UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS

TP2: Paginador xv6

José Carlos de Oliveira Júnior
Victor Andrés Caram Guimarães

8 de outubro de 2017

Belo Horizonte - MG

Introdução

O objetivo deste trabalho é explorar conceitos de memória virtual e páginas *copy on right* criando chamadas de sistema no sistema operacional *xv6* emulado no *QEMU*, bem como entender como funciona a memória virtual

O *xv6* é a re-implementação dos sistema operacional *Unix Version 6* escrito em ANSI C para sistemas com multiprocessador *x86*.

Link do repositório no GitHub: https://github.com/joseoliveirajr/xv6-public.

Parte 1: Date

Esta parte ter por propósito entender como funcionam as chamadas de sistema no xv6 seguindo o tutorial da especificação. Para implementar esta chamada de sistema, em `sysproc.c` a data foi armazenada usando a função `cmostime` definida em `lapic.c` para preencher a `struct rtcdate`. A função `argptr` trata de passar esta estrutura como argumento ao chamar a `main` do programa `date.c` que imprime a data na tela.

Parte 2: Chamadas de sistema para paginação e memória virtual

As chamadas de sistema implementadas nessa parte servem para auxiliar como no funcionamento da parte seguinte que trata de páginas *copy-on-write*. Em todas as vezes que foram criadas novas chamadas de sistema, os arquivos `user.h`, `syscall.c`, `syscall.h`, `usys.S` e `sysproc.c` passaram por alterações. Nesta documentação, outras mudanças serão explicitadas além destas quando for o caso.

Função virt2real

A função `int sys_virt2real(void)` recebe um endereço virtual e converte para um endereço real. A ideia para realizar esta conversão é localizar a página pelos 10 bits de diretório da página e 10 bits da tabela de página, para 20 bits no endereço real e usar os 12

bits do offset para localizar o endereço na página. Neste trabalho, utilizamos a função `uva2ka` para setar o ponteiro do endereço real e ter o seu resultado.

Função num_pages

A função `int sys_num_pages(void)` retorna um inteiro que indica o número de páginas que um processo faz uso. Basicamente, podemos usar o tamanho do processo em bytes e dividir por 4096 bytes que é o tamanho de uma página.

Para fazer isto, usamos a chamada `myproc` que retorna um ponteiro para uma estrutura proc com as informações do processo. Um atributo da estrutura é `sz` que possui o tamanho do processo em bytes. Basta retornar o valor deste atributo dividido por 4096.

Parte 3: Páginas copy-on-write

Nesta parte foi implementada de fato a paginação com copy-on-right. Para fazer isto, foi criada uma nova função a partir da `fork` chamada `forkcow` que será responsável por criar um processo filho com páginas copy-on-right. A única diferença é o uso da função `copyuvmcow`, uma versão modificada da função original, `copyuvm`. Das mudanças, uma delas é as páginas pai serão tratadas como ready-only. Ao mapear as páginas, não utiliza-se o parâmetro `V2P`, mas o endereço do referenciador `PTE`.

Para serem usadas na função `copyuvmcow`, foram criadas funções para gerenciar o número de processos que querem usar a TLB e o número de páginas. São três funções que recebem página referenciada. A `int cr(int pa)` retorna o contador de referências, a `void icr(int pa)` incrementa o número de referências da página e a `void dcr(int pa)` decrementa o número de referências da página. Estas funções localizadas no `kalloc.c` e utilizam da `stuct kmem` para fazer esta gestão. Ao final da função `copyuvcow` faz um flush na TLB, pois há uma mudança na tabela de páginas e, portanto, há uma diferença nas referências que podem ocasionar erros.

Por fim, é necessário fazer o tratamento de page faults pela função `void pagefault(int err_code)` em `vm.c`. Se um processo escreve em uma página com permissões de escrita ou lê um endereço virtual inválido, deve ter sua execução interrompida. Entretanto, se o processo for o primeiro a tentar escrever nessa página ela

aloca uma nova página, copia o conteúdo da memória original para a virtual, aponta a tabela de entrada para a nova página e como o processo não aponta mais para a página original, o contador de referências é decrementado. Caso o processo seja o último a escrever nessa página, remove a restrição de Read Only da página. Há um flush na TLB após todo o processo.

Conclusão

Com este trabalho prático, primeiramente foi preciso entender como funciona as chamadas de sistema no xv6. Seguindo o objetivo, foi necessário entender como funciona o endereçamento físico e virtual para processadores x86 como visto em sala de aula, bem como a conversão. Para implementar a paginação, foi necessário entender também como funciona o critério e a execução dos passos seguintes a uma ocorrência de uma page fault. O aprendizado é satisfatório a partir do trabalho fornecido, foi possível entender como funciona e como implementar diversas funções no xv6.