INE5412 – Sistemas Operacionais I
Profs. Giovani Gracioli e Márcio Bastos Castro
2022/02

Projeto de um Sistema Operacional em Nível de Usuário
Trabalho 4: Métodos Join, Suspend e Resume

1. **Descrição**

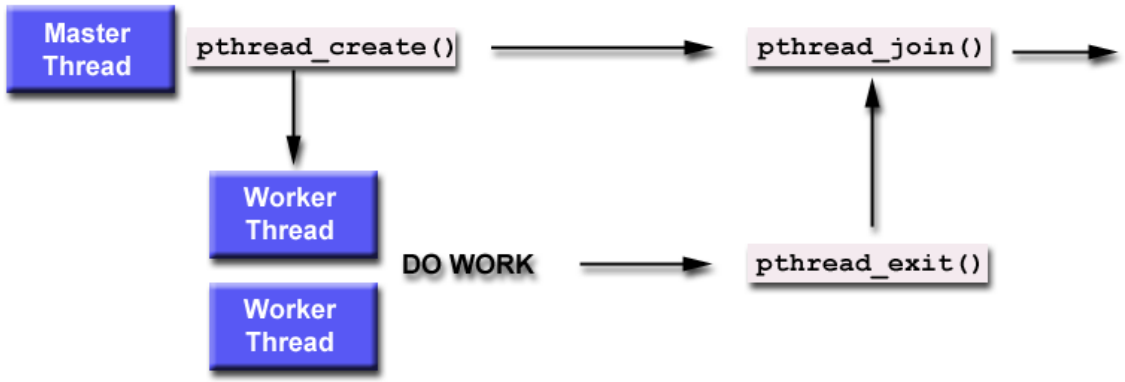
Este trabalho tem os seguintes objetivos:

1. Criação do método join da Thread. “Fazer join” de uma Thread significa suspender a Thread que chamou join até que a Thread “alvo” termine:

   Ex:

   ```
   Thread t1(&f1);
   Thread t2(&f2);
    ...
    //função f1 executada pela Thread t1
   void f1() {
           t2->join(); //suspende t1 até que t2 finalize
   ```

   A ilustração abaixo demonstra a relação de join com o ciclo de processamento tradicional usado em programas com threads, utilizando a biblioteca pthread como base.



   }

   Para mais detalhes sobre a funcionalidade de join de Thread, segue os links abaixo:

   - https://pubs.opengroup.org/onlinepubs/009695399/functions/pthread_join.html
   - https://en.cppreference.com/w/cpp/thread/thread/join
   - https://hpc-tutorials.llnl.gov/posix/joining_and_detaching/

2. Implementar a tarefa de suspender uma Thread já que a mesma não existe no código atual e é necessária pelo join();

3. Implementar a funcionalidade de resumir (resume) uma Thread, para que quando a thread "alvo" acabar seja possível colocar a Thread suspensa (que efetuou o join) de volta para a fila de prontos;

4. Utilizar o parâmetro exit_code em Thread::thread_exit().

Assim, os seguintes métodos devem ser adicionados e implementados na classe Thread:

- **int join();**

  Este método deve suspender a thread em execução até que a thread "alvo" finalize. O inteiro retornado por join() é o argumento recebido por thread_exit(), ou seja, exit_code (novo atributo **_exit_code** necessário na classe Thread – pergunta: quando o atributo _exit_code deve ser inicializado?). Como tratar a suspensão e o resumo de uma Thread?

- **void suspend();**

  Suspende a Thread até que resume() seja chamado. Como tratar as Threads suspensas dentro do SO? Deve-se usar uma nova fila? Deve -se fazer alguma alteração no estado (enum State) da Thread?

- **void resume();**

  Coloca uma Thread que estava suspensa de volta para a fila de prontos.

Os alunos têm a liberdade para adicionar métodos e atributos necessários para a implementação dos métodos/funcionalidades descritas acima.

2. Arquivo Disponibilizado

O seguinte arquivo foi disponibilizados neste trabalho:

- main_class.h (main_class.cc é o mesmo do trabalho anterior): implementação de uma aplicação exemplo, usando o novo método join da class Thread e com diferentes valores passados para thread_exit():

A saída esperada do programa exemplo é:

```
main: inicio
main: 0
main: 1
main: esperando Pang...
  Pang: inicio
  Pang: 0
     Peng: inicio
```

```
   Peng: 0
      Ping: inicio
      Ping: 0
         Pong: inicio
         Pong: 0
            Pung: inicio
            Pung: 0
Pang: 1
   Peng: 1
      Ping: 1
         Pong: 1
            Pung: 1
Pang: 2
   Peng: 2
      Ping: 2
         Pong: 2
            Pung: 2
Pang: 3
   Peng: 3
      Ping: 3
         Pong: 3
            Pung: 3
Pang: 4
   Peng: 4
      Ping: 4
         Pong: 4
            Pung: 4
Pang: 5
   Peng: 5
      Ping: 5
         Pong: 5
            Pung: 5
Pang: 6
   Peng: 6
      Ping: 6
         Pong: 6
            Pung: 6
Pang: 7
   Peng: 7
      Ping: 7
         Pong: 7
            Pung: 7
Pang: 8
   Peng: 8
      Ping: 8
         Pong: 8
            Pung: 8
Pang: 9
   Peng: 9
```

```
          Ping: 9
             Pong: 9
                Pung: 9
   Pang: fim
main: Pang acabou com exit code 0
main: esperando Peng...
       Peng: fim
main: Peng acabou com exit code 1
main: esperando Ping...
          Ping: fim
main: Ping acabou com exit code 2
main: esperando Pong...
             Pong: fim
main: Pong acabou com exit code 3
main: esperando Pung...
                Pung: fim
main: Pung acabou com exit code 4
main: fim
```

Faz parte do trabalho pensar em todas o uso das funções implementadas, não apenas os cenários descritos/apresentados neste documento e no código de aplicação exemplo.

3. Formato de Entrega

Todos os arquivos utilizados na implementação do trabalho devem ser entregues em um único arquivo .zip ou .tar.gz na atividade do moodle. Deve ser anexado um arquivo Makefile para compilar o código.

4. Data de Entrega

Conforme data e horário da tarefa do moodle.

5. Avaliação

A avaliação se dará em 3 fases

1. Avaliação de compilação: compilar o código enviado. Caso haja erros de compilação, a nota do trabalho será automaticamente zerada.

2. Avaliação de execução: para validar que a solução executa corretamente sem falhas de segmentação. Caso haja falhas de segmentação, a nota é zerada. Será também avaliado o uso de variáveis globais (-5 pontos) e vazamentos de memória (-20%).

3. Avaliação da organização do código: busca-se nesta fase avaliar a organização do código orientado a objetos. Deve-se usar classes e objetos e não estilo de programação baseado em procedimentos (como na linguagem C).

Plágio não será tolerado em nenhuma hipótese ao longo dos trabalhos, acarretando em nota 0 a todos os envolvidos.