Geek University

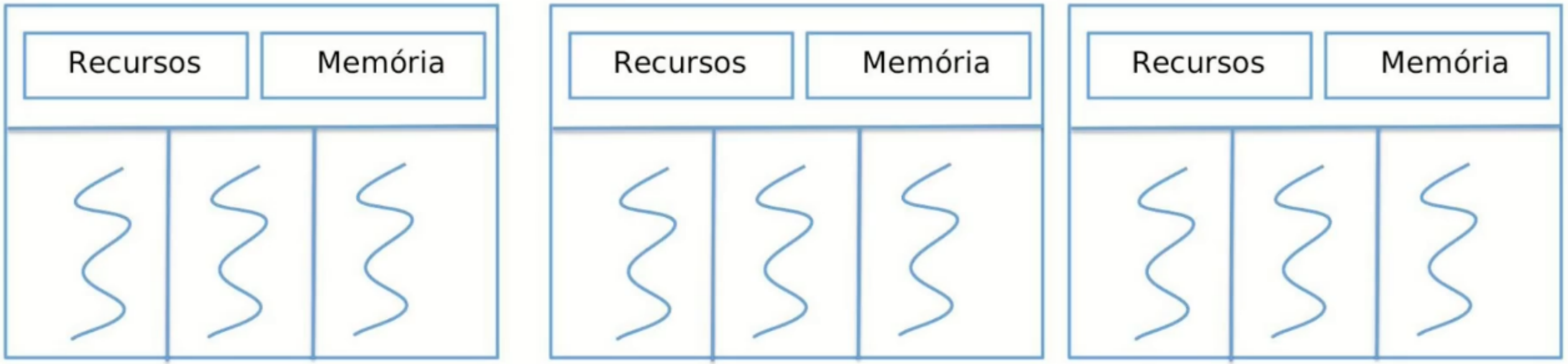
Evolua seu lado geek!

www.geekuniversity.com.br

Comunicação entre processos

Comunicação entre processos

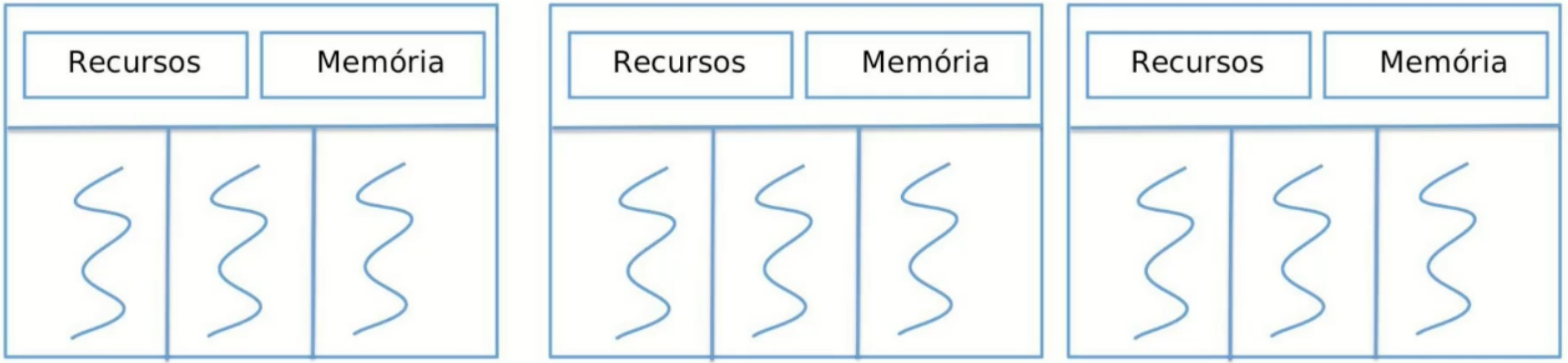
Aprendemos em aulas passadas que em um processo podemos ter várias threads que compartilham (ou podem compartilhar) recursos e memória entre si, dentro do mesmo processo.



Comunicação entre processos

Aprendemos também que uma thread ou processo não tem acesso aos recursos ou memória de outras threads ou processos.

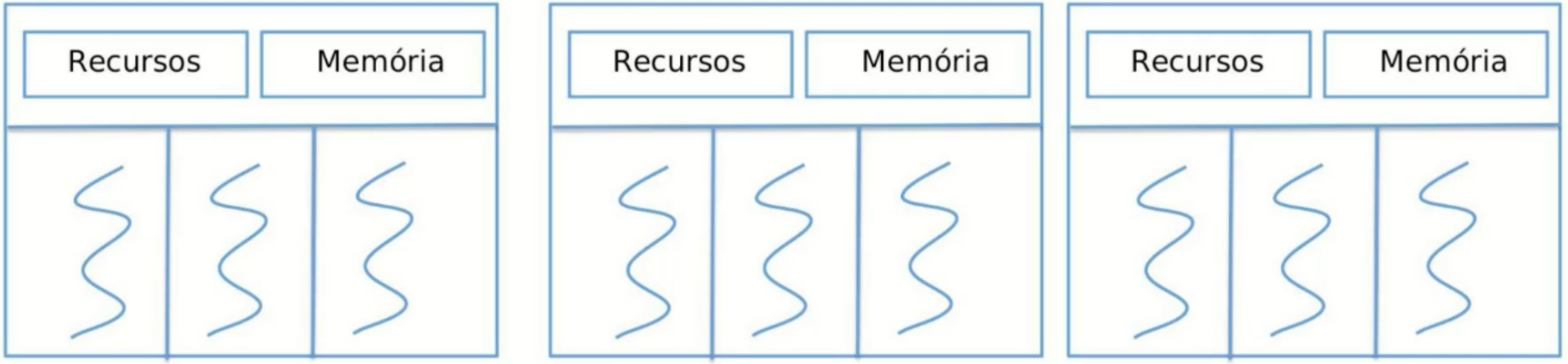
Pelo menos não de forma direta....



