Sistemas Operacionais

Especificação do Trabalho Prático (Implementação)

Profa.: Aletéia Patrícia Favacho de Araújo

Orientações Gerais

O trabalho de implementação da disciplina de SO, a ser desenvolvido em grupo com três (03) componentes, compreenderá as seguintes fases:

- Estudo teórico relacionado ao assunto do trabalho;
- Apresentação da solução teórica dada ao problema;
- Implementação da solução proposta;
- Apresentação e explicação detalhada do código-fonte implementado;
- Relatório explicando o processo de construção e o uso da aplicação.

Orientações Específicas

1. Problema

Implementação de um **pseudo Sistema de Arquivo** multiprogramado. Esse Sistema de Arquivo deve permitir que cada processo possa criar e deletar arquivos. Assim, o sistema de arquivo fará a alocação do arquivo por meio do método de alocação definido, podendo ser Alocação Contígua, Alocação Encadeada padrão (ou seja, vamos considerar que 10% do tamanho do bloco será usado para armazenar o ponteiro para o próximo bloco) e Alocação Indexada padrão (ou seja, a alocação que usará sempre um bloco adicional do disco para armazenar os índices dos blocos). Contudo, o arquivo deve ser tratado como uma unidade de manipulação, ou seja, o arquivo deve ser gravado ou deletado por inteiro. Além disso, para que possamos ter a mesma tomada de decisão, vamos considerar que o algoritmo a ser usado no armazenamento do disco (embora não seja usual ser usado na alocação do disco) seja o *first-fit* (sempre considerando a busca a partir do primeiro bloco do disco).

Além disso, o sistema de arquivos deve garantir que os processos de tempo real possam criar (se tiver espaço) e deletar qualquer arquivo (mesmo que não tenha sido criado pelo processo). Por outro lado, os processos comuns do usuário, só podem deletar arquivos que tenham sido criados por eles, e podem criar quantos arquivos desejarem, no tamanho que for solicitado (se houver espaço suficiente). Os processos de usuário vão ser identificados pela prioridade 0, e os processos de usuário vão ter prioridades entre 1 e 9. O método de alocação a ser usado deve ser: 1-contígua, 2-encadeada, e 3-indexada. É importante lembrar que se a Alocação a ser usada for a Encadeada, para cada bloco de dado requerido, 10% dele está indicando o

próximo bloco. Logo, se o arquivo precisar de 3 blocos, ele ocupará 3 blocos inteiros, e mais 30% de um quarto bloco, ou seja, para alocar esse arquivo serão necessários 4 blocos no disco. Da mesma forma, se a alocação a ser usada for a indexada, sempre deve ser alocado um bloco a mais, que é o bloco de índice. Este bloco de índice deve ser o primeiro bloco a ser alocado deste arquivo, e ele de ser indicado na alocação do disco por meio da letra “I”.

Assim, após o pseudo Sistema de Arquivo executar todos os processos, ele deve mostrar na tela do computador um mapa com a atual ocupação do disco, descrevendo quais arquivos estão em cada bloco, e quais são os blocos vazios (identificados por 0). E se tiver bloco de índice, ele deve ser identificado por meio da letra “I”.

2. Interfaces de Entrada e Saída

O sistema de arquivos terá como entrada dois arquivos com extensão .txt. O primeiro é um arquivo contendo as informações dos processos a serem simulados. O segundo arquivo, também com extensão .txt, traz a descrição das operações a serem realizadas pelo sistema de arquivos. O primeiro conterá uma linha para cada processo a ser “simulado”, e em cada linha terá o ID do processo, a prioridade do processo, e o tempo de CPU a ser dado para cada processo. O segundo arquivo .txt terá o tipo de alocação a ser usada (1-contígua, 2-encadeada ou 3-indexada), a quantidade total de blocos no disco, a especificação dos segmentos ocupados por cada arquivo, as operações a serem realizadas por cada processo.

No primeiro arquivo cada linha contém as informações de um único processo. Portanto, um arquivo com 2 linhas deve simular a existência de 2 processos, outro arquivo com 5 linhas fará com que o programa simule 5 processos, e assim por diante. Cada processo, portanto, cada linha, deve conter os seguintes dados:

<ID>, <prioridade>, <tempo de processador>

No segundo arquivo, cuja extensão também deve ser .txt, haverá a quantidade total de blocos no disco, a quantidade de segmentos ocupados, a identificação de quais arquivos já estão gravados no disco, a localização dos blocos usados por cada arquivo, a identificação de qual processo efetuará cada operação, a identificação das operações, sendo **código 0** para criar um arquivo, e **código 1** para deletar um arquivo.

