

Jocul Spanzuratoarea

UDP:
server:

Import biblioteca „socket” pentru a putea folosi functionalitatea de retea si „random” pentru a se putea alege un cuvant aleatoriu. Stabilesc adresa IP si portul serverului. Prin socket.SOCK_DGRAM arat ca folosesc UDP. In „word” stochez un cuvant aleatoriu si folosesc .lower() pentru a nu exista probleme mai tarziu la comparatii. La inceput „guessed_word” contine doar caracterul „ _ ” pentru fiecare litera din cuvantul respectiv. Setez numarul de incercari la 6. In while serverul asteapta date de la client prin server_socket. In „letter” stochez datele primite de la client. Daca „letter” se gaseste in „word” atunci actualizez „guessed_word” ca sa afiseze literele corect ghicite in cuvant. Daca „guessed_word” nu mai contine caractere „ _ ” inseamna ca toate literele au fost ghicite si clientul a castigat jocul si se afiseaza un mesaj. Daca „letter” nu se afla in „word” atunci se scade numarul de incercari. Daca „attempts” (numarul de incercari) ajunge la 0 atunci jocul se termina si se afiseaza un mesaj cu cuvantul corect. Actualizez „response” cu numarul de incercari ramase si cu starea curenta a cuvantului. Raspunsul este trimis clientului. Jocul continua pana cand clientul castiga sau a ramas fara incercari. La sfarsit se apeleaza main care contine logica principala a jocului.

client:

Stabilesc adresa IP si portul serverului. Serverul ruleaza pe acelasi localhost si portul este acela pe care serveul asteapta conexiuni. Prin socket.SOCK_DGRAM arat ca folosesc UDP. In while clientul solicita utilizatorului sa introduca o litera pentru a ghici cuvantul. Litera e stocata in „letter” si e citita prin input(). Clientul apoi trimite litera catre server prin client_socket.sendto(letter.encode(), (server_ip, server_port)). Literele sunt convertite in format binar si trimise la server. Prin client_socket.recvfrom(1024) clientul asteapta raspuns de la server. In „response” stochez raspunsul si il convertesc in sir de caractere ca sa poata fi afisat prin .decode(). Verific daca raspunsul contine „castigat” sau „pierdut”. Daca s-au gasit aceste cuvinte atunci jocul se termina prin folosirea lui break pentru a intrerupe while-ul. La sfarsit se inchide socket-ul.

```

paul@Catalin: ~
GNU nano 6.2 UDPserver.c
import socket
import random

def main():
    server_ip = '127.0.0.1'
    server_port = 12345

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    server_socket.bind((server_ip, server_port))

    words = ["masina", "calculator", "pisica", "raton"]
    word = random.choice(words).lower()
    guessed_word = ['_'] * len(word)
    attempts = 6

    while attempts > 0:
        data, client_address = server_socket.recvfrom(1024)
        letter = data.decode().lower()

        if letter in word:
            for i in range(len(word)):
                if word[i] == letter:
                    guessed_word[i] = letter

            if '_' not in guessed_word:
                response = "Ai castigat! Cuvantul era: " + word
                server_socket.sendto(response.encode(), client_address)
                break
        else:
            attempts -= 1

        if attempts == 0:
            response = "Ai pierdut! Cuvantul era: " + word
            server_socket.sendto(response.encode(), client_address)
            break

    response = "Incercari ramase: " + str(attempts) + "\n" + " ".join(guessed_word)
    server_socket.sendto(response.encode(), client_address)

if __name__ == "__main__":
    main()

```

```

paul@Catalin: ~
GNU nano 6.2 UDPclient.c
import socket

def main():
    server_ip = '127.0.0.1'
    server_port = 12345

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    while True:
        letter = input("Ghiceste o litera: ")
        client_socket.sendto(letter.encode(), (server_ip, server_port))
        response, server_address = client_socket.recvfrom(1024)
        response_str = response.decode()
        print(response_str)

        if "castigat" in response_str or "pierdut" in response_str:
            break

if __name__ == "__main__":
    main()

