Puntos que investigar antes de empezar la implementación de la práctica

a) Funciones con las que cuenta el sistema de archivos de nachos para la manipulación de archivos.

b) Comprender y conocer los parámetros de las funciones para crear, abrir, leer, escribir y cerrar un archivo en nachos.

c) Identificar las partes que forma un archivo ejecutable en nachos(es decir, la estructura interna del archivo ejecutable).

d) Identificar donde se hace la verificación de que el proceso cabe en memoria.

e) Ubicar y comprender como se carga el contenido del archivo ejecutable en memoria.

f) Identificar de donde se puede obtener el nombre del archivo ejecutable a partir de la línea de comandos del administrador de memoria.

Archivo de intercambio en Nachos

Práctica 1

Descripción

En esta práctica se deberá generar el archivo de intercambio para cada archivo de prueba que sea ejecutado a través de la línea de comandos del administrador de memoria.

Un archivo ejecutable de nachos está formado por un encabezado y de los segmentos que contienen los datos y el código del proceso. Se debe modificar Nachos de tal manera que **antes** de que se cargue el proceso a memoria se genere el archivo de intercambio, es decir, se debe crear un archivo y escribir los segmentos de datos y código del archivo ejecutable en el archivo recién creado. El archivo de intercambio se identificará por la extensión **swp**.

Posteriormente, se deberá comparar el contenido del archivo ejecutable (despreciando el encabezado) y el contenido del archivo de intercambio y deberán ser iguales byte a byte. Esta verificación se realiza con el comando **hexdump** y se explica su funcionamiento en el archivo adjunto.

Los archivos de intercambio deberán generarse en el directorio donde se encuentra programa ejecutable de prueba.

Cada equipo deberá poner entre comentarios los nombres completos de los integrantes y la fecha de entrega de la práctica en el archivo *addrspace.h* y *addrspace.cc*

Información de ayuda

NACHOS actualmente soporta paginación para procesos que caben completos en la memoria, es decir, que el número de páginas requeridas por el proceso sea menor al número marcos en el sistema

(numPages <= NumPhysPages)

Un ejemplo de esto es ejecutar el archivo de prueba *halt*.

Para poder implementar memoria virtual es necesario eliminar esta condición ya que el proceso puede tener una cantidad de páginas superior al número de marcos. Esto implica que el proceso al no caber en la memoria no tiene que ser cargado en ella. ***Nota importante: para la práctica del archivo de intercambio esta condición NO se elimina.***

executable->ReadAt(&machine->mainMemory[noffH.code.virtualAddr]),noffH.code.size, noffH.code.inFileAddr); y en lugar de ello se tiene que inicializar el espacio de intercambio con el archivo ejecutable.

Un archivo ejecutable esta formado por segmentos de **código**, de **datos inicializados** y **datos no inicializados**. Cada segmento es una estructura que contiene su dirección virtual, su ubicación dentro del archivo y su tamaño. Dentro del constructor de *AddrSpace*, en lugar de leer estos segmentos a la memoria, se leen hacia el espacio de intercambio.

El espacio de intercambio es un archivo de la clase *OpenFile*, el cual permite leer y escribir una cantidad especifica de bytes en una posición indicada dentro del archivo.

Para poder hacer uso de un archivo, primero se tiene que crear utilizando la función miembro *Create* del sistema de archivos de la clase *FileSystem*. La función *Create* requiere el nombre del archivo y su tamaño. El nombre del archivo puede ser cualquier nombre y su tamaño tiene que ser igual al número de bytes que requiere el proceso en la memoria (considerando el tamaño del código, datos inicializados , no inicializados, la pila y que sea un múltiplo del tamaño de la página). La función *Create()* regresa un booleano indicando si se creó correctamente el archivo y posteriormente se tiene que abrir el archivo usando la función *Open()*, la cual regresa un apuntador *OpenFile* al archivo abierto, dentro del cual se puede escribir y leer finalmente.

En Nachos existe una variable global para acceder al sistema de archivos y está definida en el archivo *threads/system.cc* con el nombre *fileSystem*, la cual permitirá usar las funciones miembros *Create()* y *Open()*. El espacio de intercambio se puede definir como variable global ya que será accedido por el constructor de addrspace para inicializarlo y el manejador de excepciones para leer/escribir la página cuando ocurra un fallo.

Finalmente, dentro del constructor del *addrspace* en lugar de copiar el código y los datos hacia la memoria, se deben copiar hacia el archivo de intercambio que previamente debió haberse creado y abierto. La copia se puede realizar byte por byte o bien por bloques(búffer). Es importante que calcule correctamente el desplazamiento donde se va a escribir un segmento dentro del archivo de intercambio.

Notas importantes

- Cada que se de la orden de ejecutar un programa de prueba se debe de crear su correspondiente archivo de intercambio.
- **El nombre del archivo de intercambio debe ser el nombre del programa de prueba con la extensión swp.** Es decir, el archivo de intercambio para el programa halt deberá ser halt.swp, para matmult será matmult.swp y así respectivamente.
- El funcionamiento del administrador de memoria debe ser igual al de la práctica 0, es decir, el único programa que se ejecuta es *halt* y los demás van a marcar errores y no se ejecutarán. La diferencia es que ahora por cada programa de prueba se debe generar el archivo de intercambio.
- El código que se tiene que implementar para esta práctica es muy poco (aproximadamente 15 a 20 líneas de código), pero lo complicado es determinar en que archivo y en que lugar dentro de ese archivo es conveniente colocar el código que genera el archivo de intercambio.