Especificação do Projeto Prático da disciplina TT304 – Sistemas Operacionais

Prof. André Leon S. Gradvohl, Dr.
gradvohl@ft.unicamp.br

Introdução

O projeto prático da disciplina TT304 – Sistemas Operacionais é um dos seus componentes de avaliação. O principal objetivo desse projeto é a consolidação dos conhecimentos sobre programação com múltiplos *threads*. Para tanto, o projeto propõe o desenvolvimento de um programa que realize uma sequência de operações com matrizes de números inteiros, lidos a partir de arquivos de entradas, e posterior gravação dos resultados em arquivos de saída.

O programa deve ser desenvolvido, obrigatoriamente na Linguagem C (pura)[1] e no ambiente Linux. A definição do problema a ser resolvido, bem como as demais informações sobre o projeto estão nas seções a seguir.

Definição do problema a ser resolvido

O problema a ser resolvido é o seguinte. Dadas três matrizes de entrada, $A_{n\times n}, B_{n\times n}$ e $C_{n\times n}$, o programa deverá calcular inicialmente a matriz $D_{n\times n}$, tal que $D_{n\times n} = (A_{n\times n} + B_{n\times n})$ e gravá-la em arquivo. Em seguida, o programa deverá calcular a matriz $E_{n\times n}$tal que $E_{n\times n} = (C_{n\times n} \times D_{n\times n})$. Por último, a matriz $E_{n\times n}$ deve ser gravada em arquivo e reduzida por soma, isto é, todos os seus componentes devem ser somados, resultando em um único valor final.

O fluxo completo da execução do programa deve obedecer ao que está ilustrado na Figura 1 a seguir.

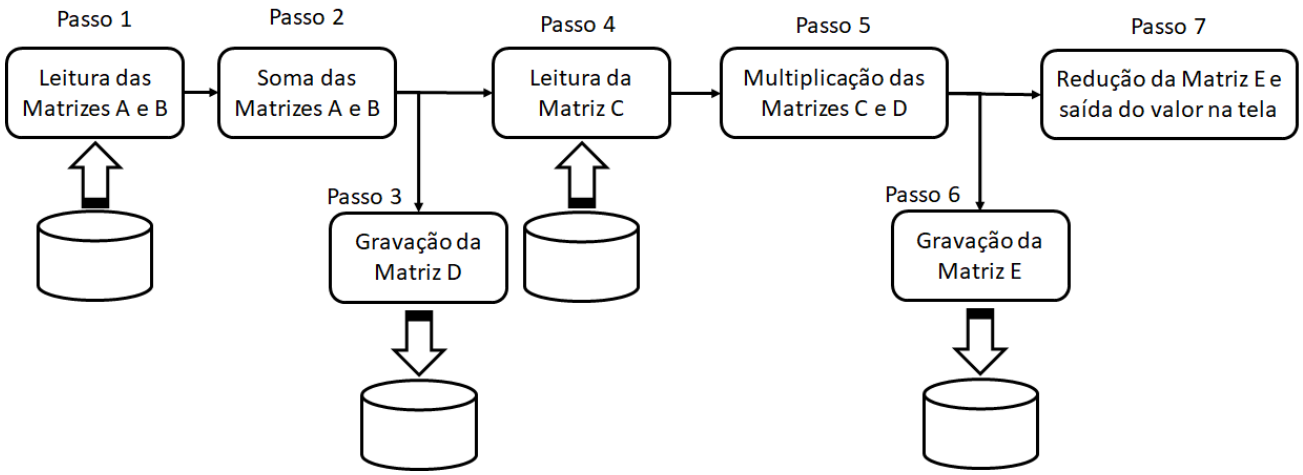


Figura 1 Passos para a execução do programa

Observe que alguns passos ou tarefas podem executadas simultaneamente por mais de um *thread*. Assim, vamos definir três tipos de *threads*: de processamento (t_p), de leitura (t_l) e de escrita (t_e).

Os *thread*s de processamento (t_p) e de leitura (t_l) podem ser executados simultaneamente com qualquer outro tipo de *thread*. O *thread* de escrita (t_e) só pode ser executado simultaneamente com outro thread de escrita se, e somente se, cada thread de escrita estiver gravando dados em arquivos diferentes.

