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 – 2023/2 – VALOR: 30 PONTOS

INTRODUÇÃO: Contexto em Automação Industrial

O processo siderúrgico de fabricação de aço é composto de diversas etapas. Uma das etapas iniciais é a obtenção de minério de ferro líquido ("ferro-gusa") a partir de partículas sólidas de minério de ferro ("finos de minério"), realizada nos altos-fornos da planta siderúrgica. O ferro-gusa é então transportado por meio de grandes recipientes denominados "panelas" ou de vagões ferroviários especiais ("carros torpedo") para as aciarias, nas quais será realizada a etapa seguinte de conversão do ferro-gusa em aço líquido (Figs. 1 e 2).

Figura 1: Carro Torpedo em uma Usina Siderúrgica.
Fonte: observatoriometroferro.ufsc.br

Figura 2: Panela recebendo ferro-gusa do alto-forno.
Fonte: www.interstahl.com

O ferro-gusa, contudo, possui em geral um alto índice de enxofre, o qual degrada algumas propriedades mecânicas do aço como ductibilidade, tenacidade, conformabilidade, soldabilidade e resistência à corrosão [1, 2]. Assim, antes de ser utilizado nas aciarias, o ferro-gusa deve sofrer um processo de remoção de enxofre, conhecido como dessulfuração. Um dos métodos mais econômicos, confiáveis e eficazes para a dessulfuração de ferro-gusa é a lança refratária de imersão [2], por meio da qual certos reagentes (p. ex. magnésio) são injetados por meio de um gás inerte (p. ex. nitrogênio) na panela ou no carro-torpedo contendo gusa (Figs. 3 e 4).

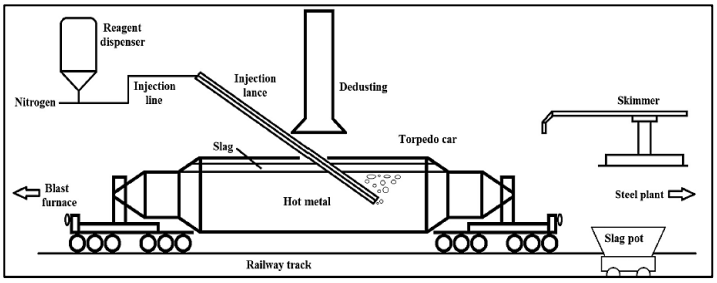


Figura 3: Esquema do processo de dessulfuração em um carro torpedo.
Fonte: [2]

Figura 4: Dessulfuração em uma panela com ferro-gusa.
Fonte: www.tebulo.com

No processo de dessulfuração, a reação química resultante pode ser representada como [2]

$$(S)_{Fe}+(O^{2-})_{escória}=(S^{2-})_{escória}+(O)_{Fe}\,.$$

Desta forma, a adição de reagentes ao ferro-gusa faz com que o enxofre presente no mesmo transforme-se em sulfetos que, por terem menor densidade, sobem à superfície do metal líquido e formam uma camada de rejeitos denominada escória, a qual então é removida da panela ou do carro-torpedo. O processo de dessulfuração é capaz de reduzir o teor de enxofre do ferro-gusa em até 0,002% [2].

DESCRIÇÃO DO TRABALHO

O setor de altos-fornos de uma indústria siderúrgica conta com dois sistemas de dessulfuração independentes que operam sobre panelas de ferro-gusa, sendo cada sistema controlado por um Controlador Lógico Programável (CLP). Em adição, um terceiro CLP monitora eventos críticos dos processos de dessulfuração em andamento, notificando os operadores sobre a ocorrência destes eventos em um pequeno painel com uma IHM (Interface Homem-Máquina) dedicada, localizado na sala de controle. Para aumentar o índice de desempenho do processo de dessulfuração, a indústria decidiu instalar dois terminais de vídeo na respectiva cabine de operação, de forma a

apresentar ao operador uma visão integrada dos dados de processo e dos alarmes críticos ocorridos durante a dessulfuração.

