pipe & fifo

中正大學，作業系統實驗室
羅習五 陽春副教授
shiwulo@gmail.com

大綱

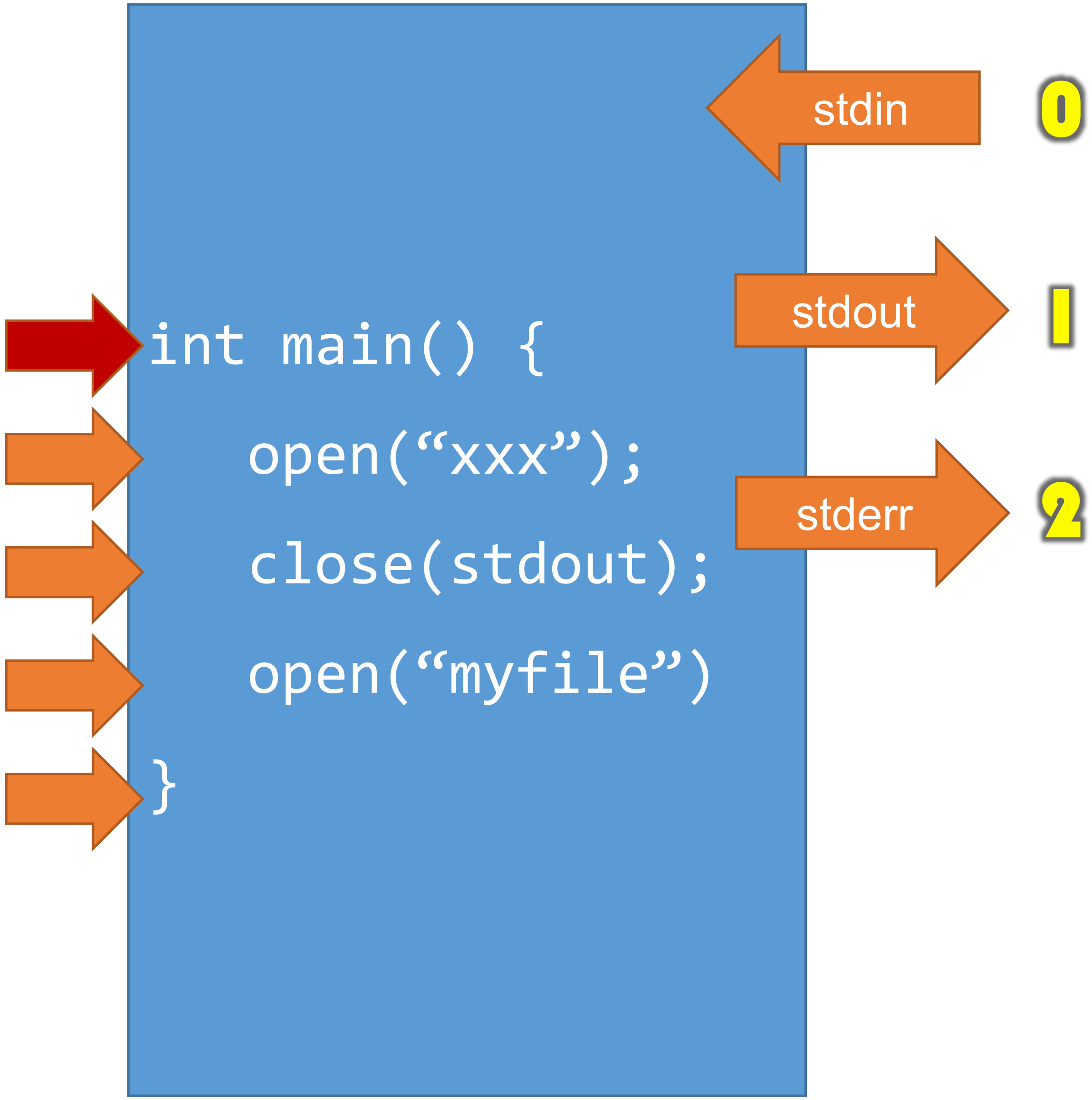
- 使用bash的pipe功能
- close，open時「作業系統」對file descript的指定方式
- pipefd函數
- 程式範例
- 使用process group，以處理ctr-c（SIGINT）
- 量測效能（SIGALARM）

使用pipe串接程式

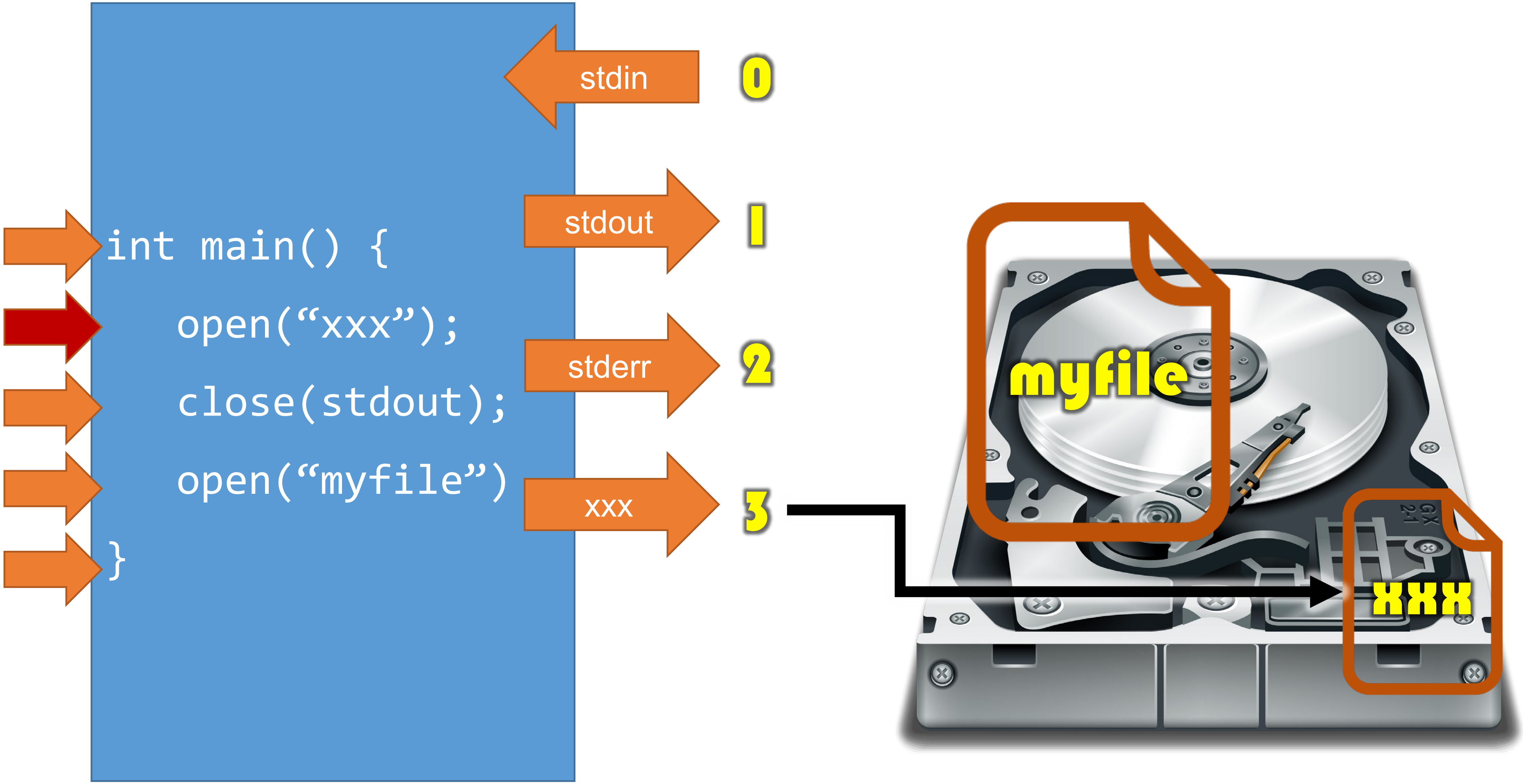
使用命令列（使用「|」）

```
shiwulo@vm:~/sp/ch11$ less fifo1.c | wc -l
22
less fifo1.c | grep "#include"
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
```

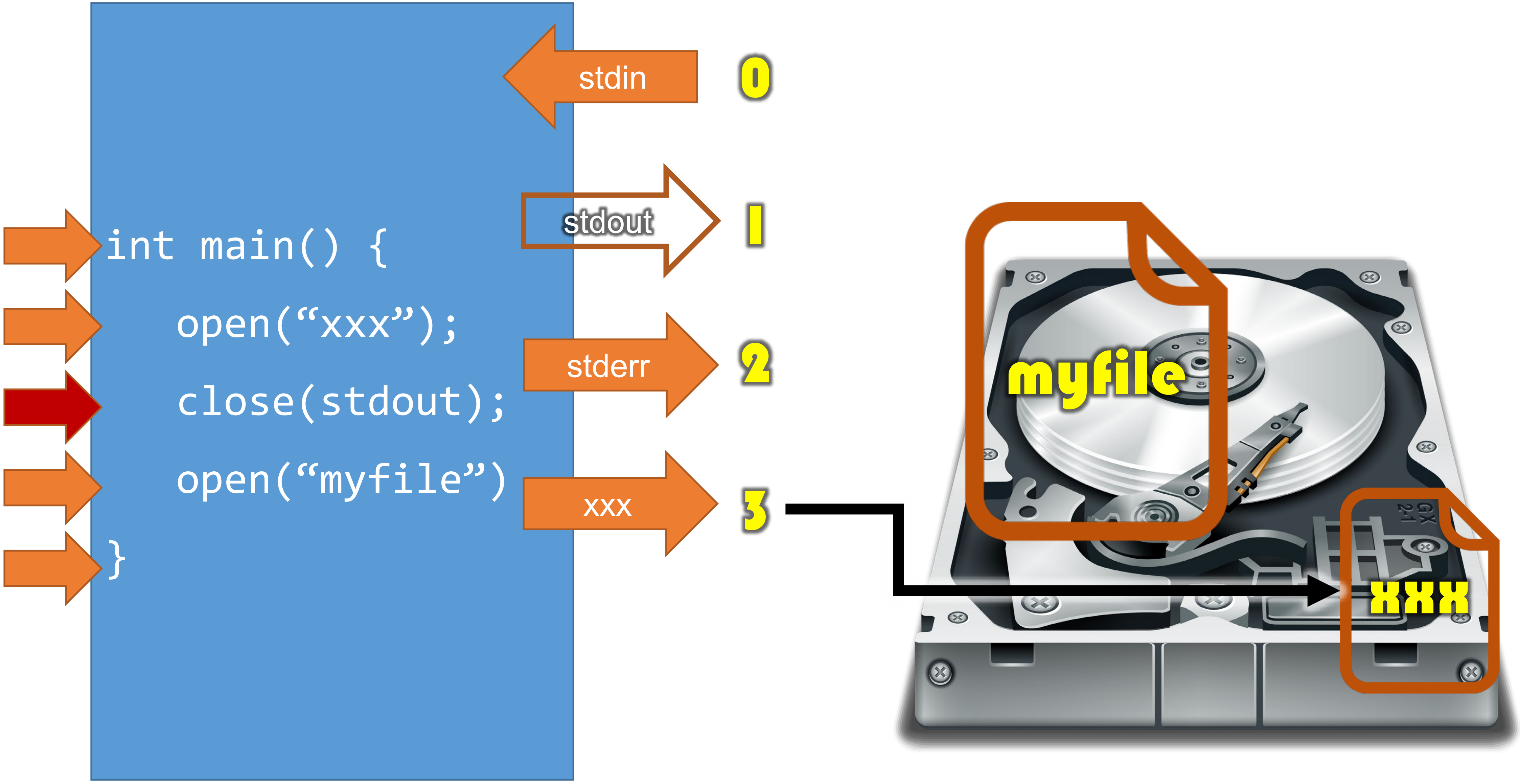
close，open時「作業系統」對file descript的指定方式



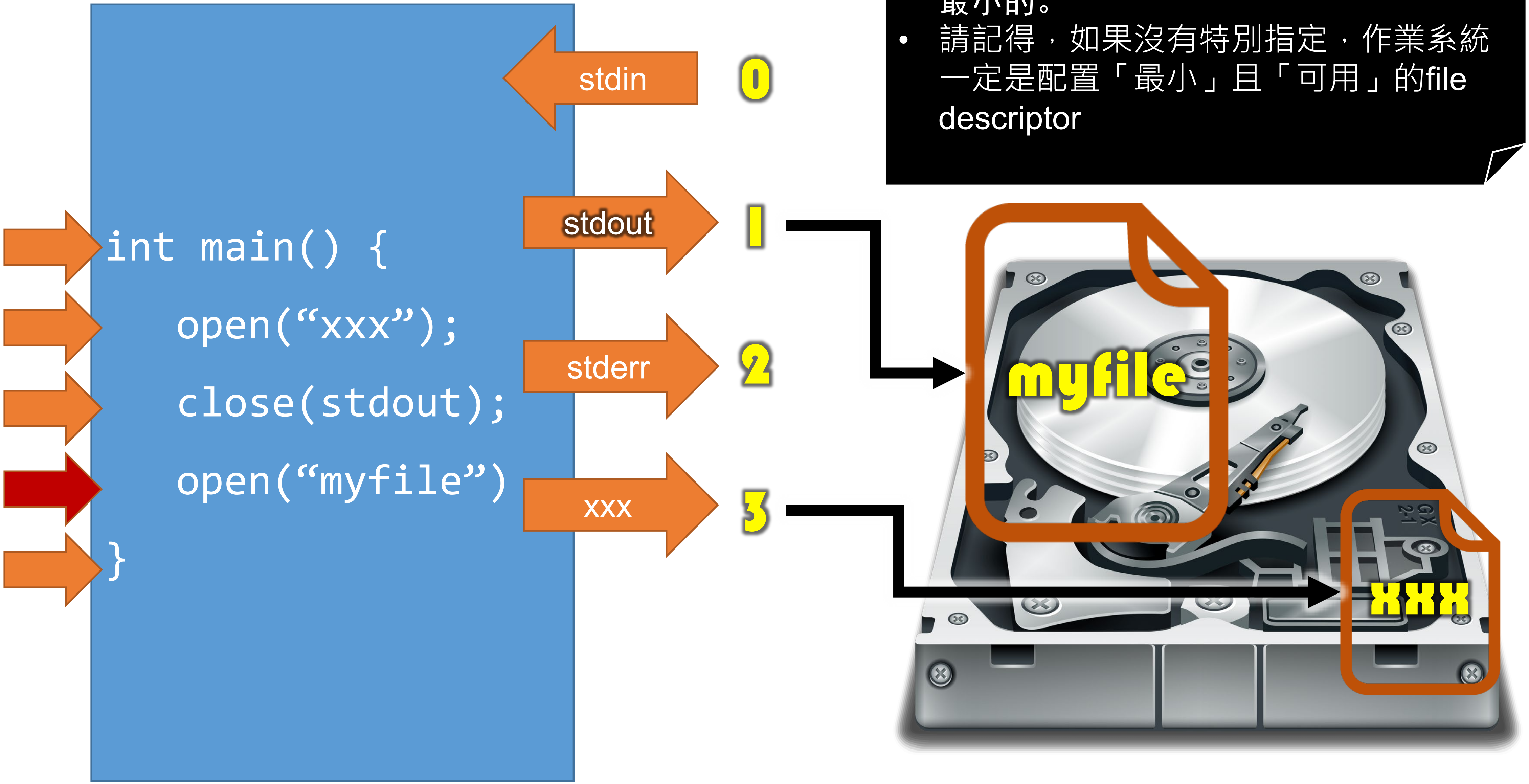
```
int main() {
    open(“xxx”);
    close(stdout);
    open(“myfile”)
}
```



```
int main() {
    open("xxx");
    close(stdout);
    open("myfile")
}
```

