Capturar contenido de un sitio web

Ya existen muchas herramientas disponibles para extraer contenido de sitios web. Sin embargo, al usarlas, no podemos comprender completamente el proceso subyacente. Si en el trabajo nos encontramos con sitios web complejos o particulares, es posible que no obtengamos los resultados deseados al utilizarlas. Necesitamos reinventar la rueda para aprender mejor y aplicar mejor estas herramientas.

También echemos un vistazo a algunas herramientas existentes.

Minero de Datos

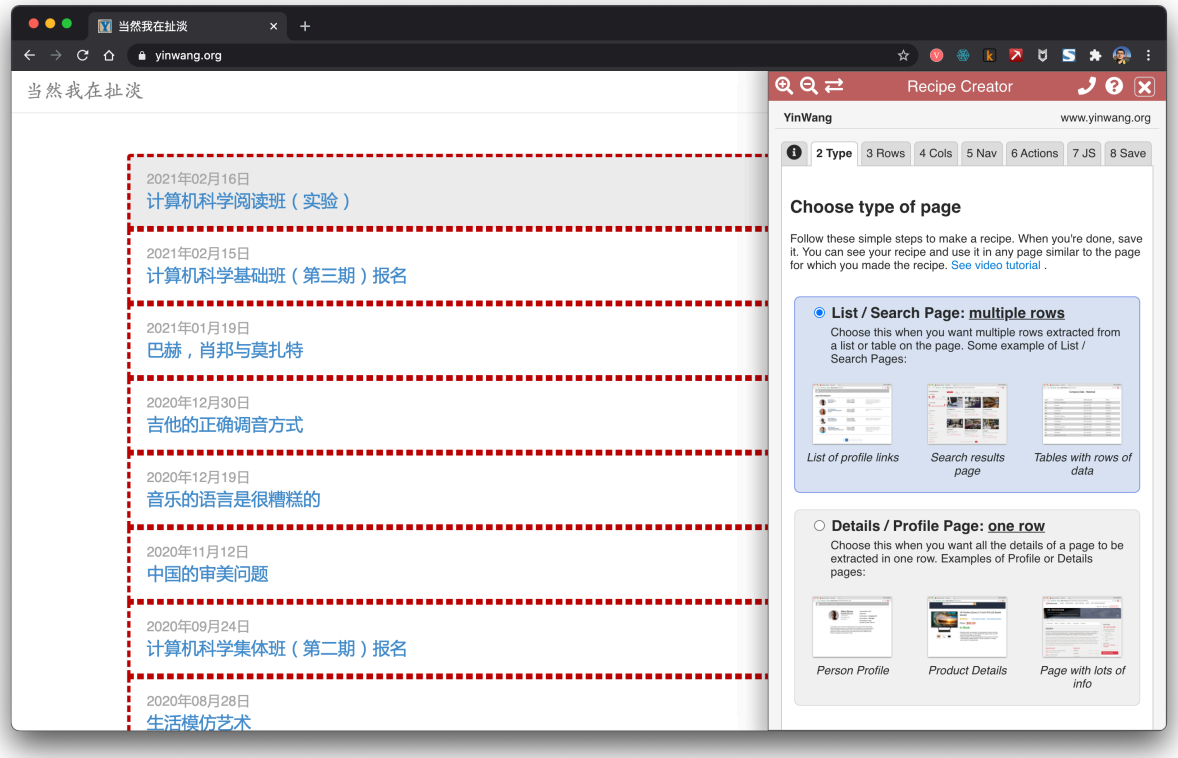

Figure 1: miner

`Data Miner` es una extensión muy útil para Chrome. Permite extraer enlaces y contenido de manera muy conveniente.

getbook

`getbook` es una herramienta muy conveniente para crear libros electrónicos.

```
pip install getbook
```

`book.json`:

```
{
  "uid": "book",
  "title": "Hola Mundo",
  "author": "Armin",
  "chapters": [
    "http://lucumr.pocoo.org/2018/7/13/python/",
    "http://lucumr.pocoo.org/2017/6/5/diversity-in-technology",
  ]
}
```

```
getbook -f ./book.json --mobi
```

De esta manera, se han convertido fácilmente algunos enlaces en un libro electrónico. Al utilizar `Data Miner` y `getbook`, uno para extraer los enlaces y otro para convertirlos en un libro electrónico, se puede crear un libro electrónico de manera muy conveniente.