Assim, a empresa em que você trabalha foi contratada para desenvolver uma aplicação de software *multithread* destinada a integrar as informações destes três CLPs, dirigindo-as para estes dois terminais de vídeo na cabine de operação dos sistemas de dessulfuração. Os dados de processo presentes nos dois CLPs dos sistemas de dessulfuração devem ser apresentados no terminal de vídeo de operação, ao passo que os alarmes críticos detectados pelo terceiro CLP devem ser exibidos no terminal de vídeo de alarmes. A aplicação a ser desenvolvida deverá ser composta pelas seguintes tarefas (onde o termo "tarefa" pode designar tanto processos quanto *threads*):

1. **Tarefas de leitura dos CLPs**: Correspondem a três tarefas que simulam a leitura das mensagens dos CLPs. Os dados dos CLPs de processo são depositados em uma lista circular em memória, ao passo que alarmes provenientes do CLP de monitoração são enviados diretamente à tarefa de exibição de alarmes (vide abaixo).
2. **Tarefa de retirada de mensagens.** Esta tarefa retira, da lista circular em memória, as mensagens de dados dos CLPs de processo e redireciona as mesmas para uma segunda lista circular em memória ou para a tarefa de exibição de alarmes, conforme os tipos destas mensagens.
3. **Tarefa de exibição de dados de processo.** Esta tarefa retira, da segunda lista circular em memória, mensagens de dados de processo e as exibe no terminal de vídeo de operação.
4. **Tarefa de exibição de alarmes.** Esta tarefa recebe mensagens de alarmes da tarefa de leitura do CLP de monitoração e da tarefa de retirada de mensagens, exibindo-as no terminal de vídeo dedicado a alarmes.
5. **Tarefa de leitura do teclado**. Esta tarefa dá tratamento aos comandos digitados pelo operador.

Tarefas de leitura dos CLPs. Duas destas tarefas simulam a leitura de dados de processo dos CLPs que controlam o processo de dessulfuração, através da geração de mensagens correspondentes com periodicidade fixa de 500ms. Todas estas mensagens devem ser depositadas em uma primeira lista circular em memória RAM com capacidade para 100 mensagens. Caso esta lista circular esteja cheia no momento de depósito de alguma mensagem, estas duas tarefas devem 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). Já a terceira tarefa de leitura do CLP simula a leitura de dados do CLP de monitoração, gerando mensagens de alarmes com periodicidade aleatória entre 1 e 5 segundos[1]. A temporização necessária a estas três tarefas pode ser implementada por qualquer método visto no curso, **exceto** a função *Sleep()*.

As mensagens de dados de processo correspondem sempre a cadeias de caracteres com tamanho fixo de 40 caracteres, no formato a seguir, onde os campos individuais são separados pelo delimitador ";" (ponto e vírgula):

NNNNN	N	NN	NNNN.N	NNNN.N	NNNN.N	HH:MM:SS
NSEQ	ID	DIAG	PRES. INTERNA	PRES. INJEÇÃO	TEMP	*TIMESTAMP*

Campo	Tamanho	Tipo	Descrição
NSEQ	5	Inteiro	Número sequencial da mensagem [1...99999]
ID	1	Inteiro	Identificação do CLP de controle da dessulfuração (1 ou 2)
DIAG	2	Inteiro	Diagnóstico dos cartões de E/S do CLP
PRES. INTERNA	6	Real	Pressão interna na panela de gusa (Pa)
PRES. INJEÇÃO	6	Real	Pressão de injeção de Nitrogênio (Pa)
TEMP	6	Real	Temperatura no interior da panela de gusa (C)
TIMESTAMP	8	Tempo	Hora, minuto e segundo

Exemplo: **"**`00455;1;23;0104.2;0123.6;1299.5;21:44:13`**"**

Na mensagem acima, o valor "55" no campo DIAG indica falha de hardware nos cartões de E/S do CLP. Neste caso, os campos de PRES. INTERNA, PRES. INJEÇÃO e TEMP devem ser preenchidos com o valor "0000.0".

Por sua vez, as mensagens de alarmes críticos devem corresponder a cadeias de caracteres com tamanho fixo de 17 caracteres, no formato a seguir, onde os campos individuais também são separados pelo delimitador ";" (ponto e vírgula):

NNNNN	AAAAAAAA	HH:MM:SS
NSEQ	ID	*TIMESTAMP*

Campo	Tamanho	Tipo	Descrição
NSEQ	5	Inteiro	Número sequencial da mensagem [1...99999]
ID	2	Alfanumérico	Identificador do alarme
TIMESTAMP	8	Tempo	Hora, minuto e segundo

Exemplo: **"**`41023;32;08:45:21`**"**

[1] Em um processo real, esta periodicidade é longa, tipicamente na escala de dezenas de minutos a horas. Aqui estamos exagerando grandemente a mesma para propósitos didáticos.

Na montagem das mensagens de dados processo e de alarmes críticos, 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 TIMESTAMP deverá ter a hora corrente. Os demais campos deverão ter seus valores gerados aleatoriamente.

