Universidade Federal de Minas Gerais Departamento de Ciência da Computação	Estrutura de Dados Profs Gisele Pappa e Wagner Meira Jr

Trabalho Prático 1

Servidor de emails

Valor: 15 pontos
Data de entrega: 16/10/2022

Neste trabalho, seu objetivo será implementar um simulador de um servidor de emails. O sistema simulado terá suporte ao gerenciamento de contas, assim como à entrega de mensagens de um usuário para outros. O foco da simulação será verificar o funcionamento correto do sistema ao executar as diversas operações do servidor (descritas a seguir) em diferentes situações.

1. Descrição

Suponhamos que você esteja desenvolvendo um novo servidor de emails para a Google, pois o servidor atual possui algumas falhas e consumo exagerado de memória. O servidor será usado por milhões de pessoas, e sabemos que um bom gerenciamento e aproveitamento de memória nesse caso é fundamental. Sua responsabilidade é fazer uma implementação eficiente e, munido do conhecimento de tipos abstratos de dados e alocação dinâmica de memória, fazer este trabalho será moleza!

A Google quer que o servidor dê suporte à entrega e consulta de mensagens de email para usuários cadastrados. Além disso, o sistema também deve suportar o gerenciamento (adição e remoção) de contas de usuários. Dessa forma, o servidor que você vai simular deverá suportar as seguintes operações:

- **Cadastrar novo usuário:** esta operação cria uma caixa de entrada para o novo usuário. A nova caixa de entrada deverá estar vazia, ou seja, sem nenhuma mensagem armazenada. Após a criação da conta e enquanto ela não for removida, todas as mensagens enviadas para esse usuário devem ser armazenadas em sua caixa de entrada.
  - **Importante:** você **não** pode assumir um número predeterminado de contas de usuário. Isso significa que o sistema não deve ter um número fixo ou máximo de contas de usuário. Cada conta criada deve ser alocada dinamicamente, e a memória utilizada por ela deve ser liberada se a conta for removida.

- **Remover usuário:** esta operação remove um usuário do sistema. Todas as mensagens em sua caixa de entrada devem ser automaticamente apagadas. Após a conta ser removida, mensagens recebidas pelo servidor e destinadas a esse usuário não devem ser aceitas.

- **Entregar email:** esta operação recebe um novo email destinado a determinado usuário. Caso o usuário não exista (ou seja, não tenha criado uma conta ou essa tenha sido removida), a mensagem deve ser recusada e gerar um erro. Se o usuário possuir uma caixa de entrada no servidor, a mensagem deve ser armazenada *em ordem de prioridade* na caixa de entrada desse usuário. Isso quer dizer que mensagens com prioridade mais alta devem ser armazenadas à frente de mensagens com prioridade mais baixa. Caso dois emails tenham a mesma prioridade, aquele que foi recebido primeiro pelo servidor deverá ficar à frente.

- **Consultar email:** esta operação recebe o identificador de um usuário e deverá retornar a próxima mensagem na caixa de entrada do usuário, *segundo a prioridade* descrita acima. Caso o usuário não exista, o servidor deve retornar um erro. Em caso de sucesso, a mensagem retornada deverá ser removida da caixa do usuário.

2. O que deve ser feito?

Você deverá implementar um simulador do sistema descrito acima. O simulador deverá suportar todas as quatro operações. Para cada operação processada, será gerada uma saída. Cada operação pode gerar um conjunto definido de saídas, especificado abaixo. O formato da saída deve corresponder **exatamente** ao formato especificado, pois será através dela que a correção do simulador será feita.

A entrada do programa de simulação será um arquivo de texto contendo, em cada linha, uma operação. A simulação deve terminar quando o final do arquivo for alcançado. Para cada linha do arquivo de entrada, ou seja, para cada operação, deverá ser impressa uma linha na saída padrão (na tela), conforme especificado abaixo. A seguir, estão especificadas as operações, conforme presentes no arquivo de entrada:

- **CADASTRA ID**
    - Operação de cadastro de novo usuário. O parâmetro ID é um valor inteiro, que tem valor $0 \leq ID \leq 10^6$, e é o identificador do novo usuário.
    - O servidor deve criar uma nova caixa de entrada vazia para o usuário ID. Enquanto essa conta não for removida, mensagens recebidas para o usuário ID deverão ser armazenadas nessa caixa de entrada, segundo a ordem de prioridade já explicada.
    - Somente uma caixa de entrada deve existir para um mesmo ID, ou seja, uma operação de cadastro deve falhar se o ID fornecido já estiver em uso.
    - Saída esperada (*em todas elas, ID deve ser substituído pelo valor correspondente*):

        - OK: CONTA ID CADASTRADA - caso a conta seja cadastrada com sucesso;
        - ERRO: CONTA ID JA EXISTENTE - caso a conta já exista.
    - Exemplo: CADASTRA 5

