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

1. Primeiro, ele executa em um conjunto de unidades de armazenamento. Esse conjunto tem no mínimo 3 e no máximo N unidades. Cada unidade é uma pendrive independente. O conjunto de unidades é aqui definida como unidade raid X. Na prática, X é um RAID 5.
2. Segundo, internamente e de forma transparente ao usuário, o mymfs é capaz de manipular arquivos de tamanho máximo de 500KB. Ou seja, internamente, e de forma transparente, dentro da unidade raid X não pode existir nenhum arquivo com tamanho superior a 500KB.
3. Terceiro, todo e qualquer dado que precise ser armazenado pelo mymfs deve ser armazenado na unidade raid X. A única exceção é o arquivo mymfs.temp que pode estar em uma pasta temp da máquina e neste arquivo pode conter apenas os nomes das unidades que compõem a unidade raid X.
4. Quarto, todos os dados de controle usados pelo mymfs (exceto o especificado no item acima) devem ficar dentro da unidade raid X, em um único arquivo chamado mymfs.config, e em formato texto; esse arquivo deve ter tamanho máximo de 50KB.
5. 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` D E F
Pré-condição: Não existe nenhum arquivo nas unidades D, E e F. Não existe um sistema de arquivos raid X mymfs nas unidades D, E, F. (note que D, E e F é apenas um exemplo, podem ser outros nomes e quantidade maior que 3)
Pós-condição: Existe um sistema de arquivos mymfs na unidade raid X, que é composta das unidades D, E, F. 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 rai X. Não existe um arquivo file.txt, pelo mymfs, dentro da unidade raid X.

Pós-condição: O arquivo file.txt está pelo mymfs dentro da unidade raid 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 raid X.
Pós-condição: Os nomes dos arquivos existentes na unidade raid X, pelo mymfs, estão listados no terminal, um por linha. (Note que são os arquivos percebidos pelo usuário). O estado da unidade raid 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 raid X. O arquivo file.txt está, pelo mymfs, dentro da unidade raid X. Não há um arquivo C:/file.txt.
Pós-condição: O estado da unidade raid 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 raid X. O arquivo file.txt está, pelo mymfs, dentro da unidade raid X.
Pós-condição: Remove o arquivo file.txt da unidade raid X. O estado da unidade raid X pelo mymfs foi alterado.

Nome: removeall
Comando de execução: `mymfs.exe X removeall`
Pré-condição: Existe um sistema de arquivos mymfs na unidade raid X.
Pós-condição: Remove todos os arquivos que estão, pelo mymfs, na unidade raid X. O estado da unidade raid 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 raid X. O arquivo file.txt está, pelo mymfs, dentro da unidade raid 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 raid 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 raid X. O arquivo file.txt está, pelo mymfs, dentro da unidade raid 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 raid 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 raid X. O arquivo file.txt está, pelo mymfs, dentro da unidade raid 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 raid X, pelo mymfs, não foi alterado.

Como ponto de partida para a implementação, sugere-se que usem o código dos alunos Gabriel Henrique

Souza Haddad Campos e Talita Arantes Melo (https://github.com/Haddadson/mymfs) da versão anterior desta disciplina, primeiro semestre de 2019. O código possui todas as operações implementadas, mas considera que o mymfs consiste em apenas 1 dispositivo e também não implementa confiabilidade e desempenho por meio do RAID 5. Apesar disso, o código pode ser estendido/evoluído para satisfazer os novos requisitos estabelecidos neste documento.

3. Confiabilidade e Desempenho

Na prática, X é um RAID 5, que tolera falha em até 1 das unidades que o compõe. Ou seja, pode-se remover 1 pendrive e nenhum dado será perdido, pois há redundância de acordo com o definido do RAID 5. No entanto, em vez de bit de paridade por setor, nossa versão simplificada do RAID funciona com compactação zip por arquivo interno do mymfs.

O RAID 5 também implementa desempenho por meio de paralelismo, então deve-se implementar um esquema multithread que faz leitura e escrita simultânea nas diversas unidades que compõe o RAID.

4. Aquecimento

Como tarefa de aquecimento, os alunos devem implementar as funcionalidades config, import, listall, export. Elas precisam estar funcionais, com a distribuição dos dados em diversas unidades e a tolerância a falhas, mas ainda não é necessário lidar com leituras e escritas simultâneas.

5. O que deve ser entregue

- Todo o projeto da implementação. Código fonte. Pode ser o link para o repositório no GitHub. A implementação deve ser *opensource* e seu código fonte deve ser entregue com uma documentação com as devidas informações, incluindo: a) o contexto do trabalho [universidade, curso/turno, disciplina/professor, aluno]; b) desenvolvedores; c) dependências; e d) licenciamento. A implementação deve ficar totalmente funcional.
- Vídeo de 3 a 5 minutos explicando a solução desenvolvida para o problema. No vídeo deve-se representar: a) o contexto do trabalho [universidade, curso/turno, disciplina/professor, aluno]; b) as principais ideias utilizadas nas implementações para que elas ficassem mais eficientes; c) as bibliotecas pesquisadas, os métodos dessas bibliotecas que foram utilizados; d) as análises realizadas para saber se o código implementado está correto; e) os pontos fortes e fracos da solução. Abaixo temos alguns exemplos de vídeos. Eles não são de trabalho de SO, mas por eles pode-se imaginar o formato do que se espera que seja entregue.
    - https://youtu.be/px9Ihw4WgNM (este feito fora da disciplina para um seminário)
    - https://www.youtube.com/watch?v=cAa18ZdLKtw (este feito na disciplina IHC)
    - https://www.youtube.com/watch?v=0agu4NU4BiA (este feito na disciplina IHC)

6. 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 e 3. Qualidade do relatório e da apresentação de acordo com o que foi especificado nas Seções 4 e 5. 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. O uso de teste unitário automatizado é obrigatório para todos os métodos que foram adicionados pelos alunos.

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.

7. 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": 30/10/2019 (pelo SGA)
- Data da entrega final do código e do relatório: 24/11/2019 (pelo SGA)
- Data da avaliação dos trabalhos em sala: 27/11/2019

8. Pontos Extras

Serão atribuídos 4 pontos extras a cada membro da equipe que implementar as operações de modo mais eficiente. Os dois critérios de eficiência são:

- Tempo
- Uso de espaço de armazenamento nas unidades que compõe a unidade X, medido por número total de arquivos.

Será considerada vencedora a implementação que permitir que, em média, todas as operações sejam executadas no menor tempo e com a menor soma total de espaço de armazenamento. Haverá um arquivo de lote com uma série de operações no sistema de arquivos *mymfs* e é sobre essa série de operações que o tempo será calculado. Se a equipe ganhar nos 2 critérios, cada membro ganha 4 pontos. Se a equipe ganhar em apenas 1 critério, cada membro ganha 2 pontos. Em caso de empate em um dado critério, os pontos extras do critério não serão concedidos.

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. Por exemplo, não é possível bonificar uma implementação que está rápida, mas está errada.

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