Geek University

Evolua seu lado geek!

www.geekuniversity.com.br

Introdução às Threads

Introdução às Threads

Threads (linha de execução) foram os primeiros recursos criados para realizar **programação concorrente**, fazendo uso então de **multi-threads**.

Para que possamos entender melhor o que são **threads**, precisamos antes compreender o que são **processos**.

Introdução às Threads

O que são processos?

O contexto de execução de um programa sendo executado é considerado um processo.

Ou seja, um processo nada mais é do que a instância de um programa de computador sendo executado.

Além disso, cada processo sendo executado tem um conjunto de recursos, como por exemplo memória, segurança e etc, que são atribuídos a este processo.

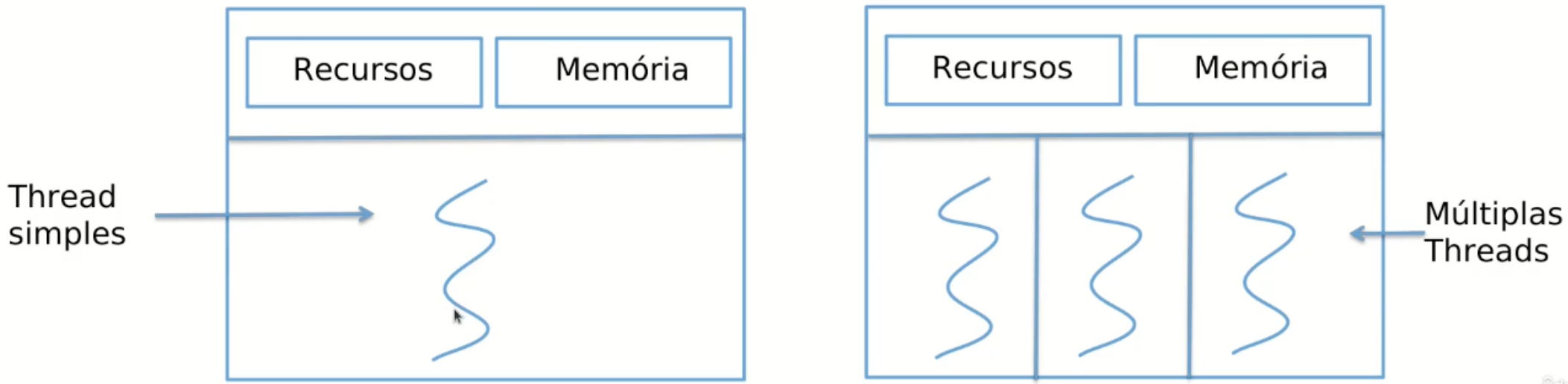
Por fim, um processo é composto por uma ou mais threads (linhas de execução).

Introdução às Threads

O que são threads?

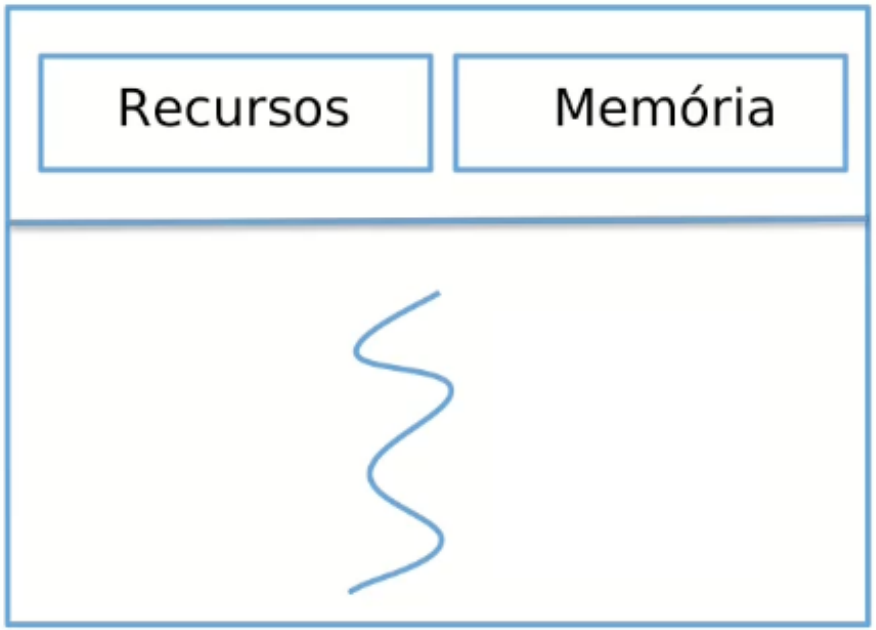
Uma thread, ou linha de execução, é a menor sequência de instruções que pode ser gerenciada pelo sistema operacional.

