

# Persistência de Arquivos: Propriedades e zip

QXD0099 - Desenvolvimento de Software para Persistência

**Universidade Federal do Ceará - *Campus* Quixadá**

Prof. Francisco Victor da Silva Pinheiro  
victorpinheiro@ufc.br



# Agenda

- Arquivos de propriedades
- Manipulação de grandes arquivos
- Checksum / Hash de grandes arquivos
- Armazenando vários arquivos em um só arquivo (arquivamento)
- Compressão de arquivos
- Encriptação de arquivos
- Leitura de arquivo ZIP

# Arquivos de propriedades

- Um mapa importante é a tradicional classe Properties, que mapeia strings e é muito utilizada para a configuração de aplicações.
- A Properties possui, também, métodos para ler e gravar o mapeamento com base em um arquivo texto, facilitando muito a sua persistência.

```
1 user.name = angelica
2 user.mail = angelica@email.com
3 user.url = http://angelica.com.br
4 user.pass = 1##5932cd
5
6 book.title = Title
7 book.author = Author
8 book.edition = Edition
9 book.published = Published By
10 book.date = Published on
11 book.code = ISBN/ISNN
12 book.read = Read More about
```



# Arquivos de propriedades

```
# Usando um dicionário para armazenar as propriedades
config = {
    "database.login": "scott",
    "database.password": "tiger",
    "database.url": "jdbc:mysql://localhost/teste"
}

login = config.get("database.login")
password = config.get("database.password")
url = config.get("database.url")

# Exemplo de como poderia ser a conexão com um banco de dados em Python usando pymysql ou mysql.connector
import mysql.connector

# Estabelece a conexão usando os valores de login, password e url
connection = mysql.connector.connect(
    host="localhost", # Retirando a parte do 'jdbc:mysql://localhost/teste'
    database="teste",
    user=login,
    password=password
)
```

# Arquivos de propriedades - gravação

```
import configparser

# Cria um objeto ConfigParser, equivalente ao Properties em Java
config = configparser.ConfigParser()

# Adiciona as propriedades (chave-valor)
config['DEFAULT'] = {
    'database': 'localhost',
    'dbuser': 'mkyong',
    'dbpassword': 'password'
}

# Tenta gravar as propriedades no arquivo "config.ini"
with open('config.ini', 'w') as configfile:
    config.write(configfile)
```

- **configparser.ConfigParser():** `ConfigParser` é uma biblioteca nativa do Python que lida com arquivos de configuração no formato `.ini`, semelhante ao arquivo `Properties` em Java.
- **config['DEFAULT']:** Define as propriedades no dicionário de configuração. No exemplo, as propriedades estão na seção `DEFAULT`, o que permite que sejam acessadas em qualquer parte do arquivo.
- **config.write(configfile):** Grava o conteúdo do dicionário de configurações no arquivo `config.ini`.

# Arquivos de propriedades - leitura

```
import configparser

# Cria o objeto ConfigParser
config = configparser.ConfigParser()

# Lê o arquivo de propriedades (config.ini)
config.read('config.ini')

# Obtém os valores das propriedades
database = config['DEFAULT'].get('database')
dbuser = config['DEFAULT'].get('dbuser')
dbpassword = config['DEFAULT'].get('dbpassword')

# Imprime os valores
print(f"Database: {database}")
print(f"DB User: {dbuser}")
print(f"DB Password: {dbpassword}")
```

- **`config.read('config.ini')`**: Lê o arquivo de configuração `config.ini` (equivalente ao `config.properties` em Java).
- **`config['DEFAULT'].get('database')`**: Obtém o valor da chave `database` na seção `DEFAULT`. Em Python, usamos o método `get()` para buscar o valor associado a uma chave.
- **`print()`**: Imprime os valores das propriedades, equivalente ao `System.out.println()` do Java.
-

# Manipulação de grandes arquivos

- `wc -l big_file.csv`: Conta o número de linhas no arquivo `big_file.csv`.
- `head -n 3 big_file.csv`: Exibe as primeiras 3 linhas do arquivo `big_file.csv`.
- `tail -n 3 big_file.csv`: Exibe as últimas 3 linhas do arquivo `big_file.csv`.
- `cat big_file.csv | less`: Exibe o conteúdo do arquivo `big_file.csv` paginado com o comando `less`, útil para navegar em arquivos muito grandes.
- `cat big_file.csv | grep "52.2479 21.0191"`: Procura e exibe as linhas no arquivo `big_file.csv` que contêm o texto "52.2479 21.0191".
- `sed "s/ /;/g" teste.txt`: Substitui todos os espaços ( ) por ponto e vírgula (;) no arquivo `teste.txt`. O comando `sed` é usado para substituições de texto.
- `ls -l | tr -s ' ' | cut -d " " -f9`: Lista os arquivos no diretório, compacta os espaços (`tr -s ' '` remove espaços extras) e depois usa `cut` para exibir apenas o nome dos arquivos (campo 9).

