Escreva programas em Pthreads para:

1. Encontrar o tempo médio necessário para o sistema criar e terminar um fio de execução.
   - O número de fios criados afeta o tempo médio? Se sim, de que maneira?

2. Implementar a regra trapezoidal.
   - Use uma variável partilhada para a soma de todos os cálculos parcelares de cada um dos fios.
   - Use ocupado-espera (busy-wait), mutexes e semáforos para forçar a exclusão mútua na seção crítica.
   - Quais as vantagens e desvantagens de cada uma das abordagens acima?

A regra trapezoidal

Se y = f (x) for uma função razoavelmente "simpática", e a <b números reais, podemos estimar a área entre o gráfico de f (x), as linhas verticais x = a e x = b, e o eixo-x dividindo o intervalo [a, b] em n sub-intervalos e aproximar a área sobre cada sub-intervalo pela área de um trapézio.
Se todos os sub-intervalo forem iguais e: ***h = (b-a) / n, xi = a + ih, i = 0,1, ..., n*** então a aproximação será: ***h [f(x0)/x2 + f(x1) + f(x2) + ⋯ + f(xn-1) + f(xn)/2].***

Objetivo: Calcular uma integral definida usando a regra trapezoidal e Pthreads.

Entrada: a, b, n
Saída: Estimativa da integral de a para b de f(x) e n trapézios
Uso: ./pth_trap <número de fios> <método>
Métodos:
1 : mutex / 2: semáforo / 3: ocupado-espera
Nota:
1. f (x) é fixa (ver abaixo)
2. n divisível pelo número de fios

Algoritmo:

```
h = (b-a)/n;
approx = (f(a) + f(b))/2.0;
for(i=1;i<=n-1;i++) {
x_i = a + i*h;
approx += f(x_i);
}
approx = h*approx;
```

f(x):

```
double f(double x) {
   double return_val;

   return_val = x*x;
   return return_val; }
```