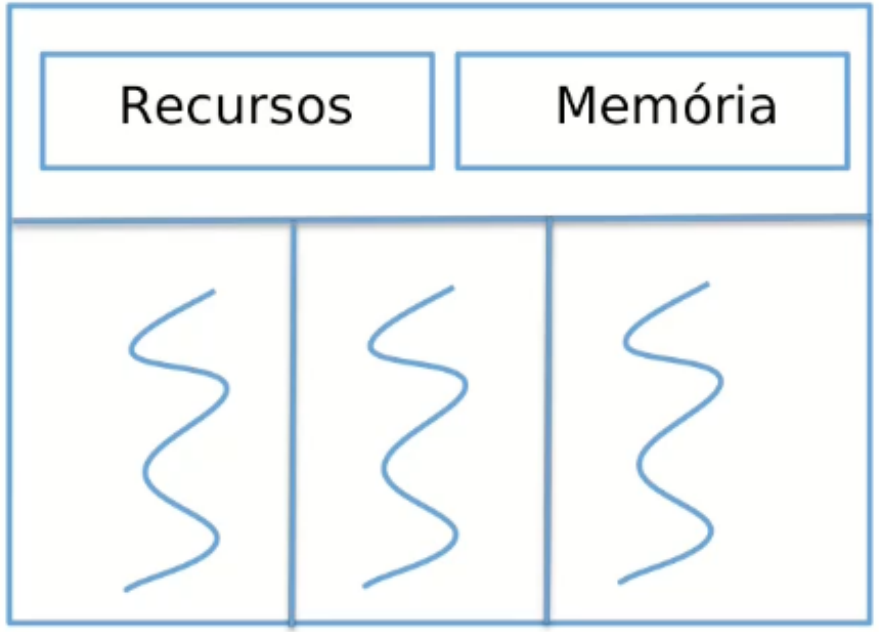
Um programa pode ser executada por uma simples thread ou por múltiplas threads.



Introdução às Threads

Quando o computador/servidor dispõe de múltiplos cores, cada thread pode ser executada em paralelo. Caso contrário, mesmo que haja múltiplas threads elas serão executadas de forma sequencial.





Introdução às Threads

Conforme vimos na seção de introdução, as threads surgiram em Python a partir da versão 1.5 e conseguimos usar seus recursos através do pacote *threading*

Introdução às Threads

Síntaxe básica de threads em Python

Introdução às Threads

Síntaxe básica de threads em Python

```python
import threading


def alguma_funcao(param):
    print('Executa algo...')
    print(f'Usa o parâmetro recebido: {param}')

    return param * param



th = threading.Thread(target=alguma_funcao, args=(42,))

th.start()
th.join()

```

Introdução às Threads

Síntaxe básica de threads em Python

```python
import threading


def alguma_funcao(param):
    print('Executa algo...')
    print(f'Usa o parâmetro recebido: {param}')

    return param * param


th = threading.Thread(target=alguma_funcao, args=(42,))
th.start()
th.join()

```

Introducão às Threads

Síntaxe básica de threads em Python

```python
import threading


def alguma_funcao(param):
    print('Executa algo...')
    print(f'Usa o parâmetro recebido: {param}')

    return param * param


th = threading.Thread(target=alguma_funcao, args=(42,))

th.start()
th.join()

```

Criamos a instância de uma thread através da classe Thread presente no pacote threading, e como único parâmetro obrigatório especificamos o nome da função que a thread deverá executar através do argumento ‘target’. Neste exemplo estamos informando também no parâmetro ‘args’ o valor 43 para servir de valor para o parâmetro ‘param’ da função.