Comunicação entre processos

Aprendemos também que uma thread ou processo não tem acesso aos recursos ou memória de outras threads ou processos.

Pelo menos não de forma direta....



Podemos acessar recursos/memória de outros processos através de dois recursos: Pipe e Queue

Comunicação entre processos

Através da classe multiprocessing.Pipe

Comunicação entre processos

Através da classe multiprocessing.Pipe

Pense neste “Pipe” como um cano onde tem uma pessoa em uma ponta e outra pessoa na outra ponta e quando uma pessoa fala a outra escuta através do cano e quando a outra pessoa responde a outra escuta.

Comunicação entre processos

Através da classe multiprocessing.Pipe

Vamos ao código fazer um exemplo…

Comunicação entre processos

E quando à Queue?

Comunicação entre processos

E quando à Queue?

Fizemos uso da Queue na seção de Threads.

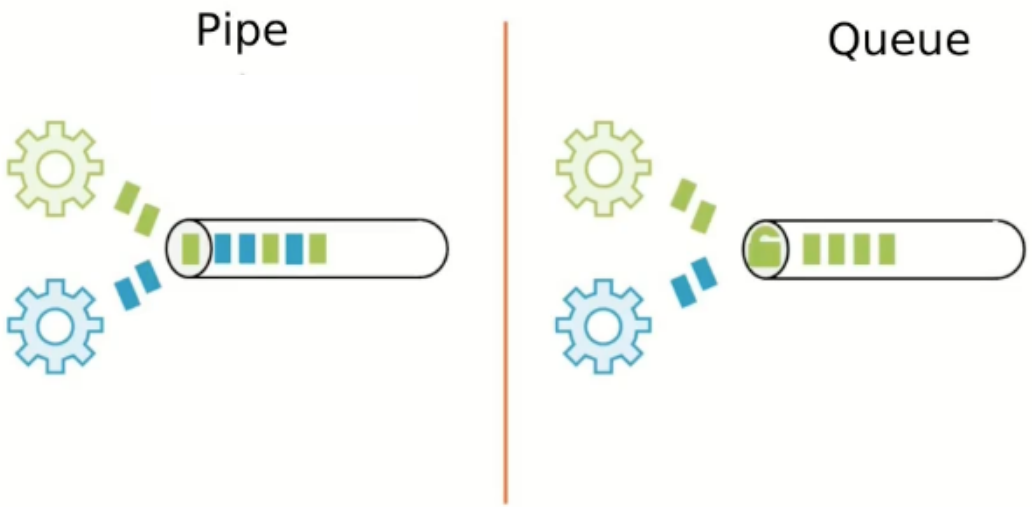
A vantagem do uso de Queues ao invés de Pipe é que as Queues permitem um maior controle sobre o processo que vai fazer uso dos dados, podendo ser realizados lock/unlock do acesso.

Comunicação entre processos

E quando à Queue?

Fizemos uso da Queue na seção de Threads.

A vantagem do uso de Queues ao invés de Pipe é que as Queues permitem um maior controle sobre o processo que vai fazer uso dos dados, podendo ser realizados lock/unlock do acesso.



Comunicação entre processos

Queue: Threads vs Processing

threading .Queue	multiprocessing .Queue
qsize()	qsize()
put()	put()
get()	get()
empty()	empty()
full()	full()
task_done()	~~task_done()~~
join()	~~join()~~

Praticamente todos os métodos de Queue que podem ser usados em Threads podem também ser usados da mesma forma em Processing....tirando os métodos: *task_done()* e *join()*

Comunicação entre processos

Queue: Threads vs Processing

threading .Queue	multiprocessing .Queue	multiprocessing .JoinableQueue
qsize()	qsize()	qsize()
put()	put()	put()
get()	get()	get()
empty()	empty()	empty()
full()	full()	full()
task_done()	~~task_done()~~	task_done()
join()	~~join()~~	join()

Para manter total compatibilidade entre as APIs de Threads e Processing, foi criada a classe *JoinableQueue* que implementa inclusive os dois últimos métodos.

Comunicação entre processos

Queue: Threads vs Processing

threading .Queue	multiprocessing .Queue	multiprocessing .JoinableQueue
qsize()	qsize()	qsize()
put()	put()	put()
get()	get()	get()
empty()	empty()	empty()
full()	full()	full()
task_done()	~~task_done()~~	task_done()
join()	~~join()~~	join()

Vamos ao código refatorar nosso exemplo com o uso de Queue...

Geek University

Evolua seu lado geek!

www.geekuniversity.com.br