```

```
paul@Catalin:~$ python3 UDPServer.c
paul@Catalin:~$ python3 UDPServer.c
paul@Catalin:~$

paul@Catalin:~$ python3 UDPClient.c
Ghiceste o litera: e
Incerari ramase: 5
-----
Ghiceste o litera: t
Incerari ramase: 5
-----
Ghiceste o litera: i
Incerari ramase: 4
-----
Ghiceste o litera: o
Incerari ramase: 4
-----
Ghiceste o litera: p
Incerari ramase: 3
-----
Ghiceste o litera: a
Incerari ramase: 3
-----
Ghiceste o litera: s
Incerari ramase: 2
-----
Ghiceste o litera: u
Incerari ramase: 2
-----
Ghiceste o litera: n
Incerari ramase: 1
-----
Ghiceste o litera: p
Ai pierdut! Cuvantul era: calculator
paul@Catalin:~$ python3 UDPClient.c
Ghiceste o litera: a
Incerari ramase: 6
-----
Ghiceste o litera: r
Incerari ramase: 6
-----
Ghiceste o litera: t
Incerari ramase: 6
-----
Ghiceste o litera: o
Incerari ramase: 6
-----
Ghiceste o litera: n
Ai castigat! Cuvantul era: raton
paul@Catalin:~$
```

TCP:

server:

Import biblioteca „socket” pentru a putea folosi functionalitatea de retea si „random” pentru a se putea alege un cuvant aleatoriu. Stabilesc adresa IP si portul serverului. Prin socket.SOCK_STREAM arat ca folosesc TCP. Serverul se leaga de adresa IP si port prin .bind() si asteapta 2 conexiuni simultane prin server_socket.listen(2). In „word” stochez un cuvant aleatoriu si folosesc .lower() pentru a nu exista probleme mai tarziu la comparatii. La inceput „guessed_word” contine doar caracterul „_” pentru fiecare litera din cuvantul respectiv. Setez numarul de incercari la 6. Afisez un mesaj pt asteptarea clientilor sa se conecteze. Serverul accepta conexiune cu Client1 prin .accept(), iar client1_socket este socket-ul pt comunicarea cu clientul si client1_address este adresa clientului. Afisez si un mesaj pentru a indica faptul ca primul client s-a conectat. Initializez client1_turn cu True pentru a indica faptul ca e randul primului client. Serverul asteapta conexiunea cu Client2 in acelasi mod ca la Client1. Initializez client2_turn cu False pentru a indica faptul ca Client2 nu incepe jocul. Incepe bucla principala a jocului care se termina atunci cand una dintre conditiile de incheiere a jocului e satisfacuta. Serverul verifica randul fiecarui client de a ghici litere in cuvant. Daca e randul primului client atunci serverul asteapta o litera de la el prin client1_socket.recv(1024). Litera e decodata si convertita in litere mici. Serverul verifica daca litera primita de la client (data1) se gaseste in cuvant si daca da atunci se actualizeaza „guessed_word” cu litera corect ghicita si se verifica daca a fost ghicit cuvantul in intregime. Daca Client1 a ghicit un cuvant sau jocul s-a terminat atunci serverul trimite un mesaj catre ambii clienti ca sa anunte castigatorul si cuvantul corect. Urmeaza randul clientului 2. Serverul asteapta sa primeasca o litera de la el, verifica daca se potriveste in cuvant si actualizeaza „guessed_word”. Daca Client2 a ghicit cuvantul sau jocul s-a terminat atunci serverul trimite un mesaj cu castigatorul si cuvantul corect. Urmeaza randul clientului 1. Daca nu s-a terminat jocul se scade numarul de incercari disponibile. Daca nu mai sunt incercari disponibile atunci serverul trimite un mesaj ca sa anunte ca s-a terminat jocul cu egalitate. Actualizez „response” pentru a primi informatii despre numarul de incercari si starea cuvantului si il trimit catre ambii clienti. Se inchid socket-urile si afisez mesaj de final.

client1:

Import biblioteca „socket” pentru a putea folosi functionalitatea de retea. Stabilesc adresa IP si portul serverului. Prin socket.SOCK_STREAM arat ca folosesc TCP. Clientul se conecteaza la server. In while clientul solicita utilizatorului sa introduca o litera pentru a ghici cuvantul. Litera e stocata in „letter” si e citita prin input(). Clientul apoi trimite litera catre server prin client_socket.sendto(letter.encode(), (server_ip, server_port)). Literele sunt convertite in format binar si trimise la server. Prin client_socket.recvfrom(1024) clientul asteapta raspuns de la server. In „response” stochez raspunsul si il convertesc in sir de caractere ca sa poata fi afisat prin .decode(). Verific daca raspunsul contine „castigat” sau „pierdut”. Daca s-au gasit aceste cuvinte atunci jocul se termina prin folosirea lui break pentru a intrerupe while-ul. La sfarsit se inchide socket-ul.

