Documentação TP2

Alunos: André Lustosa Cabral de Paula Motta Matrícula:2015097826
Daniel Ishitani Melo 2015114208

Introdução

O TP2 foi dividido em 3 partes:

i. Apenas um tutorial para entender como controlar a VM e adicionar chamada ao xv6 e criar comando para a mesma.

ii. Implementar duas chamadas: virt2real, que recebe um endereço virtual e retorna o real, e num_pages, que retorna o número de páginas usadas por um processo.

iii. Implementar a chamada forkcow (fork copy-on-write) que cria um processo filho com páginas copy on write, tal processo é read only.

Implementação

A primeira parte foi feita a partir do passo a passo da descrição, utilizamos o esqueleto do Passo 1 para sys_date no sysproc.c, adicionamos uma checagem para o retorno da argptr e então chama a cmostime() para ler a data e horário. Depois seguimos a sequência do Passo 2 para alterar os arquivos syscall.h e .c, user.h e usys.S, de forma a adicionar a nova chamada. Em seguida, foi feito o date.c, com a chamada e impressão da data no formato dia/mês/ano hora:min:sec, e adição do comando no Makefile. Após a compilação a execução teve o resultado esperado.

Na segunda parte, a implementação da sys_num_pages tem um ponteiro para um processo que recebe o retorno de myproc, então é feita a divisão do tamanho do processo recebido pelo tamanho das páginas e a função retorna esse resultado. Fizemos um teste para o caso de o tamanho do processo não ser exatamente igual ao total do tamanho da das páginas que ele usa. A num_pages.c faz a chamada e imprime o valor retornado. Além disso, foi implementada a virt2real que dado um endereço virtual, retorna o correspondente real. A função tem um ponteiro para char usado na argptr, um ponteiro para proc que recebe o retorno de myproc e dois ponteiros. Um para pde_t, pde que recebe o endereço do diretório para a tabela de páginas com índice retornado por PDX, e outro para pte_t, pgtab que recebe o endereço para a entrada na tabela com índice retornado por PTX, então retornamos o resultado de P2V com o endereço da tabela de páginas somado às flags como parâmetro.

Para a última parte do TP, páginas copy-on-write, foi feita a forkcow que é igual à fork exceto que ao copiar o estado do processo chama copyuvmcow ao invés de copyuvm. Dentro da struct kmem no arquivo kalloc.c fizemos um vetor para contar as referências a cada página, também foram implementadas as função addRefCount, minusRefCount e getRefCount para modificar e ver o contador.

Alteramos também as funções freerange para inicializar a página com 0 no contador e a função kfree para que o free só seja executado quando o processo que deu o free seja o último a referenciar a página.

No arquivo proc.c temos a função forkcow que como explicada é cópia da chamada fork, com uma mudança. Chamamos copyuvmcow ao invés de copyuvm. A copyuvmcow é definida no arquivo vm.c. A mesma se diferencia da copyuvm ao não alocar um espaço de memória novo para o processo filho. Ela, para cada página do processo pai, seta a página como Read Only e como Copy on Write e mapeia para o processo filho as mesmas páginas do processo pai, somando um ao contador de referências da página. E após isso realiza o flush da TLB.

Contudo, isso cria um problema. Quando o processo filho copy on write for tentar escrever haverá um pagefault e será necessário tratar o mesmo por uma trap do sistema.

A trap foi definida em trap.c e chama a função pagefault definida em vm.c. A função pagefault, recebe um código de erro e trata a falta da página. Ela após checar as integridades do acesso ela checa se a página possui mais de uma referência. Se isso ocorrer ela copia a página com mais de uma referência para outro local de memória, adiciona permissões de escrita, remove a flag de copy on write e reduz em um o número de referências à página original. Carrega a página em memória e da o flush da TLB. Se houver apenas uma referência, ele simplesmente seta a página como Write Enable e remove a flag de copy on write e faz o flush da TLB.

Conclusão

Com o trabalho ficou mais claro o mapeamento dos diretórios para as tabelas, das tabelas para o endereço real e da utilidade dos bits flags para as restrições de uso, tratamento de page faults e para fazer chamada copy-on-write. Aprendemos como e quando fazer flush da TLB para não gerar inconsistências e entendemos melhor as partes do código xv6 utilizadas. O trabalho teve o resultado esperado, passando nos testes do corretor disponibilizado.

Link: https://github.com/andre-motta/tp1_SO_UFMG_2017-2/tree/master/tp2/xv6-personal