EP2 - MAC0422

Diego Alvarez, Thiago I. S. Pereira

16 de outubro de 2015

Introdução

Esta apresentação tem como objetivo exemplificar a implementação do EP2 de MAC0422 - Sistemas Operacionais.
Iremos apresentar um simulador de gerência de memória, e diversos algoritmos para gerência de espaco livre e substituição de páginas.

O Simulador

O simulador foi totalmente implementado e testado ultilizando python3 em ambiente GNU/Linux

- python3 - versão 3.4.2
- Debian GNU/Linux 8.2 Jessie

O Simulador

O simulador e o shell foram implementados separadamente, o shell chamando o simulador como se fosse um programa separado, cada um tendo seus proprios módulos.

O Simulador

O simulador foi dividos nos seguintes modulos:

- **linkedlist** - Implementação da lista duplamente ligada com cabeça, usado para guardar o estado de alocação da memória
- **mem** - Abstrai o acesso a arquivos como se fosse memoria e implementa uma memoria virtual
- **mem_manager** - Cuida da alocacao da memoria para os processos e intermedia o acesso dos mesmos a memoria
- **proc_manager** - Interface que facilita a criacao, execucao e o término de processos

O Simulador

- **allocators** - Módulo que abstrai todos os algoritmos de alocação de memória em uma interface uniforme
- **pagers** - Módulo que abstrai todos os algoritmos de paginação em uma interface uniforme
- **simulator** - Tratamento de entrada e loop principal

O Simulador

O shell foi dividido nos seguintes modulos:

- **prompt** - Verifica a entrada do usuário, se todos os comandos e valores correspondem com o enunciado do EP.
- **main** - Inicia um shell e com a ajuda do modulo prompt verifica as entradas e executa o simulador com os parametros inseridos

Abstração Memória

- Ao iniciar o simulador, serão criados dois arquivos binários, para que possamos simular tanto a memória física quanto a memória virtual, estes arquivos estão em formato binário
- Para descrevermos que a memória está vazia, utilizaremos um valor fixo para isto. Como para se escrever 1 byte necessitamos de valores de 0 a 255, optamos por definir que o valor 255 representaria o número -1, como descrito no enunciado do EP.

Simulação da Memória

Abaixo observamos uma simulação de memória com tamanho de 12 bytes, contendo 12 **"ff's", que em decimal representa o valor 255**

```
$ ~ hexdump /tmp/ep2.vir
0000000 ff ff ff ff ff ff ff ff ff ff ff ff
000000c
```

Observações

Alguns detalhes de implementação que merecem atenção:

- O programa limpa os bits 'R' de acesso as páginas a cada 3 iterações
- A versão do LRU implementada foi a versão 3 conforme visto em aula
- Ao alocar os processos, o seu tamanho é arredondado para cima para um múltiplo de 16 bytes
- Quando todas as páginas estão com o bit 'R' ligado, o NRU retorna a primeira página

Testes

- Ao testar os algoritmos de paginação, o **first fit** foi usado como algoritmo de alocação
- Ao testar os algoritmos de alocação, o **FIFO** foi usado como algoritmo de paginação

