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 – 2025/1 – VALOR: 25 PONTOS

INTRODUÇÃO: Contexto em Automação Industrial

Em sistemas ferroviários, uma das principais preocupações é com a segurança das composições ferroviárias de forma a evitar acidentes, especialmente descarrilhamentos, não só em trens de passageiros como também de carga, pelos óbvios prejuízos e transtornos que acarretam.

Uma dos fatores que podem provocar descarrilhamentos é o superaquecimento dos rodeiros (conjunto de rodas) dos vagões, causado por condições anormais de operação como rolamentos danificados ou freios atuados indevidamente durante o tráfego da composição ferroviária, fazendo-o se arrastar sobre os trilhos ao invés de girar sobre os mesmos, e assim causando deformações e eventual ruptura dos rodeiros (Figs. 1 e 2).

Figura 1: Composição ferroviária com rodas travadas em um vagão, provocando superaquecimento. Fonte: MRS Logística, 2002

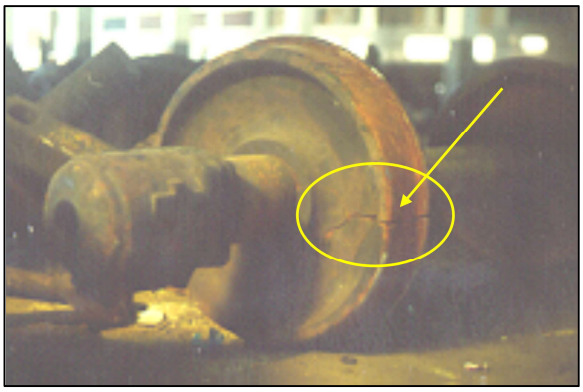


Figura 2: Rodeiro de vagão trincado. Fonte: MRS Logística, 2002

A tecnologia de "detecção de rodas quentes" (*hot wheel detection*) é comumente empregada em operadoras ferroviárias para a prevenção de acidentes, e corresponde a conjuntos de sensores instalados de forma espaçada ao longo das vias ferroviárias capazes de detectar rodas quentes, rolamentos quentes ou ambos. Cada conjunto de sensores (denominado *Hotbox*) é usualmente formado por um detector magnético de presença de rodas e por um pirômetro infravermelho (Figs. 3 e 4). Durante a passagem de uma composição ferroviária, os sinais independentes destes sensores são capturados por unidades remotas de aquisição de dados e então transmitidos destas a um sistema de controle que correlaciona, em tempo real, a presença dos rodeiros com a temperatura do pirômetro, gerando alarmes aos operadores do centro de controle ferroviário. Estes alarmes, quando não críticos, orientam os maquinistas das locomotivas, p. ex. fazendo-os desacelerar as composições ferroviárias, e, em casos críticos (tipicamente temperaturas acima de 350 C), até mesmo determinando a completa paralisação da composição ferroviária para inspeção e eventuais reparos.

Figura 3: Detectores de rodas quentes instalados em ambos os lados de trilhos ferroviários. Fonte: MRS Logística, 2002

Figura 4: Detalhamento do sensor magnético de presença (lado interno dos trilhos) e do pirômetro infravermelhoRodeiro de vagão trincado. Fonte: MRS Logística, 2002

DESCRIÇÃO DO TRABALHO

Uma empresa que atua no segmento de transporte ferroviário acaba de implantar um novo trecho de ferrovia em sua malha ferroviária. Todos os sinais de campo deste novo trecho (atuação em desvios, sinaleiros PARE/SIGA, abertura e fechamento de cancelas em passagens de nível, etc.) são capturados por estações remotas inteligentes de E/S (entradas e saídas), que, por sua vez, comunicam-se em intervalos de 2000 ms com vários Controladores Lógicos Programáveis (CLPs) localizados no Centro de Controle de Operações (CCO) da empresa. Em adição, vários detectores de roda quente (*hotboxes*) estão instalados ao longo deste novo trecho ferroviário, para detecção de aquecimento anormal dos rodeiros dos vagões. As informações destes detectores são enviadas via rede *wireless* para os CLPs a cada 500 ms.

