Instituto Federal do Espírito Santo — IFES (Campus Serra)
Coordenadoria de Informática — Cin

Trabalho de Programação 3

Simulação de Realização de Exames de Raio-X

Valor: 35 pontos.
Deadline: 11 de dezembro de 2023

Prof. Thiago M. Paixão
thiago.paixao@ifes.edu.br

1 Objetivo

Este trabalho consiste em simular o processo de realização de exames de raio-X em um hospital, com ênfase na organização da fila para laudo médico. A cada momento, pacientes chegam ao hospital e exames são realizados mediante disponibilidade de aparelhos. A IA[1] sugere diagnósticos preliminares e os exames são encaminhados para laudo de acordo com a disponibilidade da equipe de radiologistas. Métricas de desempenho, como tempo médio de laudo e exames realizados fora do prazo, são calculadas. O objetivo é entender o processo e avaliar a eficiência do sistema, contribuindo para futuras otimizações.

As principais competências a serem desenvolvidas neste trabalho incluem:

- Implementação de um sistema de simulação.
- Uso e implementação de filas.
- Implementação de módulos e TADS.
- Implementação de *logging* em arquivo texto.

2 Simulação do processo de realização de exames de raio-X

O processo de simulação é baseado em eventos que ocorrem a cada instante de tempo com uma certa probabilidade. No total, a simulação deve durar 43.200 unidades de tempo. O tempo máximo aceitável para realização de exames, para cômputo de métrica, é de 7.200 unidades de tempo. Os detalhes do processo de simulação são dados a seguir.

2.1 Dinâmica do processo de simulação

Chegada de pacientes A cada instante de tempo, no máximo 1 (um) paciente pode chegar ao hospital. A probabilidade de chegar um paciente em um instante de tempo é igual a 20%. Para cada novo paciente, é criado um registro com nome, cpf, idade e um identificador de sistema (id), e esse registro é inserido numa *lista de pacientes*. Essa lista funciona como um banco de dados em que os registros são removidos durante toda a execução do programa. Além disso, o id do paciente que chega é inserido na *fila para exame*.

Realização do exame de raio-X Havendo aparelho disponível (de um total de 5), o primeiro paciente (por meio do seu id) da *fila para exame* é alocado para realização do exame no aparelho. A duração do exame é de $t_{ex} \sim U(5, 10)$ unidades de tempo, ou seja, um número inteiro (pseudo-aleatório) gerado a partir de um distribuição uniforme com valores entre 5 e 10. Ao finalizar o exame, é gerado um registro de exame que é inserido ao final da *fila para laudo*. O registro contém o id do paciente, o

[1] Não implementaremos a IA propriamente dita, apenas simularemos seu resultado.

instante de tempo em que o registro foi gerado e a condição (normal ou patológica) sugerida pela IA (ver Tabela 1). Assumimos que aparelhos só ficam ociosos se a *fila para exame* estiver vazia.

Realização de laudos Para fins práticos, a realização de um laudo consiste em remover exames da *fila para laudo* e atualizar o status de conclusão dos exames. Havendo radiologista disponível (de um total de 3), o primeiro exame da *fila para laudo* é alocado para o radiologista livre que leva $t_{rad} \sim U(10, 30)$ unidades de tempo para preparar o laudo.

2.2 Lista de patologias

A lista de patologias, juntamente com o a probabilidade de ocorrência e o nível de gravidade é apresentada na Tabela 1. Os valores apresentados na tabela são fictícios e tem por objetivo simplesmente a realização da simulação.

Condição	Probabilidade de Ocorrência	Nível de Gravidade
Saúde Normal	0.3	1
Bronquite	0.2	2
Pneumonia	0.2	3
Fratura de Fêmur	0.15	4
Apendicite	0.15	4

Tabela 1: Lista de Patologias com Probabilidades de Ocorrência e Níveis de Gravidade.

2.3 Métricas de desempenho

Durante a simulação, várias métricas serão computadas para avaliar a eficiência e o desempenho do sistema. Estas métricas incluem:

Tempo médio de laudo Esta métrica calcula o tempo médio que os exames ocupam a *fila para laudo*. Quanto menor o tempo médio de espera, mais eficiente é o sistema.

Tempo médio de laudo por patologia Calcula o tempo médio que os exames de uma patologia específica aguardam na fila antes de serem laudados. Isso ajuda a identificar se alguma patologia está enfrentando atrasos significativos.