Las Lecciones de Física de Feynman

En el capítulo «Proyecto práctico: Convertir las páginas web de las Lecciones de Física de Feynman en un libro electrónico», aprendimos cómo convertir una página web `html` renderizada con `mathjax` en un libro electrónico. Aquí continuamos con este proyecto para ver cómo obtener todas las páginas. Las Lecciones de Física de Feynman constan de tres volúmenes. La imagen de arriba muestra el índice del primer volumen.

> http.client — Cliente del protocolo HTTP
>
> Código fuente: Lib/http/client.py
>
> Este módulo define clases que implementan el lado del cliente de los protocolos HTTP y HTTPS. Normalmente no se utiliza directamente — el módulo urllib.request lo usa para manejar URLs que utilizan HTTP y HTTPS.

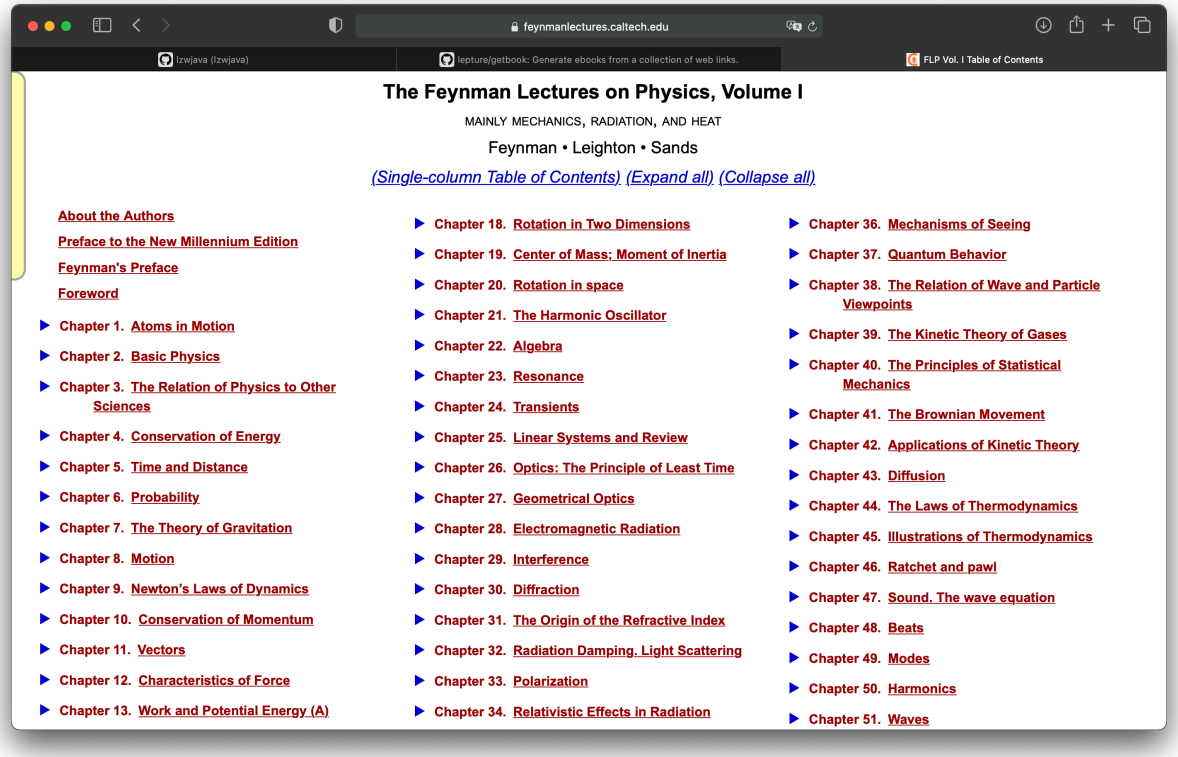
feynmanlectures.caltech.edu
lzwjava (lzwjava)
lepture/getbook: Generate ebooks from a collection of web links.
FLP Vol. I Table of Contents
The Feynman Lectures on Physics, Volume I
MAINLY MECHANICS, RADIATION, AND HEAT
Feynman • Leighton • Sands
(Single-column Table of Contents) (Expand all) (Collapse all)
About the Authors
Preface to the New Millennium Edition
Feynman's Preface
Foreword
Chapter 1. Atoms in Motion
Chapter 2. Basic Physics
Chapter 3. The Relation of Physics to Other Sciences
Chapter 4. Conservation of Energy
Chapter 5. Time and Distance
Chapter 6. Probability
Chapter 7. The Theory of Gravitation
Chapter 8. Motion
Chapter 9. Newton's Laws of Dynamics
Chapter 10. Conservation of Momentum
Chapter 11. Vectors
Chapter 12. Characteristics of Force
Chapter 13. Work and Potential Energy (A)
Chapter 18. Rotation in Two Dimensions
Chapter 19. Center of Mass; Moment of Inertia
Chapter 20. Rotation in space
Chapter 21. The Harmonic Oscillator
Chapter 22. Algebra
Chapter 23. Resonance
Chapter 24. Transients
Chapter 25. Linear Systems and Review
Chapter 26. Optics: The Principle of Least Time
Chapter 27. Geometrical Optics
Chapter 28. Electromagnetic Radiation
Chapter 29. Interference
Chapter 30. Diffraction
Chapter 31. The Origin of the Refractive Index
Chapter 32. Radiation Damping. Light Scattering
Chapter 33. Polarization
Chapter 34. Relativistic Effects in Radiation
Chapter 36. Mechanisms of Seeing
Chapter 37. Quantum Behavior
Chapter 38. The Relation of Wave and Particle Viewpoints
Chapter 39. The Kinetic Theory of Gases
Chapter 40. The Principles of Statistical Mechanics
Chapter 41. The Brownian Movement
Chapter 42. Applications of Kinetic Theory
Chapter 43. Diffusion
Chapter 44. The Laws of Thermodynamics
Chapter 45. Illustrations of Thermodynamics
Chapter 46. Ratchet and pawl
Chapter 47. Sound. The wave equation
Chapter 48. Beats
Chapter 49. Modes
Chapter 50. Harmonics
Chapter 51. Waves