Você foi contratado pela empresa de transportes ferroviários para desenvolver uma aplicação de software *multithread* responsável pela leitura dos dados das remotas de E/S e dos detectores de roda quente contidos nos CLPs, apresentando-os adequadamente em dois terminais de vídeo presentes no CCO. O primeiro destes terminais apresentará as informações de *status* da sinalização ferroviária, ao passo que o segundo é dedicado para visualização de rodas quentes. 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 dos CLPs**: Corresponde a uma tarefa que realiza a leitura dos dados de sinalização ferroviária e dos *hotboxes* contidos nos CLPs. Estas mensagens devem ser depositadas em uma lista circular em memória.
2. **Tarefa de captura de dados de sinalização ferroviária.** Esta tarefa retira, da lista circular em memória, as mensagens de dados das remotas de E/S, depositando-as em um arquivo em disco com capacidade de 200 mensagens ou, no caso de alguma mensagem indicadora de falha de hardware em alguma remota, enviando-a para a tarefa de visualização de rodas quentes (vide abaixo).
3. **Tarefa de captura de dados dos detectores de roda quente.** Esta tarefa retira, da lista circular em memória, as mensagens de dados dos *hotboxes*, repassando-as diretamente para a tarefa de visualização de rodas quentes (vide abaixo).
4. **Tarefa de exibição de dados da sinalização ferroviária.** Esta tarefa retira, do arquivo circular em disco, as mensagens de dados de sinalização ferroviária e as exibe no terminal de vídeo do CCO dedicado à sinalização ferroviária.
5. **Tarefa de visualização de rodas quentes.** Esta tarefa recebe mensagens da tarefa de captura de dados dos detectores de rodas quentes e, eventualmente, da tarefa de captura de dados de sinalização ferroviária, exibindo-as no terminal de vídeo do CCO dedicado à visualização de rodas quentes
6. **Tarefa de leitura do teclado**. Esta tarefa dá tratamento aos comandos digitados pelo operador.

Tarefa de leitura dos CLPs. Esta tarefa simula a leitura de dados dos CLPs através da geração de dois tipos de mensagens, correspondentes a (1) dados de sinalização ferroviária provenientes das remotas de E/S e (2) dados provenientes dos detectores de rodas quentes. As primeiras devem ser geradas com periodicidade aleatória entre 100 e 2000 ms, e as segundas com periodicidade fixa de 500ms. Todas as mensagens devem ser depositadas em uma única 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()*.

As mensagens de dados de sinalização ferroviária correspondem 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):

NNNNNNN	NN	N	NNN	AAAAAAAA	N	HH:MM:SS:MSS
NSEQ	*TIPO*	*DIAG*	*REMOTA*	*ID*	*ESTADO*	*TIMESTAMP*

Campo	Tamanho	Tipo	Descrição
NSEQ	7	Inteiro	Número sequencial da mensagem [1...9999999]
TIPO	2	Inteiro	Sempre "00"
DIAG	1	Inteiro	Diagnóstico da remota
REMOTA	3	Inteiro	Identificação da remota de E/S ("000 a "999")
ID	8	Alfanumérico	Identificação do sensor
ESTADO	1	Inteiro	Estado do sensor (1 = FALSO, 2 = TRUE)
TIMESTAMP	12	Tempo	Hora, minuto, segundo e milissegundo

Exemplo: "`0665543;01;0;105;SIN-0023;1;12:01:33:500`"

Na mensagem acima, o valor "1" no campo DIAG indica falha de hardware da remota. Neste caso, os campos de ID e ESTADO devem ser preenchidos com os valores "XXXXXXXX" e "0" respectivamente.

Por sua vez, as mensagens dos detectores de roda quente devem corresponder a cadeias de caracteres com tamanho fixo de 34 caracteres, no formato a seguir, onde os campos individuais também são separados pelo delimitador ";" (ponto e vírgula):

NNNNNNN	NN	AAAAAAAA	N	HH:MM:SS:MSS
NSEQ	*TIPO*	*ID*	*ESTADO*	*TIMESTAMP*

Campo	Tamanho	Tipo	Descrição
NSEQ	7	Inteiro	Número sequencial da mensagem [1...9999999]
TIPO	2	Inteiro	Sempre "99"
ID	8	Alfanumérico	Identificador do detector de roda quente
ESTADO	1	Inteiro	Estado da detecção (0 = Normal, 1 = Roda quente)
TIMESTAMP	12	Tempo	Hora, minuto, segundo e milissegundo

Exemplo: `"2058744;00;HWD-0145;0;19:45:21:553"`

Na montagem das mensagens de dados de sinalização ferroviária e de detecção de roda quente, 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.

A tarefa de leitura dos CLPs 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 esteja bloqueada pelo evento de sinalização acima descrito.