Quantidade de exames realizados após o limite de tempo estabelecido Esta métrica rastreia quantos exames foram laudados após o limite de tempo aceitável estabelecido pelo hospital. Isso ajuda a avaliar a capacidade do sistema de cumprir prazos críticos.

3 Requisitos do programa

3.1 Estrutura geral

Neste trabalho, você terá a flexibilidade de implementar os módulos e Tipos Abstratos de Dados (TADs) que considerar necessários para a simulação. No entanto, é fundamental que o arquivo `main.c` contenha a implementação da lógica principal da simulação, o que inclui o *loop* de contagem de tempo principal. A cada 10 unidades de tempo, o relatório das métricas deve ser atualizado na tela.

3.2 *Logging*

Além disso, é obrigatório implementar o registro de eventos significativos (*logging*), tais como a chegada de pacientes, o início/fim dos exames e o início/fim dos laudos. Cada novo evento deve ser registrado

em um TAD chamado `Log`. Ao concluir a execução, o registro de eventos deve ser armazenado em um arquivo denominado *log.txt*. O Código 1 fornece uma interface básica para um *log* com capacidade para 1000 eventos. Você pode partir dessa sugestão e fazer as alterações que julgar necessárias para registrar fatos relevantes da simulação (chegada de pacientes, realização de exames, relatório de métricas, etc.).

```c
typedef struct {
   char message[256];
   time_t timestamp;
} LogEvent;

typedef struct {
   LogEvent events[1000]; // Capacidade para 1000 eventos (ajuste conforme necessá
      ↪ rio)
   int count;
} Log;

Log* create_log();
void log_event(Log *log, const char *message);
void save_log_to_file(const Log *log, const char *filename);
```

Código 1: Exemplo de interface para o TAD LOG.

4 Critérios de avaliação

A avaliação deste trabalho levará em consideração os seguintes critérios:

1. Implementação e uso adequado de estruturas de dados em geral: Até **15 pontos** serão atribuídos à implementação correta das estruturas de dados necessárias para a simulação, como filas e listas, bem como ao seu uso adequado durante a simulação.
2. Lógica e organização do código: Até **5 pontos** serão concedidos pela clareza, organização e boas práticas de codificação no projeto. Isso inclui a estruturação adequada do código (modularização), nomes significativos para variáveis e funções, e formatação consistente.
3. Implementação do logging: Até **10 ponto** será atribuído pela implementação adequada do registro de eventos significativos (logging) no projeto.
4. Implementação das métricas: Até **2 pontos** serão atribuídos à implementação correta das métricas de desempenho especificadas no trabalho.
5. Documentação do código: Até **1 ponto** serão atribuídos à qualidade da documentação incorporada ao código. Certifique-se de incluir comentários significativos que expliquem a lógica por trás das implementações.
6. README.md descritivo: **Até 2 ponto** será concedido pela criação de um arquivo README.md descritivo e informativo no seu repositório. O README deve fornecer informações claras sobre como executar e utilizar o seu projeto.
7. Apresentação (se necessária) (α): Um fator real α será aplicado após a apresentação do projeto. α será um número entre 0 e 1, refletindo a qualidade da apresentação e a capacidade de explicar e defender o projeto.
8. Correção (β): Será atribuído $\beta = 1$ se não houver falhas críticas no projeto; caso contrário, $\beta = 0$. Isso avaliará se o projeto funciona conforme o esperado e se atende aos requisitos especificados.

9. Atraso (d): Não serão considerados envios fora do prazo.

A nota será calculada utilizando a equação $nota = (1 - d) \times \alpha \times \beta \times P$, onde P é a soma dos pontos dos critérios 1 a 4. É importante notar que a nota será zerada após 1 dia de atraso.

Importante: O programa será testado num ambiente Linux Ubuntu 22.04 com GCC 11. Recomendo FORTEMENTE desenvolver e testar nesse ambiente.

5 O que entregar?

1. Um link para um repositório .git com código fonte do projeto: `Makefile` e arquivos `.c` e `.h`.
2. A documentação/relatório será feita no arquivo README.md do repositório e deverá explicar o passo-a-passo para executar o programa, os principais tipos e estruturas de dados e as principais decisões de implementação.

Bom trabalho!