Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Instituto de Matemática e Estatística

Trabalho 1 – Sistemas Operacionais 1

Instruções

O trabalho poderá ser feito em grupos de até **3 integrantes, totalizando 5 grupos**. A entrega (05 de outubro de 2017) será pelo **Moodle (AVA)**, com um arquivo zipado contendo:

1. **Código fonte (baseado no código fonte que será disponibilizado para a turma):** O AMBIENTE DE COMPILAÇÃO E EXECUÇÃO DEVERÁ SER UNIX. Trabalhos feitos para Windows não receberão suporte do professor. É preciso que seja criado um Makefile para compilação do projeto. Trabalhos que apresentem qualquer erro de compilação ou execução serão imediatamente descartados da avaliação e receberão nota ZERO. Trabalhos copiados também receberão nota ZERO.
2. **PPT com a apresentação da solução:** como o algoritmo foi dividido em tarefas, incluindo tempos de execução da solução com 1 processo e com N trabalhadores.

NA DATA DE ENTREGA DO TRABALHO, CADA GRUPO DEVERÁ APRESENTAR SUA SOLUÇÃO AO PROFESSOR. APENAS 1 INTEGRANTE DO GRUPO (QUE SERÁ SORTEADO ALEATORIAMENTE) FICARÁ ENCARREGADO PELA APRESENTAÇÃO. OS DEMAIS DEVERÃO AGUARDAR FORA DE SALA.

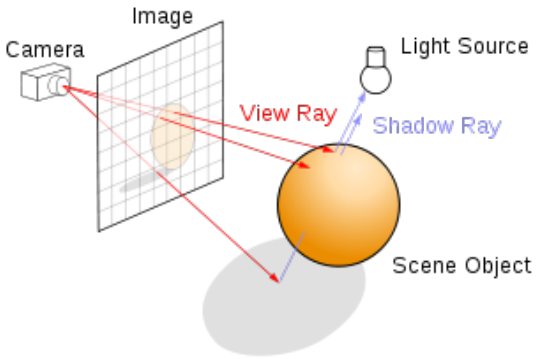
Motivação & Objetivos

Os sistemas operacionais têm como principal meta abstrair a complexidade do hardware e gerenciar de forma eficiente os recursos da máquina que o hospeda, tais como memória, discos e processador. Este último, por exemplo, executa um Processo (ou Thread) por vez, dentre um conjunto de Processos e Threads que podem ser escalonados pelo SO. Com o advento de arquiteturas "multi-core", cada núcleo pode executar um Processo ou Thread.

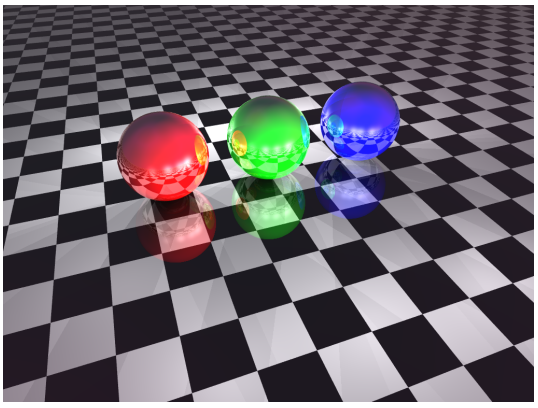
O gerenciamento de processos e threads é fundamental para o uso eficiente dos processadores por diferentes programas. Sendo assim, é crucial que os alunos entendam e pratiquem os conceitos de criação, execução e administração de processos e threads.

O Algoritmo de Traçado de Raios

Este trabalho consiste em dividir um programa sequencial em tarefas que serão executadas concorrentemente. O programa escolhido é o de renderização 3-D por Traçado de Raios (Ray-Tracing), que é um algoritmo produz uma imagem a partir de uma cena 3-D, conforme exemplifica a Figura 1 abaixo:



Configuração da cena.



Renderização.

Figura 1: Algoritmo de Sobel.

Este algoritmo pertence à classe dos algoritmos de iluminação global, muito usados em filmes de animação 3-D.

Para cada pixel da imagem original, o algoritmo simula o trajeto de todos os raios de luz a fim de compor a cor de cada pixel.

O Trabalho

O código fonte do algoritmo será disponibilizado para a turma e deverá ser usado para construção de 4 soluções diferentes, sendo a última ponto extra:

1. **Execução do algoritmo usando processos**: 1 processo pai será encarregado pela leitura da cena 3-D e pela gravação da imagem em disco. Os demais

processos filhos (trabalhadores) ficarão encarregados pelo processamento da cena 3-D e dos raios. Para isso, a imagem terá que ser dividida igualmente entre os processos, os quais processarão suas partes, escrevendo a imagem nova em uma região de memória compartilhada.

2. **Execução do algoritmo usando threads**: 1 thread será encarregado pela leitura da cena e pela gravação da imagem em disco. Os demais threads (trabalhadores) ficarão encarregados pelo processamento da cena e dos raios. Para isso, a imagem terá que ser dividida igualmente entre os threads, os quais processarão suas partes, escrevendo a imagem nova em uma região de memória compartilhada.

3. **Execução do algoritmo usando OpenMP**: 1 thread será encarregado pela leitura da cena e pela gravação da imagem em disco. Os demais threads (trabalhadores) ficarão encarregados pelo processamento da cena e dos raios. Para isso, a imagem terá que ser dividida igualmente entre os threads, os quais processarão suas partes, escrevendo a imagem nova em uma região de memória compartilhada.

4. **(1.0 EXTRA) Execução do algoritmo usando Pipe**: 1 processo será encarregado pela leitura da cena e enviará (via Pipe) cada raio aos N processos trabalhadores. Ao término do processamento, os processos copiarão (via Pipe) cada pedaço da imagem processada a um único processo que será encarregado de montar e gravar a imagem final em disco.

O objetivo destas diferentes implementações é para que o aluno entenda as diferenças de implementação de sistemas multi-processados usando processos vs. threads vs. pipe.