Tarefa de captura de dados de sinalização ferroviária. Esta tarefa deve consumir mensagens de dados de sinalização ferroviária presentes na lista circular em memória e verificar o conteúdo do campo DIAG das mesmas. Se DIAG = '1' a mensagem deve ser repassada imediatamente para a tarefa de visualização de rodas quentes por meio de *pipes* ou *mailslots*. Se DIAG ≠ '1' a mensagem deve ser depositada em um arquivo circular em disco com 200 posições, e uma notificação deste depósito deve ser feita para a tarefa de exibição de dados de sinalização ferroviária por meio de sincronização apropriada. Caso não haja espaço no arquivo circular em disco, 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) 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 leitura dos CLPs.

Tarefa de captura de dados dos detectores de roda quente. Esta tarefa deve consumir mensagens de dados dos *hotboxes* presentes na lista circular em memória e repassá-la imediatamente para a tarefa de visualização de rodas quentes, 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 leitura dos CLPs.

Tarefa de exibição de dados de sinalização ferroviária. Esta tarefa deve ser notificada da presença de uma ou mais mensagens no arquivo circular em disco, retirar as mesmas deste arquivo e exibi-las em uma janela de console exclusiva, que simulará o respectivo terminal de visualização sinalização ferroviária no CCO. As mensagens devem ser exibidas no seguinte formato:

```
HH:MM:SS:MSS NSEQ: ######## REMOTA: ## SENSOR: ######## ESTADO: TextoTextoTextoTexto
```

A notação `"########"` representa o valor do respectivo item da mensagem de dados de sinalização ferroviária, e o campo ESTADO deve ser preenchido com um texto que reflita o estado do sensor, p. ex. "Desvio atuado", "Sinaleiro em PARE", etc. Ao menos 20 textos diferentes devem ser empregados nas mensagens da console. Pesquise a Internet para obter exemplos de sinalizações ferroviárias reais.

A tarefa de exibição de dados de sinalização ferroviária 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 dos CLPs.

Tarefa de visualização de rodas quentes. Esta tarefa recebe duas mensagens diferentes, provenientes respectivamente da tarefa de captura de dados de sinalização e da tarefa de captura de dados dos detectores de

roda quente, as quais devem ser exibidas em uma janela de console exclusiva, que simulará o terminal do CCO dedicado à visualização de rodas quentes. No primeiro caso, as mensagens que indicam falha nas remotas de E/S devem ser exibidas no seguinte formato:

```
HH:MM:SS:MSS NSEQ: ######## REMOTA: ## FALHA DE HARDWARE
```

No segundo caso, o formato de exibição varia conforme o campo ESTADO presente na mensagem:

```
HH:MM:SS NSEQ: ####### DETECTOR: ######## TEMP. DENTRO DA FAIXA
```

```
HH:MM:SS NSEQ: ####### DETECTOR: ######## RODA QUENTE DETECTADA
```

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

A tarefa de visualização de rodas quentes 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 dos CLPs.

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

"c"	Notifica à tarefa de leitura dos CLPs 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.
"d"	Idem, com relação à tarefa de captura de dados de sinalização ferroviária.
"h"	Idem, com relação à tarefa de captura de dados dos detectores de roda quente.
"s"	Idem, com relação à tarefa de exibição de dados de sinalização ferroviária.
"q"	Idem, com relação à tarefa de visualização de rodas quentes.
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.

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 – 15 pontos;**
- **Entrega apenas da etapa 2, fora do prazo indicado – 8 pontos;**
- **Entrega apenas da etapa 1 – não pontuado.**

O propósito da 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 feito às pressas, próximo 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/S em disco, 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 para cada uma delas;
- Implementação da lista circular em memória RAM, com o depósito de mensagens na mesma pela tarefa de leitura dos CLPs, e o consumo destas pelas tarefas de captura de dados de sinalização ferroviária e de dados dos detectores de rodas quentes, levando em conta aspectos de sincronização eventualmente necessários;
- Temporizações provisórias da tarefa de leitura dos CLPs por meio da função *Sleep()*. Estas temporizações serão futuramente substituídas por aquelas definitivas, na etapa 2 do trabalho;

- Testes da aplicação parcial. (**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 Fig. 5 apresenta a estrutura da aplicação correspondente à primeira etapa.

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 duas "consumidoras" que acessam a lista. Assim, no caso das "consumidoras", cada uma delas deverá manter e manipular apontadores individuais para a próxima mensagem a ser consumida. Outra opção é implementar a lista circular como uma lista encadeada *(linked list)*.

