

Guião de apoio 6 ZooKeeper em Python

1. Introdução ao tema

Neste guião, serão implementadas aplicações distribuídas que utilizam os serviços do ZooKeeper para coordenar os processos participantes da mesma.

2. Instalação do Zookeeper

Para instalar o ZooKeeper e a biblioteca cliente Kazoo na máquina virtual do DI, execute os seguintes comandos num terminal:

```
$ sudo apt-get update  
$ sudo apt-get install zookeeperd  
$ pip3 install kazoo
```

3. Kazoo, uma biblioteca Python para o ZooKeeper

A construção de um programa Python que seja cliente do ZooKeeper pode ser conseguida através da classe `KazooClient` definida no módulo `kazoo.client`. Recorde o exemplo mostrado na aula TP06 e utilize-o como programa template para iniciar a resolução dos exercícios deste guião.

```
from kazoo.client import KazooClient  
  
# Criar um ZooKeeper handler  
zh = KazooClient()  
zh.start()  
  
# restante do programa  
  
zh.stop()  
zh.close()
```

4. Exercícios

1. Neste primeiro exercício, vamos implementar um **algoritmo de exclusão mútua** baseado nos serviços fornecidos pelo ZooKeeper. Basicamente, o que se pretende é que, num sistema com múltiplos processos (p. ex., vários terminais a correr o mesmo programa), apenas um processo de cada vez consiga processar uma determinada zona crítica (como a apresentada na Listagem 1).

Listagem 1 – Exemplo de uma função com uma zona crítica.

```
def critical_zone():  
    print('Comecei a executar a zona crítica')  
    time.sleep(10)  
    print('Terminei a execução da zona crítica')
```

Copie o programa template apresentado na Secção 3, acrescente o método definido na Listagem 1 e em seguida implemente o seguinte algoritmo:

1. Utilize o método `ensure_path` para garantir que o nó `'/LOCKS'` exista.
2. Utilize o método `create` para criar um nó efêmero e sequencial com o prefixo `'/LOCKS/L-'`. Guarde o identificador do nó criado, o qual é retornado por este método.
3. Crie um ciclo que:
 - a. Utilize o método `get_children` para obter a lista de nós filhos do nó `'/LOCKS'`.
 - b. Obtenha, da lista retornada no passo anterior, qual é o nó com o menor identificador.
 - c. Se o meu nó (criado no Passo 2) for o nó com o menor identificador (obtido no Passo 3b), então:
 - i. Execute o método da zona crítica.
 - ii. Utilize o método `delete` para remover o nó criado no Passo 2.
 - d. Senão:
 - i. Imprima a lista obtida no Passo 3b.
 - ii. Faça o programa dormir por 1 segundo. Após este período, ele voltará ao Passo 3.

Para avaliar o algoritmo implementado, abra três janelas de terminais e execute concorrentemente nelas o programa criado neste exercício. O resultado esperado é que o primeiro processo a ser executado consiga criar o nó e entrar na zona crítica primeiro, enquanto os outros dois processos ficam bloqueados a espera que o primeiro termine. Quando o primeiro processo terminar, um dos outros dois vai conseguir entrar na zona crítica. Somente após os dois primeiros processos terminarem é que o último processo conseguirá executar a zona crítica.

2. Neste segundo exercício, vamos implementar **um programa que utilize os watchers do ZooKeeper** para monitorizar alguns nós existentes no mesmo.

- a) Novamente, copie o programa template da Secção 3 e acrescente (após o método `start`) o código apresentado na Listagem 2.




Listagem 2 – Exemplo de funções de monitorização de nós.

```
zh.ensure_path('/PAI')

@zh.DataWatch('/NORMAL')
def watch_node (data, stat):
    print("Stat: %s\nData: %s\n" % (stat , data))

@zh.ChildrenWatch('/PAI')
def watch_children (children):
    print("Children are now: %s" % children)

while True:
    pass
```

- b) Inicialize numa janela de terminal o algoritmo implementado. Ele deverá ficar bloqueado no loop infinito.
- c) Noutra janela de terminal, abra o programa `zooinspector`, conecte-o ao serviço do ZooKeeper e crie um nó `'/NORMAL'` (através do botão ). Após criar o nó, qualquer modificação do seu valor (no lado direito da janela, sem esquecer de utilizar o botão  para gravar as modificações) causará uma notificação do ZooKeeper ao processo lançado pelo Passo 2b, o qual executará o método `watch_node` criado no Passo 2a.
- d) Ainda no programa `zooinspector`, crie um nó `'/PAI/FILHO1'` (selecione o nó `'/PAI'` na lateral esquerda e carregue no botão ). Ao criar um nó filho de `'/PAI'`, o processo lançado pelo Passo 2b será notificado e este executará o método `watch_children` criado no Passo 2a.

3. Neste exercício, vamos utilizar o ZooKeeper para **criar um protótipo de um jogo de duelo**, o qual precisa de dois jogadores para iniciar e requer que ambos saiam do jogo para que seja apresentada uma pontuação. Comece por copiar novamente o programa template da Secção 3 e em seguida:

- a) Importe a receita de DoubleBarrier do Kazoo
`from kazoo.recipe.barrier import DoubleBarrier`
- b) Após o método `start`, crie uma instância de `DoubleBarrier`. Veja a documentação desta receita em [2]. Basicamente, os argumentos a passar são:
 - a sessão cliente com o ZooKeeper
 - o path onde a barreira fará a monitorização
 - quantos jogadores são necessários para que os processos sejam liberados para iniciar o jogo.
- c) Em seguida, imprima 'Vou iniciar o jogo' e utilize o método `enter` para o processo entrar na barreira.
- d) Em seguida, imprima 'O jogo vai começar' e faça um ciclo como o da Listagem 3.

Listagem 3 – Exemplo de ciclo que se quebra com uma interrupção

```
while 1:
    try:
        pass
    except KeyboardInterrupt:
        break
```

- e) Quando houver uma interrupção, o jogo sairá do ciclo. Imprima 'Vou encerrar a minha participação no jogo' e utilize o método `leave` para o processo pedir para sair da barreira (o processo somente será liberado quando todos os jogadores também pedirem para sair).
- f) Finalmente, imprima 'O jogo foi encerrado.\nA sua pontuação foi X.'

Para executar o jogo, abra duas janelas de terminal. Execute o programa na primeira e veja que o processo fica bloqueado a espera do segundo jogador. Execute o programa na segunda janela e veja que ambos processos passam a fase de entrada na barreira e o jogo é iniciado.

Quando quiser encerrar o jogo, interrompa (Ctrl+C) a execução num dos terminais e verá que este processo pedirá para sair do jogo e ficará bloqueado a espera de que o outro processo também saia. Somente quando o outro processo também for interrompido, é que o jogo será terminado e a pontuação apresentada.

Abra uma terceira janela de terminal, inicie o `zooinspector` e execute novamente o jogo. Porém, desta vez atualize o estado do `zooinspector` (através do botão 🐞) a cada etapa do jogo, para visualizar as mudanças que acontecem na hierarquia do ZooKeeper.

5. Bibliografia e outro material de apoio

[1] <https://kazoo.readthedocs.io/en/latest/>

[2] <https://kazoo.readthedocs.io/en/latest/api/recipe/barrier.html#kazoo.recipe.barrier.DoubleBarrier>