client2: la fel ca si la client1

```
paul@Catalin: ~
GNU nano 6.2 TCPserver.c
import socket
import random

def main():
    server_ip = '127.0.0.1'
    server_port = 12345

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((server_ip, server_port))
    server_socket.listen(2) # Permite două conexiuni simultane

    words = ["masina", "pisica", "raton", "cartof"]
    word = random.choice(words).lower()
    guessed_word = ['_'] * len(word)
    attempts = 6

    print("Se asteapta conectarea clientilor...")

    client1_socket, client1_address = server_socket.accept()
    print("Clientul 1 s-a conectat.")
    client1_turn = True

    client2_socket, client2_address = server_socket.accept()
    print("Clientul 2 s-a conectat.")
    client2_turn = False

    while attempts > 0:
        if client1_turn:
            print("Clientul 1 joaca:")
            data1 = client1_socket.recv(1024).decode().lower()

            if data1 in word:
                for i in range(len(word)):
                    if word[i] == data1:
                        guessed_word[i] = data1

            if '_' not in guessed_word:
                response = "Clientul 1 a castigat! Cuvantul era: " + word
                client1_socket.send(response.encode())
                client2_socket.send(response.encode())
                break

            client1_turn = False
            client2_turn = True

        if client2_turn:
```

[Read 82 lines]

Help	Write Out	Where Is	Cut	Execute	Location	Undo
Exit	Read File	Replace	Paste	Justify	Go To Line	Redo

```
paul@Catalin: ~
GNU nano 6.2 TCPserver.c

        response = "Clientul 1 a castigat! Cuvantul era: " + word
        client1_socket.send(response.encode())
        client2_socket.send(response.encode())
        break

    client1_turn = False
    client2_turn = True

    if client2_turn:
        print("Clientul 2 joaca:")
        data2 = client2_socket.recv(1024).decode().lower()

        if data2 in word:
            for i in range(len(word)):
                if word[i] == data2:
                    guessed_word[i] = data2

            if '_' not in guessed_word:
                response = "Clientul 2 a castigat! Cuvantul era: " + word
                client1_socket.send(response.encode())
                client2_socket.send(response.encode())
                break

            client1_turn = True
            client2_turn = False

    attempts -= 1

    if attempts == 0:
        response = "Egalitate! Cuvantul era: " + word
        client1_socket.send(response.encode())
        client2_socket.send(response.encode())
        break

    response = "Incercari ramase: " + str(attempts) + "\n" + " ".join(guessed_word)
    client1_socket.send(response.encode())
    client2_socket.send(response.encode())

    print("Sfarsit.")
    client1_socket.close()
    client2_socket.close()

if __name__ == "__main__":
    main()

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^^ Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-E Redo
```



```
server_ip = '127.0.0.1'
```

```
server_port = 12345
```

```
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

```
server_socket.bind((server_ip, server_port))
```

```
words = ["masina", "calculator", "pisica", "raton"]
```

```
word = random.choice(words).lower()
```

```
guessed_word = ['_'] * len(word)
```

```
attempts = 6
```

```
while attempts > 0:
```

```
    data, client_address = server_socket.recvfrom(1024)
```

```
    letter = data.decode().lower()
```

```
    if letter in word:
```

```
        for i in range(len(word)):
```

```
            if word[i] == letter:
```

```
                guessed_word[i] = letter
```

```
    if '_' not in guessed_word:
```

```
        response = "Ai castigat! Cuvantul era: " + word
```

```
        server_socket.sendto(response.encode(), client_address)
```

```
        break
```

```
    else:
```

```
        attempts -= 1
```

```
if attempts == 0:
```

```
    response = "Ai pierdut! Cuvantul era: " + word
```

```
    server_socket.sendto(response.encode(), client_address)
```

```
    break
```



```
response = "Incerari ramase: " + str(attempts) + "\n" + " ".join(guessed_word)
server_socket.sendto(response.encode(), client_address)
```