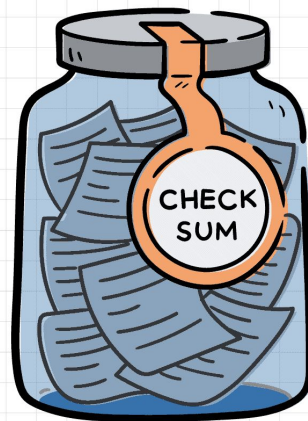
# Manipulação de grandes arquivos

- **ls -l**: Lista os arquivos e diretórios, um por linha.
- **awk '{ print \$4 ", " \$5 }' teste.txt**: Exibe o 4º e o 5º campo de cada linha do arquivo **teste.txt**, separando-os por vírgula. O **awk** é usado para manipulação de texto baseado em colunas.
- **awk -F "," '{ print \$4 ", " \$5 }' teste.csv**: Igual ao comando anterior, mas define o delimitador como vírgula (**-F ","**) e aplica a um arquivo CSV.
- **du -d1 | sort -n | cut -f2 | xargs du -hs**: Calcula o uso de disco para subdiretórios no diretório atual, ordena pelo tamanho, e depois exibe de forma legível (**-h** para "human-readable").
- **hexdump arquivo.txt**: Exibe o conteúdo do arquivo **arquivo.txt** no formato hexadecimal.
- **hexdump -C arquivo.txt**: Exibe o conteúdo do arquivo **arquivo.txt** no formato hexadecimal e caracteres ASCII, com a saída formatada.
- **hd arquivo.txt**: Uma alternativa mais curta para o comando **hexdump**, que também exibe o arquivo em formato hexadecimal.



# Checksum / Hash de grandes arquivos

- Checksum
  - `cksum teste*.*`
- MD5
  - `md5sum teste*.* > md5.txt`
  - `md5sum -c md5.txt`
- SHA1
  - `sha1sum teste*.* > sha1.txt`
  - `sha1sum -c sha1.txt`



# Armazenando vários arquivos em um só arquivo (arquivamento)

- tar
  - tar cfv teste.tar teste\*.\*
    - c – create
    - f – file
    - v - verbose



# Compressão de arquivos

- **Sem perda**

- zip, rar, 7z, gzip (um só arquivo), bzip2 (um só arquivo),...
- zip teste.zip teste\*.\*
- gzip teste.txt
- bzip2 teste.txt
- tar c teste\*.\* | gzip > teste.tar.gz
- tar cfvz teste.tar.gz teste\*.\*
- z - gzip
- tar cfvj teste.tar.bz teste\*.\*
- j – bzip2

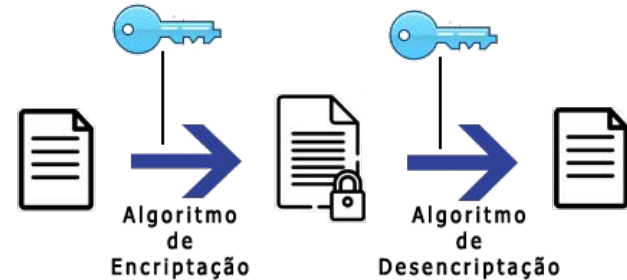


- **Com perda**

- jpg, mp3, mp4, ...

# Encriptação de arquivos

- Encriptar:
  - `gpg -c teste.txt`
- Decriptar:
  - `gpg teste.txt.gpg`
- <http://www.techrepublic.com/article/how-to-easily-encryptdecrypt-a-file-in-linux-with-gpg/>
- <http://irtfweb.ifa.hawaii.edu/~lockhart/gpg/>
- <https://help.ubuntu.com/community/GnuPrivacyGuardHowto>



# Leitura de arquivo ZIP

```
import zipfile

# Caminho do arquivo ZIP
zip_file_path = "./teste.zip"

# Abre o arquivo ZIP para leitura
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    # Lista os arquivos dentro do ZIP
    for file_name in zip_ref.namelist():
        # Abre o arquivo dentro do ZIP
        with zip_ref.open(file_name) as file:
            # Lê o conteúdo do arquivo
            for line in file:
                # Converte de bytes para string e remove quebras de
                linha

                print(line.decode('utf-8').strip())
```

- **zipfile.ZipFile()**: Abre o arquivo ZIP para leitura.
- **zip\_ref.namelist()**: Lista os nomes dos arquivos dentro do arquivo ZIP.
- **zip\_ref.open(file\_name)**: Abre um arquivo específico dentro do ZIP para leitura.
- **file.read()** ou **iteração linha por linha**: Lê o conteúdo do arquivo compactado. Em Python, as linhas são lidas como bytes, então usamos **.decode('utf-8')** para convertê-las em strings.
- **.strip()**: Remove qualquer quebra de linha ou espaços adicionais.

# Leitura de arquivo ZIP

```
import zipfile

# Caminho do arquivo ZIP
zip_file_path = "./tb1.zip"

# Abre o arquivo ZIP
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    # Pega o nome do primeiro arquivo dentro do ZIP
    file_name = zip_ref.namelist()[0]

    # Abre o arquivo dentro do ZIP
    with zip_ref.open(file_name) as file:
        # Usa o scanner (como o Scanner no Java, lendo linha por linha)
        for line in file:
            print(line.decode('utf-8').strip()) # Converte bytes para
string e remove quebras de linha
```

- **zipfile.ZipFile(zip\_file\_path, 'r')**: Abre o arquivo ZIP para leitura.
- **zip\_ref.namelist()[0]**: Obtém o nome do primeiro arquivo dentro do ZIP.
- **zip\_ref.open(file\_name)**: Abre o arquivo dentro do ZIP para leitura.
- **for line in file::** Lê o conteúdo do arquivo dentro do ZIP linha por linha.
- **.decode('utf-8').strip()**: Converte cada linha (que é lida como bytes) para uma string e remove quebras de linha ou espaços em branco adicionais.

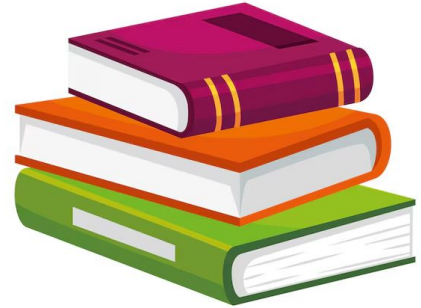
# Bibliografia Básica

- SADALAGE, P. J. E FOWLER, M. NoSQL Essencial. Editora Novatec, São Paulo, 2013.
- REDMOND, E.; WILSON, J. R. Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement. 1ª edição, 2012. The Pragmatic Programmers.
- ULLMAN, J.D.; WIDOW, J. First Course in Database Systems. 3a edição, 2007. Prentice Hall.
- HAMBRICK, G. et al. Persistence in the Enterprise: A Guide to Persistence Technologies; 1ª edição, 2008. IBM Press.
- ELMASRI, R.; NAVATHE, S. B. Sistemas de banco de dados. 4ª edição, 2009. Pearson/Addison-Wesley.



# Bibliografia Complementar

- WHITE, Tom. Hadoop: the definitive guide. California: O'Reilly, 2009. xix, 501 p. ISBN 9780596521974 (broch.).
- AMBLER, S.W., SADALAGE, P.J. Refactoring Databases: Evolutionary Database Design. 1a edição, 2011. Addison Wesley.
- SILBERSCHATZ, A.; SUDARSHAN, S. Sistema de banco de dados. 2006. Campus.
- LYNN, B. Use a cabeça! SQL. 1ª edição, 2008. ALTA BOOKS.
- SMITH, Ben. JSON básico: conheça o formato de dados preferido da web. São Paulo: Novatec, 2015. 400 p. ISBN 9788575224366 (broch.).
- HITZLER, P., KRÖTZSCH, M., and RUDOLPH, S. (2009). Foundations of Semantic Web Technologies. Chapman & Hall/CRC.
- ANTONIOU, G. and HARMELEN, F. (2008). A Semantic Web Primer. Second Edition, Cambridge, MIT Press, Massachusetts.
- HEATH, T. and BIZER, C. (2011). Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool, 1st edition.





# Obrigado!

## Dúvidas?



**Universidade Federal do Ceará - *Campus* Quixadá**

**Prof. Francisco Victor da Silva Pinheiro**  
victorpinheiro@ufc.br

