UNIVERSIDADE FEDERAL DE MINAS GERAIS

AUTOMAÇÃO EM TEMPO REAL

Trabalho Prático | Etapa 2

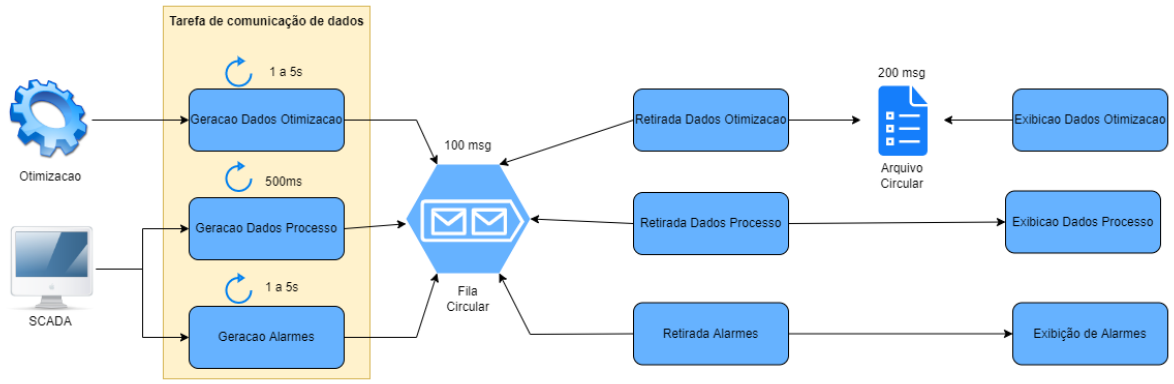
Marcone Márcio da Silva Faria

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 segunda parte do trabalho com a arquitetura multithread da aplicação previamente estruturada na etapa 1, foram criadas três threads secundárias para geração dos dados (uma para cada um dos tipos: otimização, processo e alarmes), a temporização para a geração das mensagens levando em consideração cada um dos tipos, a criação do arquivo circular em disco inserindo e retirando dados do sistema de otimização, a implementação dos mecanismos de IPC entre as tarefas de retirada de mensagens e as respectivas tarefas de exibição de dados de processo e alarme.

Por fim, foram feitas correções gerais no funcionamento do sistema em relação à primeira versão, além de melhorias gerais relacionadas à arquitetura e execução dos processos.

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. Essa função também foi usada para criar o arquivo de mensagens de otimização.
- **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.
- **Escrever Dados Arquivo:** chamado pela thread de retirada de dados de otimização, essa função é utilizada para escrever no arquivo circular. Ao atingir a posição 200 do arquivo a posição 1 é sobrescrita com o dado mais recente. O arquivo é bloqueado usando os recursos LockFile e UnlockFile, assim não há colisões nas escritas e leituras.

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

- **Tarefa 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 de Otimização:** geram os dados de otimização do sistema de acordo com uma temporização randômica de 1 a 5 segundos.
- **Tarefa de Geração de Dados de Processo:** gera todos os dados de processo do sistema com uma periodicidade padrão de 500ms.
- **Tarefa de Geração de Alarmes:** geram os alarmes do sistema de acordo com uma temporização randômica de 1 a 5 segundos.

- **Tarefa de retirada de Dados de Otimização:** retira, da lista circular, as mensagens de dados do sistema de otimização depositando os dados no arquivo circular.
- **Tarefa de retirada de Dados de Processo:** retira, da lista circular, as mensagens do sistema SCADA depositando no mailslot para enviar pro terminal de exibição.
- **Tarefa de retirada de Alarmes:** retira, da lista circular, as mensagens do sistema SCADA depositando no mailslot para enviar pro terminal de exibição.
- **Tarefa de Exibição de Dados de Otimização:** lê do arquivo circular a mensagem e imprime no terminal o formato:

  `NSEQ:###### SP (TEMP):######C SP (PRE):######psi VOL:#####m3 HH:MM:SS`

  Quando a leitura do arquivo chega na linha 200 ela recomeça da linha 1. Esse mecanismo é controlado por meio de um operador de módulo 200. Caso seja bloqueado apresenta a mensagem de bloqueio/desbloqueio em vermelho e verde respectivamente.
- **Tarefa de exibição de Dados do Processo:** recebe, da thread cliente contida no mailslot `hMailslotClienteProcesso`, as mensagens do processo contidas na lista circular e imprime no terminal no formato:

  `HH:MM:SS NSEQ:###### PR (T):######psi TEMP:######C PR (G): ######psi NIVEL:######cm`
- **Tarefa de exibição de Alarmes:** recebe, da thread cliente contida no mailslot `hMailslotClienteAlarme` as mensagens do processo contidas na lista circular e imprime no terminal no formato:

  `HH:MM:SS NSEQ:###### TEXTOTEXTOTEXTOTEXTOTEXTOTEXTOTEXTO PRI: ###`

  Sendo o campo de texto um dos dez tipos de alarmes existentes no sistema relacionado com cada um dos IDs de alarme disponíveis.

A aplicação foi então desenvolvida no Visual Studio Community utilizando a linguagem de programação C++. Uma única solução nomeada `TrabalhoPratico` foi criada e nela existem 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 de otimização:

Thread secundária do tipo produtora responsável por gerar as mensagens de otimização 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 na mensagem. Dentro desse loop a mensagem é produzida seguindo o formato padrão:

`NSEQ|TIPO|SP_PRESS|SP_TEMP|VOL|Time Stamp`

em seguida, essa mensagem é armazenada na lista circular caso a memória não esteja cheia.

2.4. Tarefa de geração de dados de processo:

Funciona exatamente da mesma forma que a thread de geração de dados de otimização, com a diferença de que são produzidas as mensagens do tipo processo no formato:

`NSEQ|TIPO|PRESSAO_T|TEMP|PRESSAO_G|NIVEL|Time Stamp`

2.5. Tarefa de geração de alarmes:

Funciona exatamente da mesma forma que a thread de geração de dados de otimização, com a diferença de que são produzidas as mensagens do tipo alarme:

`NSEQ|TIPO|ID|PRIORIDADE|Time Stamp`

2.6. 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 A. A thread começa sua execução em um loop `while` enquanto a tecla `ESC` não é pressionada. Em seguida, caso a tecla O 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 escrita no arquivo em disco utilizando a função de escrever dados no arquivo sendo em seguida liberado os objetos de sincronização utilizados.

2.7. Tarefa de retirada de dados de processo:

Thread secundária do tipo consumidora responsável por retirar as mensagens de dados de processo produzidas pela tarefa de geração de dados e mostrar no terminal B. A thread começa sua execução criando o mailslot do tipo cliente realizando-se uma verificação de sucesso ou erro dessa operação em seguida a thread executa um loop `while` enquanto a tecla `ESC` não é pressionada. Em seguida, caso a tecla P 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 processo ela é então escrita no mailslot do tipo cliente sendo em seguida liberado os objetos de sincronização utilizados.

2.8. Tarefa de retirada de alarmes:

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

2.9. Tarefa de exibição de dados de otimização:

O processo de exibição dos dados de otimização mostra a mensagem do estado de bloqueio/desbloqueio da thread no terminal B, além de ler as mensagens dos dados de otimização do arquivo em disco. Para isso a thread executa um loop `while` verificando o bloqueio e desbloqueio da mesma pela tecla T. Caso esteja desbloqueada, chama a funcao LerDadosArquivo que faz a leitura do arquivo circular e imprime na tela os dados. Para nao haver conflito com a escrita no arquivo, são usadas as funções LockFile e UnlockFile.

2.10. Tarefa de exibição de dados do processo:

O processo de exibição dos dados de processo mostra a mensagem do estado de bloqueio/desbloqueio da thread no terminal A, além de ler as mensagens dos dados de processo a partir dos dados enviados pela thread cliente do mailslot de dados de processo. Para isso a thread executa um loop `while` enquanto a tecla `ESC` não é pressionada, recebendo os dados do mailslot a partir da função `LerMailslot` armazenando a mensagem em um buffer sendo posteriormente impressa no terminal seguindo o formato especificado anteriormente.

2.11. Tarefa de exibição de alarmes:

Funciona da mesma maneira que a thread de exibição dos dados do processo, mostrando a mensagem do estado de bloqueio/desbloqueio da thread no terminal C além de mostrar as mensagens do tipo alarme.

