

# EXEC6-EDUARDO - CRIPTOGRAFIA E SOCKET

## CRIPTOGRAFIA

```
/home/edu/Documentos/repositories/criptografia/py3/edukeyPri.txt
• edu@fedora:~/Documentos/repositories/criptografia/py3$ py geraChaves.py

/home/edu/Documentos/repositories/criptografia/py3/geraChaves.py:9: SyntaxWarning: invalid escape sequence '\c'
  end = input('Endereco do arquivo (c:\chaves\): ')
\-----//
**Prj Banco de Dados Distribuidos**
\-----//
Gerador de chaves assimetricas
Digite as seguintes informacoes
Tamanho da chave: 512
Endereco do arquivo (c:\chaves\): /home/edu/Documentos/repositories/criptografia/py3/
Nome do arquivo: edukey
Chaves geradas com sucesso
/home/edu/Documentos/repositories/criptografia/py3/edukeyPub.txt
/home/edu/Documentos/repositories/criptografia/py3/edukeyPri.txt
```

- requisita os inputs de tamanho, endereço e nome das chaves
- gera as chaves
- cria o arquivo pub (chave publica)
- codifica o expoente e modulo da chave publica para o formato PEM
- cria o arquivo pri (chave privada)
- codifica o expoente e modulo da chave privada para o formato PEM
- resulta nas chaves publicas e privadas do usuario em formato txt

## CIFRAR MSG

```
• edu@fedora:~/Documentos/repositories/criptografia/py3$ py cifrarMsg.py
/home/edu/Documentos/repositories/criptografia/py3/cifrarMsg.py:8: SyntaxWarning: invalid escape sequence '\c'
  arqnomepub = input('Endereco da chave publica (c:\chaves\myPub.txt): ')
/home/edu/Documentos/repositories/criptografia/py3/cifrarMsg.py:10: SyntaxWarning: invalid escape sequence '\m'
  arqnomemsg = input('Endereco e nome da mensagem (c:\msg.txt): ')
\-----//
**Prj Banco de Dados Distribuidos**
\-----//
Cifrador de mensagens
Digite as seguintes informacoes
Endereco da chave publica (c:\chaves\myPub.txt): /home/edu/Documentos/repositories/criptografia/py3/edukeyPub.txt
Mensagem a ser cifrada: Ana linda
Endereco e nome da mensagem (c:\msg.txt): uuu.msg.txt
Mensagem cifrada no arquivo uuu.msg.txt
```

- requisita os inputs
- abre o arquivo com a biblioteca rsa
- junta e carrega a chave para decodificar a chave
- codifica a mensagem para PEM usando a chave
- cifra a msg e salva

## DECIFRAR MSG

```
/home/edu/Documentos/repositories/criptografia/py3/decifrarMsg.py:8: SyntaxWarning: invalid escape sequence '\c'
  arqnomepri = input('Endereco da chave privada (c:\chaves\myPri.txt): ')
/home/edu/Documentos/repositories/criptografia/py3/decifrarMsg.py:9: SyntaxWarning: invalid escape sequence '\m'
  arqnomemsg = input('Endereco e nome da mensagem a ser decifrada (c:\msg.txt): ')
\-----//
**Prj Banco de Dados Distribuidos**
\-----//
Decifrador de mensagens
Digite as seguintes informacoes
Endereco da chave privada (c:\chaves\myPri.txt): /home/edu/Documentos/repositories/criptografia/py3/edukeyPri.txt
Endereco e nome da mensagem a ser decifrada (c:\msg.txt): /home/edu/Documentos/repositories/criptografia/py3/msg.txt
Mensagem decifrada: EduGostoso
○ edu@fedora:~/Documentos/repositories/criptografia/py3$
```

- lê o arquivo da mensagem
- carrega a chave privada
- decodifica para o formato PEM usando chave privada
- abre o arquivo com a msg
- carrega a mensagem cifrada e logo a após decifra

## SOCKET

### SERVER\_UDP

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL  GITLENS
○ edu@fedora:~/Documentos/repositories/socket$ cd udp/
○ edu@fedora:~/Documentos/repositories/socket/udp$ python3 server_udp.py
('127.0.0.1', 54661) b'Teste S2'
█
```

- passa o host, que no caso é vazio indicando que escutara toda as interfaces
- passa a porta que vai escutar
- configura para usar ipv4, associa o socket ao endereço e porta especificados
- loop principal
  - aguarda mensagens 1024 bytes
  - exibe o endereço do client e mensagem recebida
- fecha a conexão

### CLIENTE\_UDP

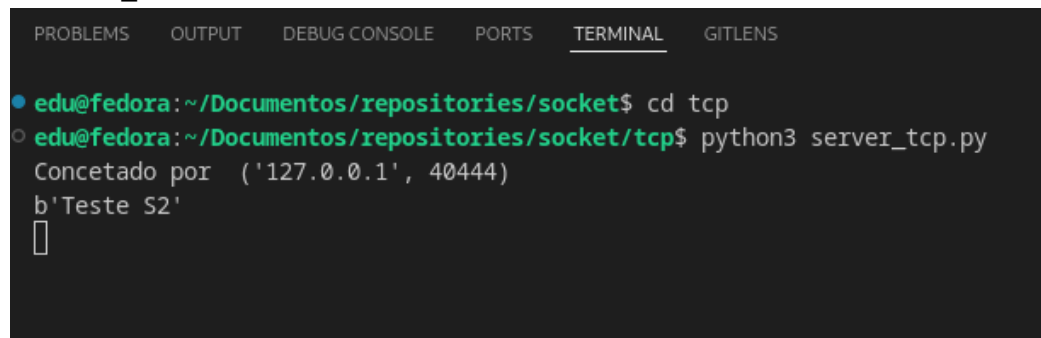
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL  GITLENS
○ edu@fedora:~/Documentos/repositories/socket/udp$ python3 client_udp.py
Para sair use CTRL+X

Teste S2
█
```

- importa a configuração inicial de server e port

- cria o socket UDP
- loop que faz a comunicação
  - envia a mensagem para o servidor
  - converte a string para bytes

## SERVER\_TCP



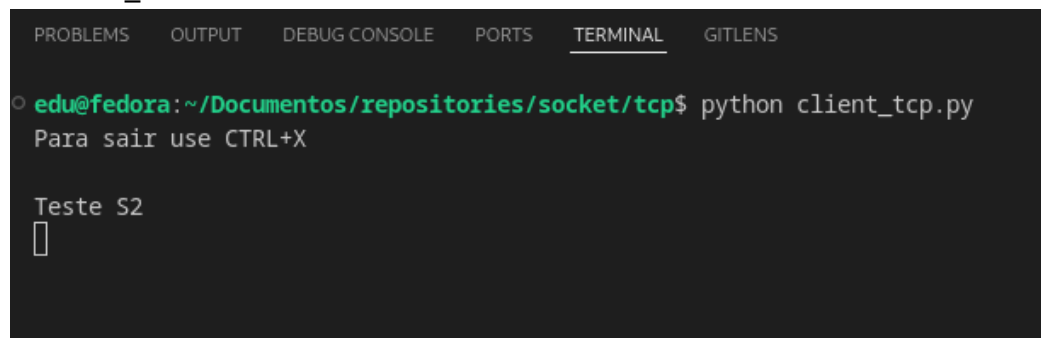
```

PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL  GITLENS
• edu@fedora:~/Documentos/repositories/socket$ cd tcp
○ edu@fedora:~/Documentos/repositories/socket/tcp$ python3 server_tcp.py
Concetado por ('127.0.0.1', 40444)
b'Teste S2'

```

- passa host e porta
- cria o socket TCP
  - ipv4
  - tipo TCP
  - passa a origem
- loop conexão
  - retorna um novo socket e endereço do client
  - imprime informações do client
- loop recebimento de dados
  - recebe dados do client ate 1024 bytes
  - retorna vazio se a conexão for fechada
  - imprime as mensagens recebidas

## CLIENTE\_TCP



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL  GITLENS
○ edu@fedora:~/Documentos/repositories/socket/tcp$ python client_tcp.py
Para sair use CTRL+X

Teste S2

```

- cria o socket TCP, usando ipv4, conecta conexão com servidor especifico
- loop
  - envia as mensagens
  - requisita a mensagem pro usuario
  - envia a mensagem para o servidor
  - converte a string para bytes
- encerra a conexão

## SERVER\_THREAD\_TCP

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL  GITLENS

• edu@fedora:~/Documentos/repositories/socket$ cd tcp_thread/
○ edu@fedora:~/Documentos/repositories/socket/tcp_thread$ python3 server_thread_tcp.py
Concetado por ('127.0.0.1', 57272)
Concetado por ('127.0.0.1', 45078)
b'Teste 1 S2'
b'Teste 2 S2'
█
```

- recebe o socket de conexão e o endereço do client
- permanece em loop recebendo mensagens
- se receber mensagens vazia, encerra a conexão

## CLIENT\_THREAD\_TCP

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL  GITLENS

• edu@fedora:~/Documentos/repositories/socket$ cd tcp_thread/
○ edu@fedora:~/Documentos/repositories/socket/tcp_thread$ python client_tcp.py
Para sair use CTRL+X

Teste 1 S2
█
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS  TERMINAL  GITLENS

• edu@fedora:~/Documentos/repositories/socket$ cd tcp_thread/
○ edu@fedora:~/Documentos/repositories/socket/tcp_thread$ python client_tcp.py
Para sair use CTRL+X

Teste 2 S2
█
```

- cria o socket TCP, usando ipv4, conecta conexão com servidor específico
- loop
  - envia as mensagens
  - requisita a mensagem pro usuário
  - envia a mensagem para o servidor
  - converte a string para bytes
- encerra a conexão