Além disso, para as operações de criação deve constar o nome do arquivo a ser criado, e a quantidade de blocos ocupada pelo arquivo. Por outro lado, na operação de deletar um arquivo, deve constar apenas o nome do arquivo a ser deletado. Dessa forma, a organização interna do arquivo de entrada deve ser:

- Linha 1: Tipo de Alocação do Arquivo (sendo 1 “Alocação Contígua”, 2 “Alocação Encadeada”, 3 “Alocação Indexada”);

- Linha 2: Quantidade de blocos no disco simulado;

- Linha 3: Quantidade de arquivos já gravados no disco (n);

- A partir da Linha 4 até Linha *n* + *3*: arquivo (a ser identificado por uma letra), número do primeiro bloco gravado, quantidade de blocos ocupados por este arquivo. Para estes arquivos deve sempre ser considerada a Alocação Contígua;

- A partir da Linha n + 3: cada linha representa uma operação a ser efetivada pelo sistema de arquivos. Para isso, essas linhas vão conter: <ID_Processo>, <Código_Operação>, <Nome_arquivo>, <se_operacaoCriar_numero_blocos>.

3. Exemplo de Execução do Pseudo Sistema de Arquivos

Considere que os arquivos passados para entrada tenham sido os arquivos *processes.txt* e *files.txt*, conforme mostrados abaixo. O comando para execução deve ser:

Nome_Arquivo_Executavel *processes.txt files.txt*

Assim, sendo, considere que o arquivo *processes.txt* tenha duas linhas:

1, 0, 3
2, 1, 2

E o arquivo *files.txt* contém as linhas:

1
10
3
X, 0, 2
Y, 3, 1
Z, 5, 1
1, 0, A, 5
1, 1, X
0, 0, B, 2
1, 0, D, 3
2, 0, E, 2
1, 1, A
2, 1, A

Assim, esses arquivos, ao serem usados como entrada para o programa desenvolvido, devem garantir que seja exibido na tela, o mapa atual do disco simulado e a quantidade de blocos livres do disco, ou seja, algo como:

Mapa de ocupação do disco:

X	X	X	Y	0	Z	0	0	0	0

Quantidade de Blocos Livres: 5

Em seguida, após a execução de todos as operações, o sistema deve mostrar na tela o resultado de cada operação realizada ou não. Indicando que a operação teve sucesso, quando for o caso. E mostrando o erro, quando a operação não puder ser realizada, conforme mostrado a seguir:

Sistema de arquivos

1º Operação - Operação 1 do Processo 1 – Criar o arquivo A => Falha
O processo 1 não pode criar o arquivo A (falta de espaço)
(esta parte não precisa aparecer...não há espaço porque o arquivo indicou, na primeira linha, que a Alocação deve ser contígua - 1 na primeira linha).

2º Operação - Operação 2 do Processo 1 – Deletar o arquivo X => Sucesso
O processo 1 deletou o arquivo X.
(esta parte não precisa aparecer...o Processo 1 é do tipo "tempo real", por isso ele conseguiu deletar um arquivo, mesmo que este arquivo não tenha sido criado por ele).

3º Operação - Operação 1 do Processo 0 – Criar o arquivo B => Falha
O processo 0 não existe
(esta parte não precisa aparecer...não tem nenhum processo com ID 0 no arquivo de processos), logo ele não pode executar nenhuma operação.

4º Operação - Operação 3 do Processo 1 – Criar o arquivo D => Sucesso
O processo 1 criou o arquivo D (blocos 0, 1 e 2).

5º Operação - Operação 1 do Processo 2 – Criar o arquivo E => Sucesso
O processo 2 criou o arquivo E (blocos 6 e 7).

6º Operação - Operação 4 do Processo 1 – Deletar o arquivo A ==> Falha
O processo 1 já encerrou a sua execução.
(esta parte não precisa aparecer...o processo 1, conforme o arquivo de processos, tem apenas 3 tempos de CPU. Logo, após executar 3 operações ele é encerrado. Assim, não tem como ele executar uma quarta operação).

7º Operação - Operação 2 do Processo 2 – Deletar o arquivo A ==> Falha
O Processo 2 não pode deletar um arquivo que não foi criado por ele.
(esta parte não precisa aparecer...o Processo 2 é processo do tipo "usuário", por isso ele só pode deletar os arquivos que tiverem sido criados por ele).