3. Resultados:

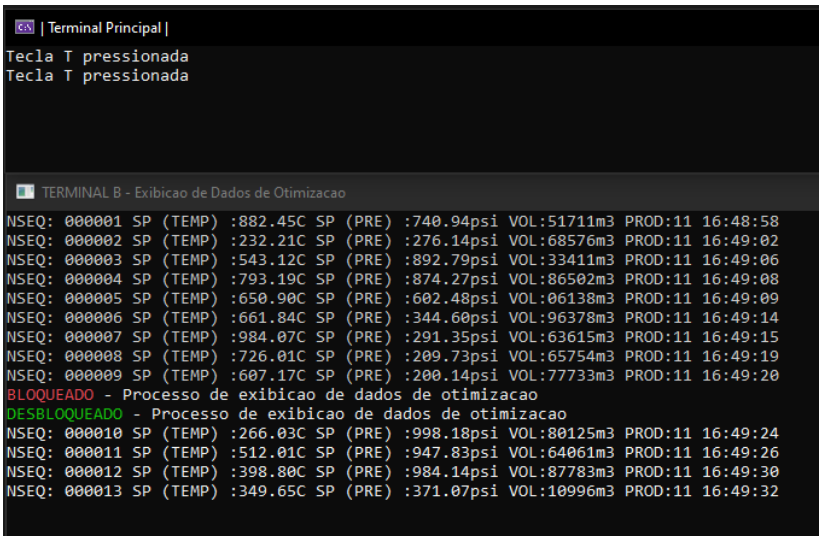
O Terminal Principal, Terminal A, Terminal B e Terminal C podem ser vistos em execução nas imagens anexadas neste documento. O Terminal Principal exibe o estado de inicialização e finalização da aplicação, também mostrando as mudanças de estados das tarefas e avisos de memória cheia tanto para a lista circular quanto para o arquivo em disco. O Terminal A exibe o estado de bloqueio/desbloqueio do processo de exibição dos dados de processo além das mensagens do tipo processo lidos diretamente da lista circular, o Terminal B exibe o estado de bloqueio/desbloqueio do processo de exibição dos dados de otimização além das mensagens lidas diretamente do arquivo em disco e o Terminal C exibe o estado de bloqueio/desbloqueio do processo de exibição de alarmes e as mensagens do tipo alarme lidos diretamente da lista circular.

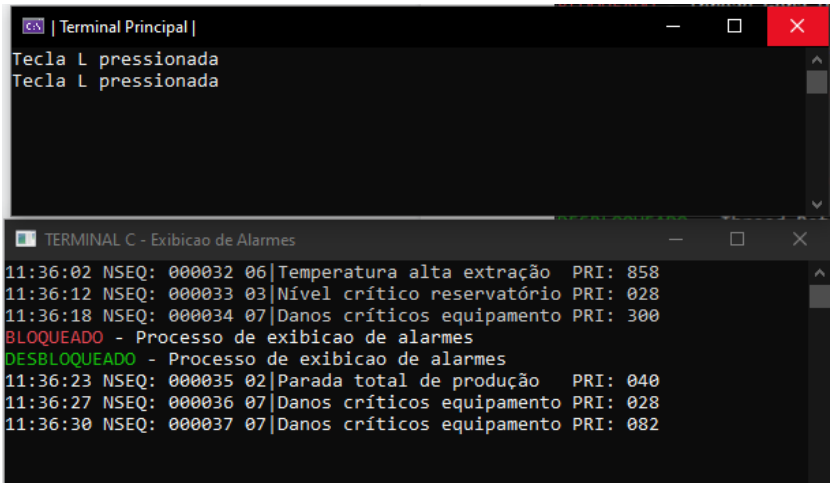
4. Conclusão:

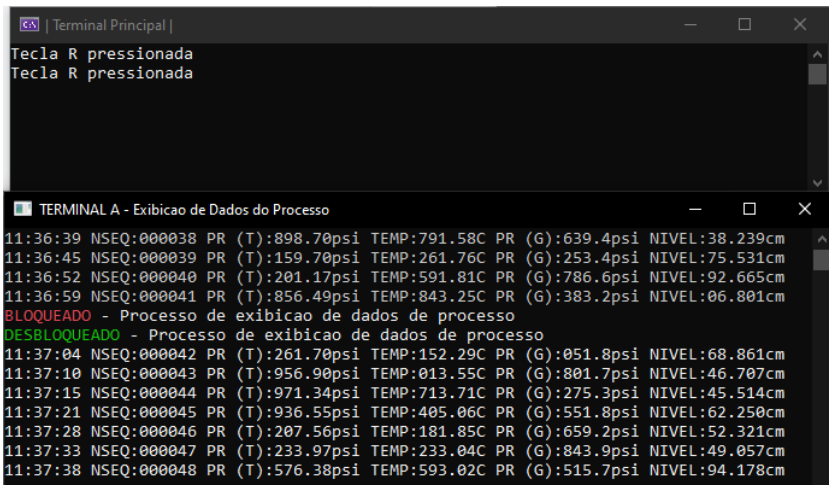
Todas as especificações do projeto foram atendidas e estão descritas ao longo do trabalho: bloqueios, exibições de mensagens, memórias, etc. Algumas melhorias foram implementadas, como o melhor controle da cor do texto exibido no terminal principal, mas o bloqueio e desbloqueio da retirada dos dados (otimização, processo e alarmes) ainda não funciona perfeitamente, apesar da implementação parecer correta.

5. Anexos:

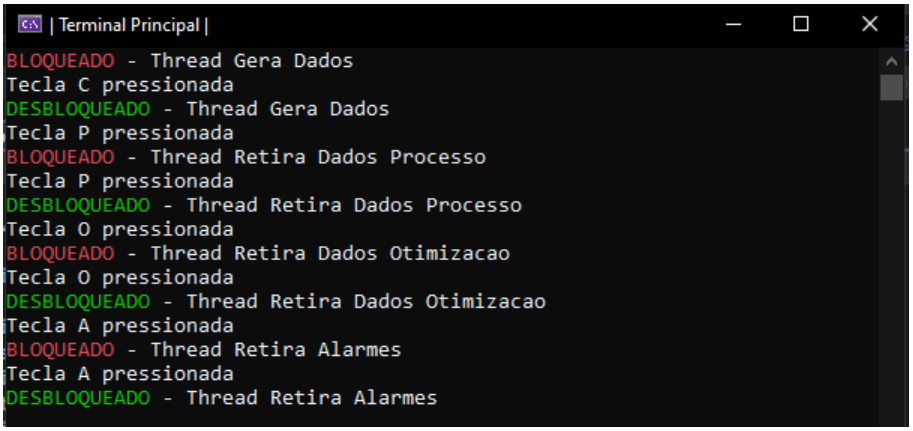
Bloqueio e Desbloqueio dos processos de exibição:



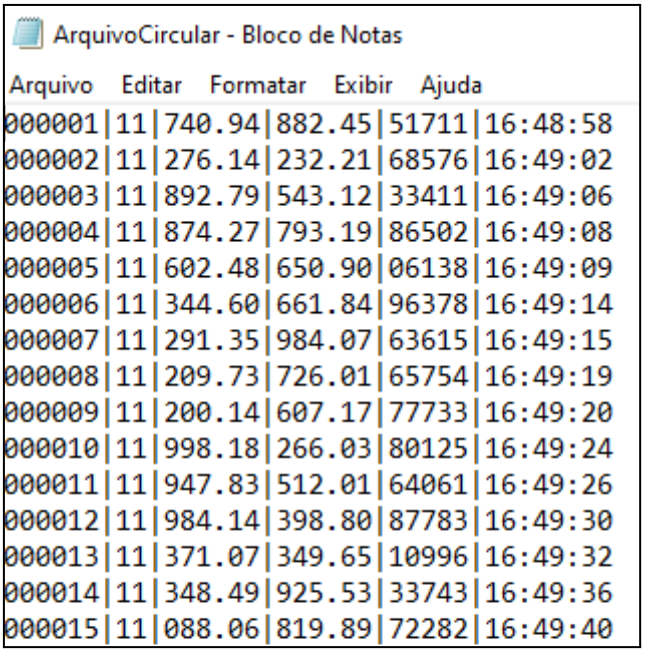




Bloqueio e desbloqueio dos processos de geração de dados e retiradas



Arquivo circular exemplo:



Arquitetura da solução no visual studio: