```
#include <stdio.h>
#include <pthread.h>

//Definindo constantes para poder alterar aqui.
#define TAM_VET 100000
#define QTD_THREADS 4

int id[QTD_THREADS];
int vetor1[TAM_VET];
int vetor2[TAM_VET];
pthread_mutex_t mutex;

//Essa estrutura armazena os índices "inicio" e "fim" para cada thread processar um intervalo do vetor.
typedef struct {
   int inicio;
   int fim;
} Total;

//Calcula o produto de dois vetores.
int ProdutoVetorial(int inicio, int fim) {
   int soma = 0;
   for (int i = inicio; i < fim; i++) {
      soma += vetor1[i] * vetor2[i];
   }
   return soma;
}
/*Cada thread calcula sua soma parcial chamando ProdutoVetorial e,
```

```
em seguida, adiciona essa soma ao resultado protegido por mutex.
*/
void * funcao(void * args){
   Total *total = (Total *)args;
   int soma_parcial = ProdutoVetorial(total->inicio, total->fim);
   int resultado = 0;

   pthread_mutex_lock(&mutex);
   resultado += soma_parcial;  // Atualiza o resultado global
   pthread_mutex_unlock(&mutex);

   return NULL;
}

int main(){
   int resultado;

pthread_mutex_init(&mutex, NULL);

   // Inicializando os vetores
// Exemplo: todos os elementos do vetor1 são 3
// Exemplo: todos os elementos do vetor2 são 4
   for (int i = 0; i < TAM_VET; i++) {
      vetor1[i] = 3;
      vetor2[i] = 4;
   }

   pthread_t threads[QTD_THREADS];
```

```
    Total totais[QTD_THREADS];
    int intervalo = TAM_VET / QTD_THREADS;

    // Criando as threads
    for (int i = 0; i < QTD_THREADS; i++) {
        totais[i].inicio = i * intervalo;
        totais[i].fim = (i + 1) * intervalo;
        pthread_create(&threads[i], NULL, funcao, &totais[i]);
    }

    // Aguardando a conclusão das threads
    for (int i = 0; i < QTD_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }

    printf("Resultado do Produto Vetorial: %d\n", &resultado);

    pthread_mutex_destroy(&mutex);
    return 0;
}
```