



Definição do Trabalho 1: Programação Paralela

Este trabalho visa explorar o uso de padrões para programação *multithread*. Cada grupo irá explorar (pelo menos) os modelos de programação *produtor/consumidor*, *fork-join* e *pool de threads* na solução do problema.

Descrição do problema: Sistema de acionamento automático em carros autônomos

Automóveis autônomos contam com uma grande quantidade de **sensores (produtores)**, que permitem reconhecer o ambiente bem como as condições internas do veículo. A quantidade de *threads* de sensores deve ser parametrizável por **N_SENsoRES**. Um dado sensorial deve ser um inteiro aleatório entre 0 e 1000, obtido em intervalos de 1s a 5s.

Os **atuadores**, que são responsáveis por controlar as ações do veículo, são acessados por meio de uma **tabela compartilhada**. Os **atuadores** e **níveis de atividade** dos atuadores são representados por inteiros. O acesso a um atuador é feito por meio de uma tabela do tipo **Map<Integer,Integer>** ou **Dict[int,int]**. O primeiro inteiro representa o atuador, enquanto o segundo representa o nível de atividade. A quantidade de atuadores deve ser parametrizável por **N_ATUADORES**, e todos inicializados com atividade 0;

Uma forma de transformar os **dados sensoriais** em comandos para os **atuadores** é passando as informações para uma **Central de Controle (consumidor)**.

Logo, para cada dado sensorial *ds* recebido, a central define um **atuador** (*ds % N_ATUADORES*) e **nível de atividade** (valor aleatório entre 0 e 100).

Para garantir bom desempenho e tempos de resposta, a Central de Controle opera sobre diversas **unidades de processamento (pool de threads)**, permitindo alterações paralelas no comportamento do conjunto de atuadores.

Uma unidade de processamento que deseja alterar o comportamento de um atuador deve alterar o inteiro correspondente ao nível de atividade **e mantê-lo no valor desejado por pelo menos 2s** (valor aleatório entre 2s e 3s). As unidades de processamento também são responsáveis por enviar ao painel do veículo que o atuador está sofrendo modificações no seu nível de atividade (imprimir na tela “Alterando: <atuador> com valor <nível de atividade>\n”). Esse valor deve ser mantido no painel por 1s (depois de imprimir a mensagem, nada deve ser impresso por 1s).

Para cada tarefa recebida pela unidade de processamento, são duas subtarefas (**divisão**), que podem ser executadas de forma independente (**fork**): **a mudança do nível de atividade do atuador** e **o envio de mudança ao painel**. Cada subtarefa apresenta um resultado: se ocorreu falha (chances aleatórias, 20% de chance de falha). No término da execução das duas atividades (**junção**), caso alguma delas sofreu uma falha (**combinação**

Ilustração de Apoio

Entrega

O trabalho consiste em:

1. Implementar um programa que simule a execução do serviço automotivo, seguindo os requisitos descritos acima;
2. Breve relatório que indique as principais decisões e estratégias de implementação utilizadas, instruções sobre como compilar e executar o código produzido, além de exemplos de saídas de execução com diferentes parametrizações. Ao final, discuta quais conclusões você observa ao variar parâmetros do sistema.

O trabalho pode ser realizado em **grupos de até 3 participantes**. O trabalho será apresentado em sala de aula.

O código-fonte e relatório devem ser enviados pelo Moodle para análise e avaliação.

Os nomes dos participantes do grupo devem constar no relatório entregue no Moodle. Participantes com nomes não referenciados não serão considerados como membros do grupo.