Projeto da Disciplina de Sistemas Operacionais (IBM1042)

Prof. Dr. Cléver Ricardo Guareis de Farias

1. Objetivo

Implementar em Java um sistema de gerenciamento de memória contígua com alocação dinâmica de partições. Este sistema deverá ser utilizado para simular a alocação e a liberação de espaços de memória de forma simplificada.

2. Visão geral

A aplicação será formada por uma interface (gráfica ou linha de comando) com o usuário (UserInterface) e pelo gerente de memória (ContiguousAllocationManager) (veja Figura 1). O gerente de memória implementa a interface ManagementInterface, a qual é utilizada pelo usuário para interagir com o gerente.

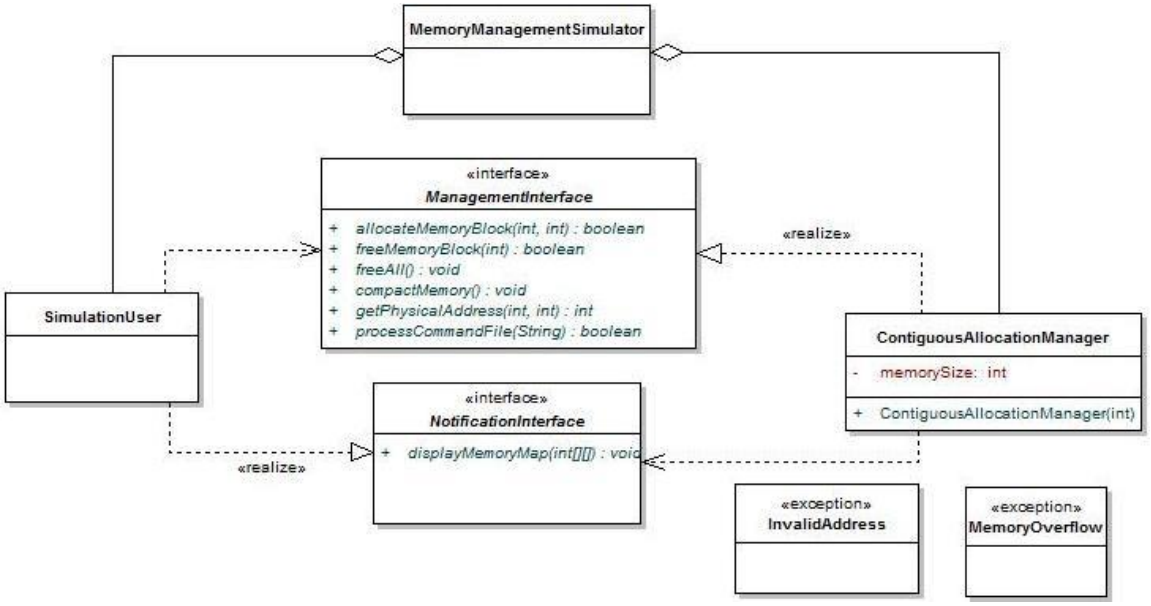

Figura 1. Arquitetura do sistema de gerência de arquivos

O gerente de memória controla inicialmente um único bloco de memória (tamanho máximo fixo e definido pelo construtor do gerente de memória). A cada requisição de alocação de memória o gerente deve verificar se há memória suficiente para atender à requisição. Caso não haja memória, o gerente deve lançar uma exceção (MemoryOverflow) para indicar que a solicitação excede a capacidade de memória disponível. Caso contrário, o gerente deverá tentar localizar, segundo um determinado critério, um bloco de memória livre grande o suficiente para atender à requisição. Neste caso, um novo bloco deve ser criado a partir da divisão de um bloco pré-existente. Na divisão, a parte de menor endereçamento deve ser alocada para atender à solicitação, enquanto que a parte de maior endereçamento permanece disponível (como um novo bloco) para atender outras

requisições. Cada bloco de memória contém três informações: identificador de processo ao qual o bloco está alocado (-1 se o bloco estiver livre), endereço base do bloco e tamanho do bloco. Caso a tentativa de localizar de um bloco grande o suficiente falhe, isto é, há memória disponível, mas não de forma contígua, a requisição falha (retorna falso). O sistema deverá retornar falso também caso haja a tentativa de alocar memória para um processo que já possui memória alocada.