As tarefas de leitura dos CLPs devem sincronizar suas execuções com a tarefa de leitura do teclado através de um par de objetos "evento" (um par para cada tarefa):

- O primeiro evento sinalizará que a tarefa de leitura do CLP 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 retirada de mensagens. Esta tarefa deve consumir mensagens de dados de processo presentes na primeira lista circular em memória, identificar o conteúdo do campo DIAG das mesmas e, conforme seu valor, depositá-la em um segundo arquivo circular em memória com 50 posições (DIAG diferente de "55") ou repassá-la para a tarefa de exibição de alarmes, por meio de *pipes* ou *mailslots* (DIAG igual a 55). Caso não haja espaço na segunda lista circular em memória, a 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).

A tarefa de retirada de mensagens deve sincronizar sua execução com a tarefa de leitura do teclado através de um par de objetos "evento", nas mesmas condições descritas para as tarefas de leitura dos CLPs.

Tarefa de exibição de dados de processo. Esta tarefa deve retirar mensagens de dados de processo da segunda lista circular em memória, exibindo-as em uma janela de console exclusiva que simulará o terminal de vídeo da operação. As mensagens devem ser exibidas no seguinte formato:

```
HH:MM:SS NSEQ: ###### PR INT: ####.# PR N2: ####.# TEMP: ####.#
```

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 as tarefas de leitura dos CLPs.

Tarefa de exibição de alarmes. Esta tarefa recebe mensagens de alarmes tanto da tarefa de retirada de mensagens quanto da tarefa de leitura do CLP de monitoração, exibindo-as em uma janela de console exclusiva que simulará o terminal dedicado a alarmes. As mensagens de alarme devem ser exibidas nos seguintes formatos, dependendo do tipo de alarme recebido:

Alarme crítico proveniente do CLP 3

```
HH:MM:SS NSEQ: ##### ID: ## TextoTextoTextoTextoTextoTexto
```

Mensagem de dados com campo DIAG = 55

```
HH:MM:SS NSEQ: ##### FALHA DE HARDWARE CLP No. #
```

Como antes, a notação "`#####`" representa o valor do respectivo item da mensagem de dados de processo. No caso das mensagens de alarmes críticos de processo, a notação "`TextoTexto...`" representa a informação textual correspondente ao código do alarme. Ao menos 10 textos diferentes devem ser empregados nestas mensagens. Pesquise a Internet para obter exemplos de textos referentes ao processo de dessulfuração de ferro-gusa.

A tarefa de exibição de alarmes deve sincronizar sua execução com a tarefa de leitura do teclado através de um par de objetos "evento", nas mesmas condições descritas para as tarefas de leitura dos CLPs.

Tarefa de leitura do teclado. Esta tarefa realiza o tratamento dos seguimtes caracteres do teclado:

"1"	Notifica à tarefa de leitura do CLP #1 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.
"2"	Idem, com relação à tarefa de leitura do CLP #2.
"m"	Idem, com relação à tarefa de leitura de CLP de monitoração de alarmes críticos.
"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 exibição de alarmes.
ESC	Notifica todas as demais 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: 30 pontos;**
- **Entrega apenas da etapa 2, no respectivo prazo indicado – 20 pontos;**
- **Entrega apenas da etapa 1 – não pontuado.**

O propósito desta divisão em etapas é permitir o desenvolvimento do trabalho de forma progressiva, utilizando os conceitos vistos na disciplina à medida que forem dados, e também de evitar que este desenvolvimento seja integralmente feito apenas às vésperas da data-limite de entrega. A etapa 1 aborda os conceitos de criação de processos e *threads* e de sincronização, e a etapa 2 complementa a anterior adicionando funcionalidades correspondentes aos recursos de temporização e IPC (*Inter-Process Communication*).

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 primeira lista circular em memória RAM, com o depósito de mensagens na mesma pelas tarefas de leitura dos CLPs e o consumo destas pela tarefa de retirada de mensagens, levando em conta aspectos de sincronização eventualmente necessários;
- Temporizações provisórias das tarefas de leitura dos CLPs por meio da função *Sleep()*. Estas temporizações serão substituídas pelas definitivas na etapa 2 do trabalho.

O diagrama de relacionamentos na Fig. 3 apresenta a estrutura da aplicação correspondente à primeira etapa.

