LAB 2

No inicio deste Lab trabalharemos primeiramente com a alocação das paginas físicas, portanto as funções abaixo deveram ser implementadas.

A partir daqui trabalharemos com novos tipos de dados, tais como o PTE e PDE , para a compreensão do funcionamento dessas estruturas, e mais informações sobre a manipulação de memoria estude os capítulos 5 e 6 do manual do Intel 386, principalmente os subcapítulos 5.2 e 6.4 que tratam do funcionamento das tabelas de memoria, e do gerenciamento de acesso de paginas.

- boot_alloc

Nesta função após recebida a quantidade bytes a serem alocados, devemos separar paginas suficientes para estes.

Para isto deveremos seguir os seguintes passos:

1. Verificar o numero de bytes, caso seja 0, nenhuma pagina deve ser alocada.
2. Verificar se há espaço para alocação.
3. Separar o numero de paginas suficientes para os bytes solicitados, sempre arrendondando para o teto.
4. Mover o ponteiro que indica qual a próxima posição livre após n paginas.
5. Retornar este ponteiro.

- men_init parte 1

Aqui faremos a alocação do vetor responsável pelo controle das paginas físicas, sendo que em cada posição deste vetor se encontra uma estrutura PageInfo, a qual contem informações sobre cada pagina física.

Dado que o SO utiliza um vetor chamado pages, teremos duas simples funções:

1. Fazer a alocação deste vetor utilizando a função boot_alloc implementada anteriormente.
2. Realizar a limpeza deste vetor com memset, para que quaisquer lixo de memoria não nos cause problemas futuramente.

- page_init

Nesta função faremos a inicialização de todas as paginas de memoria física.

Utilizando a função page2pa, a qual faz a transformação do descritor PageInfo em um endereço físico, devemos realizar a alocação respeitando as seguintes condições :

1. A pagina 0 não deve ser alocada.
2. Os endereços referentes a memoria base não devem ser alocados.
3. A memoria estendida também não deve ser alocada.

Após respeitadas tais condições as paginas restantes devem ter seu contador de referencia zerado, e a mesma colocada no vetor de paginas livres, page_free_list.

- page_alloc

O page_alloc como o nome sugere, aloca uma pagina para quem o executa, portanto, os seguintes elementos devem ser encontrados em seu escopo:

1. Verificação da existência de paginas disponíveis.
2. Uma pagina da lista de paginas livres deve ser removida desta lista.
3. A lista de paginas livres deve ser atualizada.
4. O ponteiro referente a próxima pagina livre, da pagina selecionada, deve ser setado para NULL.
5. Caso alloc_flags & ALLOC_ZERO a pagina toda deve ser setada com "0", lembre-se que a alocação ocorre no endereço virtual do kernel.

- page_free

Esta é a função responsável em dizer ao SO que uma pagina previamente utilizada esta agora disponível novamente. Portanto o page_free deve seguir os seguintes passos:

1. Verificar se o contador de referencias da pagina é 0.
2. Caso seja, a mesma deve ser devolvida a lista de paginas livres.
3. Caso contrario, a pagina ainda não esta pronta para ser liberada, pois algum processo ainda á esta utilizando.

Agora com as funções responsáveis pela alocação física implementadas, podemos partir pra a alocação virtual.

- pgdir_walk

O pgdir_walk tem como função estabelecer um link entre um endereço virtual e uma pagina de memoria. Para isto é necessário analisarmos em qual diretório de paginas este endereço reside, e qual o deslocamento dentro deste diretório.

Para que isto seja realizado com sucesso, o pgdir_walk deve trabalhar da seguinte maneira:

1. Verificar se a pagina esta presente no diretório.
2. Criar ou não esta pagina se solicitado.
3. Atualizar as entradas e permissões no diretório;
4. Devolver o PTE da pagina a função principal.

- boot_map_region

Aqui faremos o mapeamento dos endereços virtuais para os endereços físicos utilizados durante o boot.

Com os endereços físicos e virtuais iniciais, e com a quantidade de paginas necessárias, implementaremos as seguintes instruções:

1. Utilizando o pgdir_walk, obteremos o PTE do primeiro intervalo do endereço virtual (sempre múltiplo de PGSIZE).
2. Atribuiremos a este PTE um endereço físico, junto com as permissões solicitadas.
3. Repetiremos estes passos, avançando de pagina em pagina, até a quantidade desejada.

- page_lookup

page_lookup tem como objetivo obter o descritor de uma pagina de memoria a partir de seu endereço virtual.

Assim com um endereço virtual, podemos obter o PTE com o pgdir_walk, e deste PTE extraímos seu descritor, com esta ideia em mente faremos o seguinte:

1. Através do pgdir_walk conseguiremos o PTE deste endereço virtual.
2. Utilizando o pa2page obteremos nosso PageInfo.
3. Caso necessário o endereço deste PTE deve ser salvo.

- page_remove

O page_remove remove o mapeamento entre um endereço virtual e uma pagina de memoria física, para que tal endereço físico possa ser usado futuramente por outro processo.

Utilizaremos das funções anteriores pare que esta remoço ocorra sem problemas, seguindo a seguinte ordem:

1. Caso a pagina exista, uma referencia a esta deve ser decrementada.
2. Caso as referencias a esta pagina cheguem a 0, a mesma deve ser liberada.
3. Caso removida, o PTE deve ser atualizado, já que esta, a partir de agora, não pertence a um diretório.

4. A entrada na tabela de paginas deve ser atualizada, utilizando o comando tlb_invalidate.

- page_insert

No page_insert é realizado o mapeamento das paginas de memorias físicas aos endereços virtuais dos processos.

Algumas verificações devem ser feitas com fim de evitar erros durante a execução do programa, tais como a limpeza de uma pagina caso esta já esteja presente. Portanto:

1. Deve ocorrer uma verificação se a pagina solicitada já esta alocada e presente.
2. Caso esteja, suas permissões devem ser atualizadas;
3. Caso contrario a pagina deve ser inserida, com as devidas permissões e seu contador de referencias incrementado.
4. Porem se a pagina não estava alocada previamente, e não foi possível sua alocação, deve-se retornar -E_NO_MEM indicando que o limite de memoria do SO foi atingido.