo con peticiones de reserva o liberación de memoria por procesos, donde cada línea será una petición. Debe verificar que la entrada sea correcta y que un proceso no pueda liberar memoria que no tiene asignada y en ese caso reportar un error en el historial. La forma de ingresar el parámetro será:
   --peticiones <Nombre archivo> o –p < Nombre archivo >
2. La segunda parte de la entrada consistirá en el algoritmo que se desee usar, ya sea **B**est-Fit, **F**irst-Fit y **N**ext-Fit. La forma de ingresar el parámetro será:
   --algoritmo <B o F o N> o –a <B o F o N>
3. La tercera parte de la entrada consistirá en la cantidad de N MegaByte que se deseen usar. La forma de ingresar el parámetro será:
   --cantidadmemoria <Cantidad de megas> o –c <Cantidad de megas>
4. La cuarta parte es opcional, consistirá en el nombre del archivo de historial que se desee, en el caso de no usar esta opción el nombre por defecto debe ser la fecha y hora de la ejecución del programa. La forma de ingresar el parámetro será:
   --salida <Nombre archivo> o –s <Nombre archivo>
5. La quinta parte, el programa debe poder correr en 2 modos, hablador y silencioso, donde en modo hablador avisará de todos los cambios que se vayan haciendo y en cada paso escribirá el espacio de direcciones de memoria que está usando cada proceso. En modo silencioso, simplemente entregará el archivo de salida, sin imprimir nada por pantalla. La forma de ingresar el parámetro será:

   --modo <H o S> o –m <H o S>

Para obtener los parámetros de entrada debe usar la función “getopt”, si no se tienen los suficientes parámetros para realizar el procesamiento de datos, el programa debe detener su ejecución y arrojar un error explicativo sobre los parámetros que faltan.

Procesamiento de datos:

Dados los parámetros de entrada se creará el espacio de memoria del tamaño pedido, se seleccionará el algoritmo definido y se empezará a leer el archivo de entrada línea a línea. El formato de las líneas del archivo de peticiones de entrada es el siguiente:

- <Nombre de proceso>;<Tipo de petición>;<Cantidad de KiloBytes o Dirección de memoria>;
- Tipo de petición puede tomar 2 valores “allocate” o “free”, donde allocate significa reservar una cantidad de KiloBytes en memoria, y free es liberar el trozo de memoria que corresponda a la dirección de memoria entregada.
- El nombre del proceso no tiene ninguna restricción más que no poder poseer un “;” como parte del nombre.

Cuando ya no queden bloques lo suficientemente grandes para resolver una petición de memoria pero quede suficiente espacio libre para aceptar la petición, se debe desfragmentar la memoria (compactar), en caso de que no quede suficiente espacio libre, se debe negar la petición y debe ser escrito en el historial, con el formato <Nombre Proceso>;Se rechaza allocate por falta de memoria;<Cantidad de KB pedidos>.

Salida

La salida esperada para cuando el programa está en modo “Silenciado” es el archivo de historial. Para cuando el programa esté en modo “Hablador”, el programa debe avisar de todos los cambios que se realicen y debe para cada cambio mostrar el espacio de memoria asignado a cada proceso, es decir, si el proceso “P1” pidió 300 KB, y posteriormente el proceso “P2” pidió 200KB, y nuevamente “P1” pide 100KB, con el algoritmo First-Fit, con un espacio de memoria de 1MB, la dirección de memoria en KiloByte sería de la siguiente forma:

P1: 0-299, 500-599

P2: 300-499

Es decir debe seguir el siguiente formato:

<Nombre proceso>: <Espacio de memoria asignado 1>, <Espacio de memoria asignado 2>, …,<Espacio de memoria asignado M>

Donde M es la cantidad de bloques de memoria que un proceso tiene asignado.

Al final del archivo de historial, debe escribir cuantas veces fue necesario hacer desfragmentación, cuanto espacio quedó ocupado y cuanto espacio quedó libre.

Detalles sobre la revisión

- Su programa será probado con varias entradas y diferentes tamaños de entradas.

Detalles de la entrega

- Este laboratorio debe ser entregado el Domingo 5 de Julio a las 23:59 al correo cristian.jeldes@usach.cl con el asunto “Laboratorio3-SOV-<RUT>-<Nombre Apellido>”.
- Se debe entregar un solo archivo comprimido con extensión .tar.gz con todos los archivos que corresponden al laboratorio.
- Por cada día de atraso se descontará 1 punto, con 3 días de atraso la nota será un 1.0.
- El programa será revisado en la distribución de Linux Ubuntu versión 14.04 LTS de 32 bit.

- El proceso de compilación debe ser realizado mediante un MakeFile que deberá compilar el

programa.

- El directorio de desarrollo debe contener la siguiente estructura de directorios:

```
Lab3
|...build
   |...      mainProgram (Programa principal)
|...src
   |...      main.c
   |...      otraCosa.h
   |...      otraCosa.c
   |...      …
|...MakeFile
|...README
```

- Los nombres son sólo un ejemplo, pero debe tener en cuenta que el archivo ejecutable es 1:
    - El programa principal.
- Debe crear un archivo de texto README con instrucciones de compilación y ejecución.