Universidade do Estado de Santa Catarina – Bacharelado em Ciência da Computação
Sistemas Operacionais – Prof. Rafael R. Obelheiro – Semestre: 2020.1

Trabalho 2 – Gerência de Memória

Sobre o trabalho

- O trabalho é individual.
- Exercícios de implementação devem ser realizados na linguagem C, e serão testados no sistema operacional Linux. As submissões estarão sujeitas a controle de plágio usando ferramentas de detecção de similaridade.
- Exercícios teóricos devem ser submetidos em formato PDF. Para os exercícios de implementação deve ser submetido o código fonte, em formato compilável (ou seja, os arquivos `*.c`, e não listagens em PDF). Submeta um único arquivo ZIP contendo todas as suas respostas (teóricas e de implementação).
- O trabalho deverá ser entregue até **SEGUNDA-FEIRA, 10 DE AGOSTO**. O Moodle aceitará submissões até 23h59min, sendo automaticamente bloqueado após a data limite. É **RESPONSABILIDADE DOS ALUNOS** garantir que o trabalho seja entregue no prazo.
- Em caso de dificuldades ou dúvidas na interpretação do trabalho, entre em contato com o professor (`rafael.obelheiro@udesc.br`).

Exercícios

1. Considere um sistema de paginação simples, com endereços virtuais de 15 bits e páginas de 4 KB, onde o processo P1 tem a seguinte tabela de páginas:

	pág. física	válido
0	6	1
1	7	1
2	3	0
3	3	1
4	0	1
5	1	0
6	9	1
7	4	0

   (a) Acesse o endereço `https://bit.ly/3hXW3Zb` para obter um conjunto de 5 números inteiros aleatórios entre 1000 e 32000. Liste quais foram os números obtidos.

   (b) Considere os números gerados como endereços virtuais gerados por P1, e determine os endereços físicos correspondentes. Caso ocorra falta de página, aloque uma página física livre do conjunto $\{1, 2, 4, 8\}$ (você pode escolher qualquer uma) para a página virtual que gerou a falta, e determine o endereço físico resultante.
   IMPORTANTE: Suas respostas devem demonstrar claramente todos os passos da tradução de endereços. Respostas que contenham apenas o endereço físico resultante serão desconsideradas.

2. Considere um sistema de paginação multinível com endereços virtuais de 32 bits e páginas de 1 KB. Cada entrada de tabela de páginas ocupa 32 bits. O espaço ocupado por uma tabela de páginas deve ser igual ou inferior a uma página.

   (a) Quantos níveis de tabelas de páginas são necessários para atender às especificações acima?

   (b) Qual o tamanho (em número de entradas) das tabelas de páginas em cada um desses níveis? (Dica: nem todas as tabelas podem ter o mesmo tamanho.)

   (c) Em qual dos níveis da hierarquia seria melhor colocar a tabela de páginas com menos entradas? Justifique sua resposta.

   (d) Decomponha o endereço hexadecimal `0xbadcoffe` nos componentes de número de página e deslocamento de acordo com a estrutura de endereço virtual identificada nos itens anteriores. (Dica: use a representação binária do endereço para fazer a decomposição.)

3. Escreva um programa em C no Linux que simule os algoritmos de substituição de páginas FIFO e MRU. O programa deve atender aos seguintes requisitos:

   R1. O programa recebe o número de páginas físicas na memória (*npf*) como um parâmetro na linha de comando; ou seja, o programa deve ser invocado como

   ```
   $ ./prog npf
   ```

   O espaço de endereçamento virtual será de $10 \cdot npf$ páginas.

   R2. O programa deve ler da entrada padrão uma sequência de números de página, a qual é encerrada por -1. Você pode assumir que um número de página cabe em um `long`.

   R3. O programa deve manter um controle de quais páginas lógicas estão alocadas na memória física; as páginas físicas são numeradas de 0 a $npf - 1$, e inicialmente estão todas vazias.

   R4. O programa deve verificar se a página lida está presente na memória física (a entrada -1 deve ser desconsiderada). Se não estiver, ocorre uma falta de página.

   R5. O tratamento de faltas de página deve ser o seguinte:
   - a página lógica deve ser alocada em qualquer página física livre, se houver;
   - caso a memória física esteja toda ocupada, deve ser escolhida uma página vítima de acordo com o algoritmo de substituição de páginas, e a página física correspondente é alocada para a página lida.

   R6. O programa deve implementar os algoritmos de substituição de páginas FIFO e MRU. O tempo pode ser representado por um contador de dados lidos (a primeira página representa $t = 1$, a segunda página $t = 2$, e assim sucessivamente).

   R7. Ao final da execução, o programa deve imprimir o número de faltas de página para cada algoritmo.

   O exemplo a seguir mostra um exemplo de entrada e a saída esperada do programa:

   ```
   $ ./prog 3
   1
   2
   3
   1
   4
   1
   3
   2
   3
   -1

   FIFO:   7 faltas da pagina
   MRU:    5 faltas de pagina
   ```

   As faltas de página para essa entrada são as seguintes ("x" indica que a referência causou uma falta de página):

FIFO	x	x	x		x	x		x	x	total = 7 faltas
página	1	2	3	1	4	1	3	2	3	
MRU	x	x	x		x			x		total = 5 faltas