Mapa de ocupação do disco:

D	D	D	Y	0	Z	E	E	0	0

Quantidade de Blocos Livres: 3

4. Estudo Teórico para a Solução

Cada grupo deverá buscar a solução para a sincronização das tarefas, quando se fizer necessário, de acordo com o problema proposto. É responsabilidade de cada grupo estudar a maneira mais eficiente para implementar o pseudo Sistema de Arquivo. Contudo, é importante ressaltar que para o problema de compartilhamento de recursos não há soluções mágicas, as soluções possíveis são exatamente as mesmas vistas em sala de aula: semáforos (baixo nível), e monitores (alto nível, mas devem ser implementados pela linguagem de programação, pois o compilador deve reconhecer o tipo monitor).

5. Implementação

O trabalho valerá 10 pontos, sendo 9 pontos atribuídos para a precisão dos algoritmos (de acordo com a especificação definida nas seções anteriores), e 1 ponto dado à qualidade do código. Portanto, o código-fonte deve seguir as boas práticas de programação: nomes de variáveis auto-explicativos, código comentado e identado. As métricas de medida a serem usadas para avaliar a qualidade do código serão baseadas naquelas descritas em [2]. Dessa forma, funções grandes, classes com baixa coesão, código mal-identado e demais práticas erradas acarretarão no prejuízo da nota.

Contudo, é fundamental que apenas os padrões das linguagens, sem bibliotecas de terceiros, sejam utilizados. Ou seja, se você for executar um programa escrito em Java no Linux, que eu precise apenas do SDK padrão para recompilá-lo, se o programa for escrito em C, que eu possa compilá-lo com o padrão -ansi ou c99, etc. As especificações de padrão devem ser explicitadas. Isto é para evitar problemas de executar na máquina de vocês e não executar na minha. Padronização é fundamental!

A implementação poderá ser feita em qualquer linguagem e utilizando qualquer biblioteca, desde que o código seja compatível com ambiente UNIX. Programas que utilizarem bibliotecas ou recursos adicionais deverão conter no relatório informações detalhadas das condições para a reprodução correta do programa. Além disso, bibliotecas proprietárias não são permitidas.

6. Responsabilidades

Cada grupo é responsável por realizar, de maneira exclusiva, toda a implementação do seu trabalho e entregá-lo funcionando corretamente. Além disso, o grupo deverá explicar detalhadamente todo o código-fonte desenvolvido. É fundamental que ***todo o grupo*** esteja presente no dia/horário definido para apresentar e explicar o código-fonte. A ausência de qualquer aluno, na data/horário da apresentação do seu grupo, implicará em ausência de nota para este aluno nesta atividade.

7. Material a ser entregue

O grupo deverá, além de apresentar o código-fonte executando, entregar todos os arquivos do código desenvolvido e entregar um relatório (mínimo de 2 e máximo de 5 páginas) contendo, obrigatoriamente, **no mínimo**, os seguintes itens:

- Descrição das ferramentas/linguagens usadas;
- Descrição teórica e prática da solução dada;
- Descrição das principais dificuldades encontradas durante a implementação;
- Para todas as dificuldades encontradas, explicar as soluções usadas;
- Descrever o papel/função de cada aluno na realização do trabalho;
- Bibliografia.

Assim, cada grupo deverá submeter no Aprender 3 dois arquivos. Um arquivo .zip com todos os arquivos necessários para executar o código, e um arquivo .pdf com o relatório sobre o desenvolvimento realizado.

8. Cronograma

O trabalho deverá ser entregue/apresentado no dia **14/09/2022** (Grupos 1, 2 e 3), e no dia **16/09/2022** (Grupos 4 e 5) . A ordem de apresentação será por meio da ordem numérica de cada grupo. A apresentará de cada grupo será pela plataforma ZOOM, e cada grupo terá uma sala virtual para realizar a apresentação. A apresentação será no horário da aula, ou seja, das 8:00 às 9:50. Nenhum trabalho será recebido fora do prazo, por qualquer que seja o motivo. Assim, os grupos devem apresentar exatamente no dia especificado anteriormente. É obrigatória a presença de **todo o grupo** para a explicação do código-fonte desenvolvido e das decisões tomadas durante o desenvolvimento.

9. Referências

[1] Stallings, W. *Operating Systems Internals and Design Principles*, Pearson Education, 2009.

[2] Martin, R. C. *Clean Code. A Handbook of Agile Software Craftsmanship*, Prentice Hall, 2008.