Pontifícia Universidade Católica de Minas Gerais
Departamento de Engenharia de Software e Sistemas de Informação
Disciplina: Sistemas Operacionais
Professor: Lesandro Ponciano

Trabalho Prático – MY Micro File System (mymfs)

1. Objetivos

Permitir que os alunos exercitem seus conhecimentos sobre os conceitos e os algoritmos estudados na disciplina. Possibilitar que o aluno implemente mecanismos e políticas fundamentais para o funcionamento de sistemas operacionais, incluindo controle e gerência de recursos.

2. O que deve ser feito

Os alunos devem implementar um sistema de arquivos chamado *mymfs*, que executa em uma camada acima do sistema de arquivos local da máquina. A implementação deve ser feita em linguagem C ou C++. Implementações em outras linguagens não serão aceitas, exceto linguagem de máquina colocada dentro do código C/C++ para otimização de desempenho. Pode-se usar bibliotecas C e C++. Em casos de dúvidas nesse aspecto, o professor deve ser consultado.

O mymfs deve seguir as seguintes restrições associadas à sua construção. Primeiro, ele executa em uma unidade específica, essa unidade é uma pendrive, aqui definida como unidade X. Segundo, internamente e de forma transparente ao usuário, ele é capaz de manipular arquivos de tamanho máximo de 500KB. Ou seja, internamente, e de forma transparente, dentro da unidade X não pode existir nenhum arquivo com tamanho superior a 500KB. Terceiro, todo e qualquer dado que precise ser armazenado pelo mymfs deve ser armazenado na unidade X. Quarto, todos os dados de controle usados pelo mymfs devem ficar dentro da unidade X, em um único arquivo chamado mymfs.config, e em formato texto; esse arquivo deve ter tamanho máximo de 50KB. Quinto, o arquivo de código executável deve ter nome mymfs.

O mymfs deve prover ao usuário as seguintes funcionalidades: config, import, listall, export, nlines, remove, removeall, grep, head100 e tail100. Cada uma dessas funcionalidades são executadas de forma parametrizada na chamada do executável. Seguir a especificação a seguir:

Nome: config
Comando de execução: `mymfs.exe X config`
Pré-condição: Não existe nenhum arquivo na unidade X. Não existe um sistema de arquivos mymfs na unidade X.
Pós-condição: Existe um sistema de arquivos mymfs na unidade X. No terminal, confirmação da configuração ou erro associado à pré-condição.

Nome: import
Comando de execução: `mymfs.exe X import file.txt`
Pré-condição: Existe um arquivo file.txt no local indicado fora do mymfs. Existe um sistema de arquivos mymfs na unidade X. Não existe um arquivo file.txt, pelo mymfs, dentro da unidade X.
Pós-condição: O arquivo file.txt está pelo mymfs dentro da unidade X. No terminal, confirmação da importação ou erro associado à pré-condição.

Nome: listall
Comando de execução: `mymfs.exe X listall`
Pré-condição: Existe um sistema de arquivos mymfs na unidade X.
Pós-condição: Os nomes dos arquivos existentes na unidade X, pelo mymfs, estão listados no terminal, um por linha. (Note que são os arquivos percebidos pelo usuário). O estado da unidade X pelo mymfs não foi alterado.

Nome: export
Comando de execução: `mymfs.exe X export file.txt C:/file.txt`
Pré-condição: Existe um sistema de arquivos mymfs na unidade X. O arquivo file.txt está, pelo mymfs, dentro da unidade X. Não há um arquivo C:/file.txt.
Pós-condição: O estado da unidade X pelo mymfs não foi alterado. O arquivo file.txt está em "C:" No terminal, confirmação da exportação ou erro associado à pré-condição.

Nome: remove
Comando de execução: `mymfs.exe X remove file.txt`
Pré-condição: Existe um sistema de arquivos mymfs na unidade X. O arquivo file.txt está, pelo mymfs, dentro da unidade X.
Pós-condição: Remove o arquivo file.txt da unidade X. O estado da unidade X pelo mymfs foi alterado.