A cada requisição de liberação de memória o gerente de memória deverá disponibilizar o bloco de memória liberado para novas requisições. Durante esta operação, o gerente deverá verificar se o bloco liberado é adjacente a algum outro bloco de memória disponível e, neste caso, agrupar os blocos em um único bloco.

As seguintes operações são definidas na interface ManagementInterface:

- allocateMemoryBlock, utilizada para alocar um bloco de memória para um processo;
- freeMemoryBlock, utilizada para liberar um bloco de memória utilizado por um processo;
- freeAll, utilizada para liberar todos os blocos de memória ocupados por processos;
- compactMemory, utilizada para redistribuir todo o conteúdo da memória de modo a criar um único bloco de memória livre (processos movidos para a baixa parte da memória);
- getPhysicalAddress, utilizada para traduzir um endereço lógico de um processo em um endereço físico correspondente;
- processCommandFile, utilizada para processar um arquivo texto contendo a especificação de um conjunto de comandos. A Figura 2 apresenta o formato dos comandos que podem estar armazenados em um arquivo. Cada linha do arquivo deve armazenar um único comando.

```
command := allocateBlock <process_id> <size> |
           freeBlock <process_id> | freeAll | compact
```

Figura 2. Comandos válidos

A seguinte operação é definida na interface NotificationInterface:

- displayMemoryMap, utilizada notificar a alocação atual dos blocos de memória aos processos. Esta operação deverá ser invocada sempre que o gerente de memória processar uma requisição que altere a alocação (operações *allocateMemoryBlockk*, *freeMemoryBlock* e *freeAll*) ou a localização (operação *compactMemory*) dos blocos de memória. Para cada bloco de memória definido, será informado o processo ao qual o bloco está alocado (o número -1 deverá ser utilizado para representar um bloco livre), o endereço base do bloco e o tamanho do bloco.

3. Grupos e atribuições

As seguintes políticas de localização de blocos livres devem ser implementadas:

- FF: First-Fit
- NF: Next-Fit
- BF: Best-Fit

- WF: Worst-Fit

Cada grupo deverá implementar duas políticas de localização de blocos livres. A Tabela 1 apresenta as atribuições de cada grupo.

Tabela 1. Definição de Grupos

Grupo	Integrantes	Políticas
1	Diego Henrique Ferreira Emylin Paula Freire Sousa	FF, BF
2	Ana Katariny de Souza Cacheta Milena Gomes Delfini	FF, WF
3	Júlia Carmona Almeida Chaves Juliana Santos Silva	NF, BF
4	Denise Gazotto Dezembro Filipe Lôbo Del Monte	NF, WF
5	Matheus Garcia Brochi Rafael Macedo Pott	FF, BF
6	Danilo Sanches Lemes de Sousa Flavio Augusto Theodoro	FF, WF
7	Heloisa Ferreira Spagnoli Thais Sarraf Sabedot	NF, BF
8	Andre Luis Antoneli Senju José Carlos Bueno de Moraes	NF, WF
9	Gustavo Jorge Zanin Pedro César de Freitas Penarbel	FF, BF
10	Gabriel Andrade Gasparini Guilherme Del Tedesco Chagas	FF, WF
11	Thaís Danielle Miquelim	NF, BF
12	André Yuji Oku Emerson Takeshi Urushibata	NF, WF
13	Felipe Dominguez Crespo Hirata	FF, BF
14	Thiago Yuji Kusabara Tiago Mendes da Silva	FF, WF
15	Francisco Barbosa Junior Victor Silva Marassato	NF, BF

4. Observações Finais

O projeto deverá ser entregue até o dia 23 de junho de 2013 por e-mail (farias@ffclrp.usp.br).

Além dos aspectos funcionais do projeto, serão também avaliadas a estrutura, documentação e usabilidade da aplicação desenvolvida.