- **REMOVE ID**
    - Operação de remoção de conta e usuário. O parâmetro ID é o identificador do usuário cuja conta deve ser removida.
    - A conta ID deve ser removida, e todas as mensagens na caixa de entrada desta conta devem ser apagadas.
    - Após a conta ser removida, deverá ser possível cadastrar uma nova conta com identificador ID.
    - Saída esperada (*em todas elas, ID deve ser substituído pelo valor correspondente*):
        - OK: CONTA ID REMOVIDA - caso a conta seja removida com sucesso;
        - ERRO: CONTA ID NAO EXISTE - caso a conta não exista.
    - Exemplo: REMOVE 5

- **ENTREGA ID PRI MSG FIM**
    - Operação de solicitação de entrega de uma nova mensagem. O parâmetro ID é o identificador do destinatário da mensagem. Caso esse usuário esteja cadastrado no servidor, a mensagem deve ser armazenada em sua caixa de entrada segundo as prioridades já especificadas.
    - O parâmetro PRI é um inteiro entre 0 e 9 e indica a prioridade da mensagem. Valores mais altos indicam maior prioridade.
    - MSG é a mensagem a ser recebida: um conjunto de palavras separadas por espaço em branco e terminadas com a palavra reservada FIM.
    - Saída esperada *(em todas elas, ID deve ser substituído pelo valor correspondente)*:
        - OK: MENSAGEM PARA ID ENTREGUE - caso a mensagem seja entregue com sucesso;
        - ERRO: CONTA ID NAO EXISTE - caso a conta do destinatário não exista.
    - Exemplo: ENTREGA 5 1 bom dia meu amigo, como voce esta? FIM

- **CONSULTA ID**
    - Operação de consulta de email. O parâmetro ID é o identificador do usuário para o qual a consulta foi feita.
    - Esta operação deve imprimir a primeira mensagem na caixa de entrada do usuário, segundo a ordem de prioridade já especificada. A mensagem deve então ser *removida* da caixa de entrada, de tal forma que a próxima consulta para esse mesmo usuário retorne a *próxima* mensagem segundo as mesmas prioridades.
    - Saída esperada (*em todas elas, ID deve ser substituído pelo valor correspondente*):

- CONSULTA ID: MSG - imprime a primeira mensagem na caixa de entrada do usuário, segundo a ordem de prioridade. Por exemplo, para o usuário do exemplo anterior e que possua apenas aquela mensagem em sua caixa de entrada: CONSULTA 5: bom dia meu amigo, como voce esta?;
  - CONSULTA ID: CAIXA DE ENTRADA VAZIA - caso não haja nenhuma mensagem na caixa de entrada especificada;
  - ERRO: CONTA ID NAO EXISTE - caso a conta não exista.
- Exemplo: CONSULTA 5

3. O que implementar:

Você deverá implementar pelo menos dois Tipos Abtratos de Dados (TADs)/ Classes: o primeiro para o email (ou mensagem) e o segundo para a caixa de entrada. Você precisará ainda de outros, que implementem estruturas de dados adequadas para armazenar emails na caixa de entrada (*) e para armazenadas as caixas de entrada (**).

* Lembre-se de que as mensagens devem ser armazenadas segundo a ordem de prioridade especificada, de tal maneira que uma consulta seja capaz de retornar a mensagem adequada (aquela de maior prioridade e que tenha sido recebida primeiro). Qual estrutura de dados é adequada para esta tarefa?

** Você deve armazenar somente tantas caixas de entrada quanto sejam as contas cadastradas. Um vetor não é uma estrutura de dados válida para esta tarefa, visto que é uma estrutura com tamanho fixo. Que estrutura você poderia utilizar, sendo capaz de adicionar ou remover elementos dinâmica e facilmente?

4. Entregáveis

Você deve utilizar a linguagem C ou C++ para o desenvolvimento do seu sistema. O uso de estruturas pré-implementadas pelas bibliotecas-padrão da linguagem ou terceiros é terminantemente vetado.

Você DEVE utilizar a estrutura de projeto abaixo junto ao Makefile disponibilizado no Moodle:

- TP

  |- src

  |- bin

  |- obj

  |- include

  Makefile

A pasta **TP** é a raiz do projeto; a pasta **bin** deve estar vazia; **src** deve armazenar arquivos de código (*.c, *.cpp, ou *.cc); a pasta **include**, os cabeçalhos (*headers*) do projeto, com extensão ***.h**, por fim a pasta **obj** deve estar vazia. O **Makefile** disponibilizado no *Moodle* deve estar na **raiz do projeto**. A execução do **Makefile** deve gerar os códigos objeto ***.o** no diretório **obj** e o executável do TP no diretório **bin**.