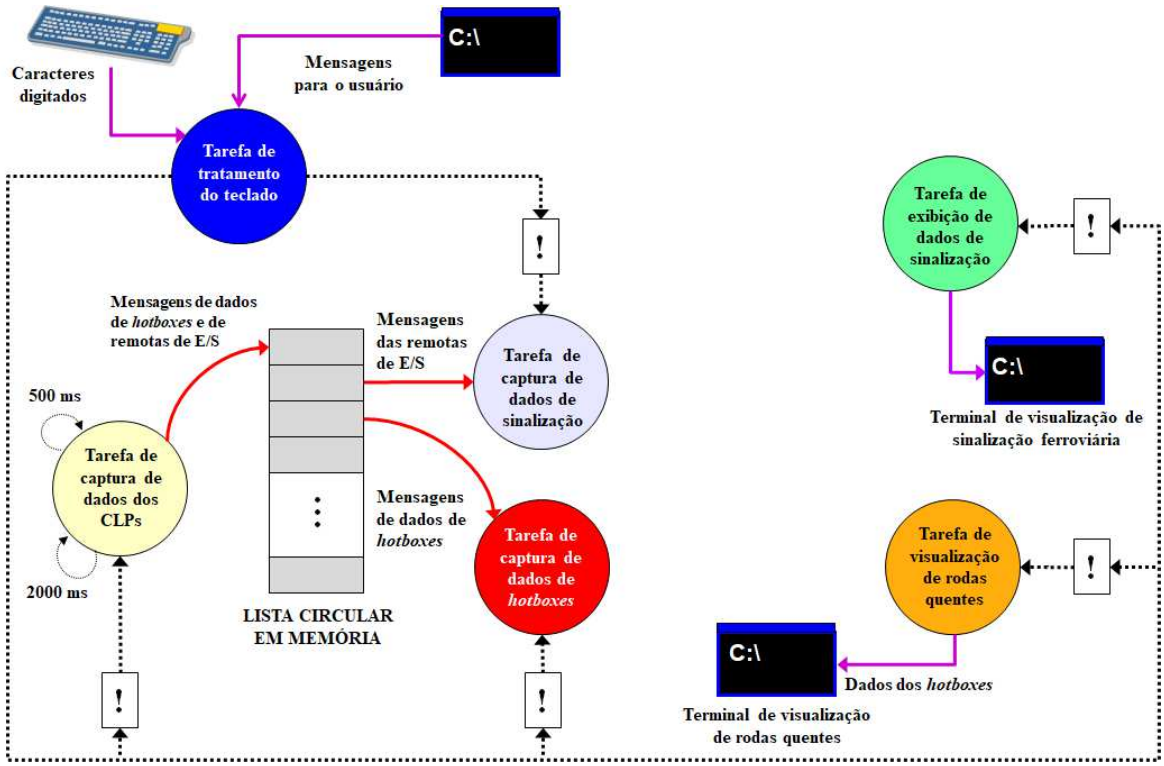


Figura 5: Diagrama de relacionamentos – Etapa 1. A legenda sobre a notação empregada encontra-se após a Fig. 6.

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 sinalização ferroviária e de visualização de rodas quentes deverão apenas emitir uma mensagem simples em suas respectivas janelas de console, indicando periodicamente 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 da tarefa de leitura dos CLPs, nas periodicidades indicadas na descrição das mesmas, sem o uso da função *Sleep()*;
- Criação do arquivo circular em disco, inserção/retirada no mesmo das mensagens de dados de sinalização ferroviária, e o sincronismo entre a tarefa de captura de dados de sinalização ferroviária e a tarefa de exibição destes dados;

- Implementação do mecanismo de IPC (*Inter-Process Communication*) entre as tarefas de captura de dados de sinalização ferroviária, de captura de dados dos *hotboxes*, e de visualização de rodas quentes;
- Testes finais da aplicação como um todo. Como no caso da Etapa 1, seja cuidadoso e minucioso em seus testes.

O diagrama de relacionamentos na Fig. 6 apresenta a estrutura da aplicação correspondente à segunda etapa.