Figure 2: fl

> Ver también: Se recomienda el paquete Requests para una interfaz de cliente HTTP de nivel superior.

Parece que `rquests` es una interfaz de nivel superior.

```
import requests

def main():
    r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
    print(r.status_code)

main()

```shell
401

import requests

def main():
    r = requests.get('https://github.com')
    print(r.status_code)
    print(r.text)

main()

```html
200
<html>
  ...
</html>
```

Probé y confirmé que la interfaz de `requests` funciona correctamente.

```
     <div class="toc-chapter" id="C03">
        <span class="triangle">
         â ¶
        </span>
        <a class="chapterlink" href="javascript:Goto(1,3)">
```

```
 <span class="tag">
  Capítulo 3.
 </span>
 La Relación de la Física con Otras Ciencias
</a>
<div class="sections">
 <a href="javascript:Goto(1,3,1)">
  <span class="tag">
   3-1
  </span>
  Introducción
 </a>
 <a href="javascript:Goto(1,3,2)">
  <span class="tag">
   3-2
  </span>
  Química
 </a>
 <a href="javascript:Goto(1,3,3)">
  <span class="tag">
   3-3
  </span>
  Biología
 </a>
 <a href="javascript:Goto(1,3,4)">
  <span class="tag">
   3-4
  </span>
  Astronomía
 </a>
 <a href="javascript:Goto(1,3,5)">
  <span class="tag">
   3-5
  </span>
  Geología
 </a>
```

```
        <a href="javascript:Goto(1,3,6)">
         <span class="tag">
          3-6
         </span>
         Psicología
        </a>
        <a href="javascript:Goto(1,3,7)">
         <span class="tag">
          3-7
         </span>
         ¿Cómo llegó a ser así?
        </a>
       </div>
      </div>
```

Este es el código `html` de la tercera sección en la página del índice. Quiero extraer los enlaces de cada sección desde aquí. `<a href="javascript:Goto(1,3,7)">`, se puede ver que es un enlace de `javascript`.

```
https://www.feynmanlectures.caltech.edu/I_03.html
```