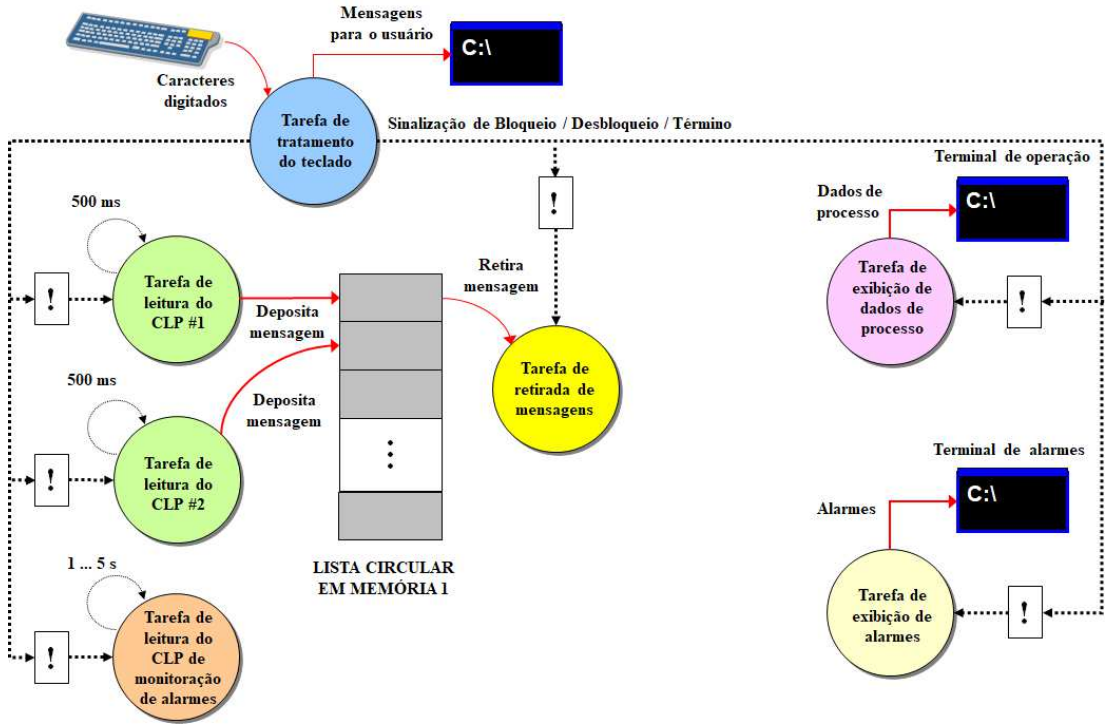


Figura 3: Diagrama de relacionamentos – Etapa 1. A legenda sobre a notação empregada encontra-se após a Fig. 4.

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 lista circular em memória, há uma tarefa "produtora" e duas "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 a lista circular como uma lista encadeada (*linked list*).

Para fins de certificação de conclusão desta etapa:

- Todas as tarefas devem indicar, em suas respectivas consoles, seus estados de bloqueio/desbloqueio;
- A tarefa de retirada de mensagens deverá exibir, na console principal, as mensagens retiradas da primeira lista circular em memória;
- As tarefas de exibição de dados de processo e de exibição de alarmes deverão apenas emitir uma mensagem simples indicando que estão em execução.

ETAPA 2 – Temporizações, IPC e conclusão da aplicação

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

- Temporizações definitivas das tarefas de leitura dos CLPs, nas periodicidades indicadas na descrição das mesmas, sem o uso da função *Sleep()*;
- Criação da segunda lista circular em memória e inserção/retirada na mesma das mensagens de dados de processo;
- Implementação do mecanismo de IPC (*Inter-Process Communication*) entre (1) a tarefa de leitura do CLP de monitoração e a tarefa de exibição de alarmes e (2) a tarefa de retirada de mensagens e a tarefa de exibição de alarmes, conforme especificado em suas respectivas 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 4 apresenta a estrutura da aplicação correspondente à segunda etapa.

