

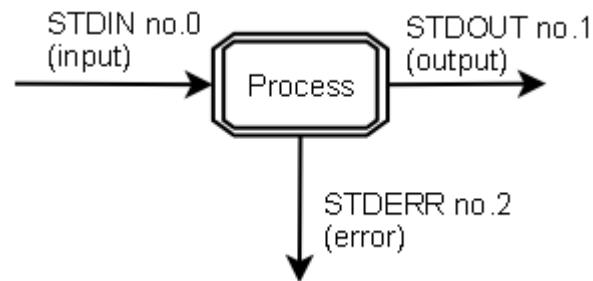


# Script Pipeline Construction

李義安 工程師

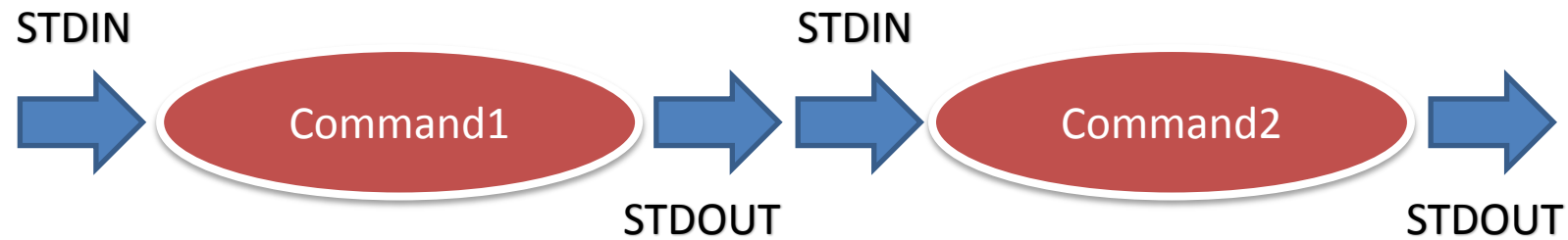
# Stdin , stdout & stderr

- 0 : stdin (standard input)
- 1 : stdout (standard output)
- 2 : stderr (standard error)



- > : redirect
- 1> : redirect STDOUT
- 2> : redirect STDERR
- &> : redirect all

# Stdin , stdout & stderr



# Link Scripts

---

- Use |
- Use ;
- Ex : qstat | grep ngs
- Ex : ls ; ngsnodes

# Job Dependency

---

- Use ; to split scripts
- Use dependency parameter in queue system
- Ex : ls => ls -a = ls ; ls -a
- qsub \$job1
- 13111
- qsub -W depend=afterok:13111 \$job2
- 13112

# Job Scripts Example

---

```
bwa mem -M -R "@RG\tID:miseq\tSM:AD002chr22\tPL:illumina" -t 40 -K 10000000  
/work1/jimmy200340/course/ref/GRCh38/GRCh38_latest_genomic.fna  
/work1/jimmy200340/course/AD002_chr22.fastq.gz | samblaster -M -e | samtools view -bS  
-o /work1/jimmy200340/course/test.bam
```

