# TP2 — Paginador xv6

Bruno Cesar Pimenta Fernandes — 2013007161

- Repositório, com a documentação atualizada (README.md)

## Introdução

Neste trabalho prático, modificamos o sistema xv6, aplicando conceitos de memória virtual aprendidos em aula (em particular, o copy-on-write).

## Parte 1: Comando date

Na primeira parte do trabalho, foi implementado o comando date no xv6. Foi seguido o tutorial como especificado.

Na implementação da função sys_date() em sysproc.c, foi usada a função cmostime() da biblioteca date.h, que armazena a data e o horário atual do sistema (em UTC) nos campos da struct rtcdate passada como parâmetro.

Na implementação do comando date, escrito em date.c, basta imprimir na saída os campos da rtcdate recebidos pela chamada de sistema implementada.

## Parte 2: Chamadas de sistema auxiliares

Para implementar a chamada num_pages() basta seguir o tutorial e na implementação em sysproc.c, na função sys_num_pages(), dividir o tamanho do processo atual (indicado por myproc()->sz) pelo tamanho das páginas do sistema, que está definido na macro PGSIZE.

Para implementar a chamada virt2real(), em sys_virt2real(), foi feito algo parecido com a implementação de walkpgdir(), usando o diretório do processo atual para encontrar a tabela de páginas correspondente e a partir dela encontrando o endereço real que corresponde ao endereço virtual passado como parâmetro.

## Parte 3: Páginas Copy-on-Write

Para implementar as páginas copy-on-write:

- Foi adicionado um campo na struct kmem em kalloc.c, pgref_cnt. Este campo é um array que mantem os contadores de referências das molduras das páginas. Ele é inicializado junto com a estrutura do alocador de memória física. Cada vez que uma moldura é alocada, seu contador é setado para 1. Cada vez que a moldura é desalocada, o contador decrementa, e a moldura só é liberada quando o contador é 0.
- forkcow() é uma versão do fork() que utiliza copyuvmcow() ao invéz de copyuvm() (do arquivo vm.c) para referenciar a tabela do parent ao invéz de copiá-la.
- Foi definido o macro PTE_COW como 0x800 para servir de flag para as páginas copy-on-write sendo usadas nas chamadas forkcow(). Observe que 0x800 pode ser usado para referenciar um bit livre da

tabela de páginas (AVL na figura).

- copyuvmcow() seta a flag PTE_COW (como dito acima) e remove a flag de permissão de escrita PTE_W, para que a tabela seja read-only. Ao contrário da original, essa função não copia para uma nova tabela, ela apenas retorna uma referência à do parent. Além disso, ela incrementa o contador de referências de moldura, já que um novo processo child está sendo criado. Note que antes de retornar, é dado o flush na TLB para garantir que não haja inconsistências.

- Para tratar os pagefaults, foi adicionada uma cláusula no switch de trap.c que indica que um pagefault ocorreu (trapno == T_PGFLT). Além disso, como especificado, devemos verificar se o pagefault é de escrita utilizando as flags do trap(tf->err & 0x2)`.

- A função que trata as pagefaults foi definida em vm.c. A pagefault_handler() primeiro garante que a flag PTE_COW está setada, e pega o contador de referência correspondente. Se o contador for igual a 1, não é necessário criar uma nova página. Basta remover a flag PTE_COW e setar a página como writeable usando a flag PTE_W. Se for maior que 1, uma nova página é criada. Nesse caso, é decrementado o contador de referências da página original, pois, obviamente, o child passa a referenciar uma página diferente.

## Conclusão

Apesar de algumas dificuldades com o código, o trabalho ajudou bastante a aprender sobre o sistemas de paginação e processos, não só no xv6, mas sim em geral.