METAS

Implemente um aplicativo de telefonia simulado usando Python + Twisted.
Diferentes implementações com diferentes complexidades são propostas para atingir o objetivo. Qualquer que seja o tipo de implementação escolhido, o mesmo conjunto de pré-requisitos se aplica. Existem também alguns pontos de bônus que podem ser implementados, se desejado, e o tempo permitir.

PRÉ-REQUISITOS

1. Um sistema limpo do CentOS 7 acessível via SSH.
2. O desenvolvimento pode ser feito em uma VM ou contêiner em execução no seu próprio computador.
3. O código produzido deve ser mantido em um repositório Git. Cabe a você onde o repositório irá viver: no GitHub, GitLab, em um diretório local no seu computador, etc.

IMPLEMENTAÇÃO BÁSICA

Implemente um aplicativo de fila de call center que gerencia um conjunto de operadores de call center e entrega as chamadas recebidas para serem "atendidas" por eles.

Descrição

1. Escreva um aplicativo de call center simulado. É chamado de "simulado" porque não realmente lida com chamadas reais feitas por softphones ou aparelhos telefônicos. Em vez disso, ele lida com comandos digitados em stdin e relata seus resultados imprimindo mensagens para stdout.

2. O aplicativo aceita comandos como "ligar", "atender", "rejeitar" e "desligar". Essa
Os comandos tentam simular ações executadas por um softphone ou telefone real.
(Dica: use o módulo “cmd” da biblioteca padrão do Python.)

3. O aplicativo deve aceitar os comandos:
call <id> makes application receive a call whose id is <id>.
answer <id> makes operator <id> answer a call being delivered to it.
reject <id> makes operator <id> reject a call being delivered to it.
hangup <id> makes call whose id is <id> be finished.

4. Um operador tem um ID (como "A", "B", ...) e um estado. Seu estado está "disponível" quando ocioso, "Tocando" quando uma chamada foi entregue, mas ainda não foi atendida por ela e "Ocupado" quando está atendendo uma chamada.

5. Quando o aplicativo recebe uma chamada, ele deve imprimir em stdout:
Chamada <ID da chamada> recebida

6. A chamada recebida deve ser entregue a uma operadora que esteja disponível, o aplicativo deve definir o estado do operador como "tocando" e imprimir em stdout:
Chamada <ID da chamada> tocando para o operador <ID do operador>

7. O aplicativo agora deve esperar e detectar quando o operador atende a chamada e, em seguida, defina o estado do operador como "ocupado" e imprima em stdout:
Chamada <ID da chamada> atendida pelo operador <ID do operador>

8. Se o operador rejeitar a chamada, o aplicativo deverá definir o estado do operador como “Disponível” novamente e imprima para stdout:
Chamada <ID da chamada> rejeitada pelo operador <ID do operador>

9. O aplicativo deve detectar quando uma chamada atendida for concluída, definir os respectivos como "disponível" novamente e imprima em stdout:
Chamada <ID da chamada> concluída e operador <ID do operador> disponível

10. Se nenhum operador estiver disponível quando uma chamada for recebida, ela deverá ser efetuada ao final de um fila, aguardando até que um operador esteja disponível para atender e o aplicativo deve imprimir em stdout:
Chamada <ID da chamada> aguardando na fila

11. Assim que um operador estiver disponível, o aplicativo deve entregar a chamada em espera na primeira posição da fila a ser respondida por esse operador, conforme descrito na etapa 6

12. Se uma chamada não for atendida por um operador, o aplicativo deve tentar entregá-la ao próximo operador disponível e repita a etapa 6 até que seja atendida.

13. Se uma chamada for concluída antes de ser atendida por um operador, o aplicativo deverá imprimir para stdout: Chamada <ID da chamada> perdida

Testes
1. Configure o aplicativo para ter 2 operadores cujos IDs são A e B.

2. Digite "ligar 1" para iniciar uma ligação e você deverá ver:
Chamada 1 recebida
Chamada 1 tocando para o operador A

3. Digite “resposta A” para o operador A responder e você deverá ver:
Chamada 1 atendida pelo operador A

4. Digite “hangup 1” para finalizar a ligação e você verá:
Chamada 1 concluída e operador A disponível

5. Digite "ligar 2" para iniciar uma ligação e você deverá ver:
Chamada 2 recebida
Chamada 2 tocando para o operador A

6. Digite "ligar 3" para iniciar outra chamada e você verá:
Chamada 3 recebida
Chamada 3 tocando para o operador B

7. Digite “resposta A” para a operadora A atender a chamada e você deverá ver:
Chamada 2 atendida pelo operador A

8. Digite “resposta B” para o operador B atender a chamada e você deverá ver:
Chamada 3 atendida pelo operador B

9. Digite “ligue 4”, depois “ligue 5” e você verá:
Chamada 4 recebida
Chamada 4 em espera na fila
Chamada 5 recebida
Chamada 5 em espera na fila

10. Digite "hangup 3" para finalizar a chamada e você deverá ver:
Chamada 3 concluída e operador B disponível
Chamada 4 tocando para o operador B

11. Digite "hangup 2" para finalizar a chamada e você verá:
Chamada 2 concluída e operador A disponível
Ligue 5 tocando para o operador A

12. Digite “resposta B” para o operador B atender a chamada e você deverá ver:
Chamada 4 atendida pelo operador B

13. Digite "rejeitar A" para a operadora A rejeitar a chamada e você deverá ver:
Chamada 5 rejeitada pelo operador A
Ligue 5 tocando para o operador A

14. Digite "ligar 6" para iniciar uma ligação e você deverá ver:
Chamada 6 recebida
Chamada 6 em espera na fila

15. Digite “hangup 5” para finalizar a ligação e veja:
Chamada 5 perdida
Chamada 6 tocando para o operador A

16. Digite "ligar 7" para iniciar uma ligação e você deverá ver:
Chamada 7 recebida
Chamada 7 em espera na fila

17. Digite "hangup 7", "hangup 6" e "hangup 4" para finalizar as chamadas e veja:
Chamada 7 perdida
Chamada 6 perdida
Chamada 4 concluída e operador B disponível

IMPLEMENTAÇÃO AVANÇADA
Estenda a implementação básica dividindo-a em dois processos que trabalham juntos por comunicação via rede em uma arquitetura cliente-servidor.

