Trabalho de Sistemas Operacionais
Data de Entrega: Até o dia 19 de Abril de 2019.

O trabalho pode ser feito em qualquer linguagem/plataforma de programação. O trabalho é INDIVIDUAL.
O que deve ser entregue:
- Descrição do Ambiente de Desenvolvimento. Por exemplo: Máquina Windows 7 32bits, Java 1.7 e Eclipe Ganimedes.
- Instruções para compilação/execução do programa.
- Código fonte do programa.
Após a entrega, vamos marcar um dia (25 ou 26 de abril) para analisar individualmente cada trabalho na presença do aluno.

OBSERVAÇÃO: O trabalho envolve listas encadeadas. Se for feito em Java ou C#, não pode usar nenhuma das estruturas de dados já fornecidas pelas bibliotecas que acompanham a linguagem. Por exemplo, em Java, não se pode usar ArrayLists, Vectors, LinkedLists, etc. A lista encadeada deve ser implementada do zero. Da mesma forma, o algoritmo de ordenação.

Título: Ordenação paralela de uma lista encadeada.

Resumo:
Neste trabalho, o aluno terá que ler números inteiros de vários arquivos, adicioná-los em uma lista encadeada na memória, ordená-la e por último escrever a lista em um arquivo. O objetivo é usar paralelismo sempre que possível para diminuir o tempo de execução.

Entrada do programa:
A entrada fornecida ao programa serão 10 arquivos binários contendo inteiros. Por arquivo binário, devemos entender que cada 4 bytes desse arquivo correspondem a um inteiro. Não serão arquivos de texto. Para entender a natureza de um arquivo binário em C, leia o seguinte link http://www.inf.pucrs.br/~pinho/LaproI/Arquivos/ArquivosBinarios.htm . Os 10 arquivos terão todos a mesma quantidade de inteiros. Serão nomeados da seguinte forma: arquivo0.bin, arquivo1.bin, ... , arquivo9.bin. Eles devem estar no mesmo diretório do programa.

Primeira Etapa - Leitura dos Arquivos:
O programa deve criar 10 threads, um para cada arquivo. Cada thread deve ler os inteiros do arquivo e adicioná-los na lista encadeada. A lista é a mesma para todos os threads. Os threads irão concorrer pelo acesso à lista, isso deve ser controlado.

Segunda Etapa - Ordenação:
O programa deve descobrir quantos núcleos/cores tem na máquina e criar a mesma quantidade de threads. Ele então fazer a ordenação da lista encadeada em paralelo. Pode ser usado qualquer algoritmo de ordenação. Lembrem-se que nem sempre o melhor algoritmo serial será o melhor paralelo. Ao final dessa etapa, a lista encadeada deve estar ordenada

na memória.

Terceira Etapa - Análise de Desempenho da Ordenação
O programa deve começar utilizando apenas uma thread e ordenar toda a lista. A seguir, executar novamente o problema agora com duas threads. Repita essa operação, aumentando de 2 em 2 threads, a quantidade de threads de ordenação.

Saída do programa:
Com a lista ordenada, o programa deve escrevê-la em um arquivo chamado binário chamado output.bin. Em teoria, o tamanho de output.bin deve ser o somatório do tamanhos dos arquivos de entrada.

Obs. Para as etapas 2 e 3, em todas as execuções, seu programa deve calcular o tempo total de ordenação. Desconsidere os tempos de leitura dos arquivos na fase de entrada do programa, e também os tempos de escrita no arquivo na fase de saída do programa. Seu programa deve calcular apenas o tempo de ordenação, considerando as diferentes quantidades de threads a serem utilizadas.