Introducão às Threads

Síntaxe básica de threads em Python

```
1  import threading
2
3
4  def alguma_funcao(param):
5      print('Executa algo...')
6      print(f'Usa o parâmetro recebido: {param}')
7
8      return param * param
9
10
11
12 th = threading.Thread(target=alguma_funcao, args=(42,))
13
14 th.start()
15 th.join()
16
17
```

Na linha 8 executamos o método start() para informar à thread que ela pode ir para a piscina de threads (thread pool) para ser executada.

Introducão às Threads

Síntaxe básica de threads em Python

```
1  import threading
2
3
4  def alguma_funcao(param):
5      print('Executa algo...')
6      print(f'Usa o parâmetro recebido: {param}')
7
8      return param * param
9
10
11
12 th = threading.Thread(target=alguma_funcao, args=(42,))
13
14 th.start()
15 th.join()
16
17
```

Na linha 9 executamos o método join() para informar à thread ela deve suspender a execução do programa até que ela seja finalizada.

Introdução às Threads

Se analisarmos o construtor da classe Thread veremos que temos outros parâmetros opcionais...

```
class Thread:
    """A class that represents a thread of control.

    This class can be safely subclassed in a limited fashion. There are two ways
    to specify the activity: by passing a callable object to the constructor, or
    by overriding the run() method in a subclass.

    """

    _initialized = False

    def __init__(self, group=None, target=None, name=None,
                 args=(), kwargs=None, *, daemon=None):
        """This constructor should always be called with keyword arguments. Arguments are:
```

Introdução às Threads

Se analisarmos o construtor da classe Thread veremos que temos outros parâmetros opcionais...

```
class Thread:
    """A class that represents a thread of control.

    This class can be safely subclassed in a limited fashion. There are two ways
    to specify the activity: by passing a callable object to the constructor, or
    by overriding the run() method in a subclass.

    """

    _initialized = False

    def __init__(self, group=None, target=None, name=None,
                 args=(), kwargs=None, *, daemon=None):
        """This constructor should always be called with keyword arguments. Arguments are:
```

Podemos definir um grupo para a thread através do parâmetro ‘group’

Introdução às Threads

Se analisarmos o construtor da classe Thread veremos que temos outros parâmetros opcionais...

```
class Thread:
    """A class that represents a thread of control.

    This class can be safely subclassed in a limited fashion. There are two ways
    to specify the activity: by passing a callable object to the constructor, or
    by overriding the run() method in a subclass.

    """

    _initialized = False

    def __init__(self, group=None, target=None, name=None,
                 args=(), kwargs=None, *, daemon=None):
        """This constructor should always be called with keyword arguments. Arguments are:
```

Podemos definir um nome para a thread através do parâmetro ‘name’

Introdução às Threads

Se analisarmos o construtor da classe Thread veremos que temos outros parâmetros opcionais...

```
class Thread:
    """A class that represents a thread of control.

    This class can be safely subclassed in a limited fashion. There are two ways
    to specify the activity: by passing a callable object to the constructor, or
    by overriding the run() method in a subclass.

    """

    _initialized = False

    def __init__(self, group=None, target=None, name=None,
                 args=(), kwargs=None, *, daemon=None):
        """This constructor should always be called with keyword arguments. Arguments are:
```

Podemos usar kwarga para parâmetros nomeados para a função informada no target da thread através do parâmetro ‘kwargs’

Introdução às Threads

Se analisarmos o construtor da classe Thread veremos que temos outros parâmetros opcionais...

```
class Thread:
    """A class that represents a thread of control.

    This class can be safely subclassed in a limited fashion. There are two ways
    to specify the activity: by passing a callable object to the constructor, or
    by overriding the run() method in a subclass.

    """

    _initialized = False

    def __init__(self, group=None, target=None, name=None,
                 args=(), kwargs=None, *, daemon=None):
        """This constructor should always be called with keyword arguments. Arguments are:
```

Podemos definir que esta thread irá ser executada no modo ‘daemon’ se definirmos True para o parâmetro.

OBS: No modo daemon uma thread filha mantém sua execução mesmo que a thread mãe morra.

Introdução às Threads

Na próxima aula iremos criar nossa thread simples...

Geek University

Evolua seu lado geek!

www.geekuniversity.com.br