UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CAMPUS DE SÃO MATEUS
DEPARTAMENTO DE COMPUTAÇÃO E ELETRÔNICA

ALUNO: Wellerson Prenholato de Jesus

CURSO: Ciência da Computação

PROFESSOR: Wilian Hiroshi Hisatugu

DISCIPLINA: Redes de Computadores

18 de abril de 2021.

Atividade Avaliativa 2

Em primeiro lugar, de alguma maneira o computador precisa saber, ao criar um socket, como ele vai conseguir se comunicar. Os Sockets podem ter funcionalidades bem diversas - podendo se comunicar com processos dentro ou fora de uma mesma máquina.

A minha proposta para essa atividade foi desenvolver de forma simples o jogo pedra, papel e tesoura, também chamado em algumas regiões do Brasil de jokenpô. Utilizando a linguagem de programação python e com auxílio dos sockets, foi possível desenvolver a comunicação entre um cliente e um servidor, sendo eles processos de uma mesma máquina. No manual de uso explicarei com mais detalhes o funcionamento do jogo.

Seguindo a mesma ideia do jokenpô:

- Pedra ganha da tesoura (amassando-a ou quebrando-a).
- Tesoura ganha do papel (cortando-o).
- Papel ganha da pedra (embrulhando-a).

O jogo foi desenvolvido utilizando a comunicação cliente x servidor, empregando o protocolo de comunicação TCP.

Manual de uso

Inicialmente o script server.py deve ser executado utilizando a linguagem de programação python. Podemos perceber que depois de executarmos o script server.py o mesmo retorna uma mensagem aguardando a conexão de um cliente.

Em seguida, o script client.py deve ser executado. Uma mensagem será enviada pelo servidor apresentando o endereço do cliente que foi conectado.

O número da porta para conexão estando certo, a conexão é realizada após a execução dos scripts server.py e client.py, nessa mesma ordem.

Com a conexão feita, na aba do client através de um menu o usuário deve informar o tipo de palpite que ele deseja, tendo como opção 1 - Palpite aleatório, opção 2 - Informar um palpite e na opção 0 (zero) caso ele queira encerrar a conexão. Caso a opção escolhida seja a 1, um palpite aleatório é enviado para o servidor, podendo ser pedra, papel ou tesoura e o servidor responde com outro palpite aleatório (podendo ser também pedra, papel ou tesoura) juntamente com o nome do ganhador, sendo ele servidor ou cliente.

Caso a opção escolhida seja a 2, é aberto um novo menu com as opções dos palpites, sendo elas: pedra, papel e tesoura. Basta informar o número referente ao palpite e pressionar enter.

Seguindo a mesma ideia do palpite aleatório, o cliente enviará essa mensagem para o servidor e o servidor retornará com o outro palpite e o respectivo ganhador.

Com o retorno do servidor, os resultados são apresentados na aba do client, e caso o usuário queira continuar jogando, basta ele informar as opções 1 ou 2.

Caso o usuário pressione a tecla 0 (zero) em seguida a tecla enter, a comunicação entre cliente x servidor será finalizada. E uma mensagem será apresentada na aba do client e outra na aba do servidor.

Obs.: Os scripts citados aqui, foram enviados como anexo na atividade avaliativa 2 na disciplina de redes de computadores na plataforma classroom.

Explicação do código

Explicação linha a linha do arquivo **server.py**.

```python
# Bibliotecas

import socket

import random

# Identificação do HOST E PORT do servidor

HOST = 'localhost' # Identifica o nome do servidor

PORT = 5000 # Identifica a porta do servidor

addr = (HOST, PORT)

# Lista com as possíveis jogadas que podem ser escolhidas pelo servidor.

opcoesJogadas = ['Pedra', 'Papel', 'Tesoura']



# O mecanismo de Socket foi criado para receber a conexão, onde na função passamos 2 argumentos, AF_INET que declara a família do protocolo;

# Se fosse um envio via Bluetooth por exemplo, seria: AF_BLUETOOTH, e o SOCKET_STREAM, indica que será TCP/IP.

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) # SOCK_STREAM identificação do TCP

# A constante AF_INET faz parte de um grupo denominado famílias de endereços, ou address families, que constitui exatamente o primeiro parâmetro opcional do construtor socket.

# A AF_INET abrange os endereços do tipo IPv4, antigo padrão da Internet.



# Esta linha define para qual IP e porta o servidor deve aguardar a conexão.

sock.bind(addr)



# Define o limite de conexões. E no caso, estamos limitando em 5 conexões.

sock.listen(5)

print("Aguardando conexão de um cliente:")

conn, ender = sock.accept() # Conexão e endereço



print('Connectado com', ender) # Apresentação do endereço do cliente, composto de nome do host e a porta que foram conectados.

print("\nOs palpites do servidor são ALEATÓRIOS!\n")
```

```
while True: # O servidor fica liberado por tempo indeterminado ou até a conexão ser encerrada.

    # Aguarda um dado enviado pela rede de até 1024 Bytes, a função 'recv' possui somente 1 argumento que é o tamanho do Buffer.

    data = conn.recv(1024) # 1024 Byter serão recebidos do client



    # print("Resposta do cliente:", data.decode())



    if not data: # Quando não tiver mais nada dentro do data a conexão é encerrada

        print("\nConexão encerrada!\n")



        conn.close() # Serve para fechar a conexão entre as aplicações.

        break



    palpiteClient = str(data.decode()) # Utilizado para decodificar e transformar a mensagem em string enviada pelo cliente.

    palpiteServ = random.choice(opcoesJogadas) # O palpite do servidor é escolhido de forma randômica, as opções estão dentro da lista opcoesJogadas.



    print("* O Servidor respondeu:", palpiteServ) # Apresenta o palpite do servidor



    # Verifica quando o cliente ganha a jogada

    if ((palpiteClient == 'Tesoura' and palpiteServ == 'Papel') or (palpiteClient == 'Pedra' and palpiteServ == 'Tesoura') or (palpiteClient == 'Papel' and palpiteServ == 'Pedra')):

        ganhador = 'Cliente'



    # Verifica quando o Servidor ganha a jogada

    if ((palpiteClient == 'Papel' and palpiteServ == 'Tesoura') or (palpiteClient == 'Tesoura' and palpiteServ == 'Pedra') or (palpiteClient == 'Pedra' and palpiteServ == 'Papel')):

        ganhador = 'Servidor'



    # Verificação em caso de empate

    if (palpiteClient == palpiteServ):

        ganhador = 'Empate'
```

```
    # String de retorno para o cliente

    result = '- Cliente: '+ str(palpiteClient) + '\n - Servidor: ' + str(palpiteServ) + '\n=> GANHADOR: ' + str(ganhador) + '\n' # Concatenação dos resultados, sendo eles apresentados para o cliente



    # Utilizado para enviar o resultodo para o cliente.

    conn.sendall(bytes(str(result), 'utf8'))
```