[1] A linguagem C pura não deve ser confundida com a linguagem C++. Portanto, caso qualquer comando ou função da linguagem C++ seja usado, isso acarretará nota zero neste componente de avaliação.

A Tabela 1 a seguir indica os passos e tarefas que devem ser realizadas simultaneamente (com múltiplos *threads*), além do tipo e quantidade de *threads*. Usaremos o símbolo // para indicar que os passos ou as tarefas devem ocorrer simultaneamente.

Tabela 1 Passos, Tarefas, Tipos e Quantidades de Threads para a solução do problema proposto.

Passo	Tarefas	Tipos e Quantidades de *threads*
Passo 1	Leitura da Matriz A // Leitura da Matriz B	2 *threads* do tipo t_l
Passo 2	Soma das Matrizes A + B = D	T *threads* do tipo t_p
Passo 3 // Passo 4	Gravação da Matriz D // Leitura da Matriz C	1 *thread* do tipo t_e e 1 *thread* do tipo t_l
Passo 5	Multiplicação das Matrizes D x C = E	T *threads* do tipo t_p
Passo 6 // Passo 7	Gravação da Matriz E // Redução da Matriz E	1 *thread* do tipo t_e e T *threads* do tipo t_p

Entradas e saídas do programa

O programa deverá receber todos os dados necessários para seu processamento no início da sua execução. Em outras palavras, nenhuma solicitação de dados deve ser feita ao usuário durante a execução do programa e todos os dados devem ser informados na linha de comando para instanciar o programa.

Os arquivos com as matrizes A, B e C devem estar em dispositivos de armazenamento, no formato texto em linha/coluna, conforme o exemplo a seguir para uma matriz 3 x 3.

$$\begin{matrix} 1 & 5 & 6 \\ 3 & 2 & 4 \\ 9 & 7 & 8 \end{matrix}$$

Os arquivos com as matrizes D e E devem ser gravados no mesmo formato.

Ao final, o programa deve mostrar na tela o valor reduzido da matriz E, além dos tempos gastos com cada uma das operações das matrizes (soma, multiplicação e redução), além do tempo total do processamento.

Exemplo de linha de comando para instanciar o programa

A linha de comando para o programa de cada laboratório deve ter a seguinte sintaxe:

```
./programa T n arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
```

Onde:

- `./programa` é o nome do programa que resolverá o problema.
- `T` é o número de *thread*s de processamento (t_p).
- `n` é o número de linhas e colunas das matrizes.
- `arqA.dat` é o nome do arquivo que contém a matriz A.
- `arqB.dat` é o nome do arquivo que contém a matriz B.
- `arqC.dat` é o nome do arquivo que contém a matriz C.
- `arqD.dat` é o nome do arquivo que contém a matriz resultante da soma das matrizes A e B.
- `arqE.dat` é o nome do arquivo que contém a matriz resultante da multiplicação das matrizes D e C.

Observe um exemplo a seguir.

```
./prog 4 100 arqA.dat arqB.dat arqC.dat arqD.dat arqE.dat
```

Nesse exemplo, o programa `prog` foi executado com 4 *threads* de processamento e trabalhou com matrizes 100×100.

A saída esperada do programa é a seguinte:

```
Redução: 1232

Tempo soma: 0.003 segundos.

Tempo multiplicação: 0.009 segundos.

Tempo redução: 0.013 segundos.

Tempo total: 0.039 segundos.
```

Relatório do projeto

Além da implementação do programa, o projeto precisará produzir um relatório dos experimentos com os tempos de processamento do programa para as seguintes situações:

- Número de *threads* de processamento (T): T = 1, T = 2 e T = 4 *threads*.
- Tamanho das matrizes ($n \times n$): $n = 100$ e $n = 1000$

São, portanto, seis situações a serem analisadas, gerando resultados para seis experimentos. Importante lembrar que, na análise das situações com um *thread* (T = 1), deve existir apenas o *thread* principal, isto é, não devem ser criados outros *threads*, nem mesmo *threads* para leitura ou escrita.