Testes

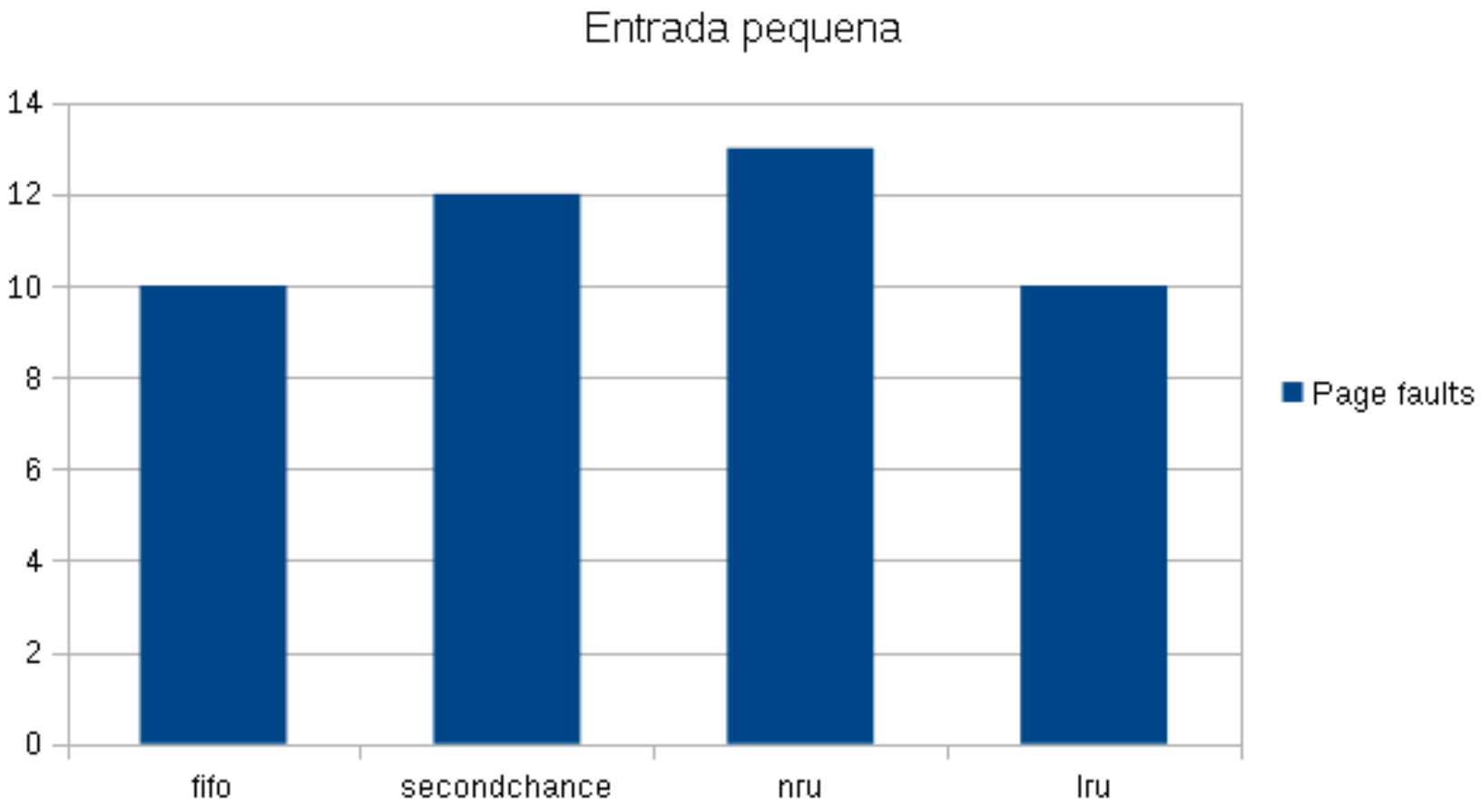
- Para comparar os algoritmos de paginação rodamos o mesmo trace para cada um e calculamos o total de **page faults**
- Para comparar os algoritmos de alocação rodamos o mesmo trace para cada um e calculamos o total de entradas na tabela de alocação
- Para os testes pequenos foi usado um tamanho de pagina de 2 bytes, e para os testes grandes um tamanho de 16 bytes

