作業系統 hw1

生機四詹育晟 b10605023

###執行方式:

假設檔名為 hw1_Q1.c，且想要測試 N=6，請在終端機中依序輸入以下指令：

gcc hw1_Q1.c -o hw1_Q1

./hw1_Q1 6

輸出結果:
0,1,1,2,3,5

3.13 (Q1)**執行邏輯:**

程式:

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
// 計算 Fibonacci 序列
void printFibonacci(int n) {
    int a = 0, b = 1, c;
    if (n <= 0) return;

    for (int i = 0; i < n; i++) {
        if (i == 0)
            printf("%d", a);
        else if (i == 1)
            printf(",%d", b);
        else {
            c = a + b;
            printf(",%d", c);
            a = b;
```

```
            b = c;
        }
    }
    printf("\n");
}
int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <non-negative integer>\n", argv[0]);
        exit(1);
    }
    int n = atoi(argv[1]);
    if (n < 0) {
        fprintf(stderr, "Error: Number must be non-negative.\n");
        exit(1);
    }
    pid_t pid = fork();
    if (pid < 0) {
        perror("fork failed");
        exit(1);
    } else if (pid == 0) {
        // Child process
        printFibonacci(n);
        exit(0);
    } else {
        // Parent process
        wait(NULL);  // 等待子程序完成
    }
    return 0;
}
```

說明：

當程式執行時，首先會進入 main 函數，接著從命令列參數中讀取使用者所輸入的數值。如果參數數量不正確，或是輸入的數值不是一個非負整數，程式就會直接結束，並顯示錯誤訊息。

一旦輸入正確，程式會呼叫 fork() 系統呼叫來建立一個子程序。從這裡開始，程式會分裂成兩個執行路徑：母程序與子程序。透過 fork() 的回傳值，程式可以分辨目前是哪一個程序在執行。

若是子程序，會呼叫 printFibonacci 函數來計算並印出 Fibonacci 數列。這個函數會從第 0 項開始逐步計算，直到印出使用者要求的第 n 項為止。為了符合題目格式，數列中的每一項之間都會用逗號隔開，例如：0,1,1,2,3。

子程序在完成 Fibonacci 數列的輸出後，會透過 exit(0) 結束執行。與此同時，母程序則在 fork() 之後呼叫 wait()，等待子程序結束。這個同步機制是為了確保輸出順序的正確性，避免母程序先結束導致子程序的輸出未完成或錯亂。當子程序完全執行完畢並結束後，母程序才會從 wait() 返回，然後整個程式結束。

範例執行:

```
xavier1021@LAPTOP-68NFQUBM:/mnt/d/os_hw1/chan$ gcc hw1_Q1.c -o output1
xavier1021@LAPTOP-68NFQUBM:/mnt/d/os_hw1/chan$ ./output1 10
0,1,1,2,3,5,8,13,21,34
```

3.14 (Q2)執行邏輯:

程式:

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

// 產生 Collatz 數列
void printCollatz(int n) {
    printf("Collatz sequence: ");
    while (n != 1) {
        printf("%d,", n);
        if (n % 2 == 0)
            n = n / 2;
```

```
        else
            n = 3 * n + 1;
    }
    printf("%d\n", n); // 印出最後的 1
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <positive integer>\n", argv[0]);
        exit(1);
    }

    int n = atoi(argv[1]);

    if (n <= 0) {
        fprintf(stderr, "Error: Please enter a positive integer.\n");
        exit(1);
    }

    pid_t pid = fork();

    if (pid < 0) {
        perror("fork failed");
        exit(1);
    } else if (pid == 0) {
        // 子程序：輸出 Collatz 數列
        printCollatz(n);
        exit(0);
    } else {
        // 母程序：等待子程序結束
        wait(NULL);
    }

    return 0;
}
```

說明：

程式從 main 函數開始執行，並檢查命令列是否正確地傳入了一個參數。如果沒有輸入參數，或是輸入的數字不是正的整數，程式就會顯示錯誤訊息並結束。這樣的設計可以避免不合法的輸入造成錯誤行為。

接著，程式使用 fork() 建立一個子程序。這個系統呼叫會複製一份當前的程式內容，建立一個與母程序幾乎相同的新程序。fork() 回傳值會讓系統知道目前是在哪一個程序中執行。若回傳值為 0，代表目前是在子程序，否則則是母程序。

在子程序中，程式會呼叫 printCollatz(n)，開始計算並輸出從使用者輸入的數字 n 開始的 Collatz 數列。這個數列的規則是：如果目前的數字是偶數，就除以 2；如果是奇數，就乘上 3 再加 1。重複這個操作直到數字變成 1 為止。在這個函式中，數列的每個數字之間會用逗號隔開，最終輸出一個像這樣的結果：35,106,53,160,...,1。