Descrição
1. Comece com uma implementação básica funcional.

2. Divida o código em duas partes: o interpretador de comandos e a fila do call center Gerente.

3. O intérprete de comando atuará como um cliente que recebe do stdin o mesmo conjunto de comandos como na implementação básica. Cada comando é então enviado ao servidor a ser tratado pelo gerenciador de filas do call center.

4. O gerenciador de filas do call center atuará como um servidor que recebe comandos de clientes. Cada comando é então tratado da mesma maneira que era no básico implementação.

5. A parte do código do interpretador de comandos deve ser transformada em um cliente TCP usando Python + Torcido. Ele deve se conectar ao servidor na porta 5678 para enviar comandos para ele. Cada O comando válido digitado em stdin deve ser convertido em uma sequência JSON e enviado ao servidor.

6. Os comandos devem ser convertidos para JSON de acordo com a tabela:
call <id> {"command": "call", "id": "<id>"}
answer <id> {"command": "answer", "id": "<id>"}
reject <id> {"command": "reject", "id": "<id>"}
hangup <id> {"command": "hangup", "id": "<id>"}

7. Dica: os métodos Cmd.cmdloop do módulo cmd do Python e reator.run do Twisted não pode ser usado simultaneamente porque ambos bloqueiam a execução do programa. UMA A possibilidade é executar o Cmd.cmdloop em outro encadeamento usando reattor.callInThread.

8. A parte do código do gerenciador de filas do call center deve ser transformada em um servidor TCP usando Python + Torcido. Ele deve aceitar conexões na porta 5678 para receber comandos de clientes conectados. Cada comando deve ser interpretado como um JSON corda e manuseado.

9. Após a manipulação de cada comando, o servidor deve enviar uma resposta ao cliente. o resposta é exatamente a mesma mensagem que seria impressa em stdout no arquivo básico implementação. Ele deve ser enviado como uma string JSON como esta:
{"response": "<mensage>"}

10. O cliente deve receber cada resposta, interpretá-la como uma string JSON e imprimir sua mensagem para stdout exatamente como seria na implementação básica.

Testes

1. Configure o gerenciador de filas do call center para ter 2 operadores.
2. Execute o gerenciador de filas do call center.
3. Execute o interpretador de comandos.
4. Execute todos os testes descritos para a implementação básica, digitando comandos no stdin do processo do interpretador de comandos e na verificação dos resultados impressos em sua stdout. Os resultados devem ser exatamente os mesmos.

TAREFAS DE BÔNUS

Essas são tarefas absolutamente não essenciais, mas que podem ser feitas para ganhar pontos extras, se desejado e o tempo permite (não faça nenhum, nenhum ou todos).

● Implementar detecção de tempo limite. Se um operador não atender ou rejeitar uma chamada dentro de 10 segundos após ser entregue, o aplicativo deve desistir, definir o estado do operador como "disponível" novamente e imprima em stdout:
Chamada <ID da chamada> ignorada pelo operador <ID do operador>

● Melhorar a implementação avançada do interpretador de comandos: em vez de executar Cmd.cmdloop em outro encadeamento, use Twisted para ler linhas de stdin e chame Cmd.onecmd como cada linha é lida.

● Empacote o aplicativo gerenciador de filas do call center como um contêiner, o comando interprete como outro e publique ambos no Docker Hub para que qualquer pessoa possa executá-los.

REFERÊNCIAS

1. A Biblioteca Padrão do Python: Suporte para intérpretes de comando orientados a linhas
Sugestão: saiba como implementar intérpretes de comando.
https://docs.python.org/2/library/cmd.html

2. A biblioteca padrão do Python: codificador e decodificador JSON
Sugestão: saiba como converter de e para JSON em Python.
https://docs.python.org/2/library/json.html

3. Fundamentos de programação de rede torcida - Jessica McKellar, Abe Fettig
Capítulos sugeridos: 1, 2 e 6.
http://shop.oreilly.com/product/0636920025016.do

4. Documentação distorcida: agendando tarefas para o futuro
Sugestão: aprenda a executar uma função após um tempo limite usando o Twisted
http://twistedmatrix.com/documents/current/core/howto/time.html

5. Exemplo torcido: Lendo linhas de stdin sem bloquear
Sugestão: aprenda a ler linhas do stdin usando o Twisted
https://twistedmatrix.com/documents/15.5.0/_downloads/stdin.py

6. Introdução à programação assíncrona e torcida
Sugestão: aprenda a usar o twistd para executar programas Twisted como daemons.
http://krondo.com/?p=2345

7. Containerize um aplicativo Python em 5 minutos
Sugestão: saiba como empacotar um aplicativo Python como um contêiner
https://www.wintellect.com/containerize-python-app-5-minutes/