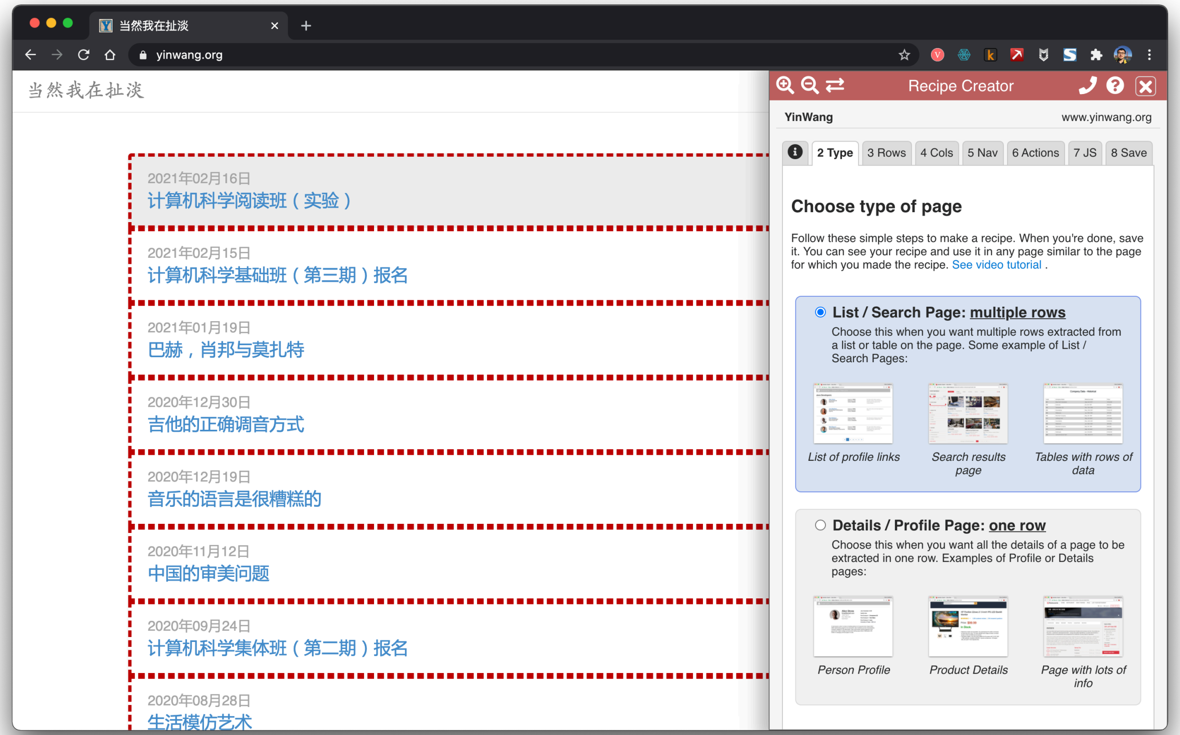
抓取網站內容

已經有很多現成的工具可以抓取網站的內容。然而，如果使用它們，我們無法更好地理解背後的過程。如果在工作中遇到複雜或特別的網站，使用這些工具可能無法得到想要的結果。我們需要造輪子，為了更好地學習它們和更好地運用它們。

也來看看現成的一些工具。

Data Miner



当然我在扯淡

yinwang.org

当然我在扯淡

2021年02月16日
计算机科学阅读班（实验）

2021年02月15日
计算机科学基础班（第三期）报名

2021年01月19日
巴赫，肖邦与莫扎特

2020年12月30日
吉他的正确调音方式

2020年12月19日
音乐的语言是很糟糕的

2020年11月12日
中国的审美问题

2020年09月24日
计算机科学集体班（第二期）报名

2020年08月28日
生活模仿艺术

Recipe Creator

YinWang www.yinwang.org

2 Type | 3 Rows | 4 Cols | 5 Nav | 6 Actions | 7 JS | 8 Save

Choose type of page

Follow these simple steps to make a recipe. When you're done, save it. You can see your recipe and use it in any page similar to the page for which you made the recipe. See video tutorial .

(●) List / Search Page: multiple rows
Choose this when you want multiple rows extracted from a list or table on the page. Some example of List / Search Pages:
List of profile links | *Search results page* | *Tables with rows of data*

( ) Details / Profile Page: one row
Choose this when you want all the details of a page to be extracted in one row. Examples of Profile or Details pages:
Person Profile | *Product Details* | *Page with lots of info*

Figure 1: miner

`Data Miner` 是 Chrome 上的一個很方便的插件。可以很方便地抓取連結和內容。

getbook

`getbook` 是一個很方便的製作電子書工具。

```
pip install getbook
```

book.json:

```
{
  "uid": "book",
  "title": "Hello World",
  "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
```

這樣就方便地把一些連結做成了電子書。通過使用 `Data Miner` 和 `getbook`，一個抓取連結，一個把連結變成電子書，就能很方便地製作電子書。

費曼物理講義

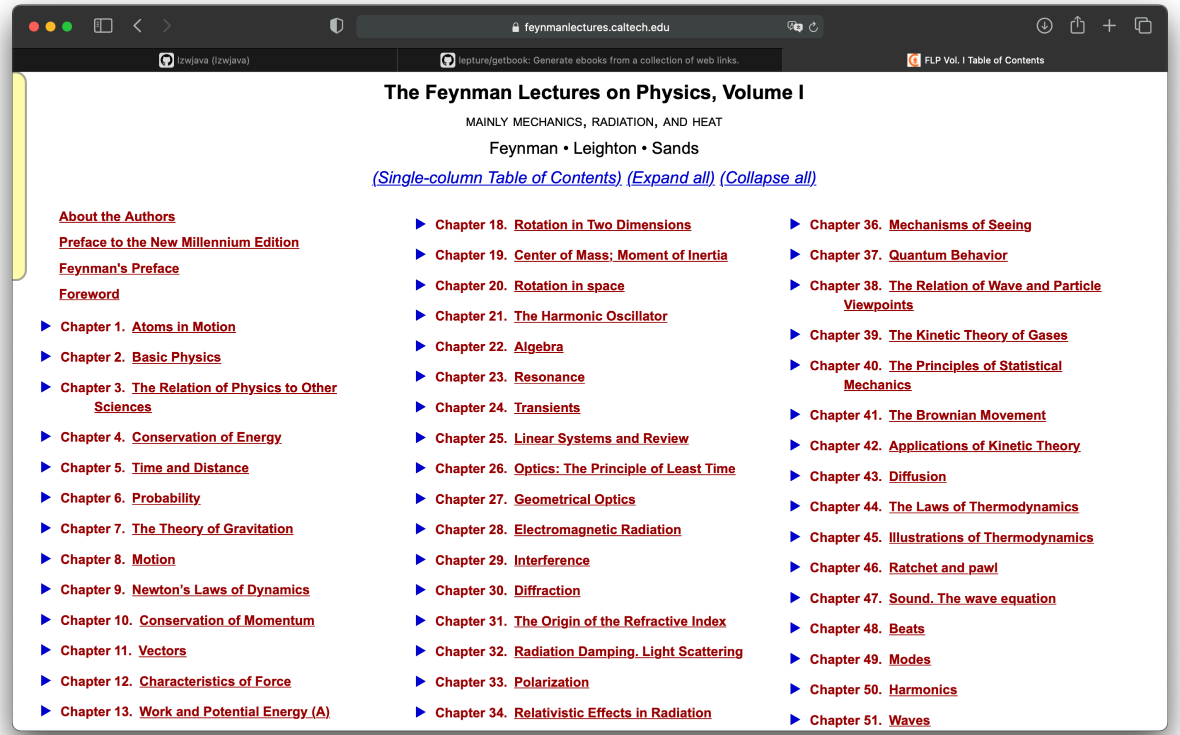
在「項目實戰：將費曼物理講義網頁做成電子書」章節中，我們學會如何把一個用 `mathjax` 渲染的 `html` 網頁做成電子書。這裡繼續這個項目，來看看如何獲取到所有的網頁。費曼物理講義有三卷。上圖是第一卷的目錄。

> http.client — HTTP protocol client
>
> Source code: Lib/http/client.py
>
> This module defines classes which implement the client side of the HTTP and HTTPS protocols. It is normally not used directly — the module urllib.request uses it to handle URLs that use HTTP and HTTPS.
>
> See also: The Requests package is recommended for a higher-level HTTP client interface.

可見 `requests` 是更高階的接口。

```
import requests

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



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)

Figure 2: fl

```
    print(r.status_code)

main()
```

```
401
```

```
import requests

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

main()
```

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

試了試，說明 requests 的接口是能用的。

```
     <div class="toc-chapter" id="C03">
        <span class="triangle">
         â ¶
        </span>
        <a class="chapterlink" href="javascript:Goto(1,3)">
         <span class="tag">
          Chapter 3.
         </span>
         The Relation of Physics to Other Sciences
        </a>
        <div class="sections">
         <a href="javascript:Goto(1,3,1)">
          <span class="tag">
           3-1
          </span>
```

```
 Introduction
</a>
<a href="javascript:Goto(1,3,2)">
 <span class="tag">
  3-2
 </span>
 Chemistry
</a>
<a href="javascript:Goto(1,3,3)">
 <span class="tag">
  3-3
 </span>
 Biology
</a>
<a href="javascript:Goto(1,3,4)">
 <span class="tag">
  3-4
 </span>
 Astronomy
</a>
<a href="javascript:Goto(1,3,5)">
 <span class="tag">
  3-5
 </span>
 Geology
</a>
<a href="javascript:Goto(1,3,6)">
 <span class="tag">
  3-6
 </span>
 Psychology
</a>
<a href="javascript:Goto(1,3,7)">
 <span class="tag">
  3-7
 </span>
```

```
          How did it get that way?
         </a>
        </div>
       </div>
```

這是在目錄頁面中，第三章節的 `html` 代碼。想從這裡抓取每一章節的連結。`<a href="javascript:Goto(1,3,7)">` 可見是一個 `javascript` 的超連結。

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

接著發現，每章節的路徑是很有規律的。`I_03.html` 表示第一卷第三章。

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

def scrape(chapter):
    if chapter < 1 or chapter > 52:
        raise Exception(f'chapter {chapter}')
    chapter_str = '{:02d}'.format(chapter)
    url = f'https://www.feynmanlectures.caltech.edu/I_{chapter_str}.html'
    print(f'scraping {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()
```

來接著寫寫抓取代碼。這裡用到了 `Process`。

```
    raise RuntimeError('''
RuntimeError:
        An attempt has been made to start a new process before the
        current process has finished its bootstrapping phase.

        This probably means that you are not using fork to start your
        child processes and you have forgotten to use the proper idiom
        in the main module:

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

        The "freeze_support()" line can be omitted if the program
        is not going to be frozen to produce an executable.
```

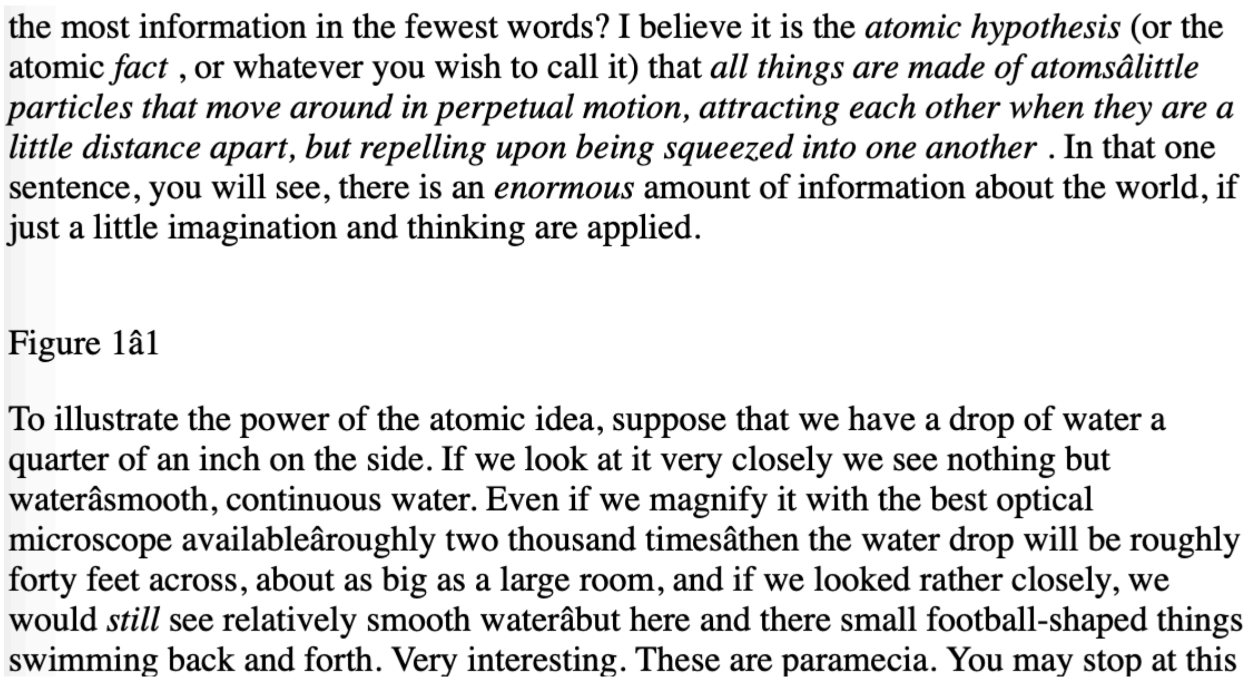
```
def main():
    for i in range(52):
        p = Process(target=scrape, args=(i+1,))
        p.start()
    p.join()

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('Time: ', 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
Time:  9.144841699
```



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

```
<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">
          Figure 1â 1
         </span>
        </div>
</div>
```

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

def scrape(chapter):
    if chapter < 1 or chapter > 52:
        raise Exception(f'chapter {chapter}')
    chapter_str = '{:02d}'.format(chapter)
    url = f'https://www.feynmanlectures.caltech.edu/I_{chapter_str}.html'
    print(f'scraping {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('Time: ', stop - start)

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

看看連結。

```
    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

```
Forbidden


You don't have permission to access this resource.


Apache/2.4.38 (Debian) Server at www.feynmanlectures.caltech.edu Port 443
```

```
% 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.
Installing collected packages: selenium
Successfully installed selenium-3.141.0
```

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

```
% chromedriver -h
Usage: chromedriver [OPTIONS]

Options
  --port=PORT                        port to listen on
  --adb-port=PORT                    adb server port
  --log-path=FILE                    write server log to file instead of stderr, increases log level to INI
  --log-level=LEVEL                  set log level: ALL, DEBUG, INFO, WARNING, SEVERE, OFF
  --verbose                          log verbosely (equivalent to --log-level=ALL)
  --silent                           log nothing (equivalent to --log-level=OFF)
  --append-log                       append log file instead of rewriting
  --replayable                       (experimental) log verbosely and don't truncate long strings so that t
  --version                          print the version number and exit
  --url-base                         base URL path prefix for commands, e.g. wd/url
  --readable-timestamp               add readable timestamps to log
  --enable-chrome-logs               show logs from the browser (overrides other logging options)
  --allowed-ips                      comma-separated allowlist of remote IP addresses which are allowed to
```

```
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

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"))
```

```
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
```

```python
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, like Gecko)

def scrape(chapter):
    if chapter < 1 or chapter > 52:
        raise Exception(f'chapter {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'scraping {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('Time: ', stop - start)

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

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

```
 errpipe_read, errpipe_write = os.pipe()
OSError: [Errno 24] Too many open files
```

```
 % ulimit a
ulimit: invalid number: a
lzw@lzwjava feynman-lectures-mobi % ulimit -a
-t: cpu time (seconds)              unlimited
-f: file size (blocks)              unlimited
-d: data seg size (kbytes)          unlimited
-s: stack size (kbytes)             8192
-c: core file size (blocks)         0
-v: address space (kbytes)          unlimited
-l: locked-in-memory size (kbytes)  unlimited
```

```
-u: processes                        2784
-n: file descriptors                 256
```

“ ‘shell 12 download_images 12 mathjax2svg latexs 128 make_svg 0 insert_svg 0 make_svg 1 insert_svg 1 make_svg 2 insert_svg 2 make_svg 3 insert_svg