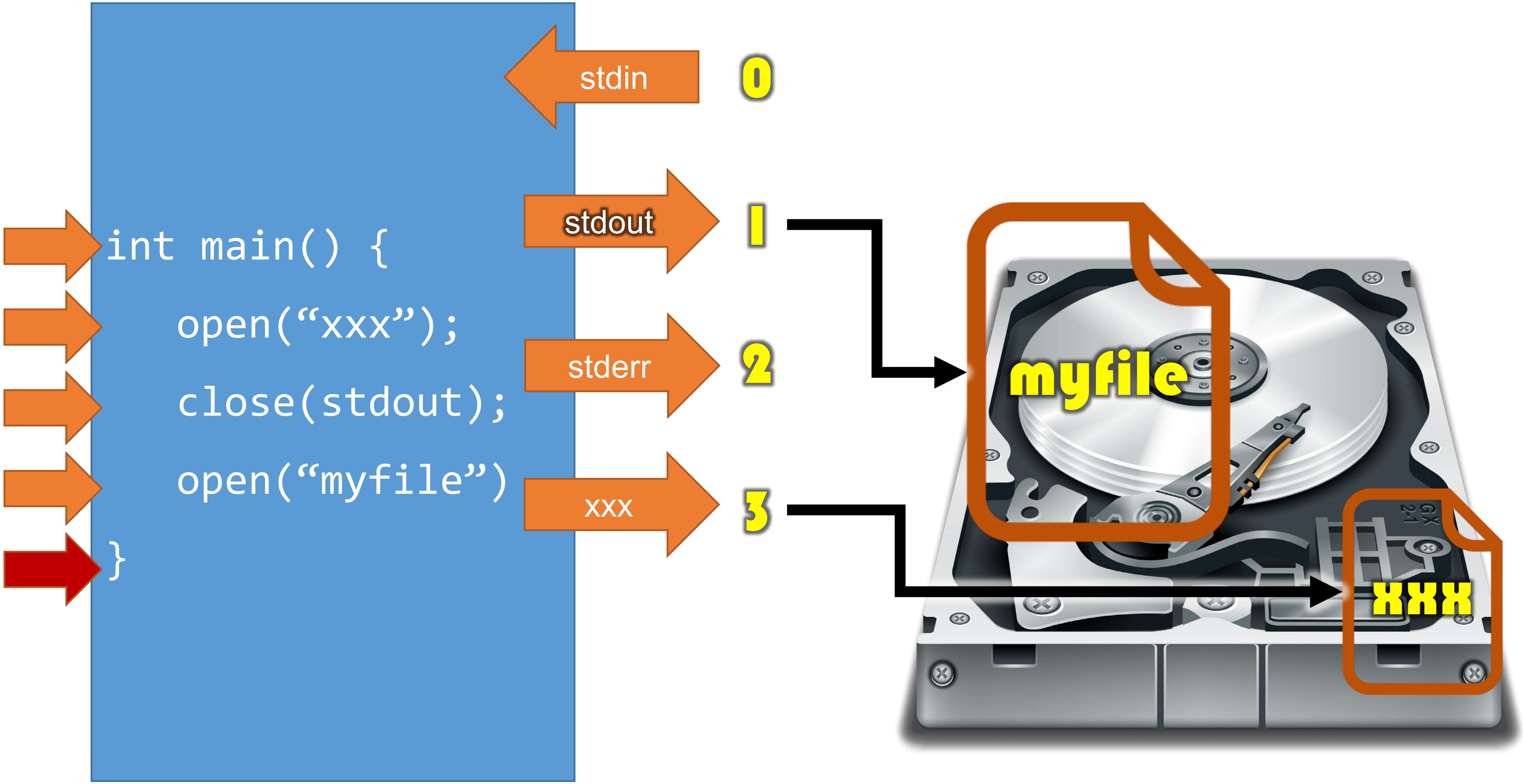


```
int main() {
    open(“xxx”);
    close(stdout);
    open(“myfile”)
}
```



```
int main() {
    open(“xxx”);
    close(stdout);
    open(“myfile”)
}
```

- 此步驟配置的file descriptor一定是編號最小的。
- 請記得，如果沒有特別指定，作業系統一定是配置「最小」且「可用」的file descriptor



```
int main() {
    open(“xxx”);
    close(stdout);
    open(“myfile”)
}
```

stdin 0

stdout 1

stderr 2

xxx 3

pipefd函數

pipe()

- #include <unistd.h>
- int pipe(int pipefd[2]);

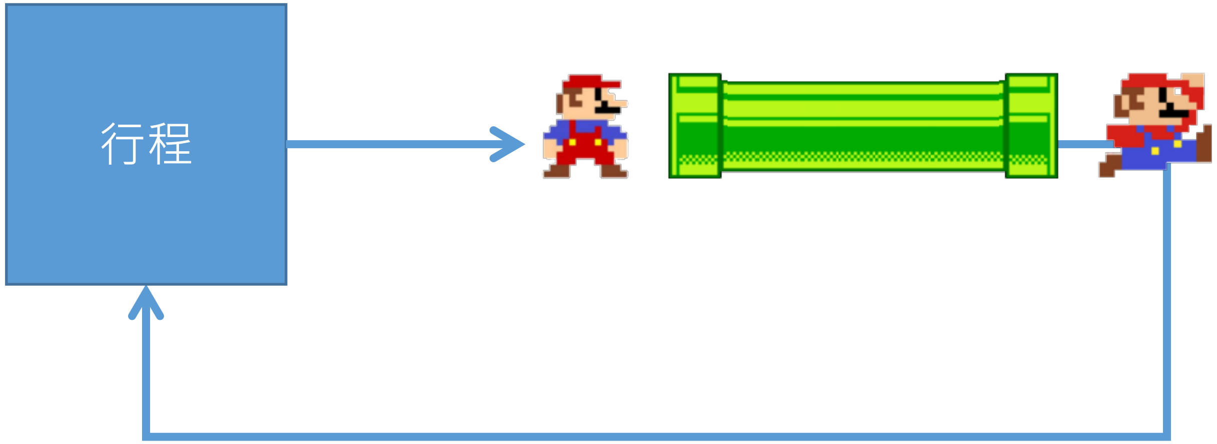
- 建立一個溝通的管道，pipefd[0]為讀取，pipefd[1]為寫入，如果發生錯誤，回傳值為-1，否則為0

使用pipe的簡單程式（pipe1.c)

```c
#include <unistd.h>
#include <stdio.h>

int main(int argc, char** argv) {
    int pipefd[2];
    char *str = "hello\n\0";
    char buf[200];
    pipe(pipefd);
    write(pipefd[1], str, strlen(str)+1));
    read(pipefd[0], buf, 200);
    printf("%s", buf);
    return 0;
}
```

示意圖



執行結果

```
shiwulo@vm:~/sp/ch11$ ./pipe1
hello
```

完整的範例

程式碼概念圖

main

1 0

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
            if (pid1 > 0)
             pid2=fork();
              if (pid2>0)
close(pipefd[0];
close(pipefd[1];
      wait();
      wait();
```

child 1

1 0

```
if (pid == 0) {
   //close stdout
   close(1);
   dup(pipefd[1]);
   //已經將pipe的fd[1]接上stdout
   //『一定要』關閉未使用的pipe fd
   //否則水龍頭會關不緊（收不到EOF）
   close(fd[0]); close(fd[1]);
   execlp( “ls”, “ls”, NULL);
}
```

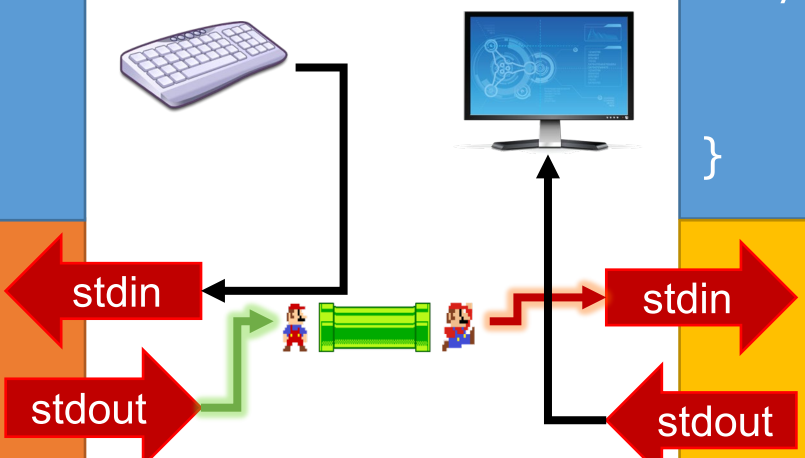
ls

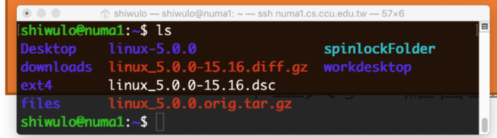
child 2

1 0

```
if(pid2 == 0) {
   //close stdin
   close(0);
   dup(pipefd[0]);
   //關上沒用的水龍頭
    close(fd[0]); close(fd[1]);
    execlp(“wc”, “wc”, NULL);
}
```

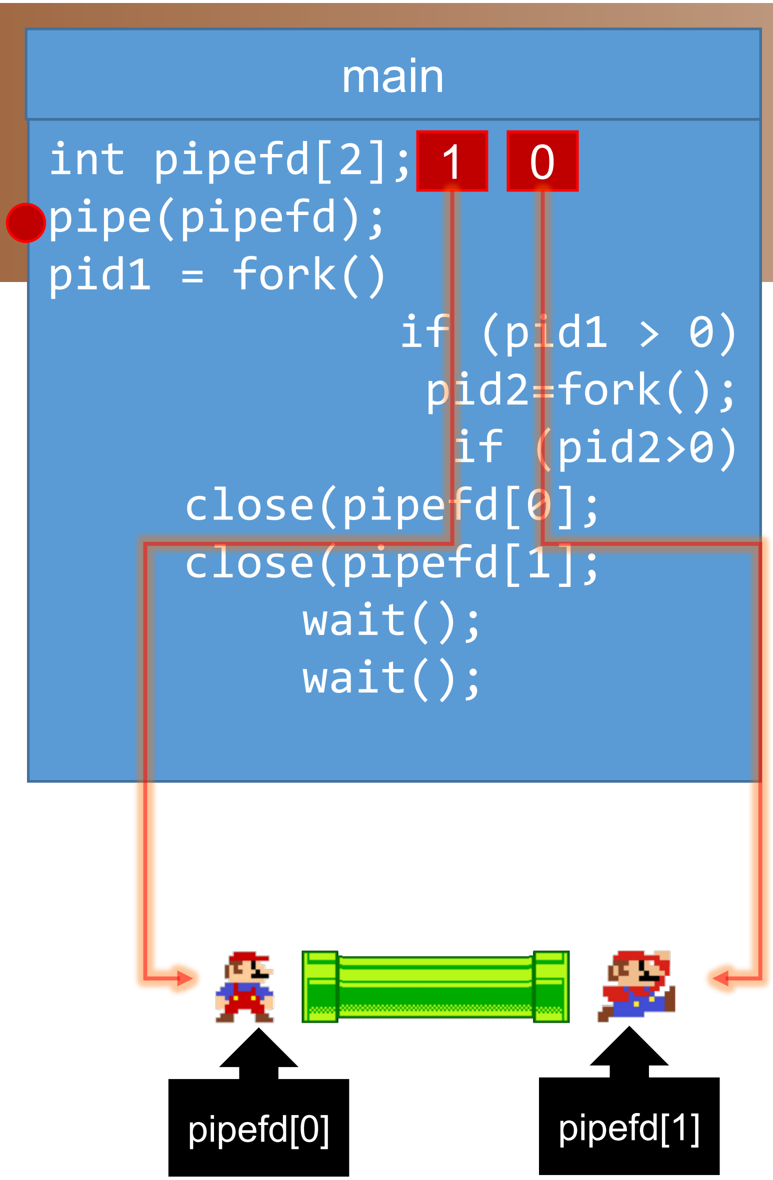
wc





