**Universidade do Vale do Itajaí**
**Disciplina: Sistemas Operacionais**
**Professor: Felipe Viel**
**Avaliação 01 – Processos, Threads, Concorrência e Paralelismo**

## Instruções

1. Esta avaliação deve ser feita individualmente e em até trio.

2. Data de entrega: 17/09/2024 até as 19:00. Trabalhos em atraso implicarão em 2,0 ponto de desconto por dia e, após 5 dias, não mais será aceita a entrega.

3. Esta avaliação tem por objetivo consolidar o aprendizado sobre conceitos de IPC, threads, concorrência e paralelismo.

4. A implementação deverá ser desenvolvida utilizando a linguagem de programação de sua preferência (C, C++, Python, Java, C#, Javascript/Node.js, Rust, etc). Porém, a utilização e suporte a threads pela linguagem escolhida é de responsabilidade do(s) aluno(s), seja usado corretamente o conceito de threads. As bibliotecas usadas devem seguir a linha da Pthreads. Bibliotecas que também implementem e que permitam usar conceitos de paralelismo também podem ser usadas, mas o aluno também é responsável pelo seu uso e apresentação. O uso de bibliotecas de Thread Pool é permitido.

5. O sistema deve ser entregue funcionando corretamente. Sistemas não compilando e executando não serão aceitos. É de responsabilidade do(s) aluno(s) apresentar a execução 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. O relatório deve seguir o formato de artigo científico ou seguindo as normas da ABNT e as orientações para produção de trabalhos acadêmicos da Univali contendo:

    - Identificação do autor e do trabalho.
    - Enunciado dos projetos.
    - Explicação e contexto da aplicação para compreensão do problema tratado pela solução.
    - Resultados obtidos com as simulações.
    - Códigos importantes da implementação.
    - Resultados obtidos com a implementação (tabelas, gráficos e etc).
    - Análise e discussão sobre os resultados finais.

8. Deve ser disponibilizado os códigos da implementação juntamente com o relatório (salvo o caso da disponibilidade em repositório aberto do aluno, que deve ser fornecido o link). O repositório deve estar aberto do momento da entrega em diante, sendo que o professor não responsabiliza caso o projeto não esteja disponível para consulta no momento da correção, sendo do(s) aluno(s) essa responsabilidade de manter disponível. Cópias entre alunos implicará em nota zero para todos os envolvidos.

9. O trabalho deverá ser apresentado em data definida pelo professor. É de responsabilidade do(s) aluno(s) explicar os conceitos, comandos, bibliotecas usadas. É de responsabilidade do(s) aluno(s) fazer a solução funcionar e ela deverá ser baixada do local de entrega no momento da apresentação. Trabalhos não apresentados terão como nota máxima 5,0, além dos descontos aplicados no restante da avaliação da implementação.

## Descrição do projeto a ser desenvolvido

### Projeto

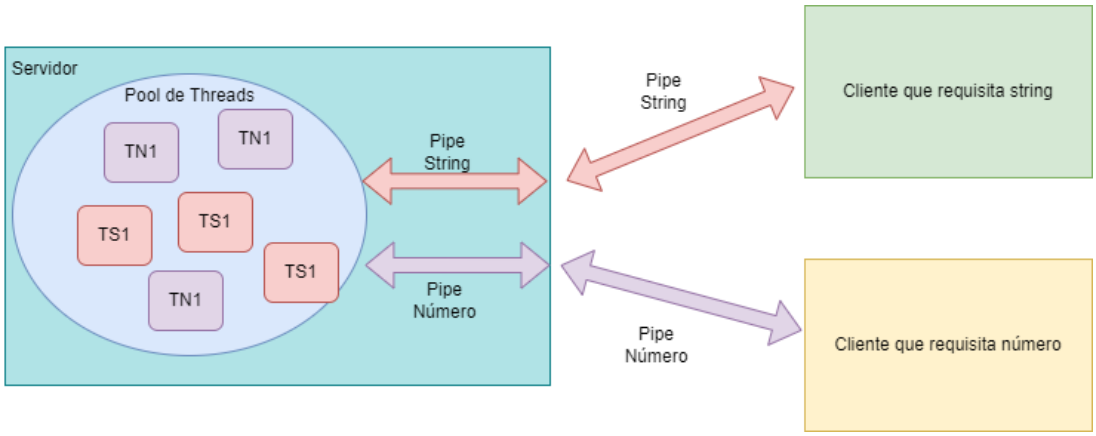
Um sistema de servidor deve sempre disponível para pode responder a requisições feitas por outros processos ou threads, podendo ser no mesmo computador ou em outros computadores. Uma abordagem útil para lidar com múltiplas requisições é utilizar Pool de Threads. Esa abordagem permite que seja diminuído a carga de trabalho em criar novas threads toda a vez que necessita atender uma requisição e torna o sistema mais “leve” comparado a uso de múltiplos processos.

Com isso, imagine que você está criando uma espécie de servidor local que funciona com pool de threads e que cada thread (worker) se comunica com processos que solicitam alguma informação. O processo “servidor” que possui os workers deve permitir enviar e receber solicitações via pipe. Os processos que gostaria de obter os serviços do servidor devem se conectar ao pipe correto. As threads da pool de threads podem ter serviços diferentes para simular um servidor como:

- Algumas threads respondem a solicitações de números e caso algum cliente queira obter isso deve se conectar ao pipe desse tipo de solicitação.
- Algumas threads respondem a solicitação de string (uma só) e caso algum cliente queira obter isso, deve se conectar ao pipe desse tipo de solicitação.

A quantidade de threads na pool é um critério que fica a sua escolha, mas lembre-se sobre dimensionar a quantidade diante do hardware que você tem disponível.

A figura abaixo ilustra o conceito do trabalho:




Na figura, dois processos clientes estão conectados ao processo servidor via pipe, porém que lida com os pipes são as threads na pool. As threads TN são as responsáveis por responder requisições de números, enquanto que as threads TS são as responsáveis por responder as requisições de string.

Pontuação extra em prova:

A fim de estimular a ampliação dos seus conhecimentos sobre paralelismo e concorrência em Sistemas Operacionais, será concedido de 0,5 à 1 ponto extra na nota da prova para os trabalhos para os que conseguirem demonstrar mais de 1 processo cliente se comunicando com o processo servidor ao mesmo tempo. Podem usar como artifícios laços de repetição para demostrar o paralelismo de tarefa no processo servidor. A pontuação (de 0,0 a 1,0) fica a critério do professor e será baseada na qualidade da entrega feita. Essa implementação não é obrigatória.