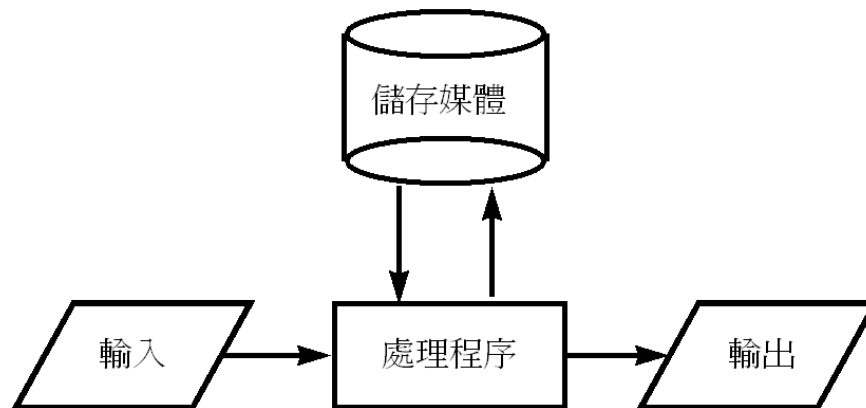
# Perl 簡介

---

- 電腦語言
- 何謂直譯式語言
- 直譯式語言的優勢
- 生物資訊的黃金組合
- 以 Perl 為核心的大型分析工具
- 第一個 Perl 程式

# 硬體 VS 軟體

電腦只是個冷冰冰的機器  
有了軟體才能協助人類做事





# 軟體有哪些

---

- 作業系統
  - Windows, Linux, MacOS, iOS, WatchOS, tvOS....
- 文書處理軟體
  - MSOffice...
- 娛樂軟體
  - Video players, Games.....
- 每一項軟體都是一組程式碼所構成
- 程式碼是透過某一組程式語言所撰寫出來的

# 高階程式語言 VS 低階程式語言

- 機械語言
  - 電腦處理器直接使用的語言
  - 每個機器的機械碼可能是不一樣的
  - CPU不同，擴充卡種類，插槽位置不同
  - 程式最終進CPU前的狀態
- 低階程式語言
  - 通常程式撰寫者較難直接使用
  - 組合語言
- 高階程式語言
  - C, Java, Basic, Perl, Python, PHP.....

# 直譯式語言？

---

- 他是一種高階程式語言
- 撰寫完之後，不必事先編譯成機械碼的一種程式語言
- 以 C, Basic 為例，程式撰寫好以後，需要編譯後，才能執行
- 直譯式語言在執行前使用者不需要編譯的動作

