EP2: MAC0422 - Sistemas Operacionais

Thiago Cunha Ferreira e Eduardo Freire de Carvalho Lima

1 Introdução

Este relatório descreve o que foi feito para o terceiro EP (Exercício-Programa) da matéria *MAC0422 - Sistemas Operacionais*, onde tivemos que:

- Alterar o gerenciamento de memória do SO, substituindo a forma de como escolhemos os buracos de memória livre para alocação, de estratégia *first fit* para *worst fit*;
- Criar um programa de análise estatística sobre os buracos na memória, analisando suas informações a cada segundo.

IMPORTANTE: DURANTE A INICIALIZAÇÃO DO MINIX, NÃO SELECIONE AS OPÇÕES 1 OU 2 DE INICIALIZAÇÃO. ESPERE OU USE A OPÇÃO 3.

O trecho de código modificado para essa parte foi marcado com uma longa sequência do símbolo # em forma de comentários. Porém, diversos outros trechos de código do MINIX não relacionados com esse EP estarão marcados da mesma forma, visto que o EP passado também exigiu a mesma marcação sobre códigos alterados e supõe-se que devemos usar as imagens anteriores de outros EP's (como visto no primeiro detalhe administrativo do exercício), a menos que claramente especificado. Assim, é importante ressaltar que **os únicos arquivos modificados relevantes para este EP são os que são mencionados na seção *Informações adicionais* deste relatório**.

2 Gerenciador de memória

Primeiramente, é importante notar que o MINIX usa um sistema de memória real com swapping e mantém uma lista ligada de "buracos", espaços de memória não alocados no momento.

Para essa primeira parte, tivemos que alterar a forma como o SO escolhe o espaço de memória para alocar novos programas, da estratégia *first fit*, que escolhe o primeiro espaço de memória encontrado maior ou igual ao tamanho requesitado, para a *worst fit*, que escolhe o maior espaço livre de memória. Para isso, ao invés de escolher o primeiro buraco encontrado com tamanho maior ou igual ao requisitado, percorre-se toda a lista e tenta achar o maior buraco, sempre deixando um ponteiro para o maior atual e o buraco anterior. Quando percorremos a lista toda, se o maior buraco for suficiente para a quantidade de memória requisitada, o usamos (aplicamos o mesmo conjunto de operações sobre o buraco que a estratégia passada).

3 Programa estatístico

Após essas mudanças, tivemos que produzir um programa, *memstat.c*, que mede algumas estatísticas de uso de memória, como número de buracos e a media, mediana e desvio padrão dos buracos, a cada segundo. Para isso, foi utilizada as informações adquiridas pela chamada de

sistema *getsysinfo*($PM\_PROC\_NR, SI\_MEM\_ALLOC, \&pmi$) de tal forma que ela armazena as informações sobre os buracos (na forma de um vetor) em uma estrutura específica.

Nesse programa, alocamos um vetor de tamanho igual ao número de buracos com base e tamanho diferentes de 0 (acessados pelo vetor citado anteriormente) que carregará o tamanho de buracos válidos, **em bytes**. Tal operação se deve à presença de vários posições desse vetor apresentarem buracos de tamanho ou base 0. Para a alocação, precisamos primeiramente percorrer o vetor de buracos para saber quantos elementos o vetor de tamanhos de buracos irá ter.

Após ter coletado o número de buracos e seus tamanhos, chamamos funções auxiliares que calculam os dados necessários. **Importante frisar que a implementação da mediana ordena o vetor para conseguir esse dado**. Entretanto, para o programa em questão, isso não afeta o resultado final.

A impressão dos resultados se dá, a cada segundo, em uma linha diferente, sendo a primeira correspondente ao cabeçalho, que indica a ordem de impressão das informações: **NÚMERO DE BURACOS - MÉDIA - DESVIO PADRÃO - MEDIANA**. Como não foi especificado uma medida para os tamanhos dos buracos, as informações relacionadas a eles estão na **unidade KB**, o que facilita a análise das informações adquiridas.

Obs: nessa etapa do programa, os buracos foram convertidos da unidade phys_clicks *para* bytes *através da operação de deslocamento de bits para a esquerda. Tal exemplo de uso dessa conversão pode ser observada no arquivo /usr/src/servers/is/dmp_pm.c, na função* holes_dmp. *Como um programa de usuário não tem acesso à constante que especifica o número de bits necessários de serem deslocados para essa operação, utilizou-se o valor correspondente para o sistema atual, que aqui é 12, como pode ser conferido no arquivo /usr/src/include/minix/const.h entre as linhas 66 e 77.*

4 Informações adicionais

- Nessa etapa, só foi modificado o arquivo */usr/src/servers/pm/alloc.c*, que contém o gerenciador dos buracos de memórias que foi alterado. Os outros arquivos relevantes foram adicionados, tanto o nosso */root/memstat.c* quanto os arquivos de teste inseridos no sistema e os resultados da execução do *script* "mkmemuse".

- Como mencionado na seção anterior, **a primeira linha do programa corresponde a um cabeçalho**. Como não sabemos se as informações estatísticas serão alimentadas em outro programa, decidimos deixar essa linha para um melhor entendimento do usuário. **Porém, se necessário removê-la, basta remover ou comentar a linha 70 do programa *memstat.c***.