Programação I / Introdução à Programação

Capítulo 7, "Files"

João Pedro Pedroso

2024/2025

Última aula

- Ordenação
- Recursão
- Séries de Fibonacci
- Listas recursivas

Hoje:

- Ler e escrever informação em ficheiros

Programas e informação

- Quando um programa está a ser executado: informação é guardada em *random access memory (RAM)*
  - RAM é rápida, mas é volátil: quando o programa termina, os dados desaparecem
  - para se guardar a informação tem de se usar um suporte *não volátil:*
    - disco rígido, pen USB, CD
    - informação é guardada em ficheiros com nomes
    - ficheiros permitem a programas guardarem informação para futuras execuções
- Funcionamento: como um bloco de notas:
  - abrir/fechar
  - modo: leitura/escrita
  - *localização* tem de ser conhecida

Ficheiros

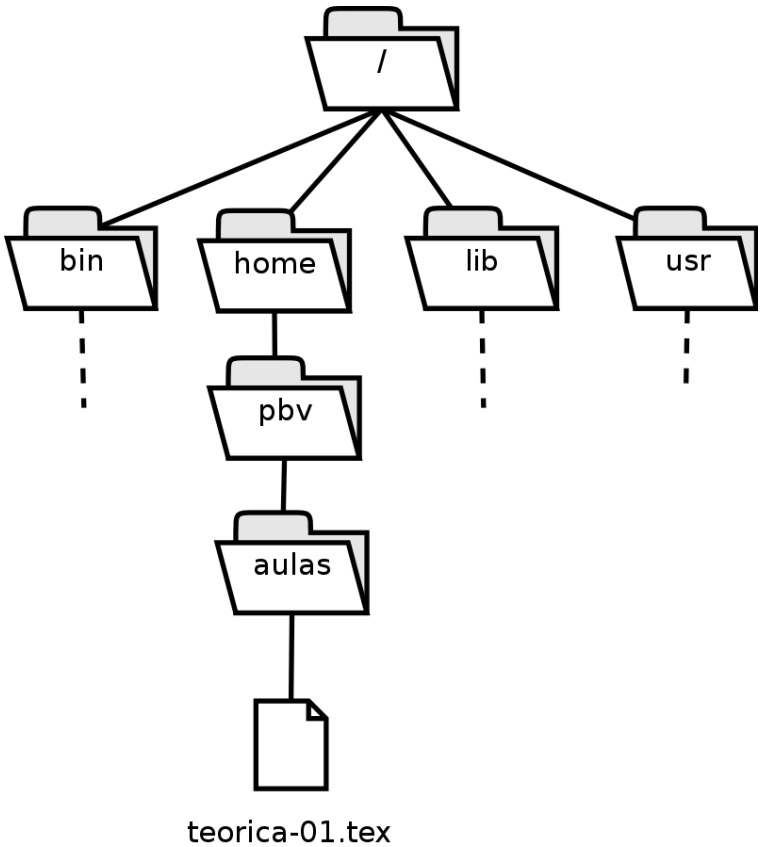
- Utilização de dados (até agora):
  - codificados diretamente no programa;
  - introduzidos pelo utilizador.
- Ficheiros:
  - imagens, textos, música, ...
  - ficheiros de texto $\rightarrow$ dados guardados são carateres

Ficheiros: caminho (*path*)

- Para abrir um ficheiro tem de se fornecer ao Python a localização do ficheiro no disco;
- Isso é feito através do *caminho (path)* para o ficheiro
- Exemplos (*caminho absoluto para ficheiro* `hello.txt`):
  - Mac: `/Users/yourname/hello.txt`
  - Windows: `C:\Users\yourname\My Documents\hello.txt`
  - Linux: `/home/yourname/hello.txt`
- (Ver primeiras aulas)

Organização de ficheiros (sistemas do tipo Unix)

- identificados por nomes
- estruturados em diretórios hierárquicos e.g. `/home/pbv/aulas/teorica-01.tex`
- permissões associadas a cada ficheiro: leitura, escrita, execução

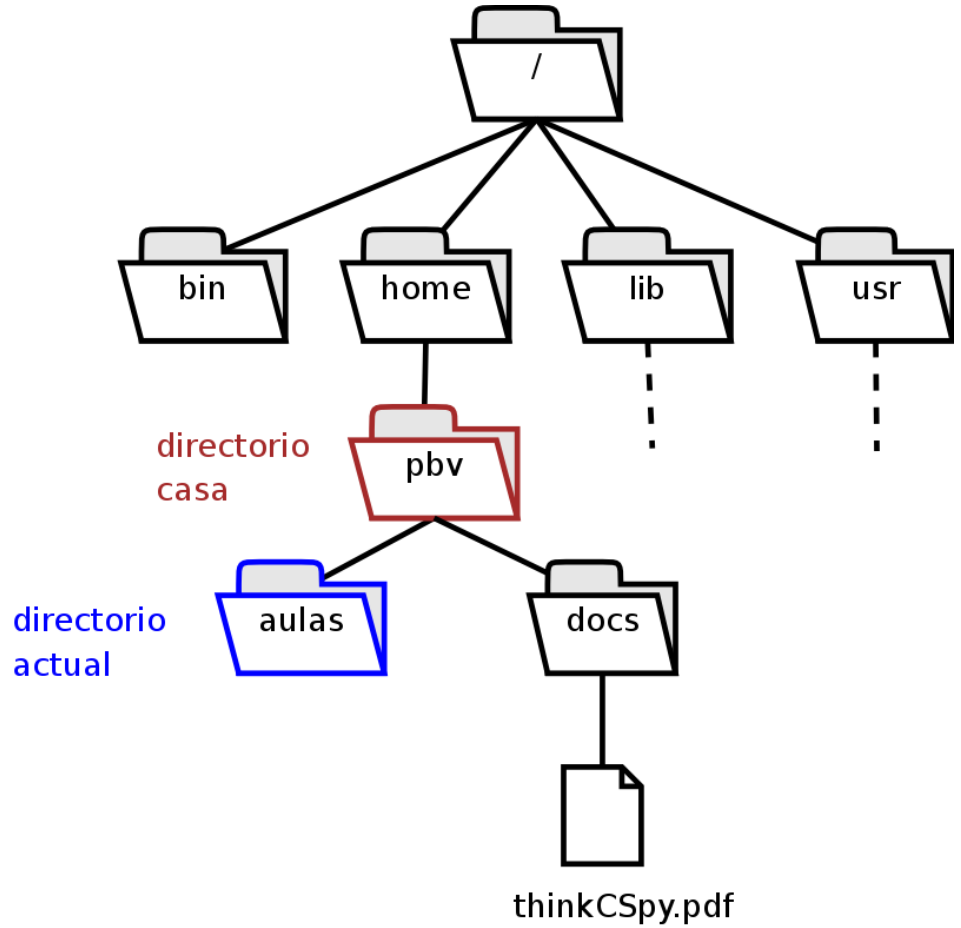


Diretório raiz

- Em sistemas unix e semelhantes, a raiz é o primeiro diretório no sistema de ficheiros
- Denotado pelo símbolo / (barra)
- Todos os ficheiros no sistema são *ramos* da raiz na hierarquia de ficheiros

Caminhos absolutos e relativos

- `/` raiz (root directory)
- `.` diretório atual
  na figura: `aulas`
- `..` diretório pai
- `~` diretório casa
  na figura: `pbv`



- Caminho para o ficheiro `thinkCSpy.pdf`:
  - absoluto: `/home/pbv/docs/thinkCSpy.pdf`
  - relativo ao diretório atual `aulas`: `../docs/thinkCSpy.pdf`
  - relativo ao diretório casa: `~/docs/thinkCSpy.pdf`

Caminho (*path*) no computador

Para localizarmos um ficheiro podemos especificar:

- caminho relativo desde o ponto onde o programa está a ser executado
  - Ex: ficheiro `hello.txt`
  - Ex: ficheiro `../hello.txt`
- caminho absoluto (relativa à *raiz* do sistema de ficheiros)
  - Ex: ficheiro `/home/jpp/P-I/hello.txt`

Regra

Regra a recordar:

- se o ficheiro e o programa Python estão no mesmo diretório, podemos usar apenas o nome do ficheiro:

```
open("myfile.txt","r")
```

- se o ficheiro e o programa Python estão em diretórios diferentes, tem de se indicar o caminho (absoluto ou relativo) até ao ficheiro:

```
open("/Users/jpp/P-I/myfile.txt", "r")
open("../P-I/myfile.txt", "r")
```

Ficheiros em Python

- Processo de utilização de ficheiros em Python:
  - abrir ficheiro → criar objeto Python para aceder ao ficheiro *file handle*
  - ler ou escrever dados;

```
1  with open("test.txt", "w") as f:
2      f.write("My first file written from Python\n")
3      f.write("---------------------------------\n")
4      f.write("Hello, world!\n")
```

- Depois disso, no terminal:

```
$ cat test.txt
My first file written from Python
---------------------------------
Hello, world!
```

- `f` → *file handle* (ou, simplificando, file/ficheiro)
- segundo argumento de open → modo de abertura
  - `"r"` (reading) → leitura (valor por omissão)
  - `"w"` (writing) → escrita
  - `"a"` (appending) → escrita adicionada no final do ficheiro

Leitura: percorrer um ficheiro

- Leitura do ficheiro que acabamos de criar:

```
1  with open("test.txt", "r") as f:
2      for line in f:
3          # Do something with the line we just read. Here we just print it.
4          print(line, end="")
```

- Notas:
  - cada linha lida é uma *string*, que inclui \n no final
  - se tentarmos ler um ficheiro que não existe → erro

```
>>> f = open("wharrah.txt", "r")
FileNotFoundError: [Errno 2] No such file or directory: "wharrah.txt"
```

Outras formas de ler

- O conteúdo de todo o ficheiro numa lista de *strings*:

```
1  with open("friends.txt", "r") as input_file:
2      all_lines = input_file.readlines()
3  all_lines.sort()
4  with open("sortedfriends.txt", "w") as output_file:
5      for line in all_lines:
6          outut_file.write(line)
```

- O conteúdo de todo o ficheiro numa só *string*:

```
1  with open("somefile.txt") as f:
2      content = f.read()
3  words = content.split()
4  print("There are {} words in the file.".format(len(words)))
```

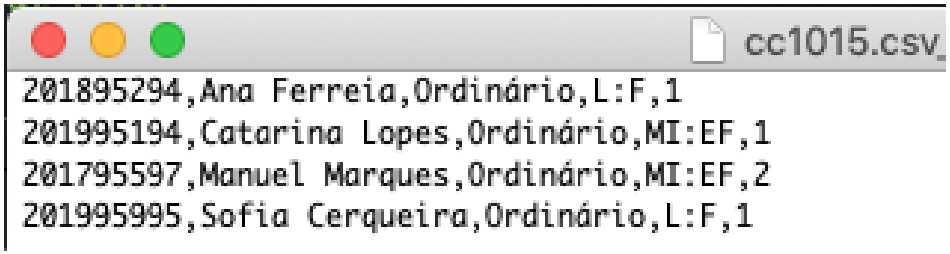
(O que fazem estes programas?)

Exemplo: remoção de linhas que começam por

```
def filter(oldfile, newfile):
    with open(oldfile, "r") as infile, open(newfile, "w") as outfile:
        for line in infile:
            # Put any processing logic here
            if line[0] != "#":
                outfile.write(line)
```

Ficheiros CSV

- *Comma-separated values:*
  - ficheiros de texto em que se usa *vírgula* para separar valores
  - formato "standard" para comunicar dados de folhas de cálculo



- Em Python: módulo `csv`
- Exemplo: leitura de um ficheiro CSV

```
import csv
with open("cc1015.csv", "r") as f:
    reader = csv.reader(f)
    for row in reader:
        print(row)
```

- Output:

```
['274595294', 'Ana Ferreira', 'Ordinário', 'L:F', '1']
['274595194', 'Catarina Lopes', 'Ordinário', 'MI:EF', '1']
['274595597', 'Manuel Marques', 'Ordinário', 'MI:EF', '2']
```