母程序則會在子程序啟動後，執行 wait(NULL)，進入等待狀態，直到子程序執行結束為止。這樣的安排能夠確保子程序先完成數列的輸出，再由母程序結束整體流程，避免兩個程序同時結束而導致輸出順序錯亂或部分遺失。最後，當子程序完成任務並結束後，母程序也會結束。

範例執行:

```
xavier1021@LAPTOP-68NFQUBM:/mnt/d/os_hw1/chan$ gcc hw1_Q2.c -o output2
xavier1021@LAPTOP-68NFQUBM:/mnt/d/os_hw1/chan$ ./output2 35
Collatz sequence: 35,106,53,160,80,40,20,10,5,16,8,4,2,1
```

3.15 (Q3)執行邏輯:

程式:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/mman.h>
#include <fcntl.h>
```

```c
#include <sys/wait.h>
#include <string.h>

#define SHM_NAME "/collatz_shm"
#define MAX_SIZE 1024

void generateCollatz(int n, char *buffer) {
    char temp[32];
    while (n != 1) {
        sprintf(temp, "%d,", n);
        strcat(buffer, temp);
        if (n % 2 == 0)
            n /= 2;
        else
            n = 3 * n + 1;
    }
    sprintf(temp, "%d", n);  // 加入最後的 1
    strcat(buffer, temp);
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Usage: %s <positive integer>\n", argv[0]);
        return 1;
    }

    int n = atoi(argv[1]);
    if (n <= 0) {
        fprintf(stderr, "Error: Please provide a positive integer.\n");
        return 1;
    }

    // a. 建立 shared memory
    int shm_fd = shm_open(SHM_NAME, O_CREAT | O_RDWR, 0666);
    if (shm_fd == -1) {
        perror("shm_open failed");
        return 1;
    }
```

```
    // 設定 shared memory 大小
    if (ftruncate(shm_fd, MAX_SIZE) == -1) {
        perror("ftruncate failed");
        return 1;
    }

    // 映射 shared memory 到虛擬記憶體空間
    char *shm_ptr = mmap(NULL, MAX_SIZE, PROT_READ | PROT_WRITE,
MAP_SHARED, shm_fd, 0);
    if (shm_ptr == MAP_FAILED) {
        perror("mmap failed");
        return 1;
    }

    // b. 建立子程序
    pid_t pid = fork();
    if (pid < 0) {
        perror("fork failed");
        return 1;
    } else if (pid == 0) {
        // 子程序：產生 Collatz 數列寫入 shared memory
        shm_ptr[0] = '\0'; // 清空記憶體
        generateCollatz(n, shm_ptr);
        exit(0);
    } else {
        // 母程序：等待子程序完成
        wait(NULL);

        // c. 讀取 shared memory
        printf("Collatz sequence: %s\n", shm_ptr);

        // d. 移除 shared memory
        munmap(shm_ptr, MAX_SIZE);
        shm_unlink(SHM_NAME);
    }

    return 0;
```

```
}
```

說明:

當程式開始執行時，它首先會檢查命令列參數是否正確，並驗證輸入是否為正整數。如果輸入不合法，程式就會直接結束並顯示錯誤訊息。

接著，程式會使用 shm_open() 建立一個名為 /collatz_shm 的共享記憶體區段，並透過 ftruncate() 設定其大小為 1024 位元組，足夠存放計算出來的數列字串。為了讓這個共享記憶體能夠被程式所操作，它再使用 mmap() 將這段記憶體映射到自己的虛擬記憶體空間，這樣接下來就可以用字串操作的方式對其進行讀寫。

完成共享記憶體的建立與設定後，程式使用 fork() 建立子程序。在子程序的執行路徑中，它會先清空共享記憶體，然後呼叫 generateCollatz() 函數，將 Collatz 數列的結果（以逗號分隔）寫入共享記憶體。這個函式的邏輯與前一題類似，不同的是它不直接印出，而是將結果格式化為字串後儲存在 buffer 裡。

在母程序的路徑中，程式會先呼叫 wait() 等待子程序執行完畢，確保共享記憶體中的資料已完整寫入。接著，母程序從共享記憶體中讀取數列，並將它印出來。最後，程式會使用 munmap() 與 shm_unlink() 將該段共享記憶體解除對映並刪除，完成資源釋放的動作，避免系統資源外洩或殘留。

範例執行:

```
xavier1021@LAPTOP-68NFQUBM:/mnt/d/os_hw1/chan$ gcc hw1_Q3.c -o output3
xavier1021@LAPTOP-68NFQUBM:/mnt/d/os_hw1/chan$ ./output3 35
Collatz sequence: 35,106,53,160,80,40,20,10,5,16,8,4,2,1
```