Para cada situação analisada, deve ser calculado exclusivamente o tempo de total de processamento, isto é, o tempo gasto nos cálculos das operações com as matrizes (soma, multiplicação e redução), incluindo os tempos gastos para a criação dos *threads* de processamento. Esses tempos devem ser comparados entre as diversas situações para obter as conclusões dos experimentos.

Para a medição do tempo de processamento, sugere-se a utilização da primitiva `clock`, conforme o exemplo a seguir.

```
#include <stdio.h>
#include <time.h>
void fun(){
    printf("A fun() iniciou. \n");
    printf("Pressione enter para parar a função. \n");
    while(1)
     if (getchar()) break;
    printf("A função parou \n");
}
int main(){
   clock_t inicio, fim;
   double tempoTotal;
   inicio = clock();
   fun();
   fim = clock() - inicio;
   tempoTotal = ((double) fim)/CLOCKS_PER_SEC;
   printf("A função fun() gastou %f segundos. \n", tempoTotal);
   return 0;
}
```

Requisitos importantes do programa desenvolvido

Há alguns requisitos fundamentais que devem ser respeitados no programa desenvolvido. Esses requisitos estão a seguir.

O primeiro requisito é que todas as matrizes precisam ser alocadas dinamicamente (com o comando `malloc` ou equivalente). Havendo qualquer tipo de alocação estática de tipos compostos (matrizes) no programa, isso implicará em nota zero no respectivo laboratório.

O segundo requisito é que cada uma das matrizes necessárias para o programa precisa ser alocada em uma única etapa, isto é, em uma única chamada do comando `malloc` ou equivalente para cada matriz. Veja uma discussão sobre a alocação em única etapa no site https://gradvohl.github.io/alocaMatrizes. Havendo qualquer tipo de alocação em duas etapas no programa (por exemplo, alocar primeiro as linhas da matriz e depois as colunas), isso implicará em nota zero no respectivo laboratório.

O terceiro requisito é que o programa deve compilar sem erros. Para tanto, as instruções para a compilação devem estar muito bem definidas. Recomenda-se que seja criado um `makefile` ou um *script* para a compilação. Se o programa não compilar, ou aparecerem erros de compilação, isso implicará em nota zero no respectivo laboratório.

O quarto requisito é que os dados também devem estar armazenados em arquivos texto, conforme estabelecido na seção "Entradas e Saídas do Programa". Se os arquivos de saída do programa não respeitarem esse requisito, isso implicará no desconto de 5 pontos da nota do projeto.

Por fim, o quinto requisito é utilizar exatamente a linha de comando indicada no exemplo informado na seção anterior. Se o programa não funcionar com a linha de comando indicada, isso implicará no desconto de 5 pontos da nota do projeto.

Os produtos do projeto e a entrega

Os produtos que devem ser entregues como resultado do projeto são os seguintes:

i. O código fonte do programa, na linguagem C, completo, documentado e pronto para ser compilado em sistemas Linux.
ii. Um relatório contendo a descrição do problema; as instruções para compilar o programa; os gráficos com os tempos de execução dos experimentos; e as conclusões a respeito dos resultados obtidos.
iii. Um vídeo mostrando o código fonte do programa, a compilação do programa, trechos dos arquivos de entrada e as execuções do programa para os seis experimentos.

Todo o código fonte documentado e o relatório devem estar disponíveis em um repositório Git. No Ambiente Virtual de Aprendizagem da Disciplina (Moodle), deve ser publicado apenas o relatório no formato *Portable Document Format* (PDF). Neste relatório deve constar o endereço do repositório Git e do *site* onde o vídeo está publicado.

Para o item (i), a sugestão é a utilização de um `makefile` que facilite a compilação em sistemas Linux.

Para o item (ii), o relatório deve estar no formato PDF. Outros formatos não serão aceitos e, se forem enviados, terão nota zero automaticamente.