Luego descubrí que la ruta de cada capítulo sigue un patrón muy claro. `I_03.html` representa el Capítulo 3 del Volumen I.

```
import requests
from bs4 import BeautifulSoup
from multiprocessing import Process

def scrape(chapter):
    if chapter < 1 or chapter > 52:
        raise Exception(f'capítulo {chapter}')
    chapter_str = '{:02d}'.format(chapter)
    url = f'https://www.feynmanlectures.caltech.edu/I_{chapter_str}.html'
    print(f'raspando {url}')
    r = requests.get(url)
    if r.status_code != 200:
        raise Exception(r.status_code)
```

```
    soup = BeautifulSoup(r.text, features='lxml')
    f = open(f'./chapters/I_{chapter_str}.html', 'w')
    f.write(soup.prettify())
    f.close()

def main():
    for i in range(52):
        p = Process(target=scrape, args=(i+1))
        p.start()
        p.join()

main()
```

Continuemos escribiendo el código de scraping. Aquí se utiliza `Process`.

```
    raise RuntimeError('''
RuntimeError:
        Se ha intentado iniciar un nuevo proceso antes de que
        el proceso actual haya terminado su fase de inicialización.

    Esto probablemente significa que no estás utilizando `fork` para iniciar tus
    procesos hijos y has olvidado usar la expresión adecuada
    en el módulo principal:

if __name__ == '__main__':
    freeze_support()
    ...

    La línea "freeze_support()" puede omitirse si el programa
    no va a ser congelado para producir un ejecutable.

```python
def main():
    for i in range(52):
        p = Process(target=scrape, args=(i+1,))
        p.start()
    p.join()
```
```

```
if __name__ == "__main__":
    main()

def main():
    start = timeit.default_timer()
    ps = [Process(target=scrape, args=(i+1,)) for i in range(52)]
    for p in ps:
        p.start()
    for p in ps:
        p.join()
    stop = timeit.default_timer()
    print('Tiempo: ', stop - start)

if __name__ == "__main__":
    main()

scraping https://www.feynmanlectures.caltech.edu/I_01.html
scraping https://www.feynmanlectures.caltech.edu/I_04.html
...
scraping https://www.feynmanlectures.caltech.edu/I_51.html
scraping https://www.feynmanlectures.caltech.edu/I_52.html
Tiempo:  9.144841699


<div class="figure" id="Ch1-F1">
        <img src="img/FLP_I/f01-01/f01-01_tc_big.svgz">
        <div class="caption empty">
         <span class="tag">
          Figura 1â 1
         </span>
        </div>
</div>

import requests
from bs4 import BeautifulSoup
from multiprocessing import Process
import timeit
```

the most information in the fewest words? I believe it is the *atomic hypothesis* (or the atomic *fact* , or whatever you wish to call it) that *all things are made of atomsâlittle particles that move around in perpetual motion, attracting each other when they are a little distance apart, but repelling upon being squeezed into one another* . In that one sentence, you will see, there is an *enormous* amount of information about the world, if just a little imagination and thinking are applied.

Figure 1â1

To illustrate the power of the atomic idea, suppose that we have a drop of water a quarter of an inch on the side. If we look at it very closely we see nothing but waterâsmooth, continuous water. Even if we magnify it with the best optical microscope availableâroughly two thousand timesâthen the water drop will be roughly forty feet across, about as big as a large room, and if we looked rather closely, we would *still* see relatively smooth waterâbut here and there small football-shaped things swimming back and forth. Very interesting. These are paramecia. You may stop at this

Figure 3: fig

```
def scrape(chapter):
    if chapter < 1 or chapter > 52:
        raise Exception(f'capítulo {chapter}')
    chapter_str = '{:02d}'.format(chapter)
    url = f'https://www.feynmanlectures.caltech.edu/I_{chapter_str}.html'
    print(f'raspando {url}')
    r = requests.get(url)
    if r.status_code != 200:
        raise Exception(r.status_code)
    soup = BeautifulSoup(r.text, features='lxml')
    f = open(f'./chapters/I_{chapter_str}.html', 'w')
    f.write(soup.prettify())
    f.close()

def main():
    start = timeit.default_timer()
    ps = [Process(target=scrape, args=(i+1,)) for i in range(52)]
    for p in ps:
        p.start()
    for p in ps:
```

```
        p.join()
    stop = timeit.default_timer()
    print('Tiempo: ', stop - start)

if __name__ == "__main__":
    main()
```