```
shiwulo@numa1:~$ ls
Desktop    linux-5.0.0                 spinlockFolder
downloads  linux_5.0.0-15.16.diff.gz  workdesktop
ext4       linux_5.0.0-15.16.dsc
files      linux_5.0.0.orig.tar.gz
shiwulo@numa1:~$ 
```

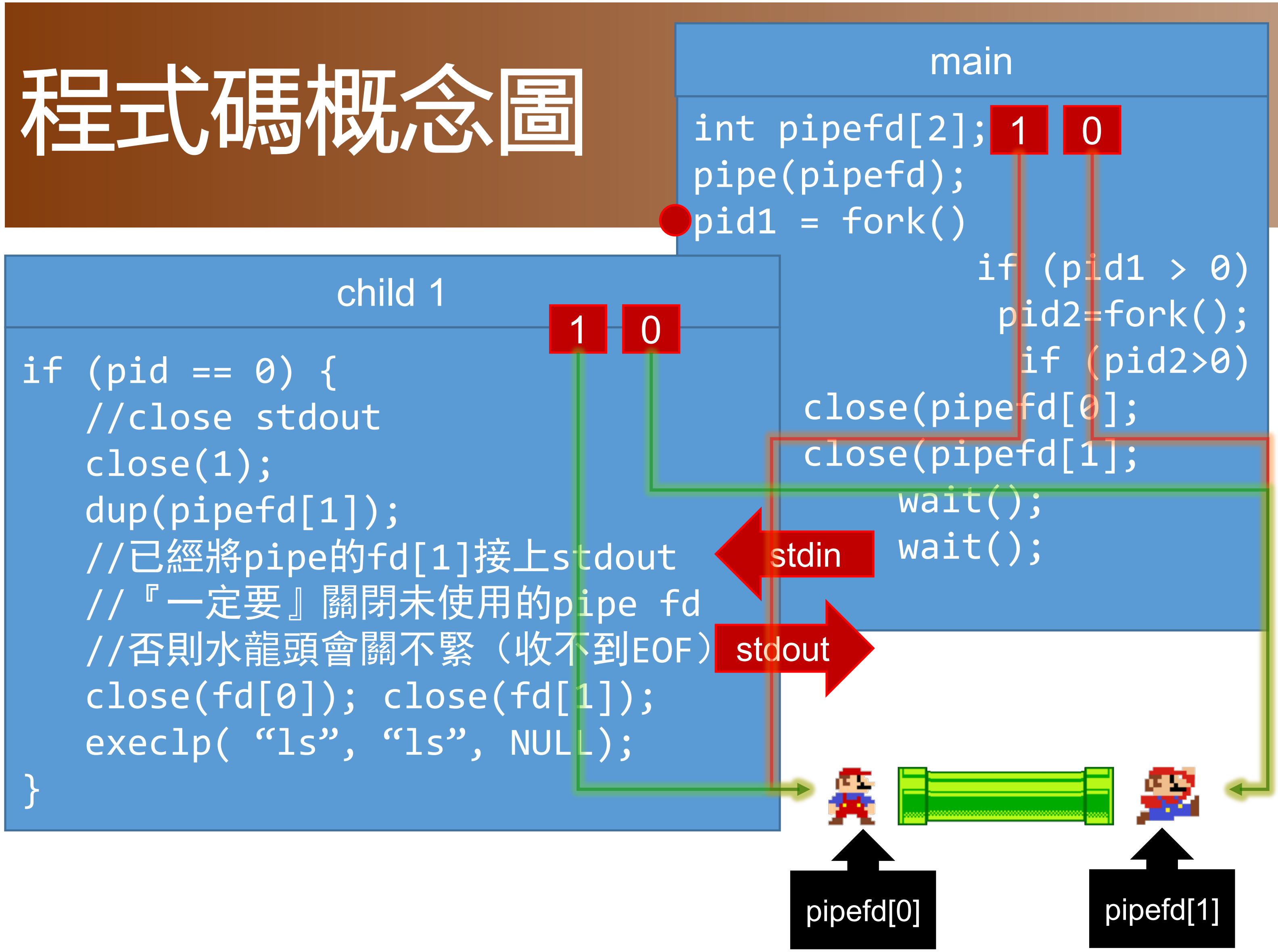
程式碼概念圖



main

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
              if (pid1 > 0)
               pid2=fork();
                if (pid2>0)
     close(pipefd[0]);
     close(pipefd[1]);
          wait();
          wait();
```

程式碼概念圖



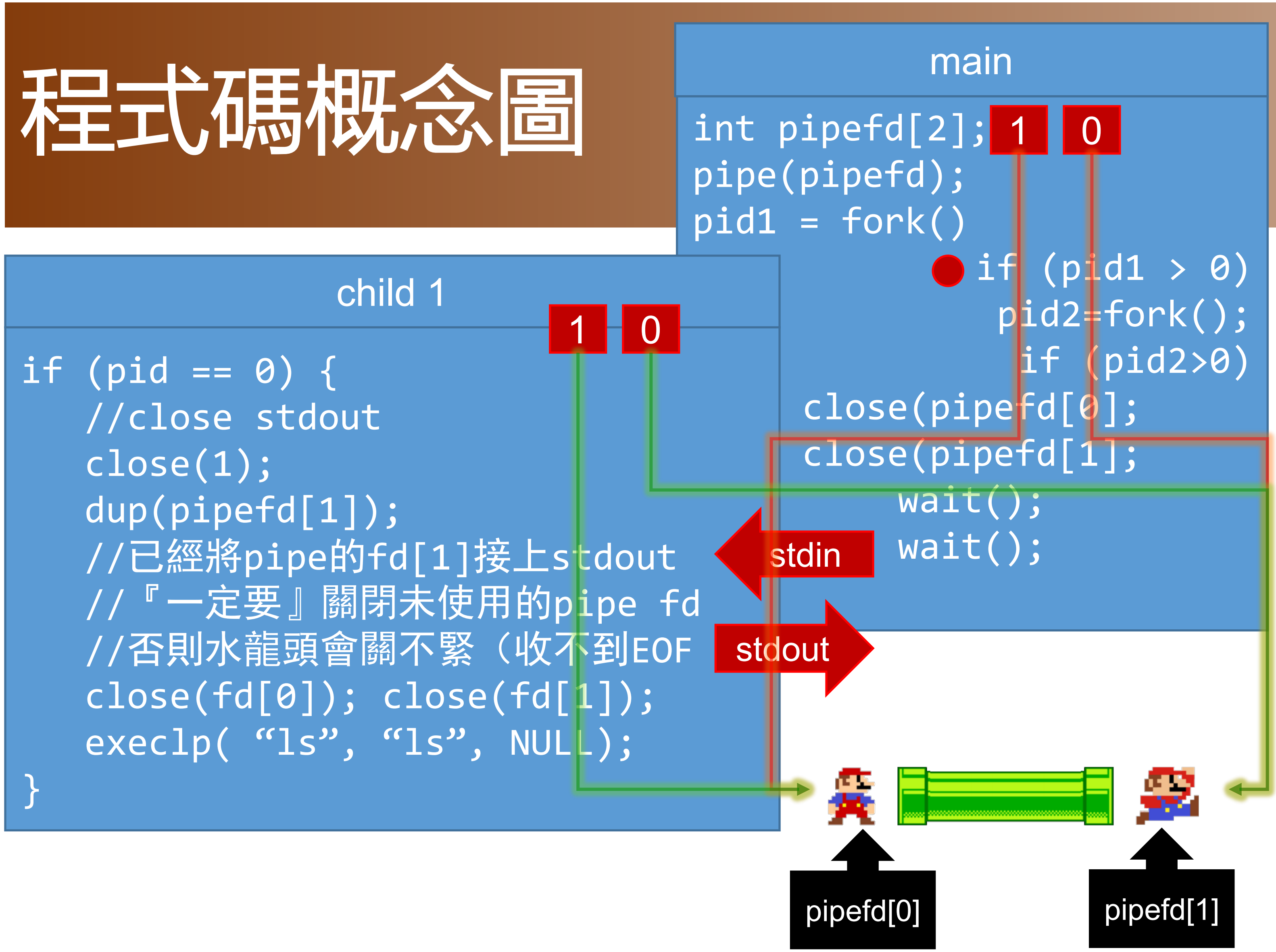
main

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
             if (pid1 > 0)
              pid2=fork();
               if (pid2>0)
close(pipefd[0]);
close(pipefd[1]);
      wait();
      wait();
```

child 1

```
if (pid == 0) {
    //close stdout
    close(1);
    dup(pipefd[1]);
    //已經將pipe的fd[1]接上stdout
    //『一定要』關閉未使用的pipe fd
    //否則水龍頭會關不緊（收不到EOF）
    close(fd[0]); close(fd[1]);
    execlp( “ls”, “ls”, NULL);
}
```

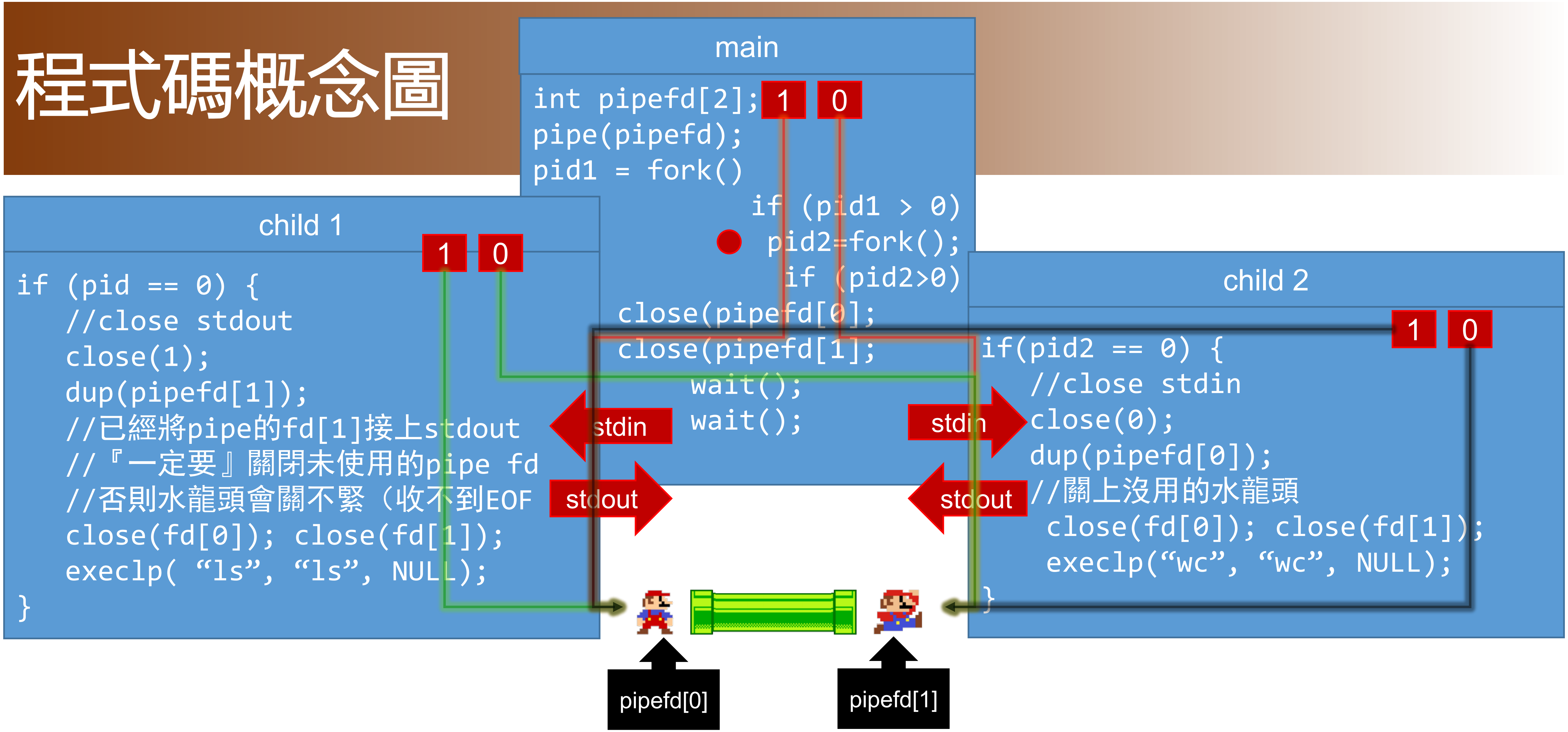
程式碼概念圖



```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
         if (pid1 > 0)
           pid2=fork();
             if (pid2>0)
     close(pipefd[0];
     close(pipefd[1];
           wait();
           wait();
```

```
if (pid == 0) {
   //close stdout
   close(1);
   dup(pipefd[1]);
   //已經將pipe的fd[1]接上stdout
   //『一定要』關閉未使用的pipe fd
   //否則水龍頭會關不緊（收不到EOF
   close(fd[0]); close(fd[1]);
   execlp( “ls”, “ls”, NULL);
}
```

程式碼概念圖



main

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
              if (pid1 > 0)
               pid2=fork();
                if (pid2>0)
     close(pipefd[0];
     close(pipefd[1];
           wait();
           wait();
```

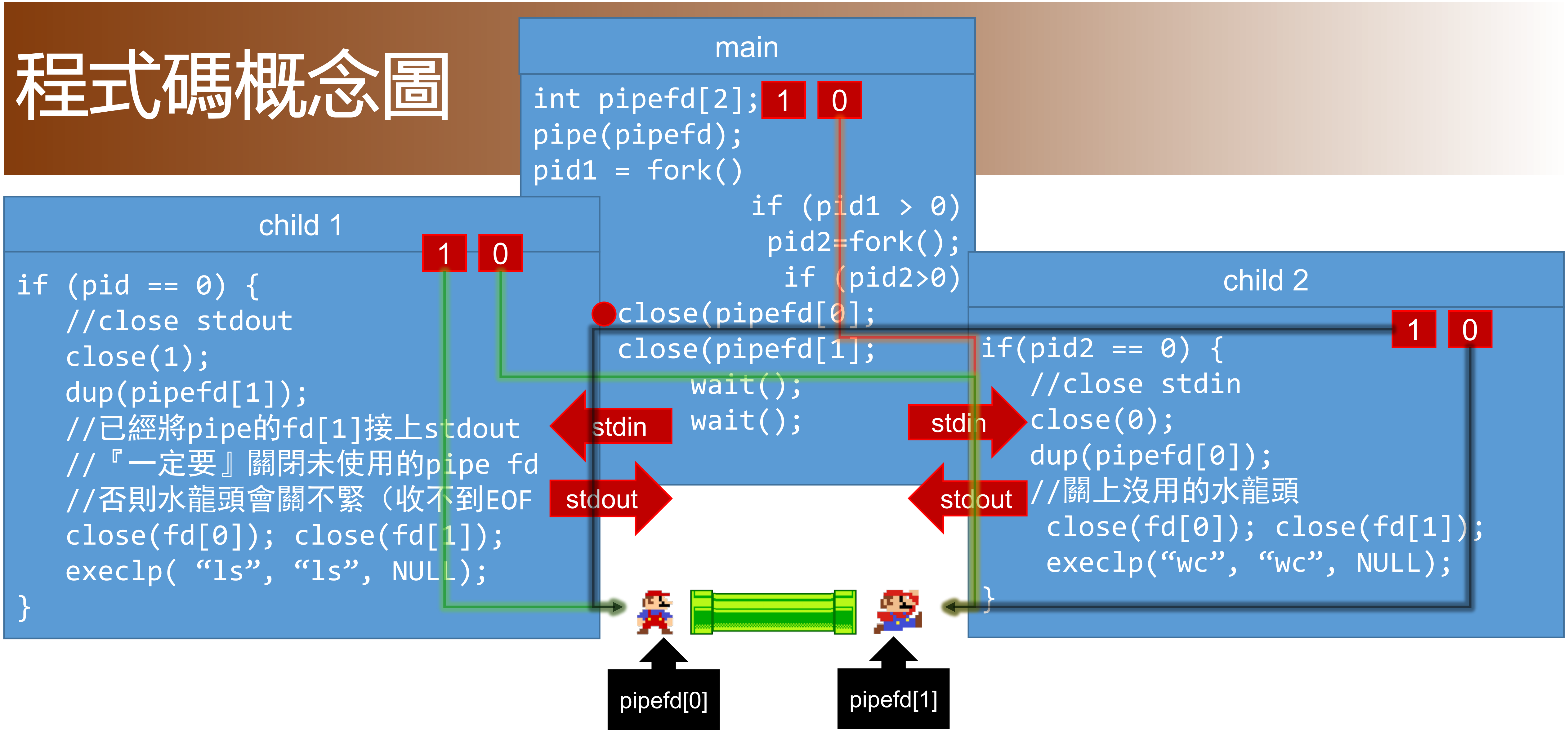
child 1

```
if (pid == 0) {
    //close stdout
    close(1);
    dup(pipefd[1]);
    //已經將pipe的fd[1]接上stdout
    //『一定要』關閉未使用的pipe fd
    //否則水龍頭會關不緊（收不到EOF
    close(fd[0]); close(fd[1]);
    execlp( “ls”, “ls”, NULL);
}
```

child 2

```
if(pid2 == 0) {
    //close stdin
    close(0);
    dup(pipefd[0]);
    //關上沒用的水龍頭
     close(fd[0]); close(fd[1]);
     execlp(“wc”, “wc”, NULL);
}
```

程式碼概念圖



main

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
            if (pid1 > 0)
             pid2=fork();
              if (pid2>0)
close(pipefd[0];
close(pipefd[1];
     wait();
     wait();
```

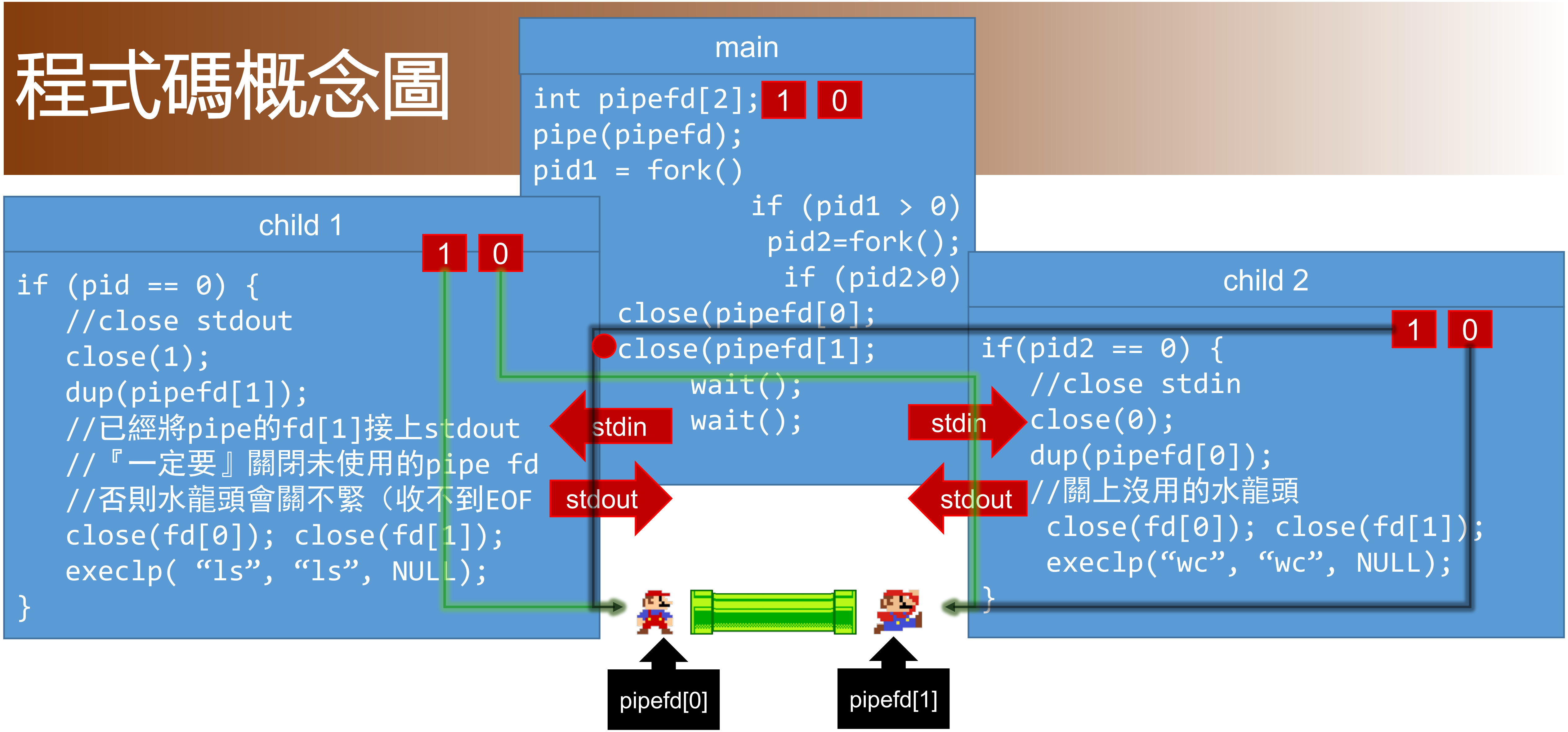
child 1

```
if (pid == 0) {
   //close stdout
   close(1);
   dup(pipefd[1]);
   //已經將pipe的fd[1]接上stdout
   //『一定要』關閉未使用的pipe fd
   //否則水龍頭會關不緊（收不到EOF
   close(fd[0]); close(fd[1]);
   execlp( “ls”, “ls”, NULL);
}
```

child 2

```
if(pid2 == 0) {
   //close stdin
   close(0);
   dup(pipefd[0]);
   //關上沒用的水龍頭
    close(fd[0]); close(fd[1]);
    execlp(“wc”, “wc”, NULL);
}
```

程式碼概念圖



main

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
              if (pid1 > 0)
               pid2=fork();
                if (pid2>0)
     close(pipefd[0];
     close(pipefd[1];
          wait();
          wait();
```

child 1

```
if (pid == 0) {
   //close stdout
   close(1);
   dup(pipefd[1]);
   //已經將pipe的fd[1]接上stdout
   //『一定要』關閉未使用的pipe fd
   //否則水龍頭會關不緊（收不到EOF
   close(fd[0]); close(fd[1]);
   execlp( “ls”, “ls”, NULL);
}
```

child 2

```
if(pid2 == 0) {
   //close stdin
   close(0);
   dup(pipefd[0]);
   //關上沒用的水龍頭
    close(fd[0]); close(fd[1]);
    execlp(“wc”, “wc”, NULL);
}
```

程式碼概念圖

main

```
int pipefd[2]; 1  0
pipe(pipefd);
pid1 = fork()
              if (pid1 > 0)
               pid2=fork();
                if (pid2>0)
close(pipefd[0];
close(pipefd[1];
     wait();
     wait();
```

child 1

1 0

```
if (pid == 0) {
   //close stdout
   close(1);
   dup(pipefd[1]);
   //已經將pipe的fd[1]接上stdout
   //『一定要』關閉未使用的pipe fd
   //否則水龍頭會關不緊（收不到EOF）
   close(fd[0]); close(fd[1]);
   execlp( “ls”, “ls”, NULL);
}
```

stdin

stdout

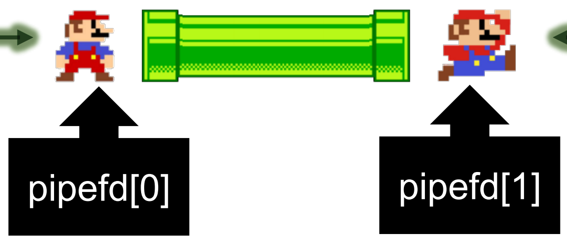
child 2

1 0

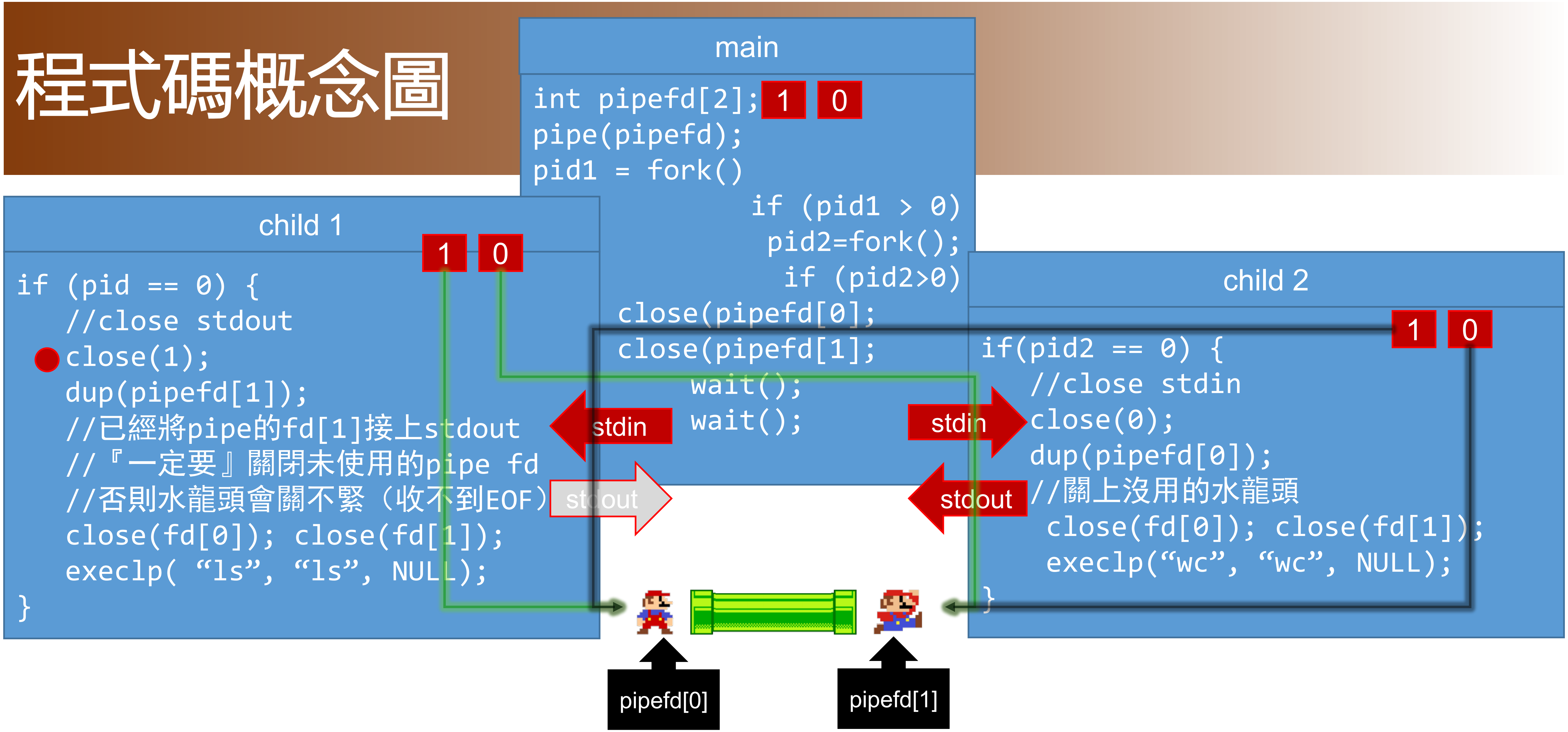
```
if(pid2 == 0) {
   //close stdin
   close(0);
   dup(pipefd[0]);
   //關上沒用的水龍頭
    close(fd[0]); close(fd[1]);
    execlp(“wc”, “wc”, NULL);
}
```

