UNIVERSIDADE FEDERAL DE MINAS GERAIS

AUTOMAÇÃO EM TEMPO REAL

Trabalho Prático | Etapa 1

Marcone Márcio da Silva Faria

Mércia Caroline Nogueira Martins

Conteúdo:

1. Introdução:

O presente trabalho tem como contexto a integração das informações de um sistema de otimização de gas lift de uma das plataformas offshore de uma empresa petrolífera com as informações já existentes no sistema SCADA direcionando tudo para três painéis de projeção de imagens presentes na sala de controle da plataforma, apresentando dados do processo de extração do óleo cru (Terminal A), dados de otimização (Terminal B) e mensagens de alarmes industriais (Terminal C).

Para o funcionamento desses processos, será desenvolvido uma aplicação multithread responsável pela leitura de dados do sistema de otimização e do sistema SCADA e distribuí-los adequadamente para os três painéis de projeção. Nessa primeira parte do trabalho foi estruturada a arquitetura multithread da aplicação, o disparo da aplicação

executando processos e threads, a implementação do mecanismo de bloqueio/desbloqueio das threads e seus encerramentos mediante um conjunto de objetos do tipo evento, a implantação da lista circular em memória e as temporizações provisórias da tarefa de comunicação de dados.

2. Desenvolvimento:

2.1. Arquitetura:

A aplicação foi estruturada a partir da criação de threads e funções que juntos possibilitam a exibição das mensagens e alarmes na tela principal, bem como realizar o controle de exibição, geração e retirada dos mesmos. Abaixo podemos verificar uma descrição das funções criadas para a solução:

- **Criar Objetos:** chamado pela thread principal criando todos os objetos de sincronização utilizados na solução, como mutexes, semáforos e sinalizadores de eventos.
- **Fechar Handlers:** chamado pela thread principal fecha todos os handlers abertos pelos objetos durante a execução.
- **Criar Threads Secundárias:** chamado pela thread principal criando todas as threads secundárias que serão detalhadas a seguir.
- **Criar Processos de Exibição:** chamado pela thread principal criando os processos de exibição da solução possibilitando a abertura dos três terminais onde é mostrado o bloqueio/desbloqueio das tarefas de exibição da solução.
- **Gerar Dados de Otimização:** chamado pela thread de geração de dados gerando os dados de otimização da solução em um período de 1000ms.
- **Gerar Dados de Processo:** chamado pela thread de geração de dados gerando os dados de processo da solução em um período de 1000ms.
- **Gerar Alarmes:** chamado pela thread de geração de dados gerando os alarmes da solução em um período de 1000ms.

Ademais, também foram criadas oito threads, cada uma com uma função específica para o funcionamento da solução como mostrado abaixo:

- **Tarefa de captura de dados do teclado:** responsável por ler os comandos digitados pelo operador e sinalizar ações específicas em outras threads: bloqueio, desbloqueio e encerramento.
- **Tarefa de geração de dados:** gera todos os dados e alarmes do sistema com o auxílio das funções de geração já especificadas acima.

- **Tarefa de retirada de dados de otimização:** retira, da lista circular, as mensagens de dados do sistema de otimização depositando os dados no terminal principal.
- **Tarefa de retirada de dados de processo:** retira, da lista circular, as mensagens do sistema SCADA depositando os dados no terminal principal.
- **Tarefa de retirada de alarmes:** retira, da lista circular, as mensagens do sistema SCADA depositando os dados no terminal principal.
- **Tarefa de exibição de dados de otimização:** apresenta uma mensagem informando o bloqueio/desbloqueio da thread.
- **Tarefa de exibição de dados do processo:** apresenta uma mensagem informando o bloqueio/desbloqueio da thread.
- **Tarefa de exibição de alarmes:** apresenta uma mensagem informando o bloqueio/desbloqueio da thread.

A aplicação foi então desenvolvida no Visual Studio Community utilizando a linguagem de programação C++. Consistindo em quatro projetos: `TrabalhoPratico` (onde está localizada a thread principal e a instanciação das threads secundárias), `ExibicaoAlarmes`, `ExibicaoDadosProcesso` e `ExibicaoDadosOtimizacao` que diz respeito às threads de exibição de alarmes, exibição de dados do processo e a exibição dos dados de otimização respectivamente. A seguir, temos uma especificação mais detalhada a respeito das threads utilizadas.

2.2. Tarefa de captura de dados do teclado:

Após a criação dos objetos, dos processos de exibição e das threads secundárias a thread principal chama o esse processo que entra em loop e espera que o operador digite uma tecla. Cada um das entradas válidas sinaliza um evento com reset automático, exceto pela tecla 'Esc' que dispara um evento de reset manual, por meio da função SetEvent(). Na tabela abaixo é possível verificar todas as entradas e suas respectivas funções:

c	Notifica à tarefa de comunicação de dados que esta deve bloquear-se ou retomar sua execução normal, dependendo de seu estado anterior, ou seja, este caractere funcionará como um sinalizador *on-off*. Esta notificação deve ocorrer por meio do respectivo objeto "evento" de sincronização.
o	Idem, com relação à tarefa de retirada de dados de otimização.
p	Idem, com relação à tarefa de retirada de dados de processo.
a	Idem, com relação à tarefa de retirada de alarmes.

t	Idem, com relação à tarefa de exibição de dados de otimização.
r	Idem, com relação à tarefa de exibição de dados de processo.
l	Idem, com relação à tarefa de exibição de alarmes.
z	Notifica a tarefa de exibição de alarmes que esta deve limpar sua janela de console.
ESC	Notifica todas as tarefas do sistema que estas devem encerrar suas execuções, bem como o encerramento da própria tarefa de leitura do teclado. Esta notificação deve ocorrer por meio dos respectivos objetos "evento" de sincronização.

