Universidade do Vale do Itajaí
Curso de Bacharelado em Engenharia de Computação
Disciplina: Sistemas em Tempo Real
Professor: Felipe Viel
Avaliação 03 – Threads, concorrência e paralelismo

Instruções

1. Esta avaliação deve ser feita individualmente ou em dupla.

2. Data de entrega: 29/10/2020 até as 07:59. Trabalhos em atraso implicarão em 1 ponto de desconto por dia e, após 7 dias, não mais será aceita a entrega.

3. Esta avaliação tem por objetivo consolidar o aprendizado sobre conceitos de threads, concorrência e paralelismo e seus impactos na execução de tarefas com deadline.

4. A implementação deverá ser desenvolvida utilizando Pthreads (C) e OpenMP.

5. O sistema deve ser entregue funcionando corretamente.

6. Deve ser disponibilizado os códigos da implementação em repositório do(s) aluno(s) (github, bitbucket, etc...), deve ser fornecido o link e o repositório dever ser público.

7. A avaliação pode ser feita em máquina virtual (fornecida) ou na própria máquina, utilizando uma distribuição Linux.

    a. O uso de máquina virtual permite variar a quantidade de processadores o que permite avaliar o impacto da arquitetura na execução de tarefas.

8. O relatório deve seguir o formato disponível no Blackboard e indicado pelo professor. Só deverá ser postado o relatório no Blackboard, no link: **Local para entrega da Avaliação 3 - M2**. Os códigos e vídeos deverão ser indicados por links.

9. Deverá ser gravado um vídeo explicando a implementação
    a. O vídeo deve ser postado no material em plataformas como Youtube, Dropbox ou Google Drive. O link para o vídeo e para o código no repositório devem ser disponibilizados no relatório.
    b. Deverá ser explicado a solução, códigos, análises, resultados e conclusões obtidas.

10. Para análise temporal, utilizar a biblioteca time.h.

11. Trabalhos caracterizados como cópia terão nota zero a todos os envolvidos.

Descrição do projeto a ser desenvolvido

Projeto 1

Problemática: uma Indústria de Alimentos de Santa Catarina chamada FoodSec S.A. possui a tarefa de escanear alimentos por meio de câmeras e verificar se os mesmos estão corretos. Os alimentos podem passar por uma das três esteiras disponíveis. As três esteiras são controladas por um por um único computador centralizado. Esse computador recebe dados de um sensor em cada uma das esteiras que captura a contagem de itens que são identificados como seguros. A contagem é exibida em um display perto das esteiras (todos os displays são controlados pela mesma função, é apenas uma replicação). Diante disso, a empresa precisa que vocês implementem, por meio de aplicação para distribuição Linux, uma solução que consiga realizar as contagens nas três esteiras e exiba o resultado total (contagem esteira 1 + contagem esteira 2 + contagem esteira 3) em tempo real. A empresa comprou o computador com processador com 4 núcleos, possui uma distribuição Linux e não aceita atualizar sistema

ou equipamento, mas aceita carregar novas aplicações. Além disso, os pesos dos itens que passão por cada uma das esteiras são armazenados em um único vetor de dados. A cada 1.500 unidades de produtos, das três esteiras, é necessário atualizar o peso total de itens processados. Sendo assim, a empresa aceita uma pausa na quantidade de itens sendo contados e pesados para realizar a pesagem total, mas ela necessita saber quanto tempo é necessário para isso.

Diante da problemática apresentada, vocês terão que implementar uma aplicação (em nível de MVP) que possa lidar com tal situação usando duas abordagens: Pthreads e OpenMP. Deverá ser feitos um relatório especificando qual a taxa de atualização (capacidade das esteiras em conseguir identificar um novo item por meio do sensor), tempo de processamento da contagem, tempo que consegue atualizar o display. Além disso, como a abordagem utiliza seção crítica, é necessário implementar o uso de mutex na biblioteca Pthread. Como forma de estudar possíveis soluções, avalie o mutex dessa biblioteca usando os protocolos Herança de Prioridade e Teto de Prioridade, os quais podem ser configurados para uso na biblioteca Pthread. Avalie a abordagem single thread e multithread, especifique temporalmente a solução com, por exemplo 1 ou 2 núcleos. Além disso, ambas as bibliotecas oferecem a implementação de Barreira (problema da barreira) e será dado pontuação adicional na prova para quem utilizar de maneira apropriada (+1).

Links úteis

http://maxim.int.ru/bookshelf/PthreadsProgram/htm/r_38.html
https://www.embedded.com/effective-use-of-pthreads-in-embedded-linux-designs-part-2-sharing-resources/
https://docs.oracle.com/cd/E36784_01/html/E36868/sync-88.html
https://man7.org/linux/man-pages/man7/pthreads.7.html
https://pubs.opengroup.org/onlinepubs/7908799/xsh/pthread.h.html
https://docs.oracle.com/cd/E19455-01/806-5257/attrib-16/index.html
https://computing.llnl.gov/tutorials/pthreads/
http://maxim.int.ru/bookshelf/PthreadsProgram/examples/mutex/
https://docs.microsoft.com/pt-br/cpp/parallel/openmp/reference/openmp-clauses?view=vs-2019
https://www.openmp.org/wp-content/uploads/OpenMP4.0.0.pdf
https://www.openmp.org//wp-content/uploads/OpenMP4.0_Intro_YonghongYan_SC13.pdf
http://www.eitas.com.br/tutorial/12/31
https://www.nersc.gov/assets/Uploads/SC16-Programming-Irregular-Applications-with-OpenMP.pdf

VM Linux (Link com email @edu.univali.br)
Login: sovm
Senha: 1234
Por padrão, a VM roda com 2 GB de memória e apenas um núcleo. Podem ser modificados para testes de desempenho.

Instalação OpenMP (Pthread é nativa na maioria das distribuições)

$ sudo apt-get install libomp-dev