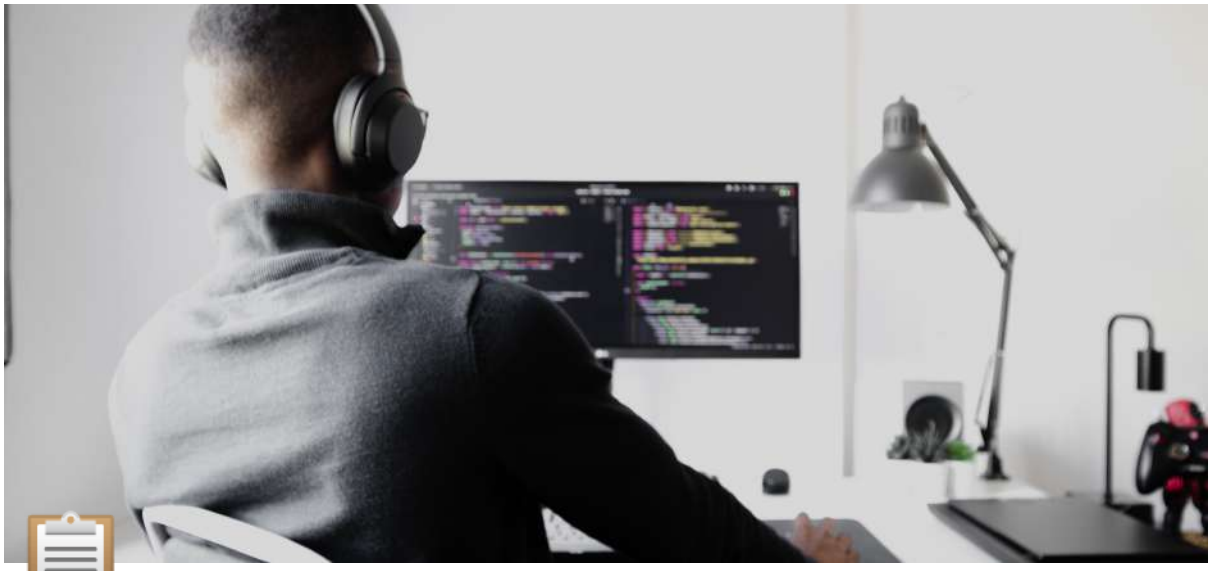


# Prog. #3 _ 1082414

## 如何執行:

### Compile:

```
$  g++ -o 1082414_prog3.out 1082414_prog3.cpp -pthread
```

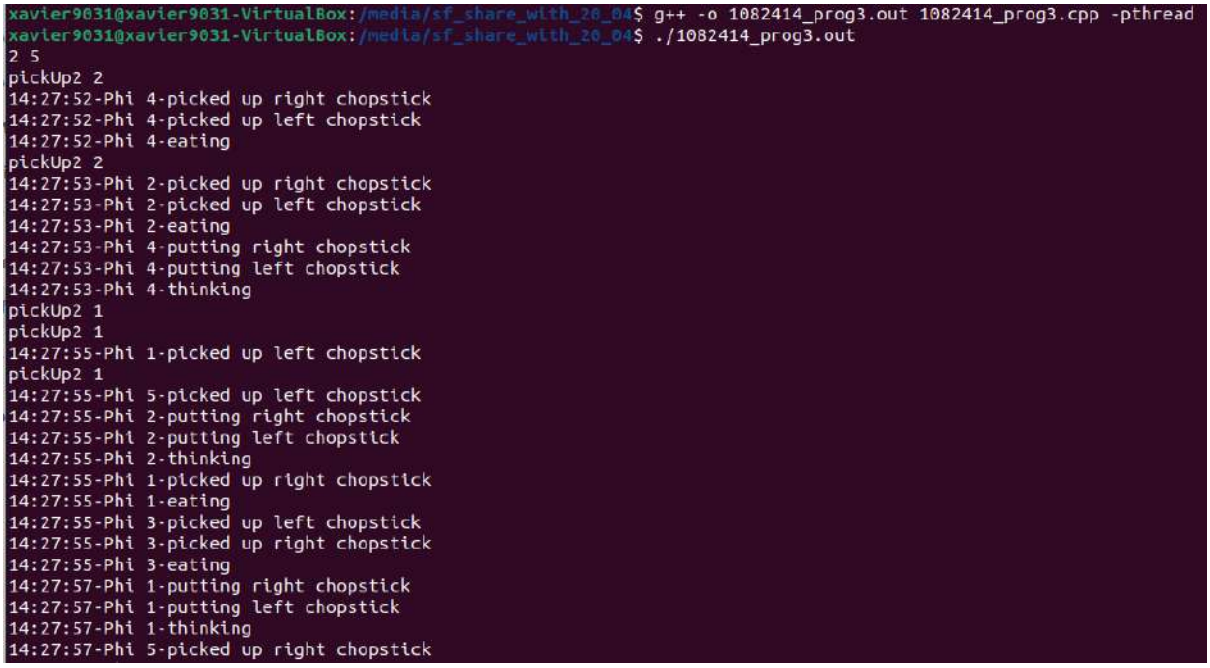
### Execute:

```
$  ./1082414_prog3.out
```

