Políticas de Ocupação de Memória em *Swapping*
Atividade de Fixação

Informações

- Trabalho **individual**;
- Entregar em <`http://trab.dc.unifil.br/moodle/`>
- Peso do trabalho no bimestre: **10%**.

1 Introdução

A ferramenta mais simples para abstração da memória principal do sistema é a *Unidade de Gerenciamento de Memória* (MMU, sigla em inglês) baseada nos registradores *base* e *limite*. Além desse componente de hardware, complementa a abstração uma política de aquinhoamento de processos, implementada em software, denominada *Swapping*. Nela, o sistema operacional mantém registros de ocupação da memória para poder decidir quais espaços serão ocupados por quais novos processos, ou aqueles que retornam do *swapping* na memória persistente (tipicamente, do disco).

Neste trabalho, você deverá programar duas estruturas de dados, e seus respectivos algoritmos, que implementam esse registro, além das políticas de aquinhoamento denominadas *first-fit* e *next-fit*.

2 Ferramentas e Instruções

Este trabalho poderá ser feito tanto em C quanto em Java, apesar de o professor preferir a primeira opção. Como incentivo, há anexado a este roteiro – em arquivo – os cabeçalhos para ambas as estruturas de dados a serem implementadas, juntamente com as interfaces das funções, na linguagem C.

Caso ainda assim prefira fazer em Java, você deverá traduzir as interfaces da linguagem C para classes de Java por conta própria. Será cobrada correção delas!

2.1 Roteiro de Atividades

A seguir, as atividades que devem ser desenvolvidas neste trabalho:

1. (30 pontos) Programe a estrutura de dados `struct reg_bitmapped` e as suas funções de manipulação, como definidas no cabeçalho `reg_bitmapped.h`. Atenção! O registro de pedaços ocupados deverá ser feito utilizando bits de `char` ou `int`, ou seja, será necessário utilizar máscaras binárias [1].

2. (20 pontos) Explique a necessidade de dividir a memória em pedaços lógicos, quando utilizamos a abordagem de registros em *bitmaps*. Explique ainda quais as considerações a ser feitas sobre os tamanhos dos pedaços.

3. (30 pontos) Programe a estrutura de dados `struct reg_nos` e as suas funções de manipulação, como definidas no cabeçalho `reg_nos.h`.

4. (20 pontos) Faça uma comparação as duas maneiras de registrar espaços ocupados por processos na memória, considerando desempenho, ocupação de espaço com o próprio registro e necessidade de dividir a memória em pedaços lógicos.

2.2 Entrega

Entregar apenas os códigos-fontes e um arquivo em PDF com as respostas das questões discursivas, compactados com `tar.gz`. Não colocar nenhum arquivo de projeto, e nem binários compilados.

Referências

[1] MICROSOFT. *Operadores bit a bit*. Microsoft Developer Network. Disponível em: <`https://msdn.microsoft.com/pt-br/library/17zwb64t.aspx`>.