```
if __name__ == "__main__":
    main()
```

UDPclient:

```
import socket
```

```
def main():
```

```
    server_ip = '127.0.0.1'
```

```
    server_port = 12345
```

```
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

```
    while True:
```

```
        letter = input("Ghiceste o litera: ")
```

```
        client_socket.sendto(letter.encode(), (server_ip, server_port))
```

```
        response, server_address = client_socket.recvfrom(1024)
```

```
        response_str = response.decode()
```

```
        print(response_str)
```

```
        if "castigat" in response_str or "pierdut" in response_str:
```

```
            break
```

```
if __name__ == "__main__":
```

```
main()
```

TCPserver:

```
import socket
```

```
import random
```

```
def main():
```

```
    server_ip = '127.0.0.1'
```

```
    server_port = 12345
```

```
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
    server_socket.bind((server_ip, server_port))
```

```
    server_socket.listen(2) # Permite două conexiuni simultane
```

```
    words = ["masina", "pisica", "raton", "cartof"]
```

```
    word = random.choice(words).lower()
```

```
    guessed_word = ['_'] * len(word)
```

```
    attempts = 6
```

```
    print("Se asteapta conectarea clientilor...")
```

```
    client1_socket, client1_address = server_socket.accept()
```

```
    print("Clientul 1 s-a conectat.")
```

```
    client1_turn = True
```

```
    client2_socket, client2_address = server_socket.accept()
```

```
    print("Clientul 2 s-a conectat.")
```

```
    client2_turn = False
```

```
while attempts > 0:
    if client1_turn:
        print("Clientul 1 joaca:")
        data1 = client1_socket.recv(1024).decode().lower()

        if data1 in word:
            for i in range(len(word)):
                if word[i] == data1:
                    guessed_word[i] = data1

            if '_' not in guessed_word:
                response = "Clientul 1 a castigat! Cuvantul era: " + word
                client1_socket.send(response.encode())
                client2_socket.send(response.encode())
                break

        client1_turn = False
        client2_turn = True

    if client2_turn:
        print("Clientul 2 joaca:")
        data2 = client2_socket.recv(1024).decode().lower()

        if data2 in word:
            for i in range(len(word)):
                if word[i] == data2:
                    guessed_word[i] = data2

            if '_' not in guessed_word:
                response = "Clientul 2 a castigat! Cuvantul era: " + word
                client1_socket.send(response.encode())
```

```
client2_socket.send(response.encode())
```

```
break
```

```
client1_turn = True
```

```
client2_turn = False
```

```
attempts -= 1
```

```
if attempts == 0:
```

```
    response = "Egalitate! Cuvantul era: " + word
```

```
    client1_socket.send(response.encode())
```

```
    client2_socket.send(response.encode())
```

```
    break
```

```
response = "Incerari ramase: " + str(attempts) + "\n" + " ".join(guessed_word)
```

```
client1_socket.send(response.encode())
```

```
client2_socket.send(response.encode())
```

```
print("Sfarsit.")
```

```
client1_socket.close()
```

```
client2_socket.close()
```

```
if __name__ == "__main__":
```

```
    main()
```

```
TCPclient1:
```

```
import socket
```

```
def main():
```

```
    server_ip = '127.0.0.1'
```

```
server_port = 12345
```

```
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
client_socket.connect((server_ip, server_port))
```

```
while True:
```

```
    letter = input("Clientul 1 - Ghiceste o litera: ")
```

```
    client_socket.send(letter.encode())
```

```
    response = client_socket.recv(1024).decode()
```

```
    print(response)
```

```
    if "castigat" in response or "pierdut" in response:
```

```
        break
```

```
client_socket.close()
```

```
if __name__ == "__main__":
```

```
    main()
```

```
TCPclient2:
```

```
import socket
```

```
def main():
```

```
    server_ip = '127.0.0.1'
```

```
    server_port = 12345
```

```
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
    client_socket.connect((server_ip, server_port))
```

```
while True:
```

```
    letter = input("Clientul 2 - Ghiceste o litera: ")
```

```
    client_socket.send(letter.encode())
```

```
    response = client_socket.recv(1024).decode()
```

```
    print(response)
```

```
    if "castigat" in response or "pierdut" in response:
```

```
        break
```

```
client_socket.close()
```

```
if __name__ == "__main__":
```

```
    main()
```