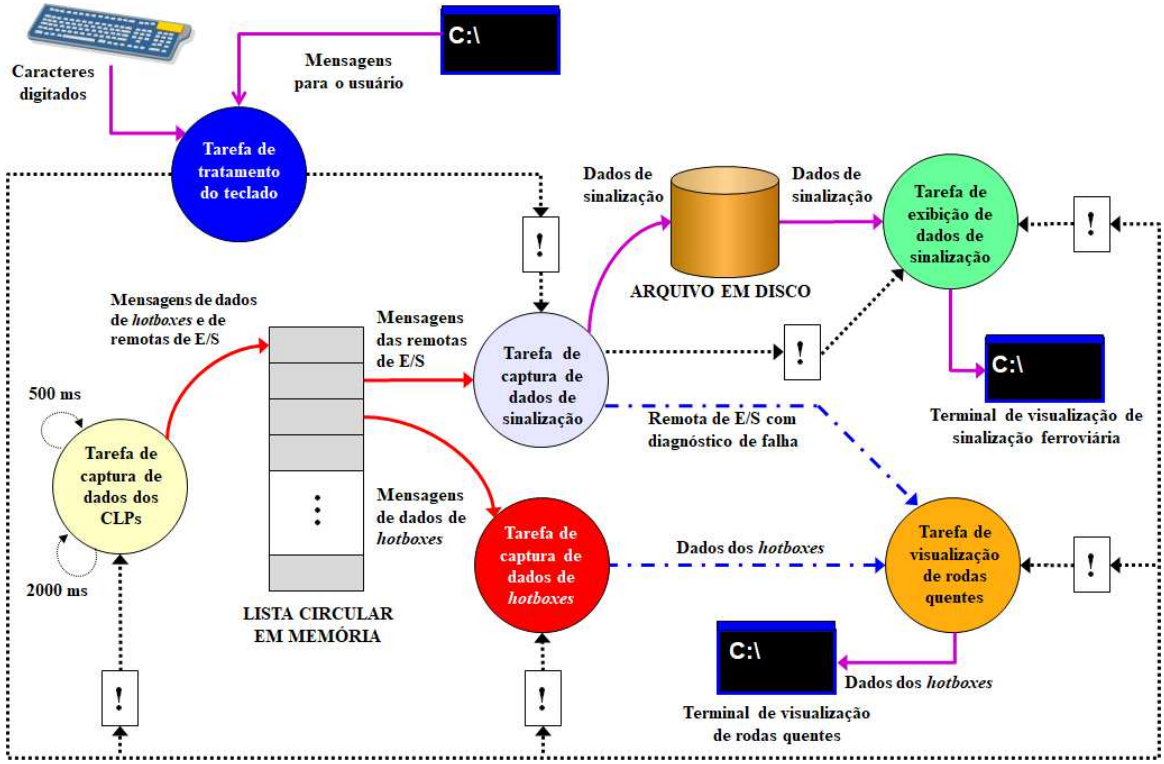
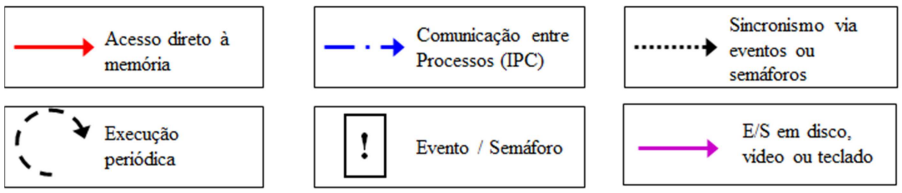


Figura 6: Diagrama de relacionamentos – Etapa 2

Legenda para as Figs. 5 e 6:



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 será reduzida para a dupla em sonjunto.
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.

5. **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, este será desconsiderado e o trabalho será considerado como **não-entregue**. O professor não enviará, aos alunos, avisos de problemas com os arquivos enviados.

6. Datas-limite de entrega:

   Etapa 1 - **23h59min** do dia **19/05/2024 (domingo)**

   Etapa 2 - **23h59min** do dia **09/06/2024 (domingo)**

   Não serão permitidos atrasos com relação à etapa 1. Quanto à etapa 2, trabalhos entregues até **29/06/2025** terão sua pontuação reduzida a 50%. Não serão aceitos trabalhos entregues após 29/06/2025.

   Atenção: submeta o trabalho exclusivamente via *Moodle*. Não serão aceitos trabalhos enviados por email, ou compartilhados em *sites* como *Dropbox*, *Google Drive*, etc., ou ainda via plataforma *Microsoft Teams*.

7. 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 exercitarem sua 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 uma pontuação 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 seja o mesmo no computador do professor. **IMPORTANTE:** Se você se esquecer deste passo, o programa não compilará no computador do professor e, assim, não poderá ser testado.
- 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 (01/06 e 22/06)	25
Entrega apenas da etapa 2, no prazo (22/06)	15
Entrega apenas da etapa 2, fora do prazo (até 29/06)	8
Entrega apenas da etapa 1	Não pontuado

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.