ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES II

Arquivos

Arquivos

Um arquivo é uma sequência de bytes armazenada em memória secundária.

Existem dois tipos de arquivos:

- Arquivos texto (.txt, .html, .py, etc.)

- Arquivos binários (.exe, .mp3, .jpg, etc.)

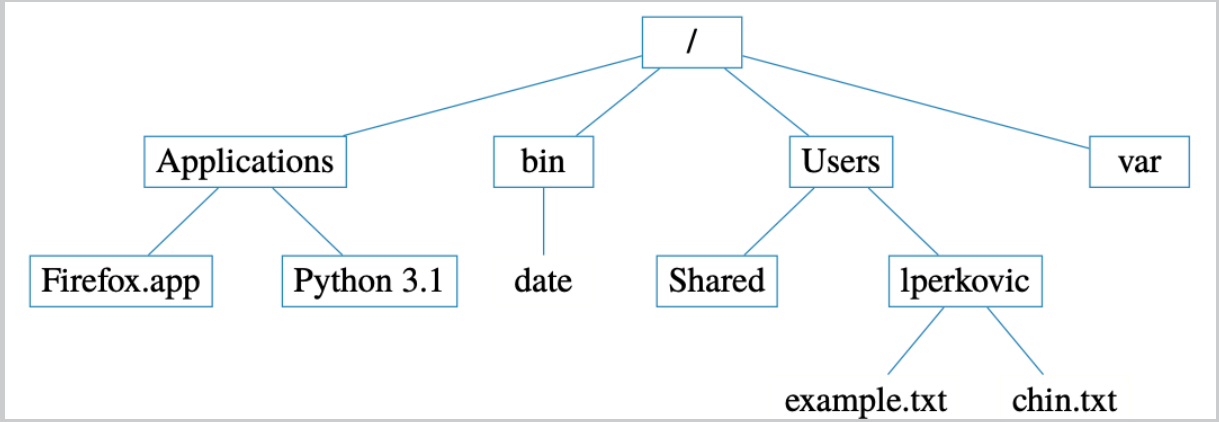
Sistema de arquivos

Componente do computador que organiza os arquivos e provê meios para criá-los, acessá-los e modificá-los.

Fornece uma visão uniforme dos arquivos, embora possam estar armazenados em diferentes dispositivos.

Arquivos são agrupados em pastas ou diretórios.

Sistema de arquivos



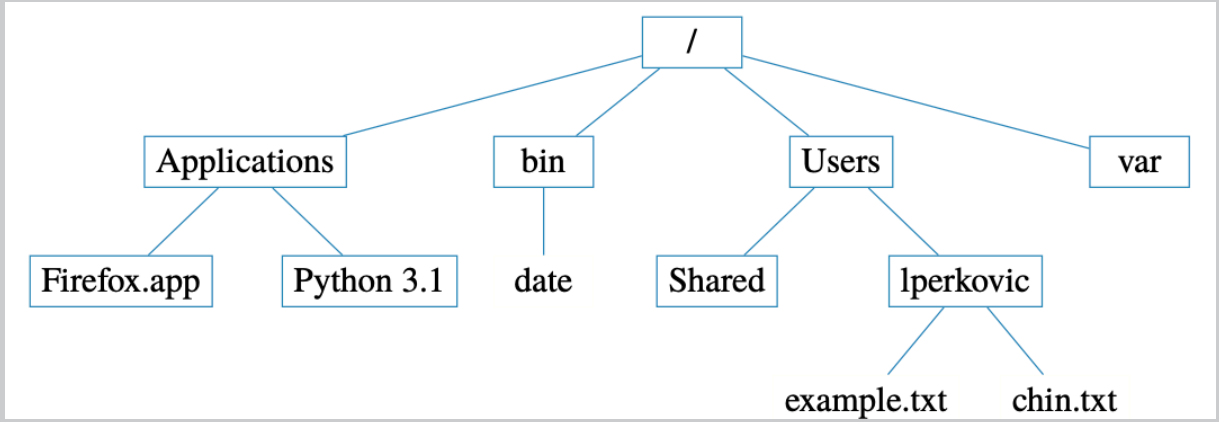
Fonte: Perkovic, 2015

Para localizar um arquivo, é necessário especificar seu nome e sua (sub)pasta.

Caminho absoluto para acessar example.txt:

/Users/lperkovic/example.txt

Sistema de arquivos



Fonte: Perkovic, 2015

A partir de um diretório, é possível acessar outra pasta por meio do caminho relativo.

Acessar example.txt a partir de Shared:

../Users/lperkovic/example.txt

Acessando arquivos

Processar um arquivo consiste em três passos:

1. **Abrir um arquivo para leitura ou escrita**
2. **Ler os dados do arquivo ou escrever nele**
3. **Fechar o arquivo**

Função open():

```
>>> open('example.txt', 'r')
```

Primeiro argumento é o caminho (absoluto ou relativo) para o arquivo, e o segundo é o modo de abertura.

Modo de abertura

Especifica a interação que será realizada com o arquivo.

Mode	Description
`r`	Reading mode (default)
`w`	Writing mode; if the file already exists, its content is wiped out
`a`	Append mode; writes are appended to the end of the file
`r+`	Reading and writing mode (beyond the scope of this book)
`t`	Text mode (default)
`b`	Binary mode

Fonte: Perkovic, 2015

No caso de arquivos binários, a sequência de bytes não é codificada ou decodificada usando alguma codificação (ASC-II, UTF-8, etc.)

Leitura ou escrita

Funções para leitura ou escrita de arquivos:

Method Usage	Explanation
`infile.read(n)`	Read n characters from the file `infile` or until the end of the file is reached, and return characters read as a string
`infile.read()`	Read characters from file `infile` until the end of the file and return characters read as a string
`infile.readline()`	Read file `infile` until (and including) the new line character or until end of file, whichever is first, and return characters read as a string
`infile.readlines()`	Read file `infile` until the end of the file and return the characters read as a list lines
`outfile.write(s)`	Write string `s` to file `outfile`
`file.close()`	Close the file

Fonte: Perkovic, 2015

Fechando um arquivo

Para fechar o arquivo, usamos o método close().

Fechar um arquivo significa avisar o sistema para liberar os recursos/informações sobre o arquivo aberto.

Exemplo

```
def readFile(filename):
    infile = open(filename, 'r')
    content = infile.read()
    infile.close()
    wordList = content.split()
    print(wordList)
    return len(wordList), len(content)

n_words, n_chars = readFile('teste.txt')
```

Escrevendo em um arquivo

Para escrever dados em um arquivo, precisamos abri-lo em modo escrita:

```
>>> outfile = open('teste.txt', 'w')
```

Usamos a função write(), que escreve strings no arquivo aberto, retornando o número de caracteres escritos.

```
>>> outfile.write('Olá classe!\n')
```

Escrevendo em um arquivo

Para escrever outros dados que não são string, devemos convertê-los ou usar o método format():

```
>>> idade = 30
>>> outfile.write('Sua idade é '+str(idade)+' anos.\n')
>>> outfile.write('Sua idade é {} anos.\n'.format(idade))
```

ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES II

Arquivos