stdin

stdout



程式碼概念圖



main

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
          if (pid1 > 0)
           pid2=fork();
            if (pid2>0)
close(pipefd[0];
close(pipefd[1];
    wait();
    wait();
```

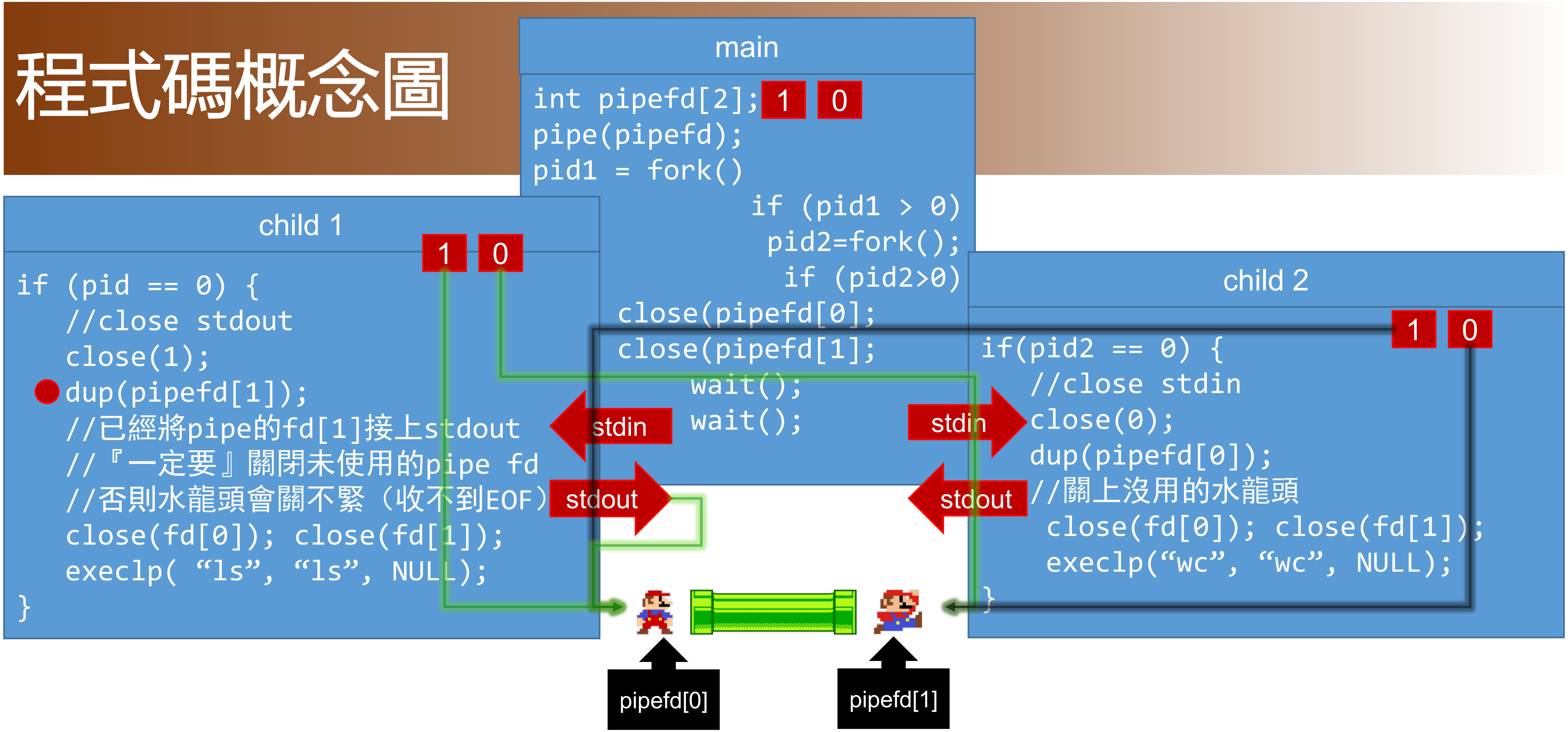
child 1

```
if (pid == 0) {
   //close stdout
   close(1);
   dup(pipefd[1]);
   //已經將pipe的fd[1]接上stdout
   //『一定要』關閉未使用的pipe fd
   //否則水龍頭會關不緊（收不到EOF）
   close(fd[0]); close(fd[1]);
   execlp( “ls”, “ls”, NULL);
}
```

child 2

```
if(pid2 == 0) {
   //close stdin
   close(0);
   dup(pipefd[0]);
   //關上沒用的水龍頭
    close(fd[0]); close(fd[1]);
    execlp(“wc”, “wc”, NULL);
}
```

程式碼概念圖



main

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
              if (pid1 > 0)
               pid2=fork();
                if (pid2>0)
     close(pipefd[0];
     close(pipefd[1];
            wait();
            wait();
```

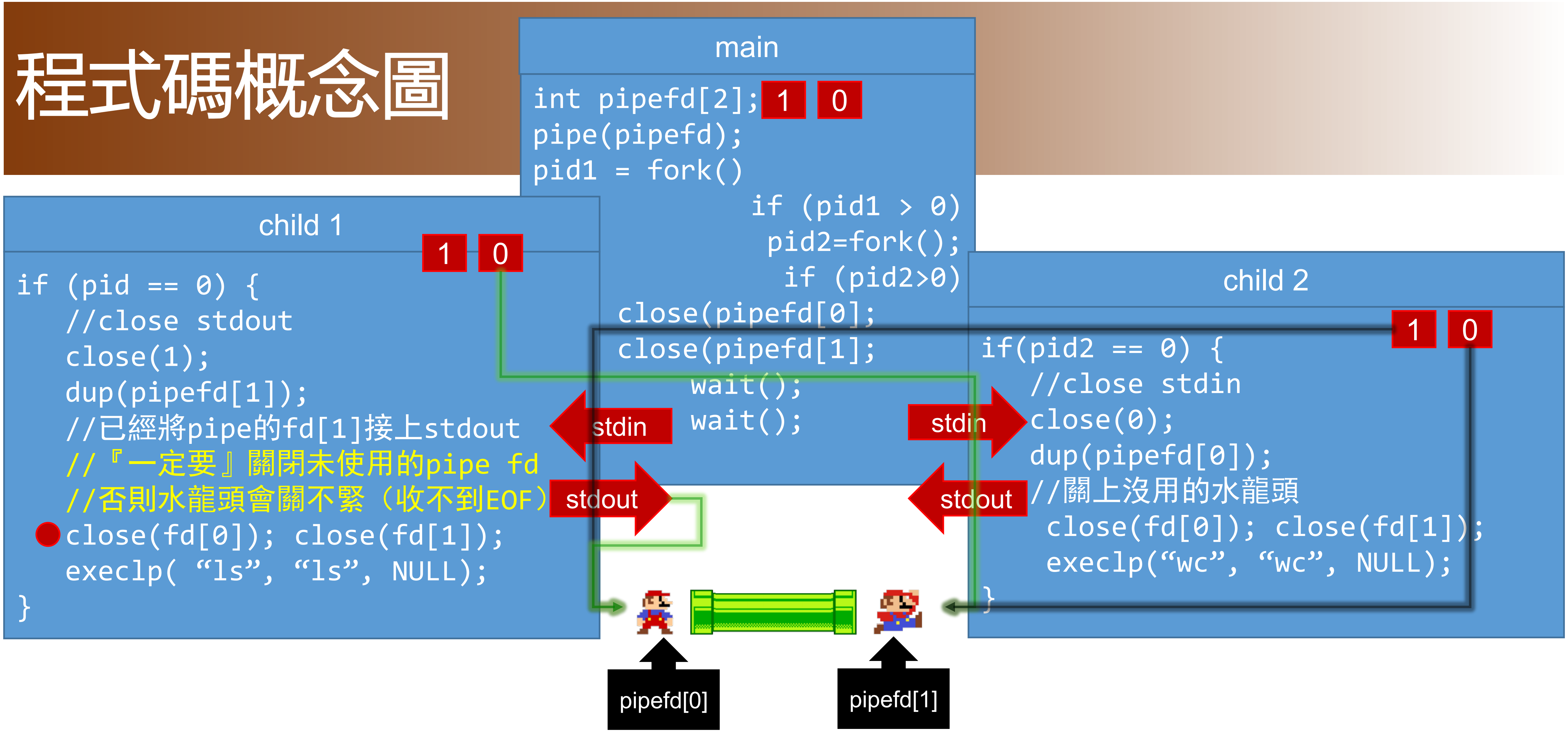
child 1

```
if (pid == 0) {
   //close stdout
   close(1);
 ● dup(pipefd[1]);
   //已經將pipe的fd[1]接上stdout
   //『一定要』關閉未使用的pipe fd
   //否則水龍頭會關不緊（收不到EOF）
   close(fd[0]); close(fd[1]);
   execlp( “ls”, “ls”, NULL);
}
```

child 2

```
if(pid2 == 0) {
   //close stdin
   close(0);
   dup(pipefd[0]);
   //關上沒用的水龍頭
    close(fd[0]); close(fd[1]);
    execlp(“wc”, “wc”, NULL);
}
```

程式碼概念圖



main

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
              if (pid1 > 0)
               pid2=fork();
                if (pid2>0)
    close(pipefd[0];
    close(pipefd[1];
          wait();
          wait();
```

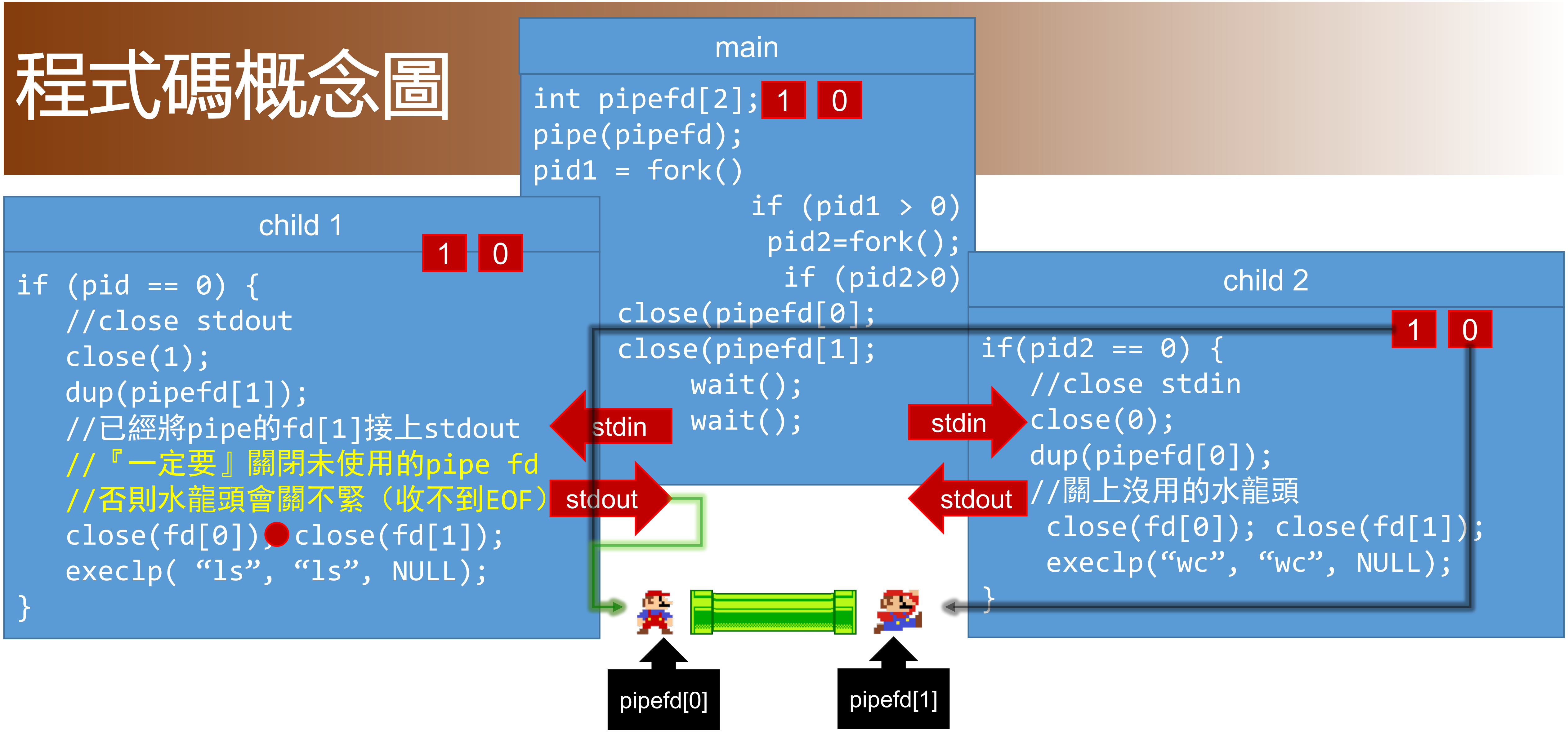
child 1

```
if (pid == 0) {
   //close stdout
   close(1);
   dup(pipefd[1]);
   //已經將pipe的fd[1]接上stdout
   //『一定要』關閉未使用的pipe fd
   //否則水龍頭會關不緊（收不到EOF）
   close(fd[0]); close(fd[1]);
   execlp( “ls”, “ls”, NULL);
}
```

child 2

```
if(pid2 == 0) {
   //close stdin
   close(0);
   dup(pipefd[0]);
   //關上沒用的水龍頭
    close(fd[0]); close(fd[1]);
    execlp(“wc”, “wc”, NULL);
}
```

程式碼概念圖



main

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
          if (pid1 > 0)
           pid2=fork();
            if (pid2>0)
close(pipefd[0];
close(pipefd[1];
     wait();
     wait();
```

child 1

```
if (pid == 0) {
   //close stdout
   close(1);
   dup(pipefd[1]);
   //已經將pipe的fd[1]接上stdout
   //『一定要』關閉未使用的pipe fd
   //否則水龍頭會關不緊（收不到EOF）
   close(fd[0])●close(fd[1]);
   execlp( “ls”, “ls”, NULL);
}
```

child 2

```
if(pid2 == 0) {
   //close stdin
   close(0);
   dup(pipefd[0]);
   //關上沒用的水龍頭
    close(fd[0]); close(fd[1]);
    execlp(“wc”, “wc”, NULL);
}
```

程式碼概念圖

main

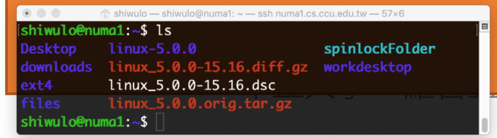
```
int pipefd[2]; 1 0
pipe(pipefd);
pid1 = fork()
              if (pid1 > 0)
               pid2=fork();
                if (pid2>0)
close(pipefd[0];
close(pipefd[1];
      wait();
      wait();
```

child 1

1 0

```
if (pid == 0) {
   //close stdout
   close(1);
   dup(pipefd[1]);
   //已經將pipe的fd[1]接上stdout
   //『一定要』關閉未使用的pipe fd
   //否則水龍頭會關不緊（收不到EOF）
   close(fd[0]); close(fd[1]);
 ●execlp( “ls”, “ls”, NULL);
}
```

ls



```
shiwulo@numa1:~$ ls
Desktop    linux-5.0.0                 spinlockFolder
downloads  linux_5.0.0-15.16.diff.gz   workdesktop
ext4       linux_5.0.0-15.16.dsc
files      linux_5.0.0.orig.tar.gz
shiwulo@numa1:~$
```

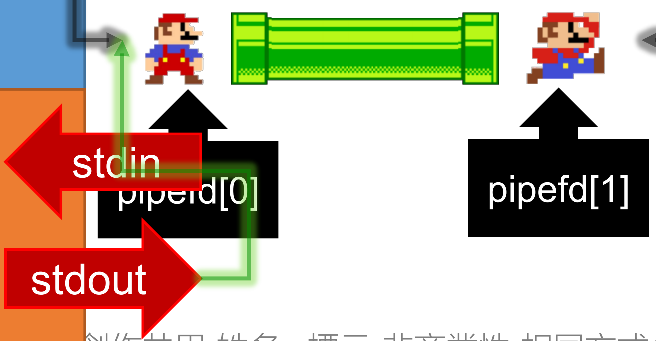
child 2

1 0

stdin

stdout

```
if(pid2 == 0) {
   //close stdin
   close(0);
   dup(pipefd[0]);
   //關上沒用的水龍頭
    close(fd[0]); close(fd[1]);
    execlp(“wc”, “wc”, NULL);
}
```



程式碼概念圖

main

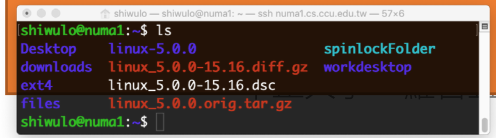
```
int pipefd[2]; 1  0
pipe(pipefd);
pid1 = fork()
           if (pid1 > 0)
            pid2=fork();
             if (pid2>0)
close(pipefd[0];
close(pipefd[1];
     wait();
     wait();
```

child 1

1 0

```
if (pid == 0) {
    //close stdout
    close(1);
    dup(pipefd[1]);
    //已經將pipe的fd[1]接上stdout
    //『一定要』關閉未使用的pipe fd
    //否則水龍頭會關不緊（收不到EOF）
    close(fd[0]); close(fd[1]);
    execlp( “ls”, “ls”, NULL);
}
```

ls



```
shiwulo@numa1:~$ ls
Desktop    linux-5.0.0                  spinlockFolder
downloads  linux_5.0.0-15.16.diff.gz  workdesktop
ext4       linux_5.0.0-15.16.dsc
files      linux_5.0.0.orig.tar.gz
shiwulo@numa1:~$ 
```

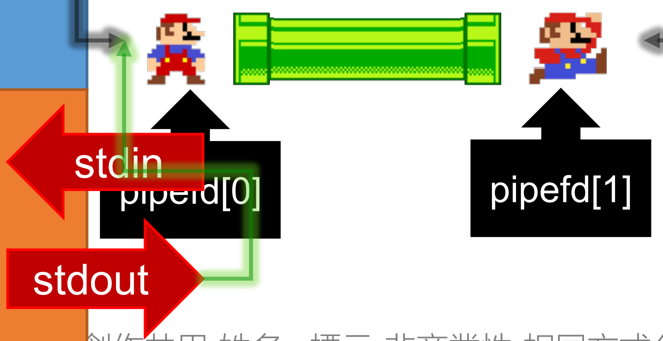
child 2

1 0

```
if(pid2 == 0) {
    //close stdin
    close(0);
    dup(pipefd[0]);
    //關上沒用的水龍頭
     close(fd[0]); close(fd[1]);
     execlp(“wc”, “wc”, NULL);
}
```

stdin

stdout



創作共用-姓名 標示-非商業性-相同方式分享
CC-BY-NC-SA

程式碼概念圖

main

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
            if (pid1 > 0)
             pid2=fork();
              if (pid2>0)
close(pipefd[0]);
close(pipefd[1];
      wait();
      wait();
```

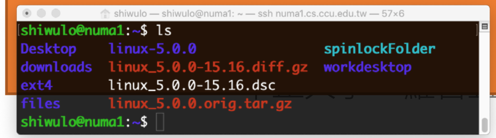
1 0

child 1

1 0

```
if (pid == 0) {
    //close stdout
    close(1);
    dup(pipefd[1]);
    //已經將pipe的fd[1]接上stdout
    //『一定要』關閉未使用的pipe fd
    //否則水龍頭會關不緊（收不到EOF）
    close(fd[0]); close(fd[1]);
    execlp( “ls”, “ls”, NULL);
}
```

ls



```
shiwulo — shiwulo@numa1: ~ — ssh numa1.cs.ccu.edu.tw — 57×6
shiwulo@numa1:~$ ls
Desktop    linux-5.0.0                  spinlockFolder
downloads  linux_5.0.0-15.16.diff.gz  workdesktop
ext4       linux_5.0.0-15.16.dsc
files      linux_5.0.0.orig.tar.gz
shiwulo@numa1:~$ 
```

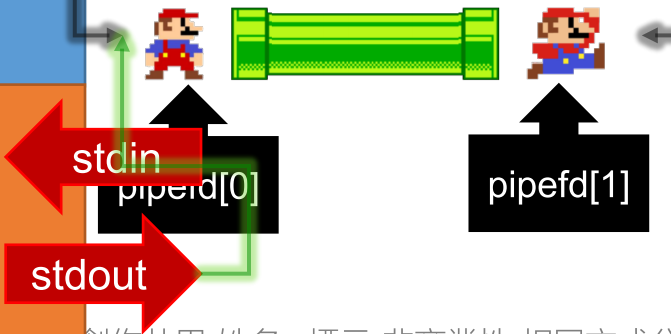
child 2

1 0

stdin

stdout

```
if(pid2 == 0) {
    //close stdin
    close(0);
    dup(pipefd[0]);
    //關上沒用的水龍頭
     close(fd[0]); close(fd[1]);
     execlp(“wc”, “wc”, NULL);
}
```



程式碼概念圖

main

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
            if (pid1 > 0)
             pid2=fork();
              if (pid2>0)
     close(pipefd[0]);
     close(pipefd[1]);
          wait();
          wait();
```

1 0

child 1

1 0

```
if (pid == 0) {
   //close stdout
   close(1);
   dup(pipefd[1]);
   //已經將pipe的fd[1]接上stdout
   //『一定要』關閉未使用的pipe fd
   //否則水龍頭會關不緊（收不到EOF）
   close(fd[0]); close(fd[1]);
   execlp( “ls”, “ls”, NULL);
}
```

ls

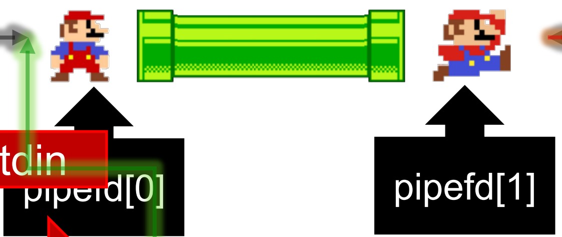
child 2

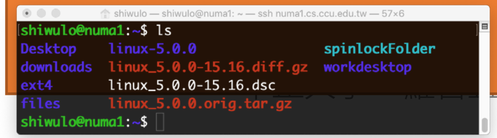
1 0

```
if(pid2 == 0) {
   //close stdin
   close(0);
   dup(pipefd[0]);
   //關上沒用的水龍頭
    close(fd[0]); close(fd[1]);
    execlp(“wc”, “wc”, NULL);
}
```

stdin

stdout





```
shiwulo@numa1:~$ ls
Desktop    linux-5.0.0                     spinlockFolder
downloads  linux_5.0.0-15.16.diff.gz  workdesktop
ext4       linux_5.0.0-15.16.dsc
files      linux_5.0.0.orig.tar.gz
shiwulo@numa1:~$
```

程式碼概念圖

main

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
             if (pid1 > 0)
              pid2=fork();
               if (pid2>0)
     close(pipefd[0];
     close(pipefd[1];
           wait();
           wait();
```

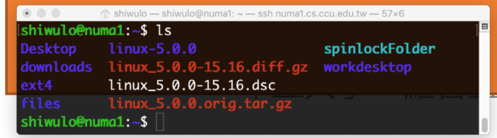
1 0

child 1

1 0

```
if (pid == 0) {
    //close stdout
    close(1);
    dup(pipefd[1]);
    //已經將pipe的fd[1]接上stdout
    //『一定要』關閉未使用的pipe fd
    //否則水龍頭會關不緊（收不到EOF）
    close(fd[0]); close(fd[1]);
    execlp( “ls”, “ls”, NULL);
}
```

ls



```
shiwulo@numa1:~$ ls
Desktop    linux-5.0.0                  spinlockFolder
downloads  linux_5.0.0-15.16.diff.gz    workdesktop
ext4       linux_5.0.0-15.16.dsc
files      linux_5.0.0.orig.tar.gz
shiwulo@numa1:~$
```

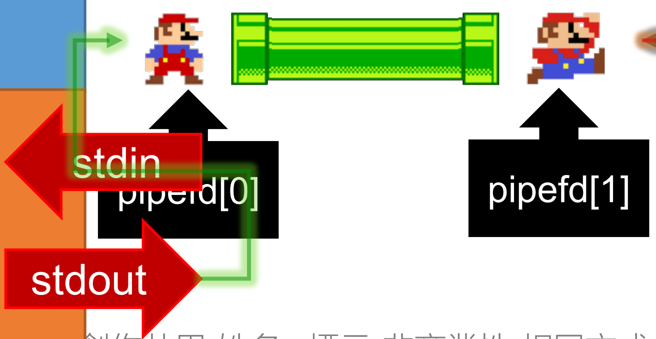
child 2

1 0

```
if(pid2 == 0) {
    //close stdin
    close(0);
    dup(pipefd[0]);
    //關上沒用的水龍頭
     close(fd[0]); close(fd[1]);
     execlp(“wc”, “wc”, NULL);
}
```

stdin

stdout



程式碼概念圖

main

```
int pipefd[2]; 1 0
pipe(pipefd);
pid1 = fork()
          if (pid1 > 0)
           pid2=fork();
            if (pid2>0)
close(pipefd[0];
close(pipefd[1];
     wait();
     wait();
```

child 1

1 0

```
if (pid == 0) {
   //close stdout
   close(1);
   dup(pipefd[1]);
   //已經將pipe的fd[1]接上stdout
   //『一定要』關閉未使用的pipe fd
   //否則水龍頭會關不緊（收不到EOF）
   close(fd[0]); close(fd[1]);
   execlp( “ls”, “ls”, NULL);
}
```

child 2

1 0

```
if(pid2 == 0) {
   //close stdin
   close(0);
   dup(pipefd[0]);
   //關上沒用的水龍頭
    close(fd[0]); close(fd[1]);
    execlp(“wc”, “wc”, NULL);
}
```

stdin

stdout

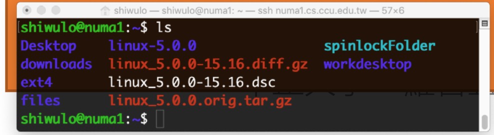
stdin

pipefd[0]

stdout

pipefd[1]

ls



```
shiwulo — shiwulo@numa1: ~ — ssh numa1.cs.ccu.edu.tw — 57×6
shiwulo@numa1:~$ ls
Desktop    linux-5.0.0                    spinlockFolder
downloads  linux_5.0.0-15.16.diff.gz  workdesktop
ext4       linux_5.0.0-15.16.dsc
files      linux_5.0.0.orig.tar.gz
shiwulo@numa1:~$ 
```

程式碼概念圖

main

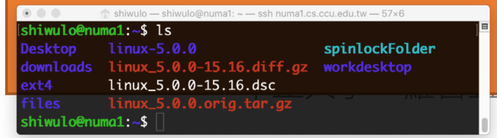
```
int pipefd[2]; 1 0
pipe(pipefd);
pid1 = fork()
            if (pid1 > 0)
             pid2=fork();
              if (pid2>0)
     close(pipefd[0];
     close(pipefd[1];
           wait();
           wait();
```

child 1

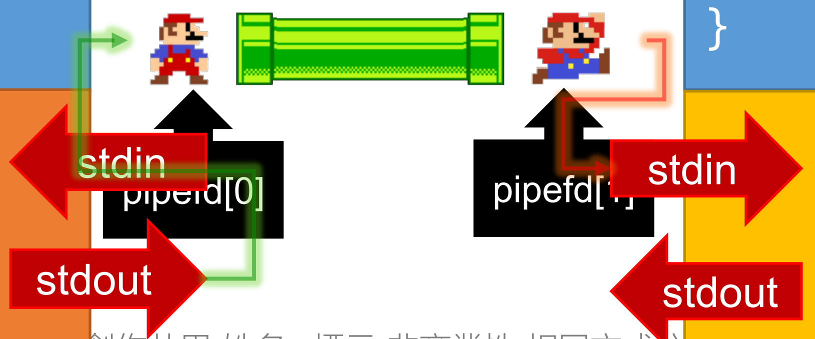
1 0

```
if (pid == 0) {
   //close stdout
   close(1);
   dup(pipefd[1]);
   //已經將pipe的fd[1]接上stdout
   //『一定要』關閉未使用的pipe fd
   //否則水龍頭會關不緊（收不到EOF）
   close(fd[0]); close(fd[1]);
   execlp( “ls”, “ls”, NULL);
}
```

ls