# 直譯式語言的優點

- 編寫上傾向容易簡單，易上手（也不太容易掌握）
- 編寫上較具彈性（其實是一種災難）
- 對進階使用者而言，支援模組架構（與容易上手概念相斥）

# 過去生物資訊的黃金組合

---

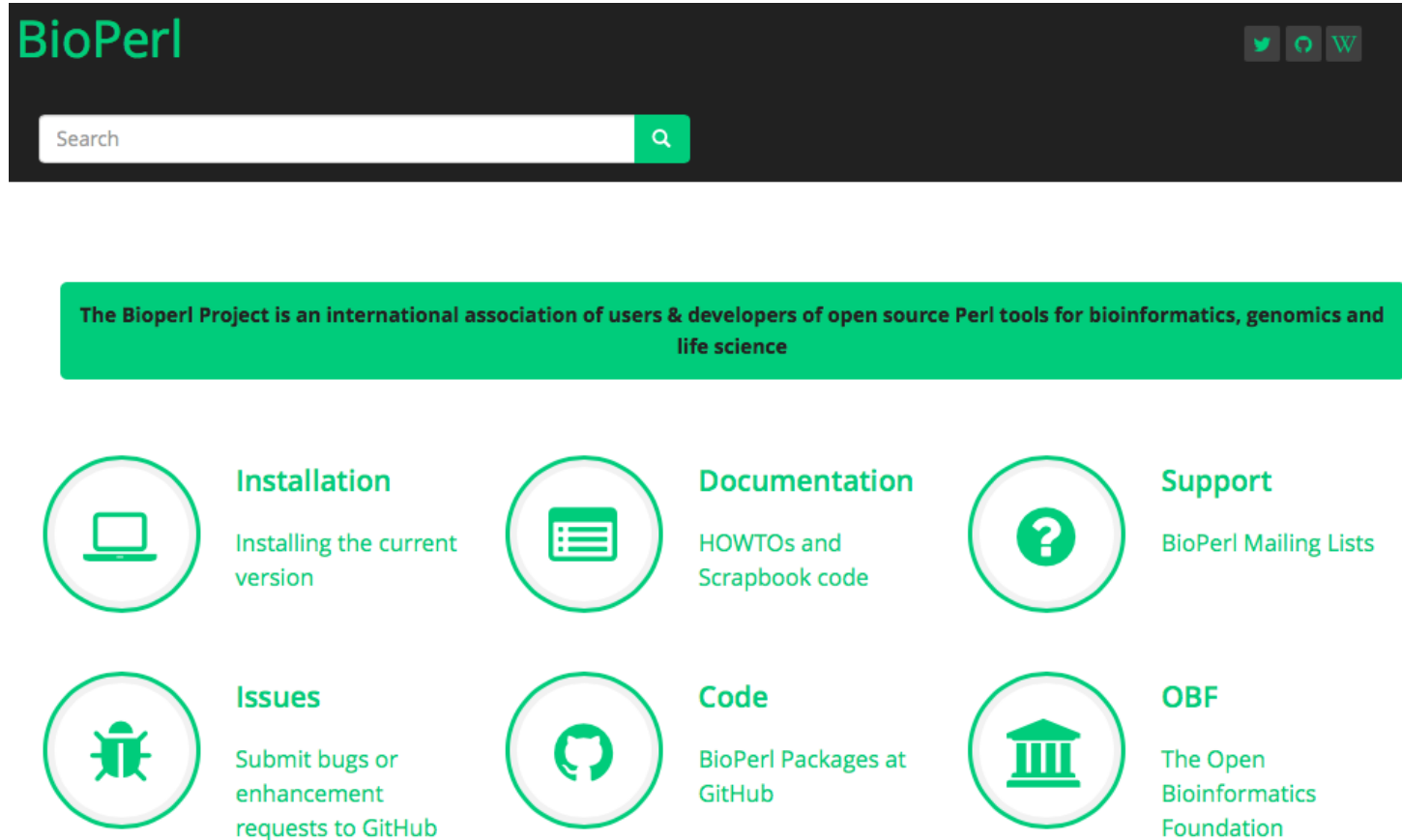
- Linux
- Perl, R
- MySQL

# 生物資訊的黃金組合

---

- Linux, MacOS
- Perl, R, Python, PHP
- MySQL, MongoDB, MariaDB...

# Why Perl?



The BioPerl Project is an international association of users & developers of open source Perl tools for bioinformatics, genomics and life science

- Installation**  
Installing the current version
- Documentation**  
HOWTOs and Scrapbook code
- Support**  
BioPerl Mailing Lists
- Issues**  
Submit bugs or enhancement requests to GitHub
- Code**  
BioPerl Packages at GitHub
- OBF**  
The Open Bioinformatics Foundation

Perl 是最早推出生命科學相關領域程式套件（BioPerl）的直譯式程式語言  
最原始的開發者包含 Sanger Institute, CSHL... 等生物資訊元老級先驅者

# 後續衍生的生物資訊分析系統

The screenshot shows the Ensembl genome browser homepage. At the top is a dark blue navigation bar with the 'e!Ensembl' logo and links for BLAST/BLAT, BioMart, Tools, Downloads, Help & Documentation, and a 'More' dropdown. Below this is a search bar with a dropdown menu set to 'All species' and a 'Go' button. A hint text below the search bar reads: 'e.g. BRCA2 or rat 5:62797383-63627669 or rs699 or coronary heart disease'. The main content area is divided into several sections. On the left, 'Browse a Genome' explains the project's goal and lists 'Popular genomes' with icons and links for Human (GRCh38.p5, GRCh37), Mouse (GRCm38.p4), and Zebrafish (GRCz10). Below this is a 'Log in to customize this list' link and an 'All genomes' section with a species selection dropdown. To the right of the 'Browse a Genome' section are four boxes: 'Still using Human GRCh37?' with a 'Go to e!GRCh37' button, 'Variant Effect Predictor' with the 'Ve!P' logo, 'Gene expression in different tissues' with a histology image, and 'Find SNPs and other variants for my gene' showing a sequence alignment. At the bottom right are two more boxes: 'Retrieve gene sequence' displaying a DNA sequence and 'Compare genes across species' with a phylogenetic tree diagram.

