UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
Departamento de Engenharia Eletrônica
DISCIPLINA: Automação em Tempo Real (ELT127)
PROFESSOR: Luiz Themystokliz S. Mendes

TRABALHO FINAL – 2022/1 – VALOR: 25 PONTOS

CONTEXTO INDUSTRIAL: INDÚSTRIA DE EXTRAÇÃO DE PETRÓLEO

A extração de petróleo de jazidas petrolíferas apresenta muitos desafios tecnológicos, devido aos vários fatores que afetam a taxa de produção tais como a geometria da jazida (em especial sua espessura e continuidade de reservatório), acumulação de hidrocarbonetos, pressão e profundidade da jazida, tipo de rocha e sua permeabilidade, saturação e propriedades do fluido petrolífero, número de poços e sua localização, etc.

Dependendo da vida útil de uma jazida petrolífera, a extração de petróleo pode ocorrer em três diferentes fases: primária, secundária e terciária [1]. Na fase primária, cerca de 5% a 20% do petróleo é recuperado com base na própria energia da jazida, ou seja, a pressão interna da jazida é suficiente para provocar espontaneamente a expulsão do petróleo. Com o passar do tempo, porém, esta energia esgota-se e então se passa à fase secundária, onde técnicas adicionais designadas como IOR (*Improved Oil Recovery*) entram em ação para aumentar a taxa de extração da jazida. Exemplos destas técnicas são a injeção de água ou gás natural na jazida, para aumentar sua pressão, e a elevação artificial de fluidos por meio de bombeamento ou injeção de gás. As técnicas de IOR respondem por 20% a 40% do petróleo extraído. Finalmente, quando o esgotamento da jazida torna o IOR ineficiente, passa-se à fase terciária onde tecnologias suplementares (referidas coletivamente como EOR, ou *Extended Oil Recovery*) são aplicadas para recuperar o petróleo residual. Exemplos destas tecnologias terciárias são processos de inundação química, processos térmicos, injeção de gás e processos microbiais.

No Brasil, a técnica de fase secundária mais empregada na extração de petróleo é o chamado *gas lift* [2], em que um gás a alta pressão é injetado continuamente ao longo do tubo de extração, diminuindo a densidade da coluna de fluido e, assim, reduzindo a pressão necessária para seu escoamento. A figura 1 apresenta um esquema desta técnica.

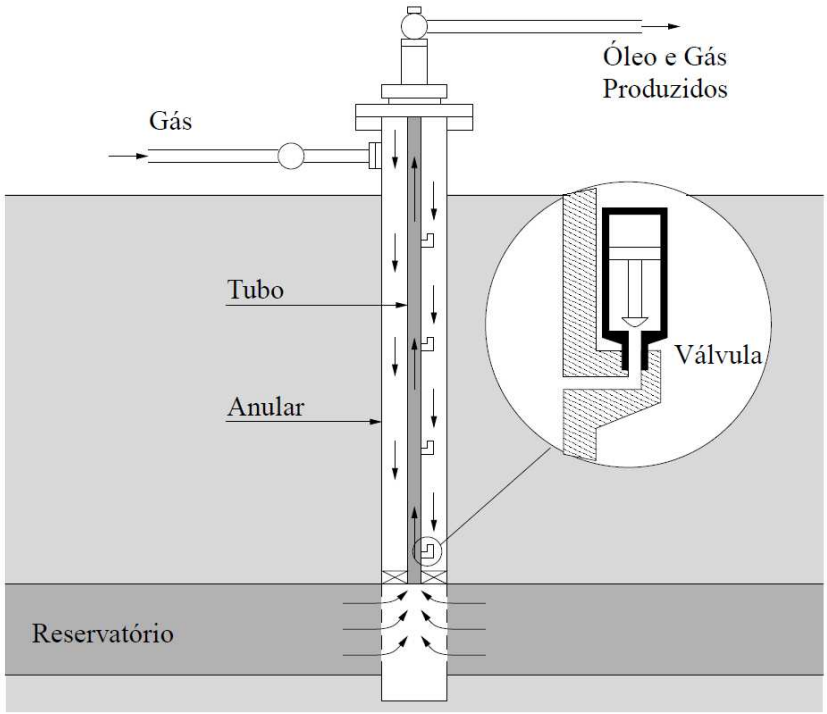


Figura 1: Extração de petróleo através de *gas lift*. Fonte: [2]

A eficiência da técnica de *gas lift* depende de diversos fatores. Em primeiro lugar, uma pressão de gás excessivamente alta pode diminuir o desempenho da extração de petróleo. Além disto, o suprimento de gás em uma plataforma petrolífera é limitado, sendo reposto somente em base periódica, e, assim, deve ser utilizado de forma eficiente para evitar a paralisação da produção caso seja totalmente consumido de forma prematura. Instalações de extração de petróleo em geral possuem sistemas computacionais de otimização para permitir o máximo desempenho da produção sob a técnica de *gas lift*. A título de exemplo, consulte www.intechww.com/wp-content/uploads/Gas-Lift-Optimization.pdf.

DESCRIÇÃO DO TRABALHO

Uma empresa petrolífera acaba de instalar um sistema de otimização de *gas lift* em uma de suas plataformas *offshore*. Este sistema adquire, do Controlador Lógico Programável (CLP) que controla as válvulas de *gas lift*, valores das variáveis de processo relevantes como pressão e temperatura no tubo de extração, nível e pressão do reservatório de gás, etc., para, em resultado, determinar o volume e pressão do gás a ser injetado no tubo de extração.

A empresa deseja integrar as informações deste sistema de otimização com aquelas já existentes no sistema SCADA (*Supervisory Control and Data Acquisition*) da plataforma petrolífera, direcionando todas estas para três painéis de projeção de imagens presentes na sala de controle da plataforma, os quais apresentam respectivamente dados do sistema de otimização, dados do processo de extração do óleo cru e mensagens de alarmes industriais. Para tal, você foi contratado para desenvolver uma aplicação de software *multithread* responsável pela leitura de dados do sistema de otimização e do sistema SCADA, e distribuí-los adequadamente para os três painéis de projeção. Os dados do sistema de otimização chegam em intervalos variáveis de 1 a 5 segundos, ao passo que os dados do sistema SCADA devem ser lidos periodicamente do SDCD a cada 500 ms. A aplicação a ser desenvolvida deverá ser composta pelas seguintes tarefas (onde o termo "tarefa" pode designar tanto processos quanto *threads*):

1. **Tarefa de comunicação de dados.** Executa a leitura de dados do sistema de otimização e do sistema SCADA, depositando-os em uma lista circular em memória.
2. **Tarefa de retirada de dados de otimização.** Esta tarefa retira, da lista circular em memória, as mensagens de dados do sistema de otimização e as deposita em um arquivo circular em disco com capacidade para 200 mensagens.
3. **Tarefa de retirada de dados de processo.** Esta tarefa retira, da lista circular em memória, as mensagens do sistema SCADA correspondentes a dados de processo e as repassa para a tarefa de exibição de dados de processo (vide abaixo).
4. **Tarefa de retirada de alarmes.** Esta tarefa retira, da lista circular em memória, as mensagens do sistema SCADA correspondentes a alarmes e as repassa para a tarefa de exibição de alarmes (vide abaixo).
5. **Tarefa de exibição de dados de otimização.** Esta tarefa retira mensagens do arquivo circular em disco e as exibe no respectivo painel de projeção de vídeo, na sala de controle.
6. **Tarefa de exibição de dados de processo.** Esta tarefa recebe mensagens de dados de processo e as exibe no respectivo painel de projeção de vídeo, na sala de controle.
7. **Tarefa de exibição de alarmes.** Esta tarefa recebe mensagens de alarmes do sistema e as exibe no respectivo painel de projeção de vídeo, na sala de controle.
8. **Tarefa de leitura do teclado**. Esta tarefa dá tratamento aos comandos digitados pelo usuário.

Tarefa de comunicação de dados. Esta tarefa simula a leitura de dados provenientes do sistema de otimização e do sistema SCADA, através da geração de mensagens correspondentes. As mensagens do sistema de otimização devem ser geradas com periodicidade aleatória entre 1 e 5 segundos. Já as mensagens do sistema SCADA podem ser de dois tipos: mensagens de dados de processo, a serem geradas com periodicidade fixa de 500ms, e mensagens de alarmes, a serem geradas com periodicidade aleatória entre 1 e 5 segundos. Todas as mensagens devem ser depositadas em uma lista circular em memória RAM, com capacidade para 100 mensagens. Caso a lista circular esteja cheia no momento de depósito de alguma mensagem, esta tarefa deve bloquear-se até que haja alguma posição livre na mesma, alertando este fato na console principal (a console associada à tarefa de leitura do teclado; vide a seguir) por meio de texto apropriado. As temporizações necessárias a esta tarefa podem ser implementadas por qualquer método visto no curso, **exceto** a função *Sleep()*. (**Nota:** como descrito adiante, a função *Sleep()* deve ser empregada apenas na Etapa 1 do trabalho).

Cada mensagem de dados do sistema de otimização deve corresponder a uma cadeia de caracteres com o tamanho fixo de 38 caracteres, no formato a seguir, onde os campos individuais são separados pelo delimitador "|" (barra horizontal):

NNNNNN	NN	NNNN.N	NNNN.N	NNNNN	HH:MM:SS
NSEQ	*TIPO*	*SP_PRESS*	*SP_TEMP*	*VOL*	*INICIO*

Campo	Tamanho	Tipo	Descrição
NSEQ	6	Inteiro	Número sequencial da mensagem [1...999999]
TIPO	2	Inteiro	Sempre "11"
SP_PRESS	6	Real	*Set Point* da pressão de injeção de gás (psi)
SP_TEMP	6	Real	*Set Point* da temperatura do gás injetado (C)
VOL	5	Inteiro	Volume total de gás a ser injetado (m^3)
Time Stamp	8	Tempo	Hora, minuto e segundo

Exemplo: `"008755|11|0343.5|0330.0|12500|14:45:00"`

As mensagens do SCADA referentes a dados de processo devem corresponder a uma cadeia de caracteres com o tamanho fixo de 46 caracteres, no formato a seguir, onde os campos individuais também são separados pelo delimitador "|" (barra horizontal):

NNNNNN	NN	NNNN.N	NNNN.N	NNNN.N	NNNN.N	HH:MM:SS
NSEQ	*TIPO*	*PRESSAO_T*	*TEMP*	*PRESSAO_G*	*NIVEL*	*Time Stamp*

Campo	Tamanho	Tipo	Descrição
NSEQ	6	Inteiro	Número sequencial da mensagem [1...999999]
TIPO	2	Inteiro	Sempre "22"
PRESSAO_T	6	Real	Pressão no tubo de extração (psi)
TEMP	6	Real	Temperatura do tubo de extração (C)
PRESSAO_G	6	Real	Pressão no reservatório de gás de injeção (psi)
NIVEL	6	Real	Nível do reservatório de gás de injeção (cm)
Time Stamp	8	Tempo	Hora, minuto e segundo

Exemplo: `"345666|22|0453.8|0076.3|0034.5|0453.2|12:37:24"`

Por fim, as mensagens de alarmes devem corresponder a uma cadeia de caracteres com o tamanho fixo de 27 caracteres, no formato definido a seguir onde os campos individuais são novamente separados pelo delimitador "|" (barra vertical):

NNNNNN	NN	NNNN	NNN	HH:MM:SS
NSEQ	*TIPO*	*ID*	*PRIORIDADE*	*Time Stamp*

Campo	Tamanho	Tipo	Descrição
NSEQ	6	Inteiro	Número sequencial da mensagem de alarme [1...999999]
TIPO	2	Inteiro	Sempre "55"
ID	4	Inteiro	Identificação do alarme
PRIORIDADE	3	Inteiro	Prioridade do alarme (1..999)
Time Stamp	8	Tempo	Hora, minuto e segundo

Exemplo: `"007459|55|0005|025|22:12:53"`

Na montagem de todas as mensagens, o campo "NSEQ" deverá ser incrementado sequencialmente a cada mensagem gerada (de forma independente para cada tipo de mensagem), voltando ao valor zero após alcançar a contagem máxima, e o campo "*Time Stamp"* deverá ter a hora corrente. Os demais campos deverão ter seus valores gerados aleatoriamente, dentro das faixas indicadas.

A tarefa de comunicação de dados deve sincronizar sua execução com a tarefa de leitura do teclado através de um par de objetos "evento":

- O primeiro evento sinalizará que a tarefa de comunicação de dados deve bloquear-se, nada mais executando até receber outra sinalização deste mesmo evento, quando então retoma sua execução normal;
- O segundo evento indica notificação de término de execução. Esta notificação deve ser atendida mesmo que a tarefa tenha sido bloqueada pelo semáforo de sinalização de bloqueio acima descrito.

Tarefa de retirada de dados de otimização. Esta tarefa deve consumir mensagens de dados do sistema de otimização, presentes na lista circular em memória, depositando-as em um arquivo circular em disco com capacidade para 200 mensagens. Caso o arquivo circular em disco esteja cheio no momento de depósito de alguma mensagem, esta tarefa deve bloquear-se até que haja alguma posição livre no mesmo, alertando este fato na console principal (a console associada à tarefa de leitura do teclado; vide a seguir) por meio de texto apropriado.

Esta tarefa deve sincronizar sua execução com a tarefa de leitura do teclado através de um par de objetos "evento", nas mesmas condições descritas para a tarefa de comunicação de dados.

Tarefa de retirada de dados de processo. Esta tarefa deve consumir mensagens de dados de processo provenientes do SCADA, presentes na lista circular em memória, repassando-as integralmente à tarefa de exibição de dados de processo. A comunicação entre estas tarefas deve ocorrer por meio de *pipes* ou *mailslots*.

Esta tarefa deve sincronizar sua execução com a tarefa de leitura do teclado através de um par de objetos "evento", nas mesmas condições descritas para a tarefa de comunicação de dados.

Tarefa de retirada de alarmes. Esta tarefa deve consumir mensagens de alarmes provenientes do SCADA, presentes na lista circular em memória, repassando-as integralmente à tarefa de exibição de alarmes. A comunicação entre estas tarefas deve ocorrer por meio de *pipes* ou *mailslots*.

Esta tarefa deve sincronizar sua execução com a tarefa de leitura do teclado através de um par de objetos "evento", nas mesmas condições descritas para a tarefa de comunicação de dados.

Tarefa de exibição de dados de otimização. Esta tarefa deve consumir mensagens do arquivo circular em disco e exibi-las diretamente em uma janela de console exclusiva, que simulará o respectivo painel de projeção de vídeo na sala de controle. As mensagens devem ser exibidas no seguinte formato:

```
NSEQ:###### SP (TEMP):######C SP (PRE):######psi VOL:#####m3 PROD:##
```

Neste formato, a notação "######" representa o valor do respectivo item da mensagem de dados de processo.

A tarefa de exibição de dados de processo deve sincronizar sua execução com a tarefa de leitura do teclado através de um par de objetos "evento", nas mesmas condições descritas para a tarefa de comunicação de dados.

Tarefa de exibição de dados de processo. Esta tarefa deve receber mensagens de dados de processo por meio de *pipes* ou *mailslots* e exibi-las diretamente em uma janela de console exclusiva, que simulará o respectivo painel de projeção de vídeo na sala de controle. As mensagens devem ser exibidas no seguinte formato:

```
HH:MM:SS NSEQ:###### PR (T):######psi TEMP:######C PR (G):######psi NIVEL:######cm
```

Como antes, a notação "######" representa o valor do respectivo item da mensagem de dados de processo.

A tarefa de exibição de dados de processo deve sincronizar sua execução com a tarefa de leitura do teclado através de um par de objetos "evento", nas mesmas condições descritas para a tarefa de comunicação de dados.

Tarefa de exibição de alarmes. Esta tarefa deve receber mensagens de alarmes por meio de *pipes* ou *mailslots* e exibi-las diretamente em uma janela de console exclusiva, que simulará o respectivo painel de projeção de vídeo na sala de controle. As mensagens devem ser exibidas no seguinte formato:

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

Como antes, a notação "######" representa o valor do respectivo item da mensagem de dados de processo. O campo "ID" da mensagem de alarme deve ser substituído por uma cadeia fixa de 30 caracteres que descreva o alarme ocorrido. Sua implementação deve prever no mínimo 10 descrições de alarme. Pesquise o processo de extração de petróleo na Internet para obter exemplos típicos de alarmes.

Esta tarefa pode ainda receber uma mensagem específica proveniente da tarefa de leitura do teclado, via *pipes* ou *mailslots*, solicitando a limpeza da janela de sua janela de console. Em adição, deve também sincronizar sua execução com a tarefa de leitura do teclado através de um par de objetos "evento", nas mesmas condições descritas para a tarefa de comunicação de dados

Tarefa de leitura do teclado. Esta tarefa aguarda os seguintes caracteres do teclado, dando aos mesmos o tratamento indicado:

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.

Esta tarefa deverá iniciar sua execução com todos os bloqueios desligados, ou seja, permitindo a plena execução de todas as demais tarefas que compõem a aplicação. Apenas quando o usuário digitar alguma das teclas acima os respectivos bloqueios deverão ser ligados.

ETAPAS DO TRABALHO

O trabalho será executado em duas etapas, sendo sua pontuação condicionada à entrega de ambas as etapas:

- **Entrega da etapa 1 e da etapa 2, nos respectivos prazos indicados: 25 pontos;**
- **Entrega apenas da etapa 2, no respectivo prazo indicado – 12,5 pontos;**
- **Entrega apenas da etapa 1 – não pontuado.**

O propósito desta divisão em etapas é permitir o desenvolvimento mais cedo do trabalho, tal que na etapa 1 sejam utilizados os conceitos vistos na disciplina até o presente momento. A etapa 2 complementa a anterior, adicionando funcionalidades correspondentes aos recursos de temporização e IPC (*Inter-Process Communication*) que ainda serão apresentados no conteúdo da disciplina.

ETAPA 1 – Arquitetura multitarefa/*multithread* e implementação da lista circular em memória

Nesta etapa será elaborada a arquitetura da aplicação, seguida do desenvolvimento e teste de partes de suas funcionalidades:

- Arquitetura multitarefa/*multithread* da aplicação. Em outras palavras, corresponde à definição de como as tarefas descritas serão modeladas em termos de processos e *threads*;
- Disparo da aplicação, com a consequente execução de seus processos e *threads*;
- Implementação do mecanismo de bloqueio/desbloqueio das *threads* e de seus encerramentos, mediante um conjunto de objetos do tipo evento;
- Implementação da lista circular em memória, com o depósito de mensagens na mesma pela tarefa de comunicação de dados, e a retirada das mesmas pelas tarefas de retirada de mensagens, levando em conta aspectos de sincronização eventualmente necessários;
- Temporizações provisórias da tarefa de comunicação de dados, por meio da função *Sleep()* com periodicidade única de 1000 ms.

Leve em conta os seguintes aspectos nesta etapa do trabalho:

1. Apesar de o programa corresponder a mais de um arquivo executável, lembre-se que, do ponto de vista do usuário, seu disparo deve ocorrer a partir de um único arquivo executável. Em outras palavras, você deve definir um processo principal a partir do qual os demais processos serão disparados por meio da função *CreateProcess()* da API Win32.
2. Na API Win32, um processo criado pode ter sua própria janela de console ou compartilhar a janela de console do processo criador. Estude a função *CreateProcess()* para entender como utilizar um ou outro caso.
3. O manuseio de uma lista circular em memória está descrito na seção "O problema dos produtores e consumidores" do livro "Programação Concorrente em Ambiente Windows". Note que, no caso da primeira lista circular em memória, há uma tarefa "produtora" e três "consumidoras" que acessam a lista. Assim, cada uma destas tarefas deverá manter e manipular apontadores individuais para a próxima mensagem a ser depositada ou retirada. Outra opção é implementar esta lista circular como uma lista encadeada (*linked list*).

Para fins de certificação de conclusão desta etapa, a tarefa de retirada de mensagens deverá exibir as mensagens retiradas na console principal, bem como seu estado de bloqueio/desbloqueio. As tarefas de exibição de dados deverão apenas apresentar uma mensagem simples em suas respectivas janelas de console, informando os seus estados de bloqueio/desbloqueio.

O diagrama de relacionamentos na Figura 2 apresenta a estrutura da aplicação correspondente à primeira etapa.

ETAPA 2 – Temporização, IPC e conclusão da aplicação

Nesta etapa a aplicação será finalizada, com o acréscimo das seguintes funcionalidades:

- Temporizações de depósito de mensagens na lista circular em memória pela tarefa de comunicação de dados, nas periodicidades indicadas na descrição da mesma;
- Criação do arquivo circular em disco e inserção/retirada no/do mesmo das mensagens de dados do sistema de otimização;
- Implementação dos mecanismos de IPC (*Inter-Process Communication*) entre as tarefas de retirada de mensagens e as respectivas tarefas de exibição de dados, conforme especificado em suas descrições;
- Testes finais da aplicação como um todo. (**Atenção**: seja cuidadoso e minucioso em seus testes. Procure testar o máximo de situações válidas e inválidas possíveis no funcionamento da aplicação. O grande cientista da computação Niklaus Wirth – criador, entre outras grandes contribuições, da linguagem Pascal – afirmava que testes apenas provam a presença de erros, nunca a ausência destes.)

O diagrama de relacionamentos na Figura 3 apresenta a estrutura da aplicação correspondente à segunda etapa.

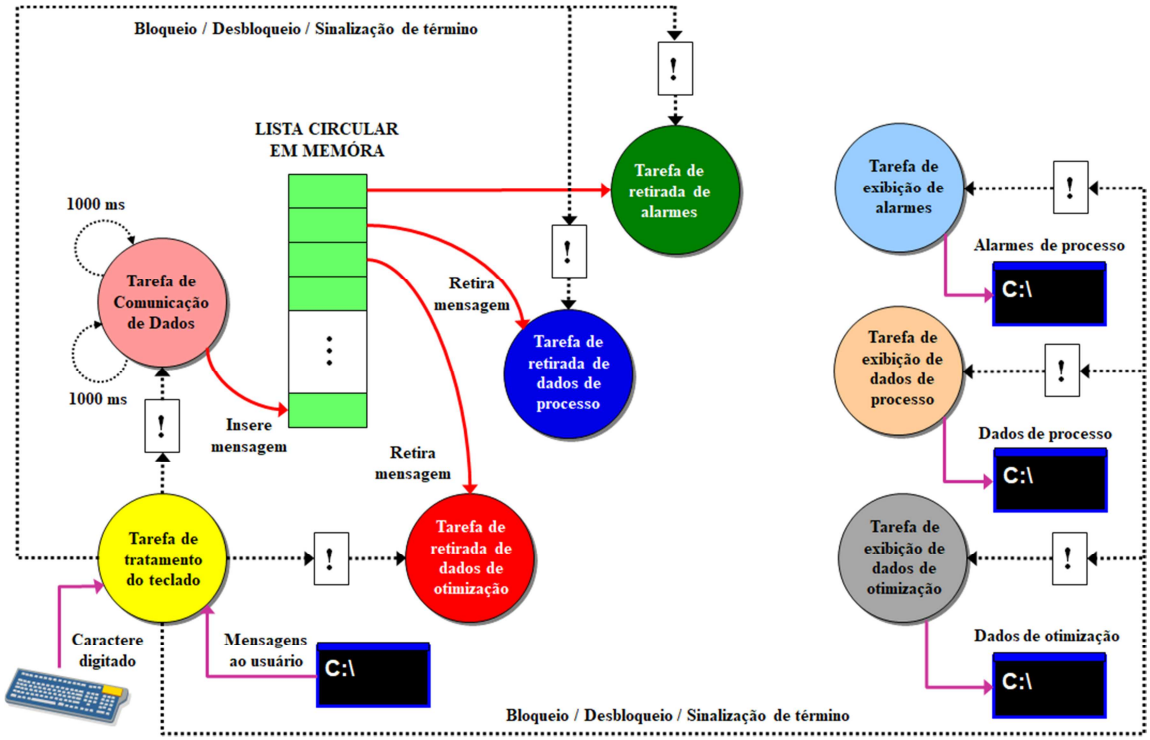


Figura 2: Diagrama de relacionamentos – Etapa 1

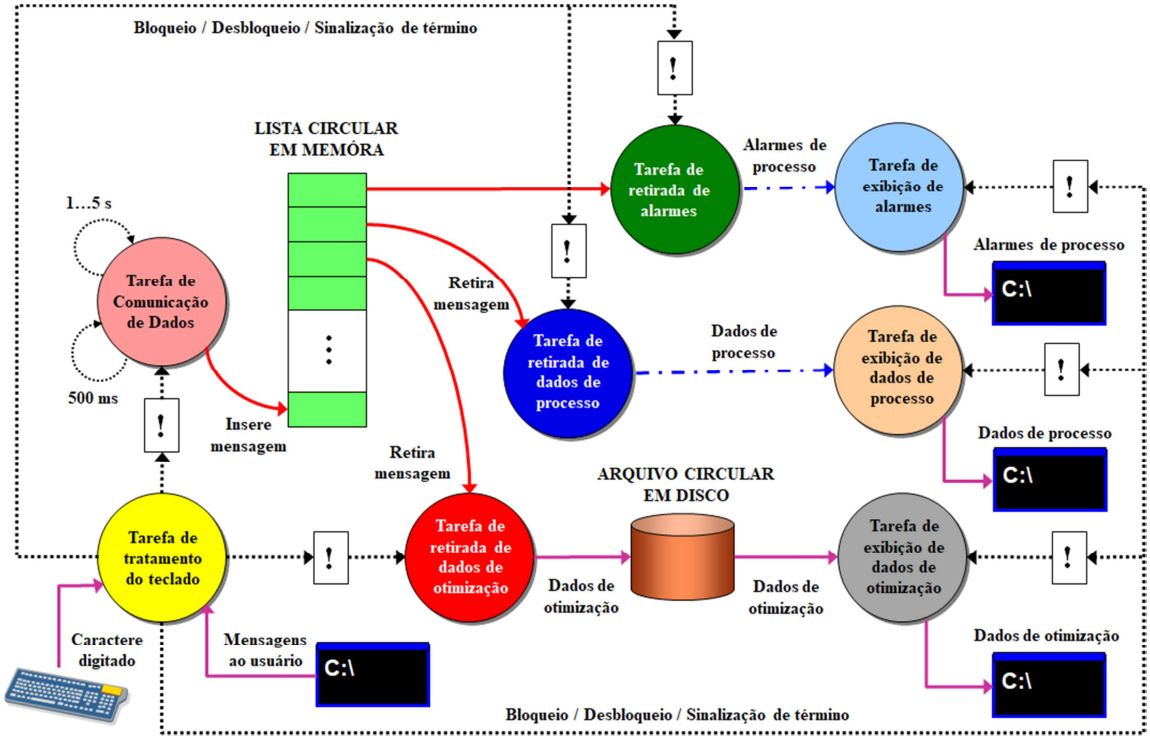
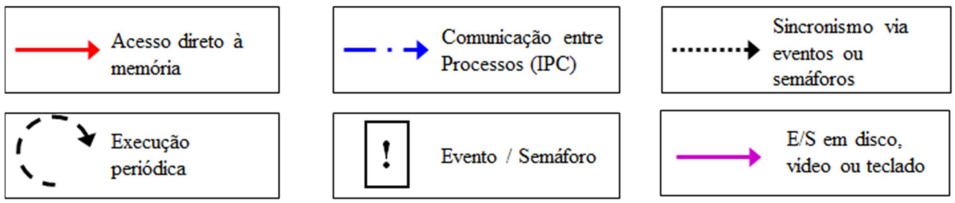


Figura 3: Diagrama de relacionamentos – Etapa 2

Legenda para as figuras 2 e 3:



INSTRUÇÕES

1. O trabalho deve ser feito de forma **individual** ou em grupo de **2 alunos** (dupla). **Se em dupla, os alunos estarão sujeitos à necessidade de comprovação de que dividiram a carga de trabalho de forma equilibrada entre si; caso o professor identifique participação desigual no trabalho, a nota do mesmo será reduzida para ambos os membros da dupla.**
2. Para desenvolver os programas, deverá ser utilizada exclusivamente a ferramenta *Microsoft Visual Studio Community Edition*, que pode ser obtida gratuitamente da Microsoft. Configure esta ferramenta para utilizar a "carga de trabalho" (*workload*) correspondente à linguagem C++.
3. O desenvolvimento da aplicação deve ser feito de modo que, nos programas-fontes, toda referência a objetos externos seja relativa ao diretório corrente (p. ex. `"..\..\teste.dat"`) ao invés de absoluta (p.ex. `"C:\programas\dados\teste.dat"`), de modo que o respectivo projeto possa ser depositado, compilado e testado pelo professor em qualquer diretório de sua escolha.
4. O trabalho, em ambas as etapas, deve ser submetido exclusivamente via *Moodle*, através dos seguintes arquivos:
   - Arquivo ZIP ou RAR contendo a pasta (diretório) que corresponde à respectiva "solução" do *Visual Studio* com todos os arquivos gerados por este, de forma que o professor possa examinar os programas-fonte, compilá-los e testar seu funcionamento em seu próprio computador. O tamanho máximo de arquivo passível de ser submetido segue as limitações do *Moodle* (20 Mb). *DICA: Você pode remover os arquivos de extensão .exe, .ipch e .db presentes em subdiretórios de sua solução, para diminuir o tamanho do arquivo ZIP ou RAR. Os mesmos serão automaticamente recriados pelo Visual Studio quando o projeto for recompilado.*
   - Documento em formato PDF, com a indicação precisa do nome e sobrenome do(s) aluno(s) autor(es) do trabalho, que descreva a aplicação desenvolvida, detalhando as *threads*/processos utilizados, seus papéis e relacionamentos, objetos de sincronização e/ou técnicas utilizadas, resultados de teste, etc., bem como quaisquer outros detalhes que auxiliem o professor no entendimento da aplicação gerada ou que esclareçam aspectos considerados relevantes pelos desenvolvedores. ATENÇÃO: Este item corresponde a 20% da pontuação do trabalho.
4. **Verifique a consistência do arquivo ZIP (RAR) antes de submetê-lo.** Muitas avaliações são prejudicadas porque o arquivo ZIP (RAR) gerado encontra-se inconsistente, p. ex. por não conter todos os arquivos necessários à reprodução do projeto original. Teste seu arquivo ZIP (RAR) descompactando-o em outro computador e reproduzindo o projeto a partir do mesmo. Se o arquivo ZIP (RAR) enviado estiver inconsistente ou incompleto, o mesmo será desconsiderado e o trabalho será considerado como **não-entregue**. O professor não enviará, aos alunos, avisos de problemas com os arquivos enviados.
5. Datas-limite de entrega:

   Etapa 1 - **23h59min** do dia **18/06/2022 (sábado)**

   Etapa 2 - **23h59min** do dia **09/07/2022 (sábado)**

   Não serão permitidos atrasos com relação à etapa 1. Quanto à etapa 2, trabalhos entregues até **16/07/2022** terão sua pontuação reduzida a 50%. Não serão aceitos trabalhos entregues após 16/07/2022.

   Atenção: submeta o trabalho exclusivamente via *Moodle*. Não serão aceitos trabalhos enviados por email, ou depositados em *sites* de compartilhamento de arquivos como *Dropbox*, *Google Drive*, etc., ou ainda postados diretamente via plataforma *Microsoft Teams*.
6. O professor não dará suporte à utilização do ambiente *Visual Studio Community Edition*. Este suporte deverá ser obtido pelo aluno por seus próprios meios. A *web* possui farto material de apoio a esta ferramenta e às linguagens C/C++. Estará, contudo, à disposição dos alunos para solucionar dúvidas sobre o enunciado do TP e sobre a matéria da disciplina em geral.

DICAS DE DESENVOLVIMENTO

1. **Não deixe a elaboração do trabalho para a última hora**. Desenvolvimento de software é uma arte complexa, na qual pequenos erros de programação podem nos custar horas ou mesmo dias de trabalho para identificá-los e corrigi-los.
2. Planeje cuidadosamente quais os processos e respectivas *thread*s que representarão as tarefas descritas anteriormente. Um bom planejamento é fundamental para o sucesso da aplicação.
3. A descrição do trabalho contém propositalmente lacunas de detalhamento com o objetivo de estimular os alunos a pesquisarem e tomarem decisões de projeto, a fim de exercitar a capacidade de agirem como projetistas de sistemas. Desta forma, exceto onde expressamente indicado no presente enunciado, há plena liberdade de escolha de recursos de sincronização, temporização, IPC, etc.

4. Praticamente todas as funções a serem executadas pelas tarefas desta aplicação já estão implementadas, em maior ou menor grau, ao longo dos diversos programas de exemplos examinados em classe, apresentados nos capítulos 2 a 6 do livro "Programação Concorrente em Ambiente Windows". Estude cuidadosamente as funções a serem executadas pelas tarefas, identifique os correspondentes programas de exemplo deste livro, e use-os como base para o desenvolvimento das tarefas.
5. No *Visual Studio*, ao invés de criar diferentes soluções (*solutions*) para cada processo executável de sua aplicação, é muito mais prático e conveniente criar uma única solução e, dentro dela, criar projetos separados para cada processo executável. Isto lhe permitirá compilar todos os projetos de uma única vez, bem como editar cada um deles em uma única instância do *Visual Studio*.
6. Seja original em sua documentação. Figuras e textos copiados deste enunciado só mostram descaso com o trabalho e prejudicarão sua pontuação.

BÔNUS ADICIONAL (6 pontos)

As seguintes características, se presentes no trabalho, e a critério exclusivo do professor, podem valer um bônus adicional de até seis (6) pontos:

- Utilização da biblioteca *Pthreads-Win32* para fins de criação de *threads* e sincronização via *mutexes* ou semáforos. Neste caso, baixe a última versão disponível desta biblioteca (2.9.1), desempacote-a sob `C:\Arquivos de Programas\pthreads-w32-2-9-1-release` em seu computador e referencie-a com este caminho no *Visual Studio Community*, para que haja coincidência com o computador do professor. **IMPORTANTE:** Se você se esquecer deste passo, o programa não compilará no computador do professor e, assim, não poderá ser testado, com consequente penalização na avaliação!
- Melhoramentos adicionais, desde que claramente documentados e considerados relevantes pelo professor;
- Grau de detalhamento e clareza da documentação do trabalho;
- Boa estruturação, organização, documentação e legibilidade dos programas-fontes.

QUADRO DE PONTUAÇÃO DO TRABALHO

A tabela seguinte apresenta os critérios de avaliação a serem considerados no trabalho.

Item	Avaliação
Criação de processos e *threads*	15%
Sincronismo entre *threads* e IPC	15%
Funcionamento da aplicação	25%
Atendimento aos requisitos do enunciado	25%
Documentação	20%

Observe, contudo, que estes critérios de avaliação estão entrelaçados entre si, de forma que, dependendo das circunstâncias, a perda de pontos em um item pode acarretar a perda automática em outros. Por exemplo, falhas de implementação no sincronismo entre *threads* podem acarretar o funcionamento incorreto da aplicação, e, assim, ambos os itens seriam prejudicados.

Lembre-se que a pontuação do trabalho está vinculada às etapas de entrega:

Item	Pontuação
Entrega da etapa 1 e da etapa 2, nos respectivos prazos (18/06 e 09/07)	25
Entrega apenas da etapa 2, no prazo (09/07)	12,5
Entrega apenas da etapa 2, fora de prazo (até 16/07)	6,25

ATENÇÃO! Caso haja evidências de improbidade acadêmica na execução do trabalho, o mesmo receberá nota zero e será encaminhado aos Colegiados de Cursos para as providências disciplinares cabíveis previstas no Regimento Geral da UFMG.

REFERÊNCIAS

[1] Amec Foster Wheeler Environment & Infrastructure UK Limited, *Oil and Gas Governance and Efficiency Study, Report 2: Review of Enhanced Oil Recovery (EOR) and Improved Oil Recovery (IOR) Technologies*, 2017. www.anp.gov.br/images/Palestras/Aumento_Fator_Recuperacao/Relatorio2_AMEC.pdf

[2] Haroldo dos Santos Rizzo Filho, "A Otimização de Gás Lift na Produção de Petróleo: Avaliação da Curva de Performance do Poço". Dissertação de mestrado, COPPE UFRJ, Rio de Janeiro, 2011. http://antigo.ppe.ufrj.br/ppe/production/tesis/haroldo_santos.pdf