未來，往哪走？

hydai@sprout, 6/13

新手包（$ 時間）

基礎建設工程

- 演算法與資料結構（Algorithm & Data structure）
- 程式設計的思緒
- 瞭解整個系統整體的運作模型
  - 應用程式
    - 視窗程式設計(ex: 小遊戲)
    - 網路程式設計(ex: BT)
  - 作業系統
    - 系統程式: 編譯器 連結器 assembler loader
    - 驅動程式
    - 基本的處理序管理，記憶體管理，檔案系統
    - CPU，記憶體，快取設計，多核心CPU
  - 硬體（e.g. 硬體描述語言），形式驗證
    - 電路設計，邏輯閘

各種 DLC（$ 更多時間）

工程DLC（通常學校不一定會教）

- 讓程式變好：
  - 軟體工程 (ex: design patterns)
  - 程式風格 (ex: 元件設計, API 設計)
- 讓事情發生：
  - 產品開發管理 (ex: project management)
  - 雖然說在這一個要素中最重要的因素是人

科學DLC（通常你有機會修到的）

- 影像處理、訊號處理、無線通訊(ex: 串流壓縮)
- 前端網頁設計、後端伺服器處理等(ex: 網站設計)
- 機器學習、資料探勘、 資訊檢索、人工智慧(ex: 我們的小遊戲喔！)
- 機器人、電腦視覺(ex: OCR)
- 人機介面(ex: UI/UX)
- 編譯器、計算機結構
- 資訊理論、程式語言、計算理論
- 系統程式、作業系統、嵌入式系統(ex: linux)
- 平行計算、分散式系統、雲端

社群DLC（我們要十個打一個）

- 參加社群，除了獲得一起前進的朋友，還有很多有趣的事情
- 會議介紹:
  - COSCUP: 開源大會(搶票大戰), 今年在 6/20 大家上囉～
  - SITCON: 學生年會, 這個的內容淺顯, 相較其他的容易聽懂
- 語言的社群: Ruby conf, PyCon
- 公民參與: g0v

參加社群，你可能會接觸的工具/平台

- 版本控制系統 (ex: git)
  - 一堆人寫程式碼，不控管太可怕了
- 程式設計師的 FB -> GitHub
  - 很多作品會用 Open Source 的授權放在這個上面
- 程式設計師的知識+/wiki -> StackOverflow
  - 這個或 GitHub 掛了的話，會暴動的喔XD

趨勢包（我的觀察啦）

Web 開發正夯 - 前端篇

- 前後端程式開發
  - 前端
    - 基礎: HTML+CSS+Javascript
    - 樣式&版面設計:
      - 程式化的 HTML: Jade
      - 程式化的 CSS: LESS, SASS
      - 完全就是程式的前端: anguler.js react.js
    - 全世界都在用的框架:
      - bootstrap

Web 開發正夯 - 後端篇

- 前後端程式開發
  - 後端
    - Javascript: NodeJS (現在併入 io.js 囉！)
    - Ruby: Ruby on Rails (RoR 在台灣社群很大！)
    - PHP: CodeIgniter, Laravel (FB 就是用 php 寫的)
    - Python: Tornado, Django (我的最愛, 私心)
    - Go: Revel (這個是我唯一沒碰過的QQ)

Web 開發正夯 - 資料庫篇

- 前後端程式開發
  - 資料庫
    - 關聯性資料庫(DBMS)
      - 最容易找到資源: MySQL (有社群版)
      - 在每個 app 都有的: sqlite
    - NoSQL(Not Only SQL)
      - MongoDB
    - Big data 相關
      - Hadoop, BigTable

聽說人人都可以寫 App

- 在 app 的世界，程式很強不再是重點
- 好的想法 + 消費者買單才是王道
- 強大的功能 + 難用的介面 = GG
- 不怎麼樣的功能 + 神好用的介面 = 你會賺很多
- 跨界合作，激發新想法（像清大有學生出來搞跨領域的社群）

很硬但是很有發展性

- 異質系統，好比說 CPU+GPU 協力運算（ex: HSA）
- 平行系統（ex: 被廠商亂用的雲端相關詞彙）
- 通常苦工會比前面所提的還來得很多
- 地雷也是很多
- 只要成功生還，有成果就有廠商對你有興趣

回歸程式本身

返璞歸真(?) 只關注程式本身

- 語言與寫法…
  - 不同的編程典範 http://en.wikipedia.org/wiki/Programming_paradigm：
  - procedural
    - object-oriented 物件導向
    - logic
    - functional
  - 不同的設計嘗試
    - generic programming
    - metaprogramming

C++ 不只是坑，他是個無底洞

- C++ 作者表示: 我也不敢說自己完全懂 C++
- 寫程式本身，不特別關注語言特性(語法班)，也可以學其它語言
- C++ 有他的歷史與特點，以及缺陷
- 沒教給的小東西: union, reference
- 沒教到的大東西: (基本上都是超級大坑，跳要時間，我們講不完
  - class (C++ 中物件導向的那一面)
  - exception handling
  - namespace management
  - template (and metaprogramming!) (寫大家用過的 STL)
  - resource&ownership management
  - C++ 的各種 idiom 等等、C++11, C++14, C++17 ...

你在課堂上看不到的 C++ - 1

- Lambda:

```
auto curry = [](auto f) {
  return [f](auto x) {
    return [f,x](auto y) {
      return f(x,y);
    };
  };
};
```

你在課堂上看不到的 C++ - 2

- http://cpptruths.blogspot.tw/2014/05/fun-with-lambdas-c14-style-part-2.html
- Overloaded Lambdas:

```
auto curry = [](auto f) {
  return [f](auto x) {
    return [f,x](auto y) {
      return f(x,y);
    };
  };
};
```

你在課堂上看不到的 C++ - 3

- http://en.cppreference.com/w/cpp/thread/future
- future from a promise:

```
std::promise<int> p;
std::future<int> f3 = p.get_future();
std::thread( [](std::promise<int> p {
        p.set_value_at_thread_exit(9);
    },
    std::move(p)
).detach();
```

C++ 由淺入深書單

- http://stackoverflow.com/questions/388242/the-definitive-c-book-guide-and-list
- 太多了，大家可以慢慢看

可能碰到的 Open Source Project

- 重要性十足的 GCC/G++, 以及準備篡位的 Clang/LLVM
- OpenGL: 寫小遊戲時已經碰到了，圖形處理函式庫
- OpenAL: 音效處理函式庫
- OpenCL: heterogeneous computing (剛才說的異質平台就會用到)
- OpenCV: 影像（視覺）處理的函式庫
- OpenMP: 平行程式設計
- C++ 很有名的函式庫：boost、Loki

更多語言，更多坑

- 強大的腳本語言: Python、Ruby
- Rust、Go
- Java 統治世界
- 老牌的 Perl、很容易有漏洞的 PHP
- 微軟的作品：C# 等以 .Net 平台為後端的語言
- 各種函數式（functional）語言:
  - Haskell, OCaml, Scheme, Racket, Scala, ...
- 新時代的組合語言: javascript 想要統治全世界
  - 前端有 angular.js
  - 後端有 node.js
  - 聽說打算弄一個 javascript OS

「提的東西很多，有些縱使看起來類似，
或是只有微妙差異，但看了以後，
或許會改變你的一生與思考」

– 一路走來所看見的事