2.3. Tarefa de geração de dados:

Thread secundária do tipo produtora responsável por gerar todas as mensagens do sistema enquanto o usuário bloquear a geração pressionando a tecla `C` ou então encerrando todos os processos com a tecla `ESC`. é constituída por um loop `for` que vai de 1 até 1000000, correspondente ao limite imposto pelo campo `NSEQ` existente em todas as mensagens. Dentro desse loop a thread chama então as funções geradoras passando como parâmetro a o valor atual da variável local do loop.

2.4. Tarefa de retirada de dados de otimização:

Thread secundária do tipo consumidora responsável por retirar as mensagens de dados de otimização produzidas pela tarefa de geração de dados e mostrar no terminal principal. A thread começa sua execução em um loop `while` enquanto a tecla `ESC` não é pressionada. Em seguida, caso a tecla C seja pressionada, a thread é então bloqueada e desbloqueada caso a tecla seja pressionada novamente.

Em seguida, caso a thread esteja desbloqueada, é verificado se o espaço para acesso da lista circular encontra-se ocupado, caso não esteja é feita uma verificação do buffer de escrita/leitura da lista circular, caso ele esteja disponível é feita estão a verificação do tipo de mensagem, caso ela seja de fato uma mensagem do tipo de dados de otimização ela é então mostrada no terminal principal, atualizando a posição ocupada da lista e fechando os objetos de sincronização e recomeçando o loop `while` do início.

2.5. Tarefa de retirada de dados de processo:

Funciona exatamente da mesma forma que a thread de retirada de dados de otimização, com a diferença de que são analisados se os dados lidos são dados de processo.

2.6. Tarefa de retirada de alarmes:

Funciona exatamente da mesma forma que a thread de retirada de dados de otimização e retirada de dados de processo, com a diferença de que são analisados se os dados lidos são alarmes.

2.7. Tarefa de exibição de dados de otimização:

A thread de exibição dos dados de otimização nesta etapa do trabalho apenas mostra a mensagem do estado de bloqueio/desbloqueio da thread no terminal B. Para isso a thread executa um loop `while` enquanto a tecla `ESC` não é pressionada, além de utilizar um mutex que escreve no terminal quando a thread é bloqueada ou desbloqueada pela tecla `T`, fechando em seguida todos os handles abertos.

2.8. Tarefa de exibição de dados do processo:

Funciona da mesma maneira que a thread de exibição dos dados de otimização, mostrando a mensagem do estado de bloqueio/desbloqueio da thread no terminal A. Para isso a thread executa um loop `while` enquanto a tecla `ESC` não é pressionada, além de utilizar um mutex que escreve no terminal quando a thread é bloqueada ou desbloqueada pela tecla `R`, fechando em seguida todos os handles abertos.

2.9. Tarefa de exibição de alarmes:

Funciona da mesma maneira que a thread de exibição dos dados de otimização e da exibição dos dados do processo, mostrando a mensagem do estado de bloqueio/desbloqueio da thread no terminal C. Para isso a thread executa um loop `while` enquanto a tecla `ESC` não é pressionada, além de utilizar um mutex que escreve no terminal quando a thread é bloqueada ou desbloqueada pela tecla `L`, fechando em seguida todos os handles abertos.

3. Resultados:

O Terminal Principal, Terminal A, Terminal B e Terminal C podem ser vistos em execução nas imagens anexadas neste documento. O Terminal Principal exibe os dados de otimização, dados de processo e os alarmes do sistema. O Terminal A exibe o estado de bloqueio/desbloqueio do processo de exibição dos dados de processo, o Terminal B exibe o estado de bloqueio/desbloqueio do processo de exibição dos dados de otimização e o Terminal C exibe o estado de bloqueio/desbloqueio do processo de exibição de alarmes.

4. Conclusão:

Todas as especificações do projeto foram atendidas e estão descritas ao longo do trabalho. Algumas melhorias poderiam ser implementadas, como o melhor controle da cor do texto exibido no terminal principal, que em alguns momentos permanece com a cor anterior

em sua maioria no momento em que são mostrados os logs informando qual a tecla foi pressionada pelo usuário. Além disso, o bloqueio e desbloqueio da retirada dos dados (otimização, processo e alarmes) não foi implementado da melhor maneira, falhando em alguns momentos. Esses erros, contudo, serão analisados criteriosamente e espera-se corrigi-los na parte 2 do trabalho.

5. Anexos: