UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
Departamento de Engenharia Eletrônica
DISCIPLINA: Automação em Tempo Real (ELT012)
PROFESSOR: Luiz Themystokliz. S. Mendes

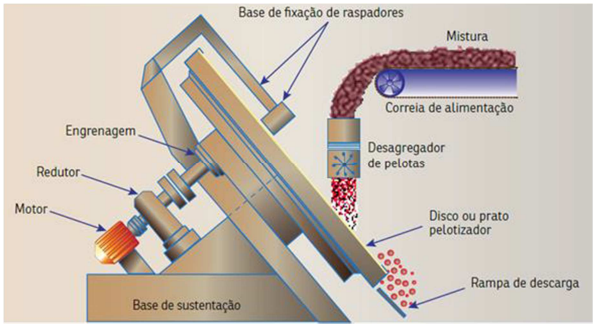
TRABALHO FINAL – 2020/2 – VALOR: 22 PONTOS

CONTEXTO INDUSTRIAL

Na indústria de mineração e beneficiamento de ferro, um dos produtos finais mais importantes para o processo de fabricação de aço são as pelotas (*pellets*), que correspondem a pequenas esferas de minério de ferro (com diâmetro entre 8 e 18mm) que resultam de um processo industrial denominado pelotização. As pelotas são empregadas nos altos-fornos de usinas siderúrgicas como matéria-prima para a fabricação do ferro-gusa, o qual, posteriormente, será transformado em aço nas unidades de aciaria destas siderúrgicas.

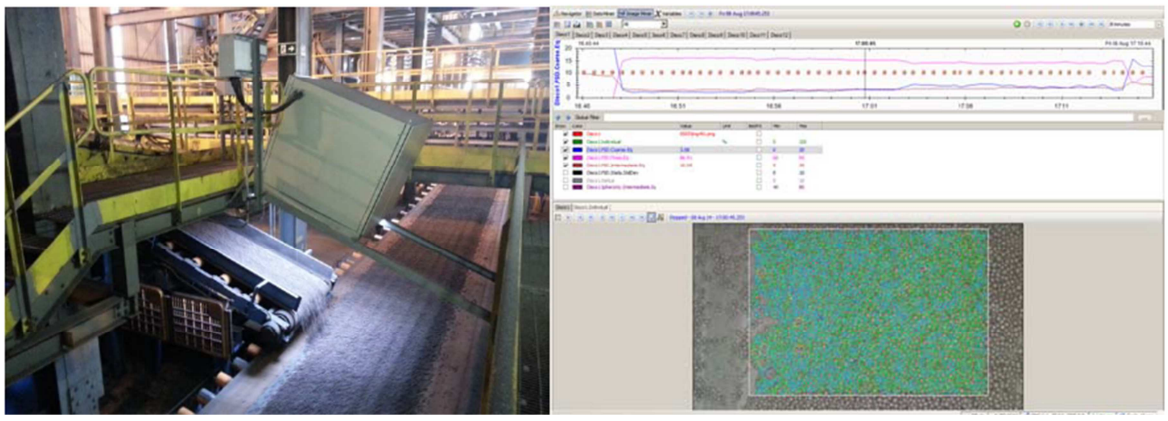
O processo de pelotização é composto de diversas etapas das quais as principais (por ordem de execução) são a moagem, o espessamento, a homogeneização, a filtragem, o pelotamento e a queima. Em particular, na etapa de pelotamento a polpa úmida de minério é derramada a uma vazão constante sobre discos de pelotamento cujo movimento contínuo de giro provoca a aglomeração da polpa na forma de pelotas "cruas", as quais, na etapa final de queima, passam por um forno do tipo grelha com o propósito de endurecimento e secagem.

Pelotas de minério de ferro geradas pelo processo de pelotização. *Crédito da imagem:* [1]



Disco de pelotamento empregado no processo de pelotização. *Crédito da imagem:* [2]

Um dos principais aspectos de qualidade a serem observados no processo de pelotização é a granulometria das pelotas "cruas", a qual deve ficar dentro de uma faixa restrita para que haja uma utilização eficiente das pelotas nos altos-fornos de usinas siderúrgicas (pelotas muito grandes ou muito pequenas diminuem o desempenho da produção de ferro-gusa), como também para otimizar a eficiência do forno de grelha (consumo de energia de ventiladores e consumo de gás). Entre as diversas tecnologias empregadas para este controle de qualidade, uma das mais modernas é a que emprega sistemas de visão computacional, os quais identificam em tempo real a variação de granularidade das pelotas "cruas" e fornecem parâmetros de controle importante para o ajuste do processo produtivo.



Sistema de visão para medição *online* da granulometria de pelotas cruas (esquerda) e software de visualização e análise das medições (direita). *Crédito da imagem:* [3]

DESCRIÇÃO DO TRABALHO

Uma empresa de mineração possui um sistema de controle automático para os discos de pelotização de uma de suas usinas de pelotização, baseado em um Controlador Lógico Programável (CLP). A empresa decidiu recentemente implantar um sistema-piloto de visão computacional para medição *online* da granulometria de pelotas cruas produzidas, a ser instalado próximo à esteira transportadora de saída de dois destes discos de pelotamento. Você foi contratado por esta empresa para desenvolver uma aplicação de software *multithread* responsável pela leitura de dados tanto do CLP quanto do sistema de medição de granulometria, apresentando-os respectivamente em dois projetores de vídeo, ambos na sala de controle dos discos de pelotização. O primeiro destes projetores apresentará dados do processo de pelotização para fins de controle e supervisão pelos operadores, ao passo que o segundo é dedicado à análise da variação de granulometria por parte de especialistas no processo de pelotização. A aplicação a ser desenvolvida deverá ser composta pelas seguintes tarefas (onde o termo "tarefa" pode designar tanto processos quanto *threads*):

1. **Tarefa de leitura do sistema de medição.** Deposita as mensagens provenientes do sistema de medição *online* de granulometria de pelotas na primeira lista circular em memória.
2. **Tarefa de leitura de dados do processo.** Deposita as mensagens provenientes do CLP na primeira lista circular em memória.
3. **Tarefa de captura de mensagens.** Corresponde a uma tarefa que retira as mensagens de dados da primeira lista circular em memória e, conforme seu tipo, envia-as para a tarefa de exibição de dados de processo ou as deposita em uma segunda lista circular em memória.
4. **Tarefa de exibição de dados do processo.** Esta tarefa recebe mensagens de dados de processo referentes aos discos de pelotamento e as exibe no projetor de vídeo da sala de controle.
5. **Tarefa de análise de granulometria.** Esta tarefa retira, da segunda lista circular me memória, as mensagens referentes ao sistema de medição de granulometria e as exibe no projetor dedicado à análise de granulometria.
6. **Tarefa de leitura do teclado**. Esta tarefa dá tratamento aos comandos digitados pelo operador.

Tarefa de leitura do sistema de medição. Esta tarefa simula a leitura de mensagens provenientes do sistema de medição de granulometria, com periodicidade aleatória entre 1 e 5 segundos. Todas as mensagens devem ser depositadas em uma primeira lista circular em memória RAM, com capacidade para 200 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. A temporização necessária a esta tarefa pode ser implementada por qualquer método visto no curso, **exceto** a função *Sleep()*.

Cada mensagem de dados de granulometria corresponde a uma cadeia de caracteres com tamanho fixo de 47 caracteres, no formato a seguir, onde os campos individuais são separados pelo delimitador ";" (ponto e vírgula):

NN	NNNN	NN	NNN.NN	NNN.NN	NNN.NN	NNN.NN	HH:MM:SS
TIPO	*NSEQ*	*ID DISCO*	*GR MED*	*GR MAX*	*GR MIN*	*SIGMA*	*TIMESTAMP*

Campo	Tamanho	Tipo	Descrição
TIPO	2	Inteiro	Sempre "00"
NSEQ	4	Inteiro	Número sequencial da mensagem [1...9999]
ID DISCO	2	Inteiro	Identificador do disco de pelotamento [1...2]
GR MED	6	Real	Valor médio da granulometria das pelotas [0,0 a 100 mm]
GR MAX	6	Real	Valor máximo da granulometria das pelotas [0,0 a 100 mm]
GR MIN	6	Real	Valor mínimo da granulometria das pelotas [0,0 a 100 mm]
SIGMA	6	Real	Desvio padrão da granulometria [0,0 a 100mm]
TIMESTAMP	8	Tempo	Hora, minuto e segundo

Exemplo: `"00;0985;02;012.50;021.44;003.71;001.42;15:27:12"`

O campo NSEQ deverá ser incrementado sequencialmente a cada mensagem gerada, voltando ao valor zero após alcançar a contagem máxima, e o campo TIMESTAMP deverá ter a hora corrente. Os demais campos (com exceção do campo TIPO) deverão ter seus valores gerados aleatoriamente nas faixas indicadas.

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

- O primeiro evento sinalizará que esta tarefa 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 evento de sinalização de bloqueio acima descrito.

Tarefa de leitura de dados de processo. Esta tarefa simula a leitura de mensagens provenientes do CLP, com periodicidade fixa de 500 ms. Todas as mensagens devem ser depositadas na mesma lista circular em memória RAM utilizada pela tarefa de leitura do sistema de medição. 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. A temporização necessária a esta tarefa pode ser implementada por qualquer método visto no curso, **exceto** a função *Sleep()*.

Cada mensagem de dados de processo deve corresponder a uma cadeia de caracteres com tamanho fixo de 57 caracteres, no formato a seguir, onde os campos individuais são separados pelo delimitador ";" (ponto e vírgula):

NN	NNNN	NN	NNNN.N	NNNN.N	NNNN.N	NN.N	NNN.N	HH:MM:SS:MSS
TIPO	*NSEQ*	*ID DISCO*	*VZ ENT*	*VZ SAIDA*	*VEL*	*INCL*	*POTÊNCIA*	*TIMESTAMP*

Campo	Tamanho	Tipo	Descrição
TIPO	2	Inteiro	Sempre "99"
NSEQ	4	Inteiro	Número sequencial da mensagem [1...9999]
ID DISCO	2	Inteiro	Identificador do disco de pelotamento [1...6]
VZ ENT	6	Real	Vazão mássica de entrada da polpa de minério (kg/h) [0...1000]
VZ SAÍDA	6	Real	Vazão mássica de saída de pelotas cruas (kg/h) [0 ... 1000]
VEL	6	Real	Velocidade de rotação do disco (cm/s) [0 ... 1000]
INCL	4	Real	Ângulo de inclinação do disco (graus) [0 ... 45]
POTÊNCIA	5	Real	Potência consumida (kWh) [0 ... 2]
TIMESTAMP	12	Tempo	Hora, minuto, segundo e milissegundo

Exemplo: `"99;3455;05;0546.4;0532.3;0010.1;12.5;0.434;13:44:12.564"`

O campo NSEQ deverá ser incrementado sequencialmente a cada mensagem gerada, voltando ao valor zero após alcançar a contagem máxima, e o campo TIMESTAMP deverá ter a hora corrente. Os demais campos (com exceção do campo TIPO) deverão ter seus valores gerados aleatoriamente nas faixas indicadas.

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 leitura do sistema de medição.

Tarefa de captura de mensagens. Corresponde a uma tarefa que deve consumir as mensagens presentes na primeira lista circular em memória. As mensagens de dados de processo devem ser enviadas à tarefa de exibição de dados de processo por meio de *pipes* ou *mailslots*. Já as mensagens do sistema de medição de granulometria devem ser depositadas em uma segunda lista circular em memória com capacidade de 100 mensagens; caso não haja espaço na lista, esta tarefa deve bloquear-se até que haja alguma posição livre no arquivo, alertando este fato na console principal (a console associada à tarefa de leitura do teclado; vide a seguir) por meio de um texto adequado. Além disto, deve ser feito o sincronismo apropriado de acesso a esta lista circular entre esta tarefa e a tarefa de análise de granulometria.

Esta tarefa deve sincronizar sua execução com a tarefa de leitura do teclado através de um par de objetos "evento" (um par para cada tarefa), nas mesmas condições descritas para a tarefa de leitura do sistema de medição.

Tarefa de análise de granulometria. Esta tarefa deve retirar mensagens medição de granulometria da segunda lista circular em memória e exibi-las em uma janela de console exclusiva, que simulará o respectivo projetor na sala de controle do pelotamento. As mensagens devem ser exibidas no seguinte formato:

```
HH:MM:SS NSEQ: #### ID: ## GMED: ###### GMAX: ###### GMIN: ###### SIG: ######
```

Como antes, a notação `"######"` representa o valor do respectivo item da mensagem de medição de granulometria.

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 leitura do sistema de medição.

Tarefa de exibição de dados de processo. Esta tarefa recebe mensagens da tarefa de captura de defeitos das tiras e as exibe em uma janela de console exclusiva, que simulará o projetor dedicado existente na sala de controle do pelotamento. As mensagens de dados de processo devem ser exibidas no seguinte formato:

```
HH:MM:SS.MSS NSEQ: #### ID: ## VZ E: ###### VZ S: ###### V: ###### ANG: #### P: #####
```

A notação `"######"` representa o valor do respectivo item da mensagem de dados de processo.

Esta tarefa recebe também uma mensagem via *pipes* ou *mailslots* da tarefa de tratamento do teclado para que sua janela de console seja limpa

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 leitura do sistema de medição.

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

g	Notifica à tarefa de leitura do sistema de medição de granulometria 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.
c	Idem, com relação à tarefa de leitura de dados de processo.
r	Idem, com relação à tarefa de retirada de mensagens..
p	Idem, com relação à tarefa de exibição de dados de processo.
a	Idem, com relação à tarefa de análise de granulometria.
l	Notifica à tarefa de exibição de dados de processo 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.

ETAPAS DO TRABALHO

O trabalho será executado em duas etapas, porém sua pontuação está condicionada à entrega de ambas as etapas. Não serão pontuados os trabalhos cuja entrega corresponda apenas à primeira ou à segunda etapa.

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

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

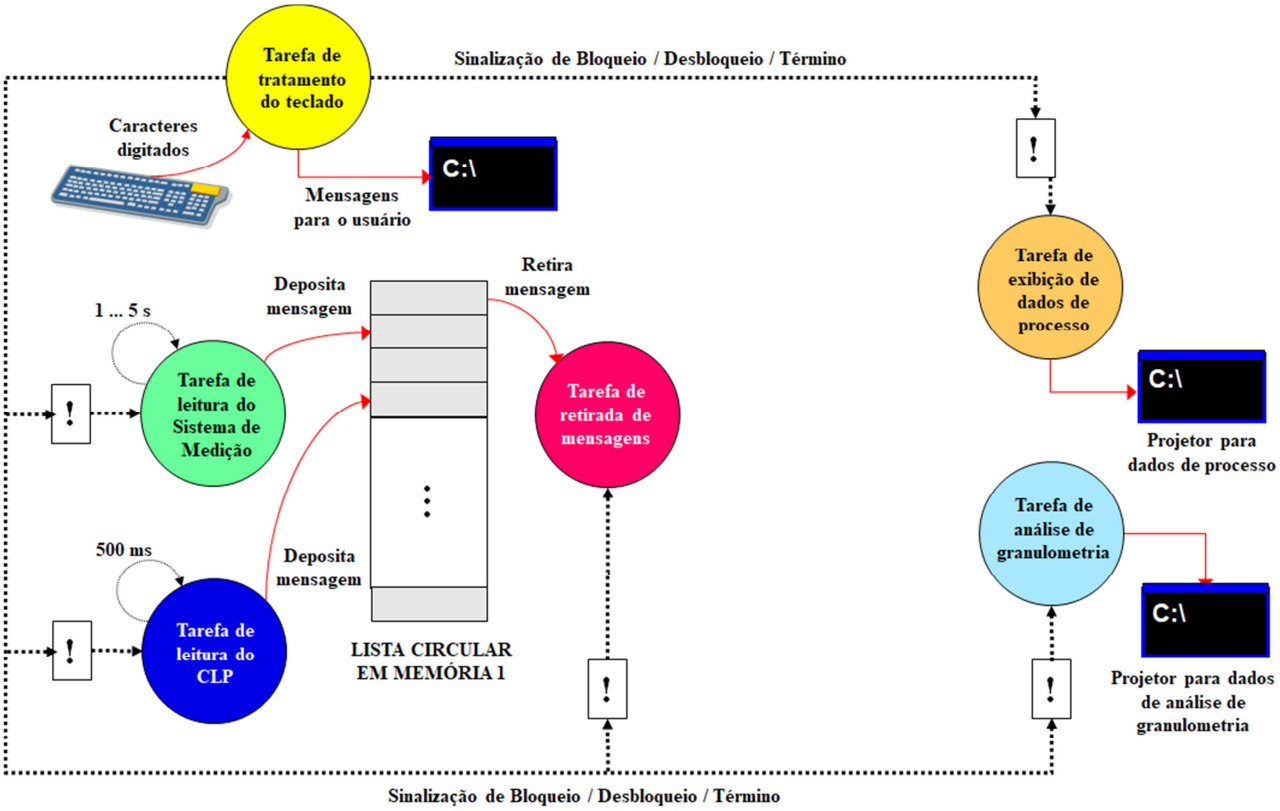
- Arquitetura multitarefa/*multithread* da aplicação. Em outras palavras, corresponde à definição de como as tarefas anteriormente descritas serão modeladas em termos de processos e *threads*;
- Disparo da aplicação com a consequente execução de seus processos e *threads*;
- Implementação do mecanismo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

Leve em conta as seguintes observações sobre esta etapa do trabalho:

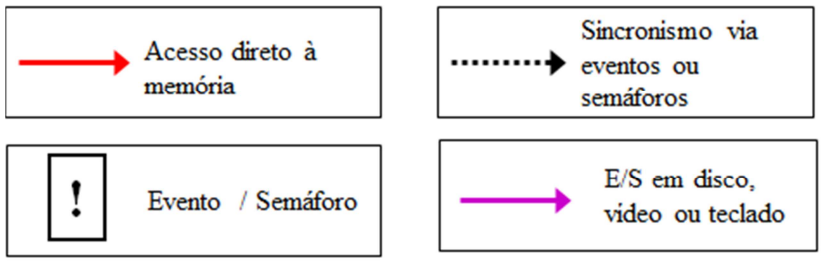
1. Observe que apesar de o programa corresponder a mais de um arquivo executável, lembre-se que, do ponto de vista do usuário, seu disparo deve ocorrer a partir de um único arquivo executável. Em outras palavras, você deve definir um processo principal a partir do qual os demais processos serão disparados 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 um *buffer* circular está descrito na seção "O problema dos produtores e consumidores" do livro "Programação Concorrente em Ambiente Windows". Note que no presente trabalho há duas tarefas "consumidoras" que retiram diferentes mensagens na lista circular em memória, e, assim, cada uma delas deverá manter e manipular apontadores individuais para a próxima mensagem a ser retirada.

Para fins de certificação de conclusão desta etapa, as tarefas de leitura do sistema de medição, de leitura de dados de processo, e de retirada de mensagens deverão exibir mensagens na console principal indicando o seu funcionamento em termos de depósito e consumo de mensagens, bem como seus estados de bloqueio/desbloqueio. As tarefas de exibição de dados de processo e de análise de granulometria deverão apenas apresentar uma mensagem simples em suas janelas de console dedicadas, informando os seus estados de bloqueio/desbloqueio. Estas mensagens intermediárias deverão ser substituídas por aquelas reais na etapa 2.

O diagrama de relacionamentos a seguir apresenta a estrutura da aplicação resultante ao final desta primeira etapa.



Legenda:

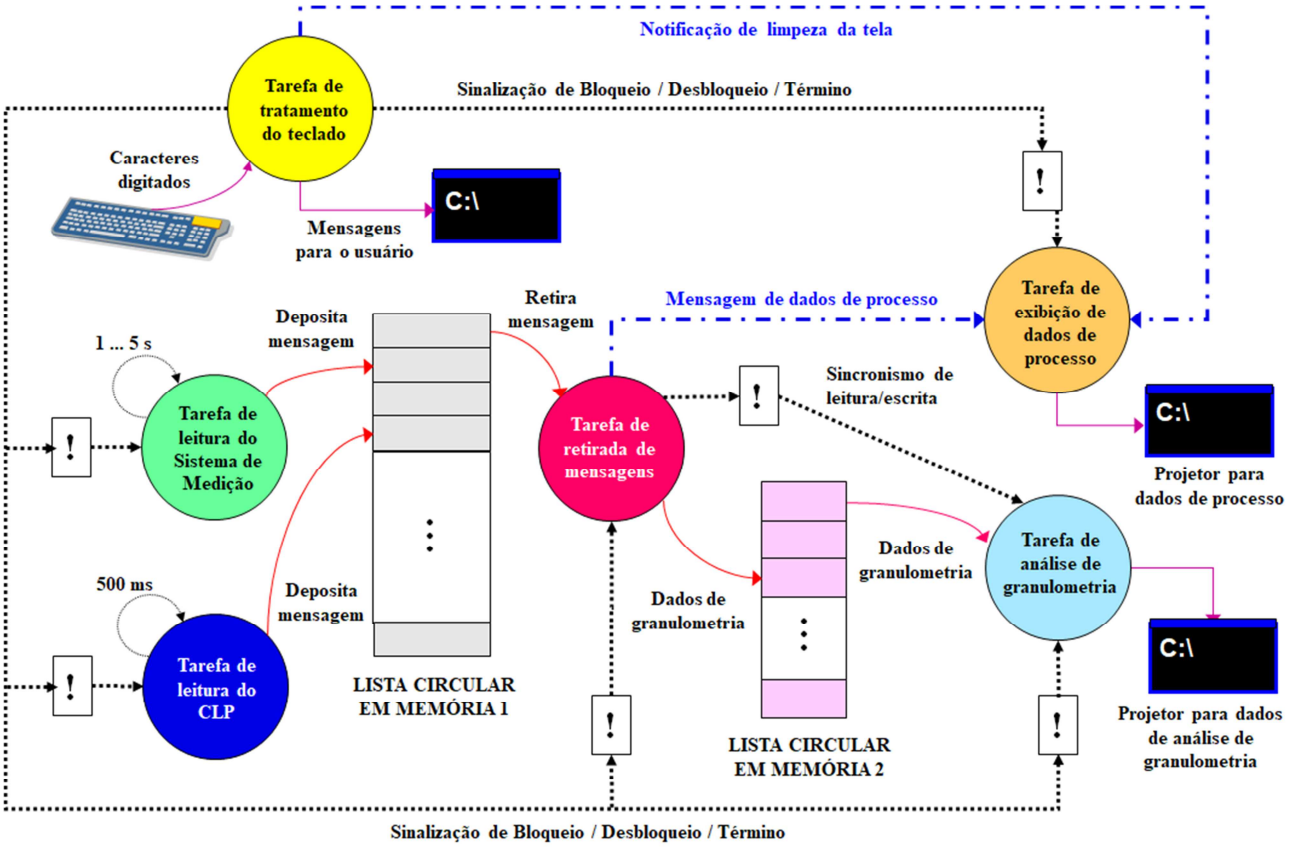


ETAPA 2 – Temporização, IPC e conclusão da aplicação

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

- Temporizações de depósito de mensagens na primeira lista circular em memória pela tarefa de leitura do sistema de medição e pela tarefa de leitura de dados de processo, como especificado na descrição das mesmas;
- Criação da segunda lista circular em memória e inserção/retirada das mensagens de dados de granulometria, respectivamente pelas tarefas de retirada de mensagens e de análise de granulometria, conforme especificado nas descrições das mesmas;
- Criação do mecanismo de sincronização de acesso a esta lista circular;
- Implementação dos mecanismos de IPC (*Inter-Process Communication*) entre as tarefas de leitura do teclado, de retirada de mensagens e de exibição de dados de processo, conforme especificado na descrição das mesmas;
- 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 a seguir apresenta a estrutura da aplicação resultante ao final da segunda etapa.



Legenda:

INSTRUÇÕES

1. O trabalho deve ser feito de forma **individual** ou em grupo de **2 alunos** (dupla). Se em dupla, e a critério exclusivo do professor, os alunos estarão sujeitos à necessidade de comprovação de que dividiram a carga de trabalho igualmente entre si.
2. Para desenvolver os programas, deverá ser utilizada exclusivamente a ferramenta *Microsoft Visual Studio Community Edition*, que pode ser obtida gratuitamente da Microsoft. Não serão considerados trabalhos executados com qualquer outra ferramenta ou com diferente versão do *Visual Studio*. 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 cada uma das etapas, deve ser submetido exclusivamente via *Moodle*, através dos seguintes arquivos:
   - Arquivo .ZIP ou .RAR contendo as pastas (diretórios) que correspondem às respectivas "soluções" 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 será de 2 Mb. *DICA: Você pode remover os arquivos de extensão .exe, .ipch e .db presentes em sub-diretórios 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 cerca de 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 de entrega:

   Etapa 1 - **23h59min** do dia **28/02/2021 (domingo)**

   Etapa 2 - **23h59min** do dia **21/03/2021 (domingo)**.

   Não serão permitidos atrasos com relação à etapa 1. Quanto à etapa 2, trabalhos entregues até 28/03/2021 terão sua pontuação reduzida a 50%. Não serão aceitos trabalhos entregues após 28/03/2021.

   Reafirma-se que a entrega de ambas as etapas é obrigatória para a pontuação do trabalho.

   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.
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 traiçoeira, 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 especificarem soluções de projeto, a fim de exercitarem a capacidade de agirem como projetistas de sistemas. Desta forma, exceto onde expressamente indicado neste 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 apresentados nos capítulos 2 a 6 do livro "Programação Concorrente em Ambiente Windows". Estude cuidadosamente as funções a serem executadas pelas tarefas, identifique neste livro os programas de exemplo correspondentes 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 itens de pontuação a serem atribuídos ao trabalho.

Item	Pontuação
Criação de processos e *threads*	4,0
Sincronismo entre *threads* e IPC	4,0
Funcionamento da aplicação	6,0
Atendimento aos requisitos do enunciado	4,0
Documentação	4,0

Observe, contudo, que os itens 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.

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] Vale.com, "Você sabe o que é pelotização?" *http://www.vale.com/brasil/PT/aboutvale/news/Paginas/voce-sabe-o-que-e-pelotizacao.aspx*

[2] Passos, L. A. S., Moreira, J. L., Jorge, A., & Cavalcante, M. V.S. "Melhoria no desempenho do processo de produção de pelotas de minério de ferro em discos de pelotização pela utilização de sistemas otimizantes com lógica nebulosa". 44º Seminário de Redução de Minério de Ferro e Matérias-primas, 15º Simpósio Brasileiro de Minério de Ferro e 2º Simpósio Brasileiro de Aglomeração de Minério de Ferro, 15 a 18 de setembro de 2014, Belo Horizonte, MG, Brasil.

[3] Gontijo, H. M., Gontijo, M. M., Freitas, R. O., Pires, L. C., Santos, W. B., Fernandes, R. B., & Ferri, L. N., "Implementação do sistema de controle avançado no forno de endurecimento". 46º Seminário de Redução de Minério de Ferro e Matérias-primas, 17º Simpósio Brasileiro de Minério de Ferro e 4º Simpósio Brasileiro de Aglomeração de Minério de Ferro, parte integrante da ABM Week, realizada de 26 a 30 de setembro de 2016, Rio de Janeiro, RJ, Brasil.