Latex & Markdown

Tópicos de Informática para Automação

Mário Antunes

December 8, 2025

Exercícios

Este guia vai levá-lo desde a configuração do seu ambiente para escrever usando LaTeX e Markdown até à construção de um serviço web que converte automaticamente ficheiros Markdown para PDF. Não é necessária experiência prévia em programação — basta seguir os passos e copiar os blocos de código exatamente.

Passo 0: Configuração do Sistema

Primeiro, precisamos de instalar as ferramentas de "Compilação".

1. Abra o seu **Terminal**.
2. Copie e cole o seguinte comando (vai precisar da sua palavra-passe):

```
sudo apt update
sudo apt install wget texlive-luatex texlive-latex-recommended \
texlive-latex-extra texlive-fonts-recommended texlive-fonts-extra \
fonts-lmodern fonts-noto fonts-noto-cjk fonts-noto-color-emoji pandoc -y
```

3. Crie uma pasta chamada `ex12` para colocar os exercícios.
4. Execute isto no seu terminal dentro da pasta `ex12` primeiro:

```
wget -O profile.png "https://placehold.co/400×400/png"
wget -O graph.png "https://placehold.co/600×400/png"
```

> **Nota:** O `texlive-full` é a versão completa da ferramenta texlive, mas é grande (vários GBs). Contém todos os pacotes que possa precisar para não encontrar erros de falta de pacotes mais tarde. Em caso de erro, considere instalá-lo.

Passo 1: Escrever um CV

Vamos criar um CV usando ambos os métodos. Crie uma pasta chamada `ex12` no seu Ambiente de Trabalho para manter as coisas organizadas.

Opção A: A Via LaTeX (Profissional)

O LaTeX tem uma classe famosa chamada `moderncv`. Tem um aspeto ótimo logo de início.

1. Crie um ficheiro chamado `my_cv.tex` dentro da sua pasta `ex12`.
2. Cole este código nele:

```
\documentclass[11pt,a4paper,sans]{moderncv}
\moderncvstyle{classic} % Options: casual, classic, banking, oldstyle, fancy
\moderncvcolor{blue}    % Options: blue, orange, green, red, purple, grey
\usepackage{graphicx} % Required for images

% Personal Data
\name{John}{Doe}
\title{Computer Science Student}
\address{123 University Ave}{Aveiro, Portugal}
```

```
\phone[mobile]{+351 912 345 678}
\email{john.doe@student.pt}

% Profile picture (height: 64pt, border: 0.4pt)
\photo[64pt][0.4pt]{profile.png}

\begin{document}
\makecvtitle

\section{Education}
\cventry{2023--2026}{BSc in Computer Science}{University of Aveiro}
{Aveiro}{}{Focus on Algorithms and Web Development.}

\section{Skills}
\cvitem{Languages}{Python, C, Java, SQL}
\cvitem{Tools}{Git, Docker, LaTeX, Linux}

\section{Experience}
\cventry{2022}{Summer Intern}{Tech Corp}{Lisbon}
{}{Helped fix bugs in the frontend application.}

\end{document}
```

3. **Compile-o:** Execute este comando no seu terminal dentro da pasta:

   ```
   pdflatex my_cv.tex
   ```

4. Verifique a pasta. Agora tem o `my_cv.pdf`.

Opção B: A Via Markdown (Rápida)

1. Crie um ficheiro chamado `my_cv.md`.
2. Cole este código:

```
# John Doe
![Profile Picture](https://placehold.co/150×150/png)

**Computer Science Student**
*Aveiro, Portugal | +351 912 345 678 | john.doe@student.pt*

---

## Education
**BSc in Computer Science** | University of Aveiro | *2023--2026*
* Focus on Algorithms and Web Development.

## Skills
* **Languages:** Python, C, Java, SQL
* **Tools:** Git, Docker, LaTeX, Linux

## Experience
**Summer Intern** | Tech Corp | *2022*
* Helped fix bugs in the frontend application.
```

3. **Compile-o:**

   ```
   pandoc my_cv.md -o my_cv_markdown.pdf
   ```

Passo 2: Escrever um Relatório de Projeto

Cenário: Está a escrever um relatório sobre "A Eficiência do Café na Velocidade de Programação dos Estudantes".

Opção A: A Via LaTeX

1. Crie um ficheiro chamado `report.tex`.
2. Cole este código:

```
\documentclass[12pt]{report}
\usepackage[utf8]{inputenc}
\usepackage{graphicx}
\usepackage{hyperref}
\usepackage{booktabs}

\title{The Efficiency of Coffee on
Student Coding Speed}
\author{Group 12: John Doe \& Jane Smith}
\date{\today}

\begin{document}

\maketitle
\tableofcontents

\chapter{Introduction}
This study analyzes whether caffeine intake
correlates with lines of code written per hour.
\begin{figure}[!htb]
    \centering
    \includegraphics[width=0.8\textwidth]{graph.png}
    \caption{Projected Coding Speed vs. Caffeine}
    \label{fig:coffee_graph}
\end{figure}

\chapter{Methodology}
We observed 10 students over 4 hours.
\begin{table}[!htb]
    \centering
    \caption{Participant Group Settings}
    \label{tab:groups}
    % Notice the @{} to remove side spacing
    \begin{tabular}{@{}llrr@{}}
        \toprule
        Group & Drink Type & Dosage (mg)
        & Participants \\
        \midrule
        A & Water & 0 & 5 \\
        B & Espresso & 120 & 5 \\
        C & Energy Drink & 160 & 5 \\
        \bottomrule
    \end{tabular}
\end{table}

\chapter{Results}
Group B wrote code 20\% faster but introduced
10\% more bugs.

\chapter{Conclusion}
Coffee increases speed but decreases accuracy.

\end{document}
```

3. **Compile-o:**

   ```
   pdflatex report.tex
   ```

```
pdflatex report.tex
```

(Nota: Executamos duas vezes para que o LaTeX possa gerar os números do Índice corretamente).

Opção B: A Via Markdown

1. Crie um ficheiro chamado `report.md`.
2. Cole este código:

```
---
title: The Efficiency of Coffee on Student Coding Speed
author: Group 12 - John Doe & Jane Smith
date: 2023-12-05
---

# Introduction
This study analyzes whether caffeine intake
correlates with lines of code written per hour.

![Projected Coding Speed vs. Caffeine](graph.png){ width=80% }

# Methodology
We observed 10 students over 4 hours.

Table: Participant Group Settings

| Group | Drink Type | Dosage (mg) | Participants |
| :--- | :--- | ---: | ---: |
| A | Water | 0 | 5 |
| B | Espresso | 120 | 5 |
| C | Energy Drink | 160 | 5 |

# Results
Group B wrote code 20% faster but introduced 10% more bugs.

# Conclusion
Coffee increases speed but decreases accuracy.
```

3. **Compile-o** (Usando a flag `--number-sections` para que pareça um relatório):

```
pandoc report.md -o report_markdown.pdf --number-sections
```

Passo 3: Escrever Slides de Apresentação

Agora precisa de apresentar as conclusões do Passo 2.

Opção A: A Via LaTeX (Beamer)

1. Crie um ficheiro chamado `slides.tex`.
2. Cole este código:

```
\documentclass{beamer}
\usetheme{Madrid}
\usepackage{booktabs} % Required for the table

\title{Coffee vs. Code}
\author{Group 12}
\date{\today}

\begin{document}

\frame{\titlepage}
```

```
\begin{frame}
\frametitle{Introduction}
\begin{itemize}
    \item Does caffeine help us code?
    \item We tested 10 students.
\end{itemize}
\begin{center}
    \includegraphics[width=0.6\textwidth]{graph.png}
\end{center}
\end{frame}

\begin{frame}
\frametitle{Methodology}
\begin{table}
    \centering
    \begin{tabular}{@{}llrr@{}}
        \toprule
        Group & Drink & Dosage & N \\
        \midrule
        A & Water & 0mg & 5 \\
        B & Espresso & 120mg & 5 \\
        \bottomrule
    \end{tabular}
\end{table}
\end{frame}

\begin{frame}
\frametitle{Results}
\alert{Result:} Speed increased by 20\%.
\end{frame}

\end{document}
```

3. **Compile-o:**

   ```
   pdflatex slides.tex
   ```

Opção B: A Via Markdown (Beamer via Pandoc)

1. Crie um ficheiro chamado `slides.md`.
2. Cole este código (Use `---` para separar slides):

```
---
title: Coffee vs. Code
author: Group 12
date: today
---

# Introduction
* Does caffeine help us code?
* We tested 10 students.

![](https://placehold.co/600×300/png)

---

# Methodology

| Group | Drink | Dosage | N |
| :--- | :--- | ---: | ---: |
| A | Water | 0mg | 5 |
| B | Espresso | 120mg | 5 |
```

```
---

# Results
**Result:** Speed increased by 20%.
```

3. **Compile-o** (Dizemos ao Pandoc para fazer output do tipo `beamer`):

```
pandoc slides.md -t beamer -o slides_markdown.pdf
```

Passo 4: O Conversor Automatizado (Docker)

Agora, vamos construir uma aplicação web real. Um utilizador faz upload de um ficheiro Markdown num website, o servidor converte-o para PDF usando Pandoc, e envia-o de volta.

4.1 Estrutura de Pastas

Crie uma nova pasta chamada `converter_app`. Dentro dela, crie duas pastas: `backend` e `frontend`. A sua estrutura deve ser exatamente assim:

```
converter_app/
├── compose.yml
├── backend/
│   ├── Dockerfile
│   ├── main.py
│   └── requirements.txt
└── frontend/
    ├── Dockerfile
    ├── index.html
    └── nginx.conf
```

4.2 Os Ficheiros

1. `converter_app/compose.yml` (Isto diz ao Docker para correr tanto a API Python como o Servidor Web).

```
services:
  backend:
    build: ./backend
    ports:
      - "8000:8000"
    volumes:
      - ./backend:/app

  frontend:
    build: ./frontend
    ports:
      - "8080:80"
```

2. `converter_app/backend/Dockerfile` (Isto instala Python + Pandoc + LaTeX dentro do contentor).

```
FROM python:3.12-trixie

# Install system dependencies (Pandoc and LaTeX)
RUN apt update && apt install -y \
    texlive-luatex texlive-latex-recommended \
    texlive-latex-extra texlive-fonts-recommended \
    texlive-fonts-extra fonts-lmodern fonts-noto \
    fonts-noto-cjk fonts-noto-color-emoji pandoc \
    && rm -rf /var/lib/apt/lists/*
```

```
WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

3. converter_app/backend/requirements.txt

```
fastapi
uvicorn
python-multipart
```

4. converter_app/backend/main.py (A lógica Python: Receber ficheiro -> Correr Pandoc -> Devolver PDF).

```
import subprocess
import os
from fastapi import FastAPI, UploadFile, File
from fastapi.responses import FileResponse
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

# Allow the frontend to talk to this backend
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_methods=["*"],
    allow_headers=["*"],
)

@app.post("/convert")
async def convert_md_to_pdf(file: UploadFile = File( ... )):
    # 1. Save the uploaded markdown file
    input_filename = "input.md"
    output_filename = "output.pdf"

    with open(input_filename, "wb") as f:
        f.write(await file.read())

    # 2. Run Pandoc command inside the container
    # pandoc input.md -o output.pdf
    try:
        subprocess.run(
            ["pandoc", input_filename, "-o", output_filename],
            check=True
        )
    except subprocess.CalledProcessError:
        return {"error": "Conversion failed"}

    # 3. Return the generated PDF
    return FileResponse(output_filename, filename="converted.pdf", media_type='applicatio
```

5. converter_app/frontend/Dockerfile

```
FROM nginx:alpine
COPY nginx.conf /etc/nginx/conf.d/default.conf
COPY index.html /usr/share/nginx/html/index.html
```

6. converter_app/frontend/nginx.conf

```
server {
```

```
    listen 80;
    server_name localhost;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }
}
```

7. converter_app/frontend/index.html (A Interface de Utilizador).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Markdown to PDF Converter</title>
    <style>
        body { font-family: sans-serif; display:
        flex; justify-content: center; align-items: center;
        height: 100vh; background: #f0f0f0; }
        .card { background: white; padding: 2rem;
        border-radius: 10px; box-shadow: 0 4px 6px rgba(0,0,0,0.1);
        text-align: center; }
        button { background: #007bff; color: white; border: none;
        padding: 10px 20px; border-radius: 5px; cursor: pointer;
        margin-top: 10px;}
        button:hover { background: #0056b3; }
    </style>
</head>
<body>
    <div class="card">
        <h1>MD to PDF Converter</h1>
        <p>Select a Markdown file to convert.</p>
        <input type="file" id="fileInput" accept=".md">
        <br><br>
        <button onclick="uploadAndConvert()">Convert & Download</button>
        <p id="status"></p>
    </div>

    <script>
        async function uploadAndConvert() {
            const fileInput = document.getElementById('fileInput');
            const status = document.getElementById('status');

            if(fileInput.files.length === 0) {
                alert("Please select a file!");
                return;
            }

            const formData = new FormData();
            formData.append("file", fileInput.files[0]);
            status.innerText = "Converting... Please wait.";

            try {
                // Send file to Backend (FastAPI)
                const response = await fetch('http://localhost:8000/convert', {
                    method: 'POST',
                    body: formData
                });

                if (!response.ok) throw new Error("Conversion failed");
```

```
                // Create a blob from the response (the PDF)
                const blob = await response.blob();
                const url = window.URL.createObjectURL(blob);

                // Force download
                const a = document.createElement('a');
                a.href = url;
                a.download = "converted_document.pdf";
                document.body.appendChild(a);
                a.click();
                a.remove();
                status.innerText = "Done!";
            } catch (error) {
                console.error(error);
                status.innerText = "Error converting file.";
            }
        }
    </script>
</body>
</html>
```

4.3 Executar a App

1. Abra o seu terminal dentro da pasta `converter_app`.
2. Execute o seguinte comando:

   ```
   sudo docker compose up --build
   ```

   (Isto vai demorar alguns minutos a descarregar a imagem Python e instalar o LaTeX dentro dela. Seja paciente).
3. Assim que parar de mexer e disser "Application startup complete":
   - Abra o seu browser (Firefox/Chrome).
   - Vá a: `http://localhost:8080`
4. Faça upload do `my_cv.md` ou `report.md` que criou anteriormente.
5. Clique em **Convert**. O browser deve descarregar uma versão PDF do seu ficheiro Markdown!