Explicação linha a linha do arquivo **client.py**.

```
# Bibliotecas

import socket

import random

# Identificação do HOST E PORT do client

HOST = '127.0.0.1' # Identifica o nome do client

PORT = 5000 # Identifica a porta do client para comunicar com o servidor



# Lista com as possíveis jogadas que podem ser escolhidas pelo cliente.

opcoesJogadas = ['Pedra', 'Papel', 'Tesoura']



# O mecanismo de Socket foi criado para receber a conexão, onde na função passamos 2 argumentos, AF_INET que declara a família do protocolo;

# Se fosse um envio via Bluetooth por exemplo, seria: AF_BLUETOOTH, e o SOCKET_STREAM, indica que será TCP/IP.

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# A constante AF_INET faz parte de um grupo denominado famílias de endereços, ou address families, que constitui exatamente o primeiro parâmetro opcional do construtor socket.

# A AF_INET abrange os endereços do tipo IPv4, antigo padrão da Internet.



# Realiza a conexão com o servidor

sock.connect((HOST, PORT)) # Parênteses duplo pq o connect tem apenas um parâmetro.



while True: # O cliente pode realizar n comunicações com o servidor, ou até essa ligação ser encerrada.

    # O usuário escolhe o tipo de jogada que será feita, sendo 1 - Palpite aleatório e 2 o usuário informa um palpite, por fim caso ele informe 0(zero) a conexão é encerrada.

    opcao1 = int(input("\n* Escolha uma opção:\n 1- Palpite aleatório\n 2- Informar um palpite\n 0- Para encerrar.\n -> Opcao: "))
```

```
    # Caso o usuário escolha a opcao 1, então o palpite será aleatório. Essa aleatoriedade é dada de acordo com a lista opcoesJogadas.
    if (opcao1 == 1):
        mensagemEnvioClient = random.choice(opcoesJogadas) # Randomização da jogada de acordo com a lista opcoesJogadas.

    # Caso o usuário escolha a opcao 2, então um menu é aberto para escolhar o palpite.
    if (opcao1 == 2):
        opcao2 = int(input("\n* Ecolha o palpite:\n 1- Pedra\n 2- Papel\n 3- Tesoura\n -> Opcao: "))

        if (opcao2 == 1):
            mensagemEnvioClient = 'Pedra' # Caso o usuário escolha a opção 1, então a jogada será Pedra.
        elif (opcao2 == 2):
            mensagemEnvioClient = 'Papel' # Caso o usuário escolha a opção 1, então a jogada será Papel.
        elif (opcao2 == 3):
            mensagemEnvioClient = 'Tesoura' # Caso o usuário escolha a opção 1, então a jogada será Tesoura.

    # Caso o usuário informe 0(zero) no momento de escolher a jogada, então a conexão é encerrada.
    if (opcao1 == 0):
        print("\nConexão encerrada!\n")
        # Utilizado para fechar a conexão entre as duas aplicações.
        sock.close()
        break

    # Utilizado para fazer o envio de dados para o servidor.
    sock.sendall(str.encode(mensagemEnvioClient)) # Enviar mensagem para o servidor

    print("\n-> Palpite enviado pelo cliente: ", mensagemEnvioClient) # Apresenta o palpite escolhido pelo cliente
    # Aguarda o retorno do servidor, um dado enviado pela rede de até 1024 Bytes, a função 'recv' possui somente 1 argumento que é o tamanho do Buffer.
    data = sock.recv(1024) # Bytes
```

```
    # Decodifica a mensagem recebida pelo servidor, coloca todas as letras em maiúsculo e por último essa mensagem é apresentada.

    print('\n*** Resultado final do jogo: \n', data.decode().upper())
```

Obs.: Caso a visualização aqui não esteja legível, os arquivos possuem os mesmos comentários.