Mira el enlace.

```
    imgs = soup.find_all('img')
    for img in imgs:
        print(img)

scraping https://www.feynmanlectures.caltech.edu/I_01.html
<img id="TwitLink" src=""/>
<img id="FBLink" src=""/>
<img id="MailLink" src=""/>
<img id="MobileLink" src=""/>
<img id="DarkModeLink" src=""/>
<img id="DesktopLink" src=""/>
<img src="img/camera.svg"/>
<img src="img/FLP_I/f01-00/f01-00.jpg"/>
<img data-src="img/FLP_I/f01-01/f01-01_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-02/f01-02_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-03/f01-03_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-04/f01-04_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-05/f01-05_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-06/f01-06_tc_big.svgz"/>
<img class="first" data-src="img/FLP_I/f01-07/f01-07_tc_iPad_big_a.svgz"/>
<img class="last" data-src="img/FLP_I/f01-07/f01-07_tc_iPad_big_b.svgz"/>
<img data-src="img/FLP_I/f01-08/f01-08_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-09/f01-09_tc_big.svgz"/>
<img data-src="img/FLP_I/f01-10/f01-10_tc_big.svgz"/>
```

https://www.feynmanlectures.caltech.edu/img/FLP_I/f01-01/f01-01_tc_big.svgz

(Nota: La URL no necesita traducción, ya que es un enlace directo a un recurso en línea).

```
Prohibido
```

No tienes permiso para acceder a este recurso.

Servidor Apache/2.4.38 (Debian) en www.feynmanlectures.caltech.edu Puerto 443

```shell
% pip install selenium
Collecting selenium
  Using cached selenium-3.141.0-py2.py3-none-any.whl (904 kB)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.9/site-packages (from selenium) (1.24.1
Installing collected packages: selenium
Successfully installed selenium-3.141.0
```

Nota: El texto en el bloque de código no se ha traducido, ya que es un comando y salida de terminal que debe mantenerse en inglés para su correcto funcionamiento.

```
export CHROME_DRIVER_HOME=$HOME/dev-env/chromedriver
export PATH="${PATH}:${CHROME_DRIVER_HOME}"

% chromedriver -h
Uso: chromedriver [OPCIONES]
```

Opciones –port=PORT puerto en el que escuchar –adb-port=PORT puerto del servidor adb –log-path=ARCHIVO escribir el registro del servidor en un archivo en lugar de stderr, aumenta el nivel de registro a INFO –log-level=NIVEL establecer el nivel de registro: ALL, DEBUG, INFO, WARNING, SEVERE, OFF –verbose registrar de manera detallada (equivalente a –log-level=ALL) –silent no registrar nada (equivalente a –log-level=OFF) –append-log añadir al archivo de registro en lugar de sobrescribirlo –replayable (experimental) registrar de manera detallada y no truncar cadenas largas para que el registro pueda ser reproducido. –version imprimir el número de versión y salir –url-base prefijo de ruta base para comandos, por ejemplo, wd/url –readable-timestamp añadir marcas de tiempo legibles al registro –enable-chrome-logs mostrar registros del navegador (anula otras opciones de registro) –allowed-ips lista de direcciones IP remotas permitidas para conectarse a ChromeDriver, separadas por comas

```python
# No es necesario traducir el bloque de código, ya que es un lenguaje de programación.

from selenium import webdriver
from selenium.webdriver.common.by import By
```

```
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support.expected_conditions import presence_of_element_located
```

Este código importa varias clases y funciones de la biblioteca Selenium, que se utiliza para automatizar la interacción con navegadores web. Aquí está el desglose de lo que hace cada importación:

- `webdriver`: Proporciona la funcionalidad principal para controlar un navegador web.
- `By`: Se utiliza para localizar elementos en la página web mediante diferentes estrategias como ID, nombre, clase, etc.
- `Keys`: Proporciona teclas especiales como ENTER, TAB, etc., que se pueden enviar durante la interacción con la página.
- `WebDriverWait`: Permite esperar a que se cumpla una condición específica antes de continuar con la ejecución del script.
- `presence_of_element_located`: Es una condición que se utiliza con `WebDriverWait` para esperar a que un elemento específico esté presente en el DOM de la página.

```
with webdriver.Chrome() as driver:
    wait = WebDriverWait(driver, 10)
    driver.get("https://google.com/ncr")
    driver.find_element(By.NAME, "q").send_keys("cheese" + Keys.RETURN)
    first_result = wait.until(presence_of_element_located((By.CSS_SELECTOR, "h3>div")))
    print(first_result.get_attribute("textContent"))

# El código en Python no necesita ser traducido, ya que es un lenguaje de programación universal.

from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support.expected_conditions import presence_of_element_located
import urllib

def main():
    driver = webdriver.Chrome()
    wait = WebDriverWait(driver, 10)
```

```
    driver.get("https://www.feynmanlectures.caltech.edu/I_01.html")
    elements = driver.find_elements(By.TAG_NAME, "img")
    # print(dir(elements[0]))
    print(driver.page_source)
    i = 0
    for element in elements:
        # src = element.get_attribute('src')
        element.screenshot(f'images/{i}.png')
        i +=1
    driver.close()
main()

from bs4 import BeautifulSoup
from multiprocessing import Process
import timeit
from pathlib import Path
from selenium import webdriver
from selenium.webdriver.common.by import By

def img_path(chapter):
    return f'./chapters/{chapter}/img'

def img_name(url):
    splits = url.split('/')
    last = splits[len(splits) - 1]
    parts = last.split('.')
    name = parts[0]
    return name

def download_images(driver: webdriver.Chrome, chapter):
    path = img_path(chapter)
    Path(path).mkdir(parents=True, exist_ok=True)

    elements = driver.find_elements(By.TAG_NAME, "img")
    for element in elements:
        src = element.get_attribute('src')
        name = img_name(src)
        element.screenshot(f'{path}/{name}.png')
```