以 Perl 為核心的生物資料  
庫資訊系統，

API 均以 Perl 程式撰寫

該系統已囊括國際上重要  
模式物種之基因體資訊  
將註解資訊以座標的方式  
儲存於 MySQL 資料庫中



# Hello World !

---

```
#!/usr/bin/perl
```

```
print "Hello World!\n";
```

# Perl 程式編譯器位置

---

```
#!/usr/bin/perl
```

```
print "Hello World!\n";
```

# 前段控制碼

---

```
#!/usr/bin/perl
```

```
print "Hello World!\n";
```

宣告 Perl 的直譯式程式擺放位置與檔案名稱

# 輸出／列出指令

---

```
#!/usr/bin/perl
```

```
print "Hello World!\n";
```

# 列出的內容以雙引號夾出

---

```
#!/usr/bin/perl
```

```
print "Hello World!\n";
```

# 換行符號

---

```
#!/usr/bin/perl
```

```
print "Hello World!\n";
```

電腦有如絕對服從之軍人，  
程式撰寫者有如指揮官，  
一個指令一個動作

沒有換行符號，就會黏在一起

# 單行指令結束位置

---

```
#!/usr/bin/perl
```

```
print "Hello World!\n";
```

# 變數

```
#!/usr/bin/perl
```

\$ 為變數的起始代碼

```
$i = 1;  
print $i, "\n";
```

```
$i++;  
Print $i, "\n";
```



# 變數

```
#!/usr/bin/perl
```

i 為軟體撰寫者給的變數名稱

```
$i = 1;  
print $i, "\n";
```

```
$i++;  
Print $i, "\n";
```

# 變數

```
#!/usr/bin/perl
```

```
$i = 1;  
print $i, "\n";
```

```
$i++;  
Print $i, "\n";
```

++ 為計算指令  
意即 \$i 加一以後  
覆寫回 \$i

# 變數

```
#!/usr/bin/perl
```

```
$i = 1;  
print $i, "\n";
```

```
$i++;  
Print $i, "\n";
```

練習一：

如果起始值為1  
第一行列印起始值後，  
第二行列印值為4  
的程式碼應該要怎樣寫？

# 陣列

```
#!/usr/bin/perl
```

```
my @array = ('a','b','c','d');
```

```
$array[3] = 'd';
```

```
print $array[0], "\n";
```

陣列的起始代碼為 @

亦可直接修改陣列位置中的任一數值

# 陣列

---

```
#!/usr/bin/perl
```

```
my @array = ('a','b','c','d');
```

```
print $array[0], "\n";
```

@後面的名稱為軟體撰寫者指定的名稱

# 陣列

```
#!/usr/bin/perl
```

```
my @array = ('a','b','c','d');
```

```
print $array[0], "\n";
```

以 **()** 與 **,** 分隔指定陣列裡的内容

文字類別需要加上 **"**  
數字類別的，**不需要**

# 陣列

```
#!/usr/bin/perl
```

```
my @array = ('a','b','c','d');
```

```
print $array[0], "\n";
```

陣列中單一變數取出

以 \$**陣列名稱**[位置碼] 將數值取出

第一位置為 0

第二位置為 1

第三位置為 2

位置碼 = 陣列個數值 - 1

# 陣列

---

```
#!/usr/bin/perl
```

```
my @array = ('a','b','c','d');
```

```
print $array[0], "\n";
```

練習二：

如果列印值為 d  
的程式碼應該要怎樣寫？



# 行程控制

---

```
for (my $i=0; $i <=
9; $i++) {
    print "$i\n";
}
```

```
foreach my $aline
(@array) {
    print $aline,
"\n";
}
```

for 條件判斷

foreach (@array)

while (IN, a file)

# 行程控制

```
for (my $i=0; $i <= 9; $i++) {  
    print "$i\n";  
}
```

```
foreach my $aline (@array) {  
    print $aline, "\n";  
}
```

```
while (my $aline=<IN>) {  
    chomp $aline;  
    print $aline, "\n";  
}
```

練習三：  
指定一個 array  
並逐列印出來

# 目錄，檔案控制，輸出與輸入

```
opendir(DIR, "/XXX/XXX/XXX");  
my @array = readdir(DIR);  
foreach my $aline (@array) {  
    print $aline, "\n";  
}  
close DIR;
```

透過 open 功能  
可開啟目錄或檔案

```
open(IN, "/XXX/XXX/A_FILE");  
while (my $aline=<IN>) {  
    chomp $aline;  
    print $aline, "\n";  
}  
close IN;
```

如果要寫入檔案時  
除了要在 open 功能中加入  
'>' 箭號之外，在輸出指令  
後方要加入檔案輸出標記

```
open(IN, ">/XXX/XXX/A_FILE");  
foreach my $aline (@array) {  
    print IN $aline, "\n";  
}  
close IN;
```

# Perl PBS Module

---

- <https://hackmd.io/@jimmy200340/rkSg-RXlv>