```
shiwulo — shiwulo@numa1: ~ — ssh numa1.cs.ccu.edu.tw — 57×6
shiwulo@numa1:~$ ls
Desktop    linux-5.0.0                  spinlockFolder
downloads  linux_5.0.0-15.16.diff.gz    workdesktop
ext4       linux_5.0.0-15.16.dsc
files      linux_5.0.0.orig.tar.gz
shiwulo@numa1:~$ 
```



child 2

1 0

```
if(pid2 == 0) {
   //close stdin
   close(0);
   dup(pipefd[0]);
   //關上沒用的水龍頭
    close(fd[0]); close(fd[1]);
    execlp(“wc”, “wc”, NULL);
}
```

wc

程式碼概念圖

main

```
int pipefd[2]; 1 0
pipe(pipefd);
pid1 = fork()
                   if (pid1 > 0)
                    pid2=fork();
                     if (pid2>0)
     close(pipefd[0];
     close(pipefd[1];
          wait();
          wait();
```

child 1

1 0

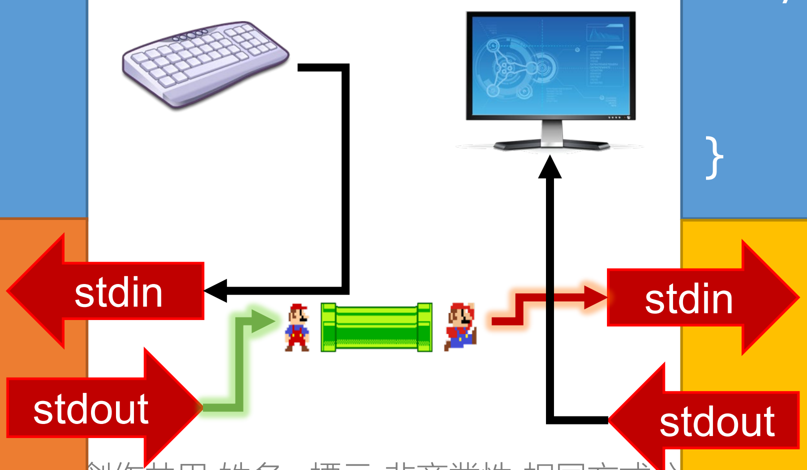
```
if (pid == 0) {
    //close stdout
    close(1);
    dup(pipefd[1]);
    //已經將pipe的fd[1]接上stdout
    //『一定要』關閉未使用的pipe fd
    //否則水龍頭會關不緊（收不到EOF）
    close(fd[0]); close(fd[1]);
    execlp( “ls”, “ls”, NULL);
}
```

終於
完成「人體蜈蚣」

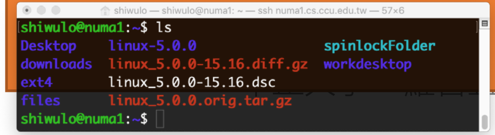
請掌聲鼓勵

1 0

```
if(pid2 == 0) {
    //close stdin
    close(0);
    dup(pipefd[0]);
    //關上沒用的水龍頭
     close(fd[0]); close(fd[1]);
     execlp(“wc”, “wc”, NULL);
}
```



ls



```
shiwulo — shiwulo@numa1: ~ — ssh numa1.cs.ccu.edu.tw — 57×6
shiwulo@numa1:~$ ls
Desktop    linux-5.0.0                  spinlockFolder
downloads  linux_5.0.0-15.16.diff.gz    workdesktop
ext4       linux_5.0.0-15.16.dsc
files      linux_5.0.0.orig.tar.gz
shiwulo@numa1:~$ 
```

wc

程式碼概念圖

main

1 0

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
            if (pid1 > 0)
             pid2=fork();
              if (pid2>0)
close(pipefd[0];
close(pipefd[1];
     wait();
     wait();
```

child 1

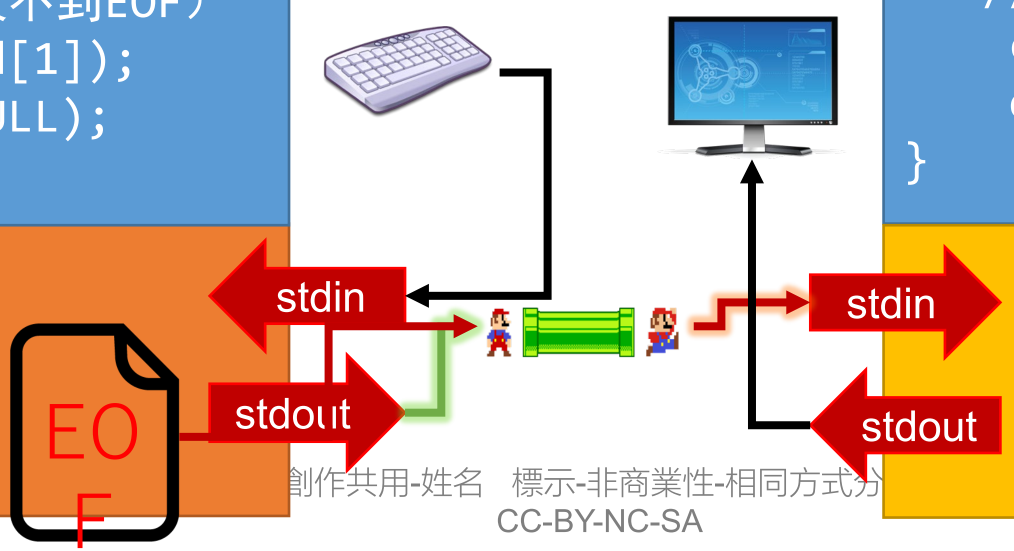
1 0

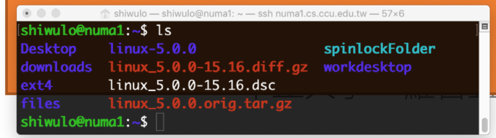
```
if (pid == 0) {
   //close stdout
   close(1);
   dup(pipefd[1]);
   //已經將pipe的fd[1]接上stdout
   //『一定要』關閉未使用的pipe fd
   //否則水龍頭會關不緊（收不到EOF）
   close(fd[0]); close(fd[1]);
   execlp( “ls”, “ls”, NULL);
}
```

child 2

1 0

```
if(pid2 == 0) {
   //close stdin
   close(0);
   dup(pipefd[0]);
   //關上沒用的水龍頭
    close(fd[0]); close(fd[1]);
    execlp(“wc”, “wc”, NULL);
}
```





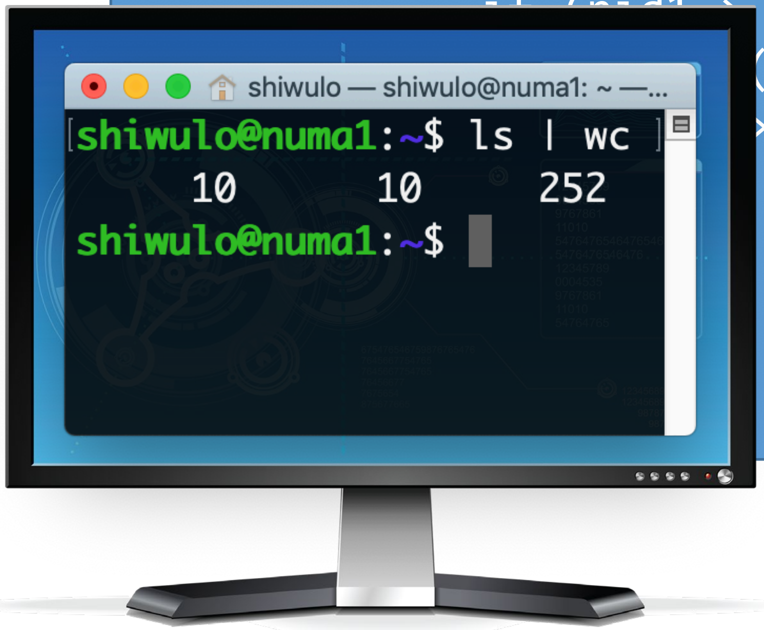
```
shiwulo@numa1:~$ ls
Desktop    linux-5.0.0                  spinlockFolder
downloads  linux_5.0.0-15.16.diff.gz   workdesktop
ext4       linux_5.0.0-15.16.dsc
files      linux_5.0.0.orig.tar.gz
shiwulo@numa1:~$
```

程式碼概念圖

main

```
int pipefd[2]; 1  0
pipe(pipefd);
pid1 = fork()
               if (pid1 > 0)
                          ();
                          >0)
```

因為收『到EOF』
所以『wc』會印出結果,
然後結束執行



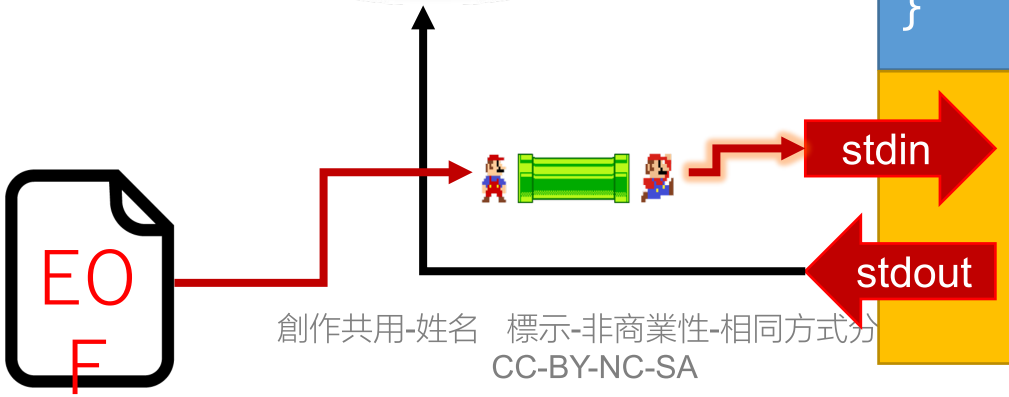
```
shiwulo@numa1:~$ ls | wc
     10      10     252
shiwulo@numa1:~$ 
```

child 2

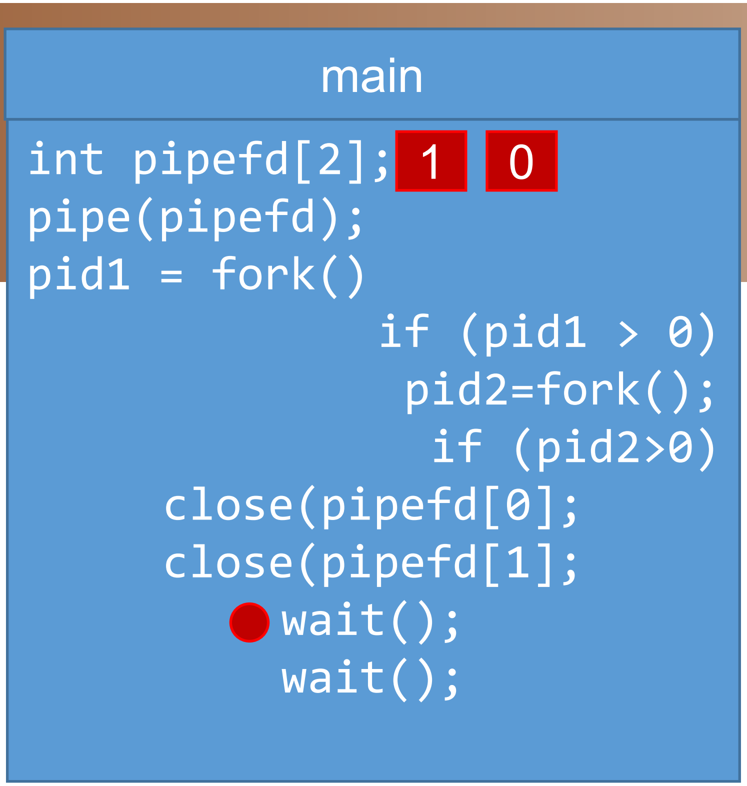
1 0

```
if(pid2 == 0) {
    //close stdin
    close(0);
    dup(pipefd[0]);
    //關上沒用的水龍頭
     close(fd[0]); close(fd[1]);
     execlp("wc", "wc", NULL);
}
```

wc



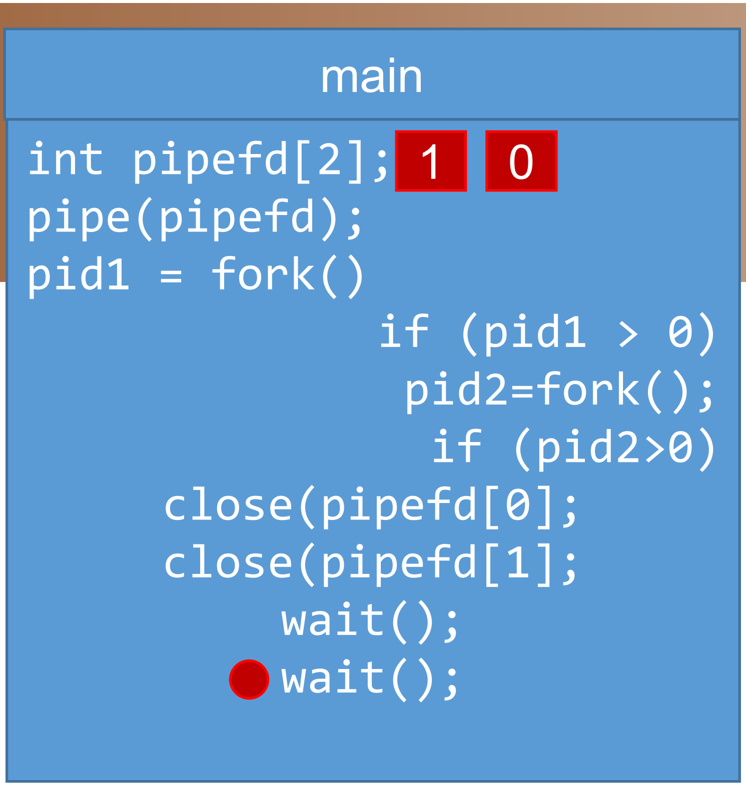
程式碼概念圖



main

```
int pipefd[2]; 1  0
pipe(pipefd);
pid1 = fork()
                  if (pid1 > 0)
                   pid2=fork();
                    if (pid2>0)
       close(pipefd[0];
       close(pipefd[1];
         ● wait();
           wait();
```