USER_AGENT = 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/605.1.15 (KHTML, como Gecko) Versión/14.0.3 Safari/605.1.15'

```
def scrape(chapter):
    if chapter < 1 or chapter > 52:
        raise Exception(f'capítulo {chapter}')
    chapter_str = '{:02d}'.format(chapter)
    url = f'https://www.feynmanlectures.caltech.edu/I_{chapter_str}.html'
    driver = webdriver.Chrome()
    driver.get(url)
    page_source = driver.page_source
    Path(f'./chapters/{chapter_str}').mkdir(parents=True, exist_ok=True)
    print(f'extrayendo {url}')

    download_images(driver, chapter_str)

    soup = BeautifulSoup(page_source, features='lxml')
    imgs = soup.find_all('img')
    for img in imgs:
        if 'src' in img.attrs or 'data-src' in img.attrs:
            src = ''
            if 'src' in img.attrs:
                src = img.attrs['src']
            elif 'data-src' in img.attrs:
                src = img.attrs['data-src']
                del img.attrs['data-src']
            name = img_name(src)
            img.attrs['src'] = f'img/{name}.png'

    f = open(f'./chapters/{chapter_str}/I_{chapter_str}.html', 'w')
    f.write(soup.prettify())
    f.close()

    driver.close()

def main():
    start = timeit.default_timer()
```

```
    ps = [Process(target=scrape, args=(i+1,)) for i in range(2)]
    for p in ps:
        p.start()
    for p in ps:
        p.join()
    stop = timeit.default_timer()
    print('Tiempo: ', stop - start)

if __name__ == "__main__":
    main()
```

```
scraping https://www.feynmanlectures.caltech.edu/I_01.html
scraping https://www.feynmanlectures.caltech.edu/I_02.html
Tiempo:  21.478510914999998
```

```
 errpipe_read, errpipe_write = os.pipe()
OSError: [Errno 24] Demasiados archivos abiertos
```

```
 % ulimit a
ulimit: número inválido: a
lzw@lzwjava feynman-lectures-mobi % ulimit -a
-t: tiempo de CPU (segundos)         ilimitado
-f: tamaño de archivo (bloques)      ilimitado
-d: tamaño de segmento de datos (kbytes) ilimitado
-s: tamaño de pila (kbytes)          8192
-c: tamaño de archivo de volcado (bloques) 0
-v: espacio de direcciones (kbytes)  ilimitado
-l: tamaño de memoria bloqueada (kbytes) ilimitado
-u: procesos                         2784
-n: descriptores de archivo          256
```

```
12
descargar_imágenes
12
mathjax2svg
latexs 128
hacer_svg 0
```

```
insertar_svg 0
hacer_svg 1
insertar_svg 1
hacer_svg 2
insertar_svg 2
hacer_svg 3
insertar_svg 3
convertir

12
descargar_imágenes
12
mathjax2svg
latex 0
latex 0
convertir
Tiempo:  11.369145162

% grep --include=\*.html -r '\$' *
43/I_43.html:un largo período de tiempo $T$, tiene un cierto número, $N$, de impactos. Si
43/I_43.html:el número de colisiones es proporcional al tiempo $T$. Nos gustaría
43/I_43.html:Hemos escrito la constante de proporcionalidad como $1/\tau$, donde
43/I_43.html:$\tau$ tendrá las dimensiones de un tiempo. La constante $\tau$ es el
43/I_43.html:hay $60$ colisiones; entonces $\tau$ es un minuto. Diríamos
43/I_43.html:que $\tau$ (un minuto) es el

□□  E21018: Fallo al crear el dominio Mobi mejorado al analizar el contenido del archivo. Contenido: 
Advertencia W28001: El lector Kindle no admite el estilo CSS especificado en el contenido. Eliminando la
Advertencia W29004: Etiqueta abierta cerrada forzosamente: <span amzn-src-id="985"> Archivo: /private/va
Advertencia W29004: Etiqueta abierta cerrada forzosamente: <p amzn-src-id="975"> Archivo: /private/var/

Advertencia W14001: Problema con el hipervínculo, aún no resuelto:  /private/var/folders/_3/n3b7dq8x665
Advertencia W14001: Problema con el hipervínculo, aún no resuelto:  /private/var/folders/_3/n3b7dq8x665
Advertencia W14001: Problema con el hipervínculo, aún no resuelto:  /private/var/folders/_3/n3b7dq8x665

<span class="disabled" href="#Ch1-F1">
          1-1
</span>
```

```
Rasterizando 'OEBPS/84b8b4179175f097be1180a10089107be75d7d85.svg' a 1264x1011
Rasterizando 'OEBPS/23a4df37f269c8ed43f54753eb838b29cff538a1.svg' a 1264x259
Traceback (most recent call last):
  File "runpy.py", line 194, in _run_module_as_main
  File "runpy.py", line 87, in _run_code
  File "site.py", line 39, in <module>
  File "site.py", line 35, in main
  File "calibre/utils/ipc/worker.py", line 216, in main
  File "calibre/gui2/convert/gui_conversion.py", line 41, in gui_convert_override
  File "calibre/gui2/convert/gui_conversion.py", line 28, in gui_convert
  File "calibre/ebooks/conversion/plumber.py", line 1274, in run
  File "calibre/ebooks/conversion/plugins/mobi_output.py", line 214, in convert
  File "calibre/ebooks/conversion/plugins/mobi_output.py", line 237, in write_mobi
  File "calibre/ebooks/oeb/transforms/rasterize.py", line 55, in __call__
  File "calibre/ebooks/oeb/transforms/rasterize.py", line 142, in rasterize_spine
  File "calibre/ebooks/oeb/transforms/rasterize.py", line 152, in rasterize_item
  File "calibre/ebooks/oeb/transforms/rasterize.py", line 185, in rasterize_external
  File "calibre/ebooks/oeb/base.py", line 1092, in bytes_representation
  File "calibre/ebooks/oeb/base.py", line 432, in serialize
TypeError: no se puede convertir un objeto 'NoneType' a bytes

% kindlepreviewer feynman-lectures-on-physics-volumn-1.epub -convert
Verificando los argumentos especificados.
Preprocesamiento en progreso.
Procesando 1/1 libro(s).
¡Libro convertido con advertencias! : /Users/lzw/projects/feynman-lectures-mobi/feynman-lectures-on-phys
Postprocesamiento en progreso.
Escribiendo archivos de salida/log en /Users/lzw/projects/feynman-lectures-mobi/output

Limpiando el manifiesto...
Recortando archivos no utilizados del manifiesto...
Creando salida AZW3...
Serializando recursos...
Dividiendo el marcado en saltos de página y límites de flujo, si los hay...
Creando salida KF8
    Generando marcado KF8...
La tabla de etiquetas no tiene aid y un tamaño de fragmento demasiado grande. Añadiendo de todos modos.
```

```
La tabla de etiquetas no tiene aid y un tamaño de fragmento demasiado grande. Añadiendo de todos modos.
La tabla de etiquetas no tiene aid y un tamaño de fragmento demasiado grande. Añadiendo de todos modos.
    Comprimiendo el marcado...
    Creando índices...
```