Paginación.

Para implementar el mecanismo de paginación optamos por dividir la memoria virtual en dos grandes partes, una para las páginas que utiliza el kernel, y otra para las páginas que utilizan los programas del usuario. Se optó por proporcionar 256 Mb de memoria para las páginas para el kernel, y 256 Mb de memoria para las páginas de los programas de usuario.

El “mapeo” de la memoria virtual es uno uno con la física.

Se eligió colocar el directorio de páginas, apuntado por el cr3 , en la posición 0x08000000, colocando a continuación de él las tablas de páginas y luego las páginas correspondientes a cada tabla.

Dentro de las funciones de inicilización , tando del kernel como de las páginas para programas de usuario, nos aseguramos que el contenido de cada entrada del direcotorio de página y de cada tabla de páginas apunte a la dirección correcta en tenga activos los bits necesarios para permitir inidicarle a procesador que la página se encuentra presente en memoria o no.

Cuando el sistema “bootea” se llama a la función \_StartCR3, que es la encargada de activar paginación alterando el contenido de cr0 y de setear la dirección del directorio de páginas en cr3, y a la función initializePaging, cuya principal función es inicializar las tablas y páginas correspondientes al kernel –poniéndolas como presentes- y se setar en la IDT el manejador se de utiliza para la page fault.

En este punto es necesario aclarar que tanto las tablas como las páginas de kernel se encuentran presentes en todo momento de ejecución del sistema y comprenden las primeras 64 entradas del direcorio de página. Es decir, que las primeras 64 tablas, y todas sus páginas asocadas corresponden al kernel. A partir de la entrada 64 del direccitorio de páginas , se cuenta con espacio disponible para tablas y páginas del usuario.

El mecanismo para asignar memoria a los procesos del usuario es el siguiente. Cuando un proceso es creado en la función CreateProcess\_in\_kernel, de llama a la función create\_proc\_ptable, presente en el archivo paging.c. Esta función de ocupa de iterar sobre un vector auxiliar denominado user\_tables, donde cada posición representa una tabla de páginas que puede ser asignada a un procesod e usuario. Como se mencionó anteriormente , la memoria disponible para ser asignada a los procesos de usuario en el sistema es de 256, por lo que sólo 64 tablas del páginas pueden ser usadas por los procesos de usuario. De esta manera, cuando un procesode usuario es creado se busca la primera entrada del direcorio de páginas que no se esté utilizando , indicada por el vector user\_tables, se pone el pid del proceso que solicitó esa tabla en el vector, para señalizar que esa tabla ya está siendo utilizada. Después de actualizar el valor de la entrada del vector, se procede a poner como presente la tabla de página en cuestión, inicializando todas las páginas correspondientes a la misma, y poniendo como presente sólo la primera, que será utilizada para el strack del proceso. Cuando la función CreateProcess\_in\_kernel debe inicializar el comienzo del stack del proceso llama a la función de parging.c get\_stack\_start, que en función del inicide del arreglos user\_tables que le fue asignado al proceso, devuelve una posición correspondiente a la dirección final de la página con índice 1023 asociada a la tabla del proceso. Esto se decidió hacer así, para manejar de una forma más simple el crecimiento y decrecimiento del stack, que se explicará en un punto siguiente.

Cambio de contexto.

Estando ya el proceso creado y teniendo ya una página asignada para su stack nos debemos asegurar que en cada cambio de contexto las páginas correspondientes al mismo, sean bajadas, cuando el proceso sea colocado como “ready”, y sean subidas , cuando sea colocado como “running”.

Para realizar esta tarea se agregaron dos funciones que son llamadas desde el manejador de la interrupción generada por el timer tick (que es la que realiza el cambio de contexto). De esta manera, antes de llamar a la función de hopOffPages, que es la que se ocupa de bajar las páginas del proceso al cual se le está quitando el procesador, se llama a la función LoadAuxStack , que cambiar el stack utilizado por el procesador a uno auxiliar para que no hayan problemas cuando se estén bajando las páginas. Una vez cambiado el stack al auxiliar, se procede a subir las páginas corresponidentes al proceso próximo a ejecutar mediante la función TakeUpPages.

Es importante mencionar en este punto el mecanismo por el cual se registra cuales son las páginas que deben ser subidas de un proceso una vez que estas fueron bajadas. Recordar cuales eran las paginas que estaban presentes para el proceso en el momento en el que se le quitó el procesador es una tarea importante para realizar el cambio de contexto. Para señalizar las páginas que estaban presentes en un momento determinado se utilizó el bit 10 de la entrada de tabla correspondiente a la pagina determinada. De esta manera, cuando un proceso necesita otra pagina, además de activar el bit 1 de la entrada de la tabla también se activa su bit 10, para así, poder “recordar” que la página estaba presente.

Dinamismo del stack.

Para manejar el crecimiento dinámico del stack de la forma más simple y directa posible, se decidió, como se mencionó anteriormente, reservar una tabla completa (con todas sus páginas), para un proceso cuando es creado. De esta manera, en el momento inicial de su creación la base del stack apunta a la última dirección de la última página correspondiente a la tabla de proceso. De esta forma, y como el stack crece “hacia abajo”, es decir, el esp se va haciendo menor a medida que el proceso va “pusheando” variables en el stack, cuando la diferencia entre el esp y la base del la paágina que se está utiliando en menor a 1K, se procede a asignar la página siguiente al proceso, de manera que éste pueda crecer así hasta que se acabe la memoria correspondiente a la tabla utilizada. En otras palabras, en este sistema , un proceso puede disponer con 4Mb de memoria como máximo.

De manera similar sistema le quitará una página al proceso si ve que la diferencia entre el esp y su dirección base es mayor a 6K.

Para hacer este chequeo del stack se utiliza la función