Prog. #1 _ 1082414

如何執行:

compile:

```
$  g++ -o 1082414_prog1 1082414_prog1.cpp -lrt
```

execute:

```
$  ./1082414_prog1
```

input_example:

```
42
```

output_example:

```
[4082 Child]  : 21
[4081 Parent] : 64
[4083 Child]  : 32
[4081 Parent] : 16
[4084 Child]  : 8
```

```
[4081 Parent] : 4
[4085 Child]  : 2
[4081 Parent] : 1
Max : 64
Order : 2
```

設計理念:

透過迴圈處理至考拉茲猜想結束，過程中經由parent process跟child process對share memory不斷的存取與計算，最終得出結果

詳細解釋:

```
/*

   compile: g++ -o 1082414_prog1 1082414_prog1.cpp -lrt

   execute: ./1082414_prog1

*/

#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/wait.h>
#include <sys/types.h>

void error_and_die(const char *msg) {//錯誤回報
 perror(msg);
 exit(EXIT_FAILURE);
}

int cof(int n){//考拉茲猜想 ( Collatz conjecture )
 if(n%2==0){
  return n/2;
 }else{
  return 3*n+1;
 }
}

int main(int argc, char *argv[]) {//主函式

 const char *name = "test";//共用記憶體物件的名字
 const size_t region_size = 16;//設定物件大小為16Byte, 可以存4個int

 int fd = shm_open(name, O_CREAT | O_RDWR, 0666);//若記憶體物件不存在(O_CREAT), 且是為了讀寫開啟
(O_RDWR), 則產生此共用記憶體物件, 目錄存取為0666
 if (fd == -1)//物件產生失敗
   error_and_die("shm_open");

 int r = ftruncate(fd, region_size);//設定此物件的大小
 if (r != 0)//設定大小失敗
   error_and_die("ftruncate");

 int *ptr =(int*)mmap(0, region_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);//建立一個包含共用記憶體
物件的記憶體對映檔案 ( 實體化 ) , 回傳一個指到使檔案的指標
 if (ptr == MAP_FAILED)//建立失敗
   error_and_die("mmap");

 close(fd);//用完了, 關掉

 //shm的變數初始化
 std::cin >> ptr[0];//用來存考拉茲猜想的值
 ptr[1] = 0;//用來存第幾次執行考拉茲猜想
 ptr[2] = 0;//用來存序列中最大的數
 ptr[3] = 0;//用來存最大的數發生在第幾次

 while(ptr[0]!=1){//當考拉茲猜想還沒結束
    pid_t pid = fork();//產生child process

    if (pid == 0) {//如果是pid是0(child process), 則執行此區段
```

```
    ptr[0] = cof(ptr[0]);//做一次考拉茲猜想,並存回shm裡
    ptr[1]++;//執行次數+1
    if(ptr[0]>ptr[2]){//若計算的數大於目前存著最大的數
     ptr[2] = ptr[0];//將這次計算的數存為最大的數
     ptr[3] = ptr[1];//並紀錄是第幾次計算出最大的數
    }
    std::cout << "["<<getpid()<<" Child]  : "<<ptr[0] << std::endl;//用getpid()取得自己的pid,印出clild的結果
      exit(0);//child process完成最後一個敘述,要求作業系統將自己刪除時結束,並回傳狀態值給parent process
    }
    else {//是parent process
       int status;
       waitpid(pid, &status, 0);//等待child process執行結束
   if(ptr[0] == 1){////當考拉茲猜想被child process算出來, 印出成果
    std::cout << "["<<getpid()<<" Parent] : "<<ptr[0] << std::endl;//用getpid()取得自己的pid,印出parent的結果
    std::cout << "Max : " << ptr[2] << std::endl;//印出過程中的最大值
    std::cout << "Order : " << ptr[3] << std::endl;//印出過程中的最大值出現在第幾次
    break;//中斷迴圈
   }
   ptr[0] = cof(ptr[0]);//做一次考拉茲猜想,並存回shm裡
   ptr[1]++;//執行次數+1
   if(ptr[0]>ptr[2]){//若計算的數大於目前存著最大的數
    ptr[2] = ptr[0];//將這次計算的數存為最大的數
    ptr[3] = ptr[1];//並紀錄是第幾次計算出最大的數
   }
   if(ptr[0] == 1){////當考拉茲猜想被parent process算出來, 印出成果
    std::cout << "["<<getpid()<<" Parent] : "<<ptr[0] << std::endl;//用getpid()取得自己的pid,印出parent的結果
    std::cout << "Max : " << ptr[2] << std::endl;//印出過程中的最大值
    std::cout << "Order : " << ptr[3] << std::endl;//印出過程中的最大值出現在第幾次
    break;//中斷迴圈
   }
   std::cout << "["<<getpid()<<" Parent] : "<<ptr[0] << std::endl;//用getpid()取得自己的pid,印出parent的結果
    }
  }
  //當考拉茲猜想被全部計算完把shm還回系統
   r = munmap(ptr, region_size);
    if (r != 0)error_and_die("munmap");
   r = shm_unlink(name);
   if (r != 0)error_and_die("shm_unlink");

   return 0;
}
```