Ficheiros CSV: escrita

- Exemplo:

```
import csv
data = [
    ['274595294', 'Ana Ferreira', 'Ordinário', 'L:F', '1'],
    ['274595194', 'Catarina Lopes', 'Ordinário', 'MI:EF', '1'],
    ['274595597', 'Manuel Marques', 'Ordinário', 'MI:EF', '2'],
    ['274595995', 'Sofia Cerqueira', 'Ordinário', 'L:F', '1'],
]

with open("test.csv", "w") as f:
    writer = csv.writer(f, delimiter=",")
    writer.writerows(data)
```

- Output: ficheiro `test.csv`

```
274595294,Ana Ferreira,Ordinário,L:F,1
274595194,Catarina Lopes,Ordinário,MI:EF,1
274595597,Manuel Marques,Ordinário,MI:EF,2
274595995,Sofia Cerqueira,Ordinário,L:F,1
```

Ficheiros guardados na Internet

- Exemplo:

```
from urllib.request import urlopen
url = "http://www.gutenberg.org/cache/epub/3333/pg3333.txt"
with urlopen(url) as response:
    for line in response:
        print(line.decode(), end="")
```

- Output:

```
The Project Gutenberg EBook of Os Lusíadas, by Luís Vaz de Camõ

This eBook is for the use of anyone anywhere at no cost and wit
[...]
Canto Primeiro

1
As armas e os barões assinalados,
Que da ocidental praia Lusitana,
Por mares nunca de antes navegados,
```

Pickle: guardar objetos Python em ficheiros

- Guardar objetos Python:

```
import pickle
data = {
    'a': [1, 2.0, 3, 4+6j],
    'b': ("character string", b"byte string"),
    'c': {None, True, False}
}
with open("data.pickle", "wb") as f:
    pickle.dump(data, f)
```

- Ler os objetos guardados:

```
import pickle
with open("data.pickle", "rb") as f:
    data = pickle.load(f)
```

- Notas:
  - escrita no modo `"wb"` → *write binary*
  - leitura no modo `"rb"` → *read binary*

O formato JSON

- JSON: *JavaScript Object Notation*
  - formato popular para trocar dados
  - permite trocar dados com outras linguagens de programação
- Python: módulo `json`
  - permite guardar grande parte de dados Python em ficheiros
  - representa-os em *strings* → *serializing/deserializing*
- Converter objetos Python para *strings* JSON:

```
>>> import json
>>> json.dumps([1, "simple", "list"])
'[1, "simple", "list"]'
```

- Guardar objetos Python em ficheiros:

```
1  import json
2  with open("test.json","w") as f:
3       json.dump([1, 'simple', 'list'],f)
```

- Ler os objetos guardados:

```
>>> with open("test.json","r") as f:
        x = json.load(f)
>>> x
```

Ficheiros e o editor Pyzo...

- O Pyzo não executa os programas a partir do diretório onde os escreve
- Isso cria limitações quando se pretende ler e escrever ficheiros, ou importar funcionalidades definidas em módulos nossos
- Para ultrapassar este problema, uma possibilidade é:
  - verificar qual é o diretório atual, fazendo na janela interativa:

```
import os
os.getcwd()
```

  - mudar o diretório atual do Python para aquele em que se escreve os ficheiros; por exemplo:

```
os.chdir('/Users/jpp/_NOW/P1/CLASSES/DEMO')
```

  - a partir desse momento, podemos ler e escrever ficheiros no mesmo lugar onde estão os programas:

```
import os
os.chdir('/Users/jpp/_NOW/P1/CLASSES/DEMO')
with open("test2.txt", "w") as f:
    f.write("My second file written from Python\n")
    f.write("---------------------------------\n")
    f.write("Hello, world!\n")
```

Módulos criados por nós e o editor Pyzo...

- O mesmo procedimento é útil para importar módulos guardados nesse diretório:

```
import os
os.chdir('/Users/jpp/_NOW/P1/CLASSES/DEMO')
from factorial import factorial

print(factorial(0))
print(factorial(1))
print(factorial(100))
```

Aula de hoje:

- Ficheiros: conceito
- Ficheiros de texto

  `with open(filename, access_mode) as file_handle:`
- Folhas de cálculo: ficheiros CSV
- Ficheiros para objetos Python: Pickle, JSON

Próximas aulas

- Conjuntos e mais tipos de dados em Python