程式碼概念圖



```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
             if (pid1 > 0)
              pid2=fork();
               if (pid2>0)
     close(pipefd[0];
     close(pipefd[1];
          wait();
        ● wait();
```

pipe的重點
複習影片

程式碼概念圖

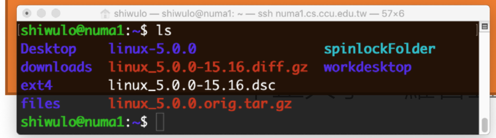
main

```
int pipefd[2];
pipe(pipefd);
pid1 = fork()
                 if (pid1 > 0)
                  pid2=fork();
                   if (pid2>0)
       close(pipefd[0];
       close(pipefd[1];
             wait();
             wait();
```

child 1

```
if (pid == 0) {
    //close stdout
    close(1);
    dup(pipefd[1]);
    //已經將pipe的fd[1]接上stdout
    //『一定要』關閉未使用的pipe fd
    //否則水龍頭會關不緊（收不到EOF）
    close(fd[0]); close(fd[1]);
    execlp( “ls”, “ls”, NULL);
}
```

ls



```
shiwulo — shiwulo@numa1: ~ — ssh numa1.cs.ccu.edu.tw — 57×6
shiwulo@numa1:~$ ls
Desktop    linux-5.0.0                   spinlockFolder
downloads  linux_5.0.0-15.16.diff.gz  workdesktop
ext4       linux_5.0.0-15.16.dsc
files      linux_5.0.0.orig.tar.gz
shiwulo@numa1:~$ 
```

child 2

```
if(pid2 == 0) {
    //close stdin
    close(0);
    dup(pipefd[0]);
    //關上沒用的水龍頭
     close(fd[0]); close(fd[1]);
     execlp(“wc”, “wc”, NULL);
}
```

wc

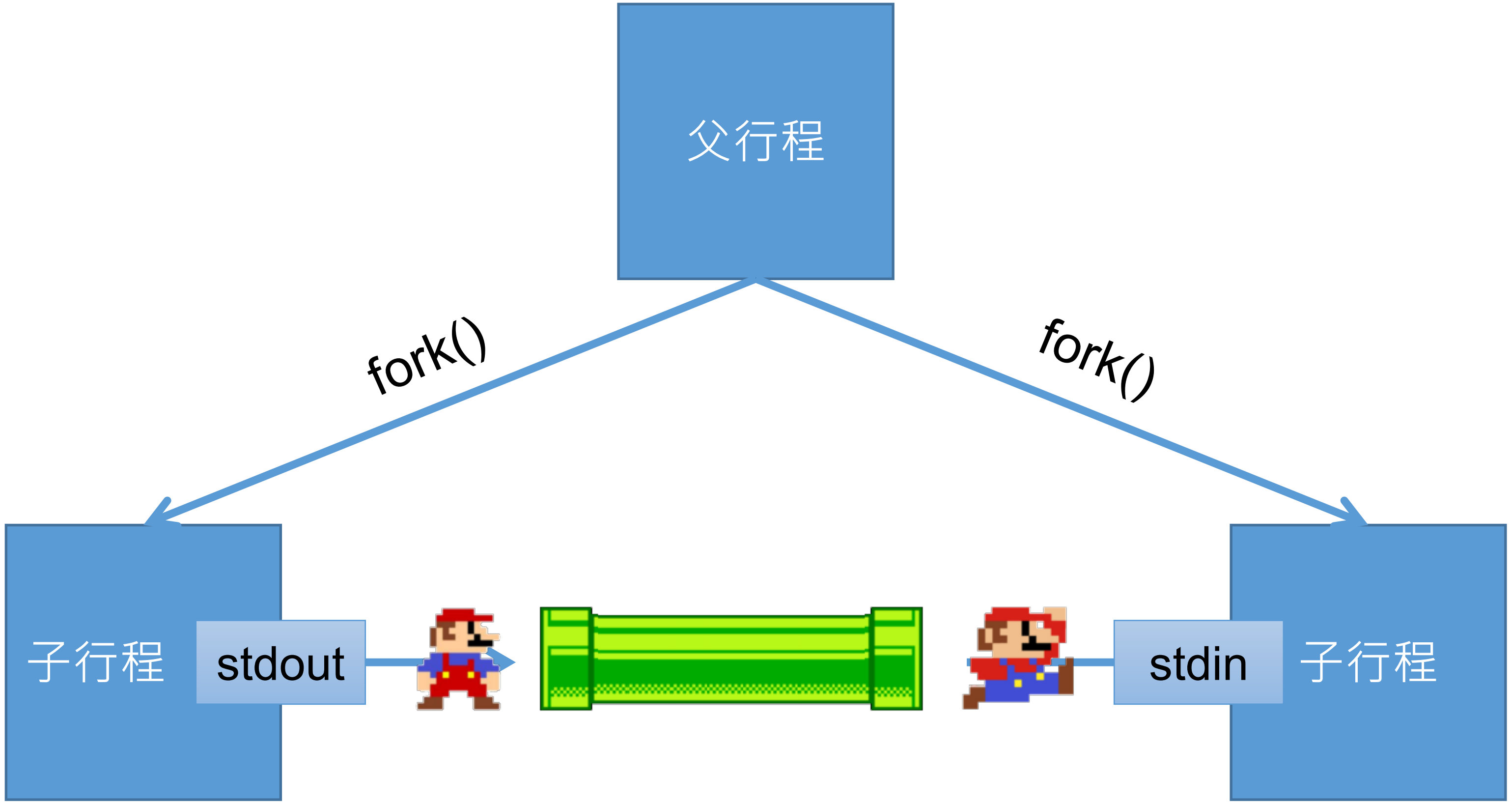
子行程間通訊（pipe4-2.c)

```c
int main(int argc, char **argv) {
    int pipefd[2];
    int ret, wstat, pid1, pid2;
    //char **param={"EXENAME", NULL};
    pipe(pipefd);
    pid1 = fork();   //產生第一個child
    if (pid1==0) {
        close(1);   //關閉stdout
        dup(pipefd[1]); //將pipefd[1]複製到stdout
        close(pipefd[1]);   //將沒用到的關閉
        close(pipefd[0]);   //將沒用到的關閉
        execlp("ls", "ls", NULL);   //執行ls，ls會將東西藉由stdout輸出到pipefd[1]
    } else printf("1st child's pid = %d\n", pid1);
    if (pid1>0) {
```

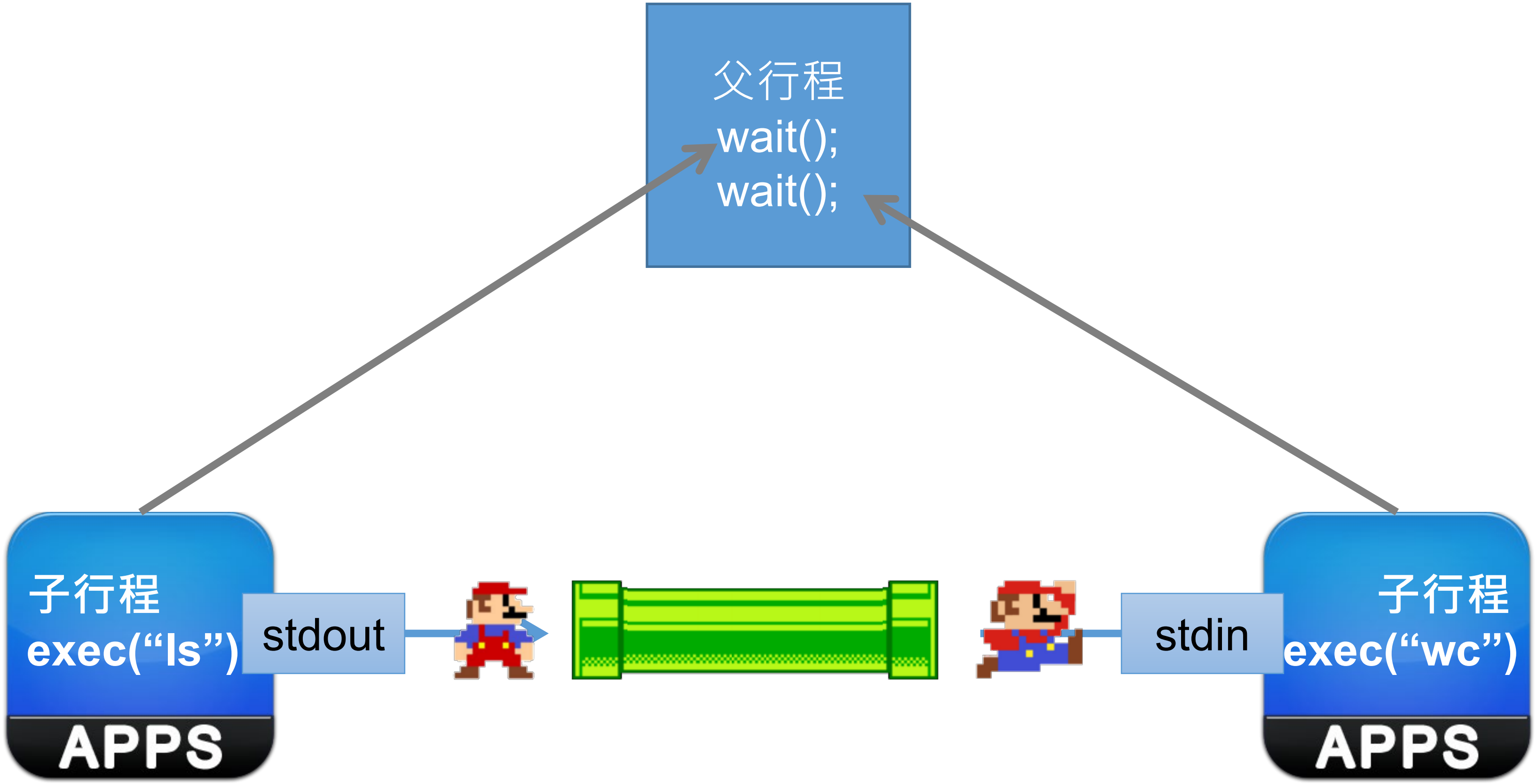
子行程間通訊（pipe4-2.c）

```
        pid2 = fork();//產生第二個child
        if (pid2==0) {
            close(0);   //關閉stdin
            dup(pipefd[0]); //將pipefd[0]複製到stdin
            close(pipefd[1]);   //將沒用到的關閉
            close(pipefd[0]);   //將沒用到的關閉
            execlp("wc","wc", NULL);   //執行wc， wc將透過stdin從pipefd[0]讀入資料
        } else printf("2nd child's pid = %d\n", pid2);
    }
    //parent一定要記得關掉pipe不然wc不會結束（因為沒有接到EOF）
    close(pipefd[0]); close(pipefd[1]);
    printf("child %d\n",wait(&wstat));
    printf("child %d\n",wait(&wstat));
}
```

示意圖



示意圖



結果

```
$ ./pipe4-2
1st child's pid = 27705
2nd child's pid = 27706
     18      18     139
child 27705
child 27706
$ ls | wc
     18      18     139
```

時間與輸出入導向

```
#include <stdio.h>
#include <time.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>


void per_sec(int signum)
{
    long          ns; // ns
    time_t         s;  // Seconds
    struct timespec spec;

    clock_gettime(CLOCK_REALTIME, &spec);

    s  = spec.tv_sec;
    ns = spec.tv_nsec;
    struct tm* lt = localtime(&s);
```

```c
#include <stdio.h>
#include <time.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>


void per_sec(int signum)
{
    long          ns; // ns
    time_t        s;  // Seconds
    struct timespec spec;

    clock_gettime(CLOCK_REALTIME, &spec);

    s  = spec.tv_sec;
    ns = spec.tv_nsec;
    struct tm* lt = localtime(&s);

    printf("Current time: %4d-%02d-%02d,%02d-%02d-%02d,%09ld\n", lt->tm_year+1900, lt->tm_mon+1, lt->tm_mday, lt->tm_hour, lt->tm_min, lt->tm_sec, ns);
    alarm(1);
}


int main(int argc, char** argv) {
    close(1);
    struct timespec spec;

    clock_gettime(CLOCK_REALTIME, &spec);

    s  = spec.tv_sec;
    ns = spec.tv_nsec;
    struct tm* lt = localtime(&s);

    printf("Current time: %4d-%02d-%02d,%02d-%02d-%02d,%09ld\n", lt->tm_year+1900, lt->tm_mon+1, lt->tm_mday, lt->tm_hour, lt->tm_min, lt->tm_sec, ns);
    alarm(1);
}


int main(int argc, char** argv) {
    close(1);
    open("/home/shiwulo/hello_timer", O_WRONLY | O_APPEND | O_CREAT,S_IRWXU);
    signal(SIGALRM, per_sec);
    alarm(1);
    getchar();
}
```

上機考提示

期中考筆試

- 考古題絕對要看

期末考上機考

- 題組一
  - 會使用getpid()、會輸出入轉向
  - 執行外部程式（即execv()）
  - 使用fork，讓child執行外部程式
  - 使用wait()，讓parent等child
- 題組二
  - 會使用pipe
  - 知道stdin、stdout、stderr的輸出入轉向
- 題組三
  - 會用inotify（強烈建議使用教學上的範例，了解事件的意義）

念念時間

- 要有興趣
- 不斷學習
- 知道將來的要過怎樣的生活
  - 物價、房價、租屋、車子
  - 伴侶、子女
  - 奢持品
- 知道如何過「那樣的生活」
  - 多少薪水，願意花這麼多的薪水聘自己嗎?
  - 一般的期盼是「十倍」的產出