Existe no Moodle um vídeo onde um dos monitores do semestre passado discute sobre a função e uso de makefiles.

5. Documentação

A documentação do trabalho deve ser entregue em formato **pdf** e também **DEVE** seguir o modelo de relatório que será postado no Moodle. Além disso, a documentação deve conter **TODOS** os itens descritos a seguir **NA ORDEM** em que são apresentados:

Título, nome, e matrícula.
Introdução: Contém a apresentação do contexto, problema, e qual solução será empregada.
Método: Descrição da implementação, detalhando as estruturas de dados, tipos abstratos de dados (ou classes) e funções (ou métodos) implementados.
Análise de Complexidade: Contém a análise da complexidade de tempo e espaço dos procedimentos implementados, formalizada pela notação assintótica.
Estratégias de Robustez: Contém a descrição, justificativa e implementação dos mecanismos de programação defensiva e tolerância a falhas implementados.
Análise Experimental: Apresenta os experimentos realizados em termos de desempenho computacional e localidade de referência, assim como as análises dos resultados.
Conclusões: A Conclusão deve conter uma frase inicial sobre o que foi feito no trabalho. Posteriormente deve-se sumarizar o que foi aprendido.
Bibliografia: Contém fontes utilizadas para realização do trabalho. A citação deve estar em formato científico apropriado que deve ser escolhido por você.
Instruções para compilação e execução: Esta seção deve ser colocada em um apêndice ao fim do documento e em uma página exclusiva (não divide página com outras seções).
***Número máximo de páginas:** 10

A documentação deve conter a descrição do seu trabalho em termos funcionais, dando foco nos algoritmos, estruturas de dados e decisões de implementação importantes durante o desenvolvimento.
Evite a descrição literal do código-fonte na documentação do trabalho.
Dica: Sua documentação deve ser clara o suficiente para que uma pessoa (da área de Computação ou não) consiga ler, entender o problema tratado e como foi feita a solução.

6. Submissão

Todos os arquivos relacionados ao trabalho devem ser submetidos na atividade designada para tal no Moodle. A entrega deve ser feita **em um único arquivo** com extensão **.zip**, com nomenclatura nome_sobrenome_matricula.zip, onde nome, sobrenome e matrícula devem ser substituídos por suas informações pessoais. O arquivo **.zip** deve conter a sua documentação no formato **.pdf** e a estrutura de projeto descrita no início da Seção 4.

7. Avaliação

O trabalho será avaliado de acordo com:

- Corretude na execução dos casos de teste - (50% da nota total)
- Apresentação da análise de complexidade das implementações - (15% da nota total)
- Estrutura e conteúdo exigidos para a documentação - (20% da nota total)
- Indentação, comentários do código fonte e uso de boas práticas - (10% da nota total)
- Cumprimento total da especificação - (5% da nota total)

Se o programa submetido não compilar[1], seu trabalho não será avaliado e sua nota será 0. Trabalhos entregues com atrasos sofrerão penalização de 2^{d-1} pontos, com d = dias de atraso.

8. Considerações Finais

1. Comece a fazer esse trabalho prático o quanto antes, enquanto o prazo de entrega está tão distante quanto jamais estará.
2. Leia **atentamente** o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.
3. Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.
4. Plágio é CRIME. Trabalhos onde o plágio for identificado serão **automaticamente anulados** e as medidas administrativas cabíveis serão tomadas (em relação a todos os envolvidos). Discussões a respeito do trabalho entre colegas são permitidas. É permitido consultar fontes externas, desde que exclusivamente para fins didáticos e devidamente registradas na sessão de bibliografia da documentação. **Cópia e compartilhamento de código não são permitidos**.

[1] Entende-se por compilar aquele programa que, independente de erros no Makefile ou relacionados a problemas na configuração do ambiente, funcione e atenda aos requisitos especificados neste documento em um ambiente Linux.

FaQ (Frequently asked Questions)

1. **Posso utilizar alguma estrutura de dados do tipo Queue, Stack, Vector, List, e etc..., do C++? NÃO**
2. Posso utilizar o tipo String? SIM.
3. Posso utilizar o tipo String para simular minhas estruturas de dados? NÃO
4. Posso utilizar alguma biblioteca para tratar exceções? SIM.
5. Posso utilizar alguma biblioteca para gerenciar memória? SIM.
6. As análises e apresentação dos resultados são importantes na documentação? SIM.
7. Os meus princípios de programação ligados a C++ e relacionados a engenharia de software serão avaliados? NÃO
8. Posso fazer o trabalho em dupla ou em grupo? NÃO
9. Posso trocar informações com os colegas sobre a teoria? SIM.
10. Posso fazer o trabalho no Windows, Linux, ou MacOS? SIM.
11. Posso utilizar IDEs, Visual Studio, Code Blocks, Visual Code, Eclipse? SIM.