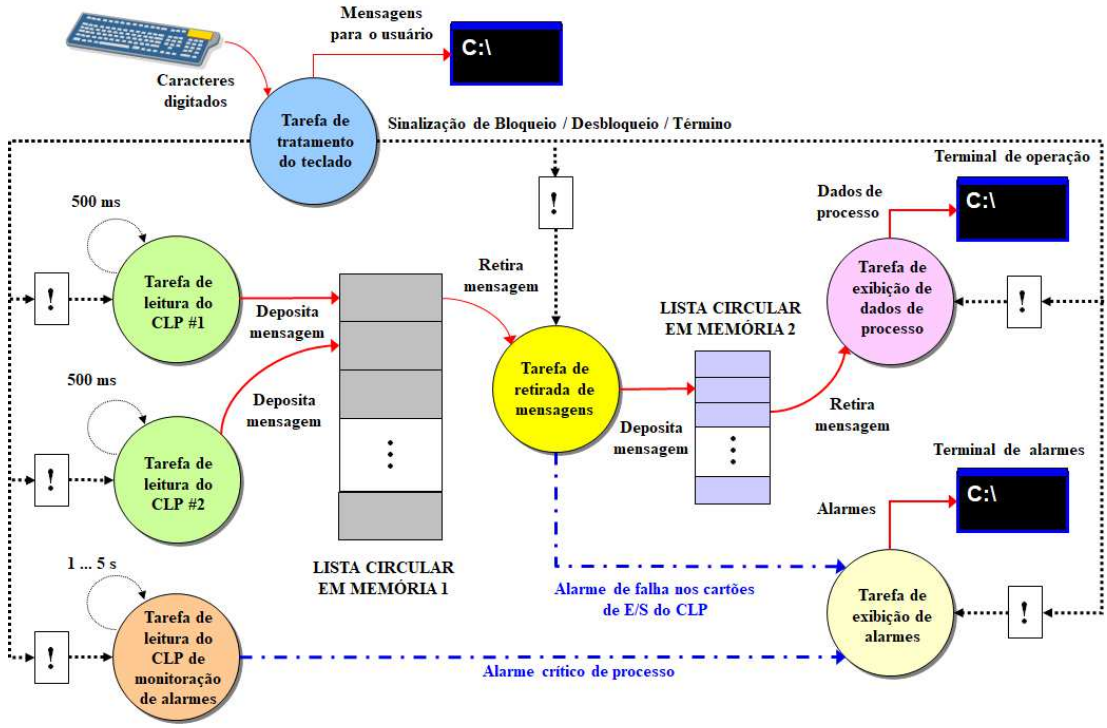
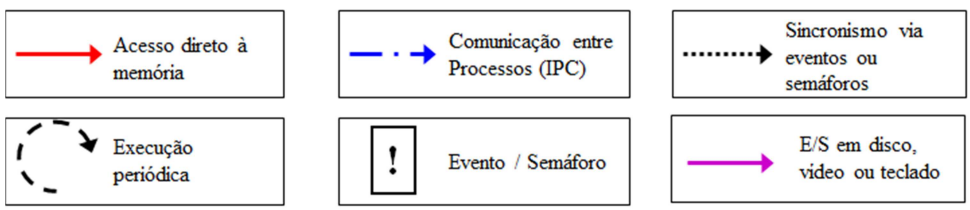


Figura 4: Diagrama de relacionamentos – Etapa 2

Legenda para as figuras 3 e 4:



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, sua nota 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 **12/11/2023 (domingo)**

   Etapa 2 - **23h59min** do dia **03/12/2023 (domingo)**

   Não serão permitidos atrasos com relação à etapa 1. Quanto à etapa 2, trabalhos entregues até **10/12/2023** terão sua pontuação reduzida a 50%. Não serão aceitos trabalhos entregues após 10/12/2023.

   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 deste 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 (até 5 pontos)

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

- Utilização da biblioteca *Pthreads-Win32* para fins de criação de *threads* e sincronização via *mutexes* ou semáforos. Neste caso, baixe a versão 2.9.1 desta biblioteca (atenção: esta não é a última versão), 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 (12/11 e 03/12)	30
Entrega apenas da etapa 2, no prazo (03/12)	20
Entrega apenas da etapa 2, fora de prazo (até 10/12)	10

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] Rafaela Pacheco Malvão dos Santos, "Desenvolvimento do modelo de previsão de enxofre na dessulfuração de gusa em carro torpedo". Dissertação de mestrado, UFF, Volta Redonda (RJ), 2016. https://sucupira.capes.gov.br/sucupira/public/consultas/coleta/trabalhoConclusao/viewTrabalhoConclusao.jsf?popup=true&id_trabalho=4359073

[2] Schrama F. N.H, Beunder E. M., Van der Berg B., Yang Y., Boom R., "*Sulphur removal in ironmaking and oxygen steelmaking*". Ironmaking & Steelmaking Processes, Products and Applications Vol. 44, Issue 4, 2017. https://www.tandfonline.com/doi/full/10.1080/03019233.2017.1303914