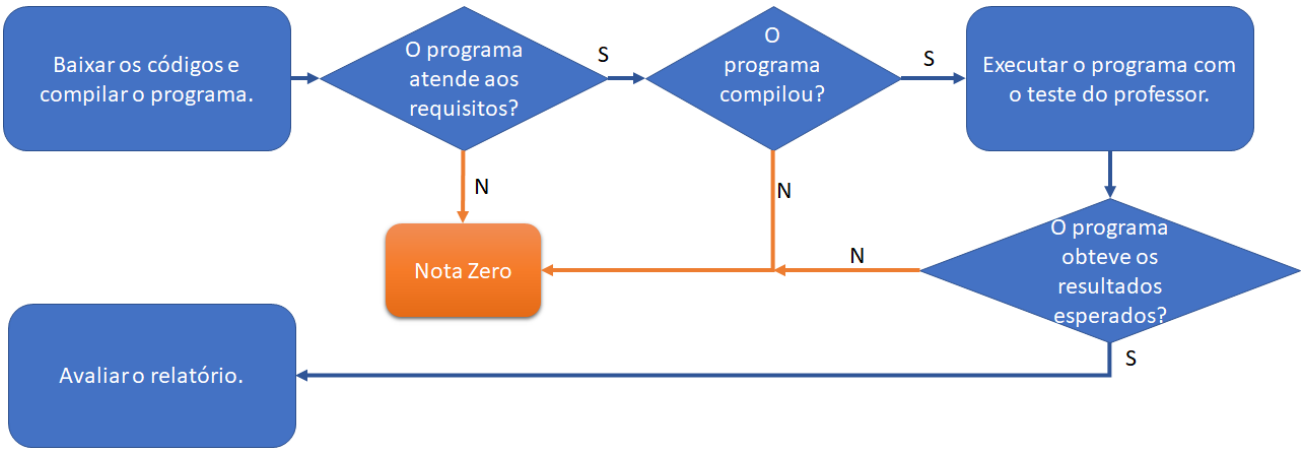
Para o item (iii), sugere-se o uso dos programas para capturar a tela do computador e criar vídeos. O vídeo deve estar publicado em um *site* a que o professor tenha acesso, independentemente de estar acessando a partir da rede da Unicamp ou não. A ausência do vídeo ou a impossibilidade de acesso acarretarão nota zero neste projeto. O vídeo poderá ser removido após a publicação das notas.

A falta de qualquer um dos itens acarretará nota zero nesse componente de avaliação.

Na data indicada no plano de ensino da disciplina, até as 23h55, o relatório deverá ser publicado no Moodle e todos os componentes do grupo devem concordar com o envio. Envios sinalizados como "rascunho" ou não confirmados por todos os membros do grupo serão considerados não enviados e terão nota zero.

Avaliação dos produtos do projeto

A avaliação do programa desenvolvido seguirá o seguinte fluxograma:



Os produtos do projeto, serão avaliados conforme os seguintes critérios.

- A qualidade dos códigos produzidos, inclusive o grau de modularidade dos códigos, e a inovação no que se refere ao uso dos núcleos de processamento disponíveis.
- A qualidade da documentação do código.
- A qualidade do texto do relatório, incluindo a correta utilização da norma gramatical da língua portuguesa.
- A estética e a qualidade do vídeo apresentado, bem como seu conteúdo e se cumpre com o que foi solicitado no início dessa seção.

A Tabela 2 a seguir informa alguns descontos na pontuação, caso determinadas situações ocorram.

Tabela 2 Situações em que podem ocorrer descontos na pontuação.

Situação	**Desconto**
Compilação com advertências (*warnings*)	-2 pontos
Código pouco ou mal documentado	-2 pontos
Código confuso ou pouco modularizado	-2 pontos
Vídeo incompleto, difícil de entender ou com baixa qualidade (de áudio ou de imagem)	-2 pontos
Relatório confuso ou com muitos erros de ortografia e gramática	-3 pontos
Código com alocação dinâmica em duas ou mais etapas	-5 pontos
Arquivos de saída fora do padrão	-5 pontos
Programa desenvolvido em outra linguagem de programação que não seja C pura.	-10 pontos

A data de entrega está estabelecida no Plano de Desenvolvimento da Disciplina.