### Input:

- 參數一 (mode) : 請輸入1~3 分別代表三種不同的哲學家模擬
- 參數二 (phi_num) : 請輸入3~11 此參數代表了哲學家的人數

### Example:




# 設計理念:

使用 pthread 來模擬哲學家問題。從命令列讀入兩個整數，第一個1~3的整數 m 表示模擬的演算法，第二個數字是一個 3~11的整數 n，程式就會產生 n 個執行緒來代表有 n 個哲學家。

根據輸入進入三種不同模式的哲學家，不同的哲學家有不同的拿筷子的模式，根據哲學家的動作將時間及狀態印出。第一種所有的哲學家都是先拿右手再拿左手，當所有的哲學家同時拿起右邊的筷子時會產生死結。第二種模式偶數號的哲學家會先拿右邊的筷子再拿左邊的筷子，單數號則相反。第三種模式哲學家只會在可以同時拿起筷子的情況下才會拿起筷子。

# 完成功能:

- 基本功能全部
- 無死結偵測

# 詳細解釋:

## 標頭檔:

```
#include <iostream>//io
#include <pthread.h>//thread相關
#include <unistd.h>
#include <time.h>//時間相關
```

```
#include <string.h>//字串相關
using namespace std;
```

## 變數設定:

```
//全域變數設定
pthread_mutex_t chopsticks[NUMBER_OF_PHILOSOPHERS];
pthread_t philosophers[NUMBER_OF_PHILOSOPHERS];
pthread_attr_t attributes[NUMBER_OF_PHILOSOPHERS];
int phi_num, phi_mode;
const int EAT_NUM = 10;//吃的次數
```

## 三種哲學家跟三種拿法:

```
void *philosopher1(void *);
void *philosopher2(void *);
void *philosopher3(void *);
void pickUp1(int , int );
void pickUp2(int , int );
void pickUp3(int , int );
```

## 哲學家通用函式:

```
void waitting(int );//等待
void think(int );//思考
void eat(int ) ;//吃飯
void putDown(int , int );//放下筷子
```

## 輸出相關函式:

```
//輸出相關
string outputFormat(int , string );
void print(string );
```

## Driven Code:

```
int main() {
  srand(0);//設定隨機種子
  cin >> phi_mode >> phi_num;
  //輸入範圍判讀
  if(phi_mode > 3 || phi_mode < 0)return 0;
  if(phi_num > 11 || phi_num < 3)return 0;
```

```
  for (int i = 0; i < phi_num; ++i) {
    pthread_mutex_init(&chopsticks[i], NULL);
  }
  for (int i = 0; i < phi_num; ++i) {
    pthread_attr_init(&attributes[i]);
  }
  int *id=(int *)malloc(sizeof(int)*phi_num);
  if(phi_mode == 1){
    for (int i = 0; i < phi_num; ++i) {
      id[i]=i;
      pthread_create(&philosophers[i], &attributes[i], philosopher1, (void *)(&id[i]));
    }
  }else if(phi_mode == 2){
    for (int i = 0; i < phi_num; ++i) {
      id[i]=i;
      pthread_create(&philosophers[i], &attributes[i], philosopher2, (void *)(&id[i]));
    }
  }else if(phi_mode == 3){
    for (int i = 0; i < phi_num; ++i) {
      id[i]=i;
      pthread_create(&philosophers[i], &attributes[i], philosopher3, (void *)(&id[i]));
    }
  }
  else{return 0;}
  for (int i = 0; i < phi_num; ++i) {
    pthread_join(philosophers[i], NULL);
  }
  return 0;
}
```