Nome: removeall
Comando de execução: `mymfs.exe X removeall`
Pré-condição: Existe um sistema de arquivos mymfs na unidade X.
Pós-condição: Remove todos os arquivos que estão, pelo mymfs, na unidade X. O estado da unidade X pelo mymfs foi alterado. No terminal, confirmação da remoção ou erro associado à pré-condição.

Nome: head100
Comando de execução: `mymfs.exe X read100 file.txt`
Pré-condição: Existe um sistema de arquivos mymfs na unidade X. O arquivo file.txt está, pelo mymfs, dentro da unidade X.
Pós-condição: No terminal, as 100 primeiras linhas do arquivo file.txt estão exibidas ou está exibido erro associado à pré-condição. O estado da unidade X, pelo mymfs, não foi alterado.

Nome: tail100
Comando de execução: `mymfs.exe X tail100 file.txt`
Pré-condição: Existe um sistema de arquivos mymfs na unidade X. O arquivo file.txt está, pelo mymfs, dentro da unidade X.
Pós-condição: No terminal, as 100 últimas linhas do arquivo file.txt estão exibidas ou está exibido erro associado à pré-condição. O estado da unidade X, pelo mymfs, não foi alterado.

Nome: grep
Comando de execução: `mymfs.exe X grep word file.txt`
Pré-condição: Existe um sistema de arquivos mymfs na unidade X. O arquivo file.txt está, pelo mymfs, dentro da unidade X.
Pós-condição: Se exite word no arquivo file.txt, está escrito no terminal "Encontrado" seguido do número da primeira linha do arquivo no qual word foi encontrado. Se não existe word no arquivo, está escrito no terminal apenas "Não encontrado". Se alguma pré-condição não foi satisfeita, um erro está exibido no termina. O estado da unidade X, pelo mymfs, não foi alterado.

3. Aquecimento

Como tarefa de aquecimento, os alunos devem implementar (e deixar totalmente funcional como especificado na Seção 2) as funcionalidades config, import, listall, export. Naturalmente, para essas funcionalidades ficarem totalmente operacionais, toda a estrutura do código precisará ser feita.

4. O que deve ser entregue

- Todo o projeto da implementação. Código fonte.
- Um relatório até 3 páginas descrevendo as ideias utilizadas nas implementações para que elas ficassem mais eficientes, as bibliotecas pesquisadas, os métodos dessas bibliotecas que foram

utilizados, as dificuldades experimentadas durante a implementação, e as análises realizadas para saber se o código implementado está correto.

5. O que será avaliado

Será avaliado se o código está correto ou não. Um código será considerado correto se ele atende ao que foi especificado na Seção 2. Qualidade do relatório e da apresentação de acordo com o que foi especificado nas Seções 2 e 4. Se ocorrer qualquer evidência de cópia de trabalhos, receberão nota zero todos os membros das equipes em que os trabalhos estiverem envolvidos em cópia.

É esperado que o aluno empregue boas práticas de engenharia de software incluindo a legibilidade do código em termos de comentários, variáveis com nomes significativos, métodos com nomes significativos. É fortemente recomendado o versionamento de código no GitHub, mantendo o repositório privado para a equipe durante o desenvolvimento. Dica: o uso de padrões de projeto pode simplificar muito o trabalho de desenvolvimento e reduzir drasticamente a quantidade de código que precisa ser escrita.

5. Equipe, avaliação e prazos

- Tamanho da equipe: 2 membros
- Valor do aquecimento: 5 pontos
- Valor do trabalho integral: 10 pontos
- Data de entrega da tarefa de “Aquecimento”: 07/05/2019 (pelo SGA)
- Data da entrega final do código e do relatório: 02/06/2019 (pelo SGA)
- Data das apresentações dos trabalhos: 03/06/2019

Serão atribuídos 2 pontos extras a cada membro da equipe que implementar as operações de modo mais eficiente. Haverá um arquivo de lote com uma série de operações no sistema de arquivos. Será considerada vencedora a implementação que permitir que, em média, todas as operações sejam executadas no menor tempo. Note que, por se tratar de uma avaliação de eficiência, todo o código deve estar correto para se habilitar a participar da competição pelos pontos extras. Não é possível bonificar uma implementação que está rápida, mas está errada.

Os casos omissos, não previstos nesta especificação, serão solucionados pelo professor.