Testes

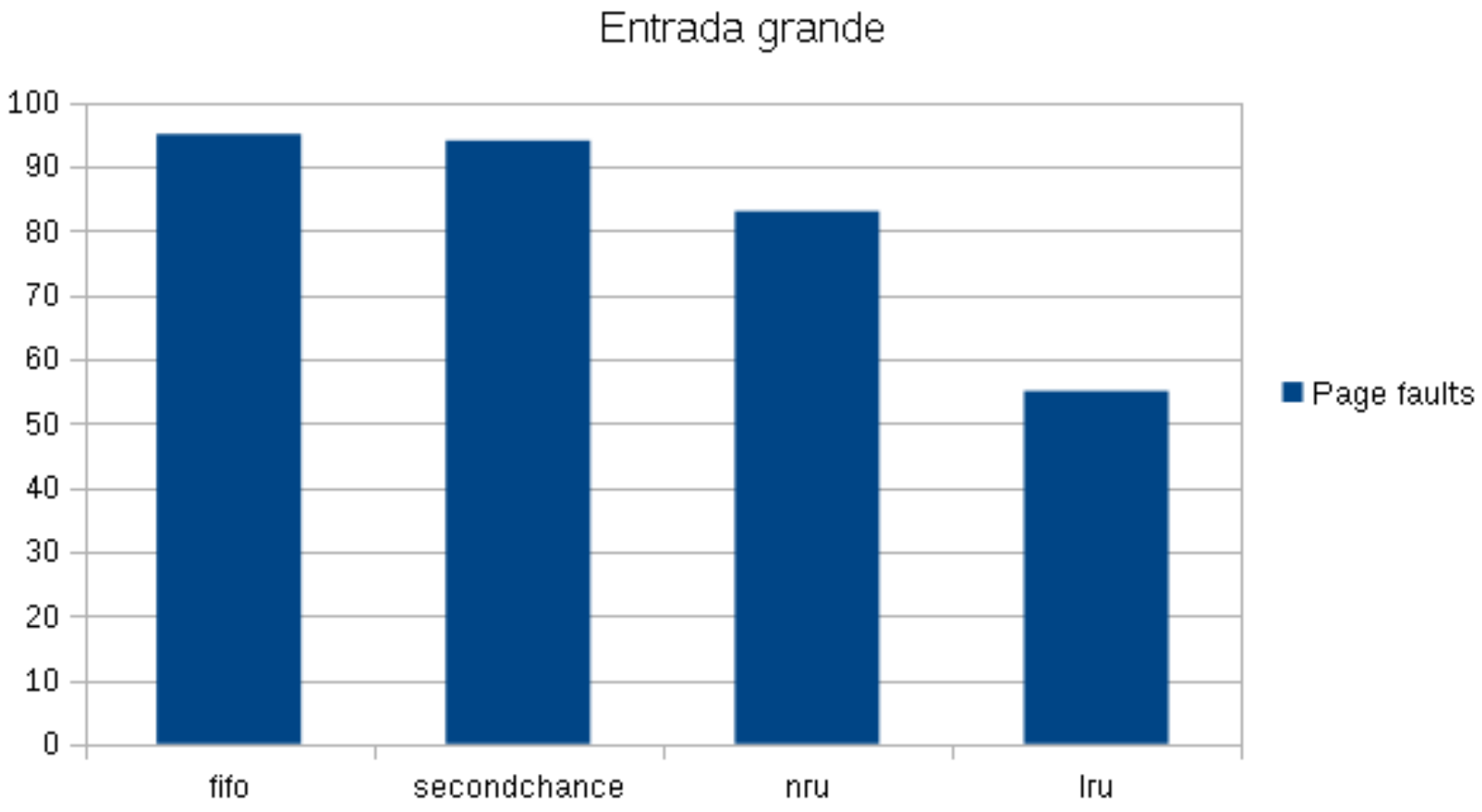
Tabela 1: Segmentos de Memória

Nº de Segmentos de Memória		
	Entrada Pequena	Entrada Grande
First Fit	28	379
Next Fit	27	409

Testes


Entrada pequena
14
12
10
8
6
4
2
0
fifo
secondchance
nru
lru
Page faults

Testes


Entrada grande
100
90
80
70
60
50
40
30
20
10
0
fifo
secondchance
nru
lru
Page faults