程式碼說明 & 如何編譯

1. void point(int x, int y)

```
21
22 void point(int x,int y){
23   cout<<"("<<x<<","<<y<<")";
24 }
25
```

用來印出座標(點)
Ex: (2,3) 、 (0,1)

2. void anotherpoint(int x,int y, int &x1,int &y1)

```
26 void anotherpoint(int x,int y,int &x1,int &y1){
27    if(rand()%2==1){ //means boat is parallels,y is the same,I want
28             if(x==0){      // 0<=x<=3
29                point(x+1,y);
30                x1 = x + 1;
31                y1 = y;
32             }
33             else if(x==3){
34                point(x-1,y);
35                x1 = x - 1;
36                y1 = y;
37             }
38             else{
39               if(rand()%2==1){
40                 point(x+1,y); //When rand()=1
41                 x1 = x + 1;
42                 y1 = y;
43               }
44               else{
45                 point(x-1,y);
46                x1 = x - 1;
47                y1 = y;
48               }
49             }
50    }
```

```
51    else{  //means boat is vertical,x is the same,I want
52             if(y==0){      // 0<=y<=3
53                point(x,y+1);
54                x1=x;
55                y1=y+1;
56             }
57             else if(y==3){
58                point(x,y-1);
59                x1=x;
60                y1=y-1;
61             }
62             else{
63               if(rand()%2==1){
64                point(x,y+1); //When rand()=1
65                x1=x;
66                y1=y+1;
67               }
68               else{
69                point(x,y-1);
70                x1=x;
71                y1=y-1;
72               }
73             }
74    }
75 }
```

此 Function 用來決定戰艦的〝另一個點〞(x1,y1)

我的設計是:利用 rand()%2 來決定戰艦應為 Ex:

(2,2)

(2,1)

(2,1) (3,1)

(X 座標相同) or (Y 座標相同)

我令: rand()%2 == 1 時,y 座標相同 (水平向的戰艦)

rand()%2 == 0 時,x 座標相同 (垂直向的戰艦)

但還有一種情況要考慮:當 random 出的第一個點 x = 0 or x =3 or y = 0 or y = 3 **(邊界)**時

首先是水平向的戰艦:

我用了 28~39 行的 if、else if 判斷式,這邊我令 x = 0 時,戰艦的另一個點的 x 座標只能 + 1,- 1 會使 X = -1 ,會超出題目的邊界,同理, x = 3 時,另一個點的 x 座標只能 - 1,+ 1 會使 x = 4,超出題目邊界,垂直向的戰艦同理 (52 ~ 61 行)。

而當 random 出的第一個點 x != 0 and x != 3 and y != 0 and y != 3 時 (38 和 62 行的 else 判斷式)

首先一樣是水平向的戰艦:

我用了 39 ~ 48 行的 if、else 判斷式,我令 rand()%2 == 1 時, 戰艦的另一個點的 x 座標會 + 1,反之,rand()%2 == 0 , 戰艦的另一個點的 x 座標會 - 1

垂直向的戰艦同理 (63 ~ 72 行)。

3. void attacked(int bombx, int bomby, int x, int y, int x1, int y1, int &num, int &atked_x, int &atked_y)

```
77 void attacked(int bombx,int bomby,int x,int y,int x1,int y1,int &num, int &atked_x,int &atked_y){
78      if((bombx==x && bomby==y) || (bombx==x1 && bomby==y1)){
79         if(atked_x == bombx && atked_y == bomby){   // For 3-c
80
81         }
82         else{
83                 num--;
84         }
85         atked_x = bombx;
86         atked_y = bomby;
87
88        if(num!=0){
89         cout << "hit"<<endl;
90        }
91        else{
92         cout<<"hit and sinking"<<endl;
93        }
94
95      }
96      else{
97        cout << "missing"<<endl;
98      }
99 }
```

當 Child、Parent **被對方攻擊**時, 會呼叫此 function

設計理念:

bombx、bomby:轟炸的座標

(x, y)(x1, y1):敵人(被轟炸方)的戰艦座標

num:剩餘 **〝未被擊中〞** 的座標

(atked_x, atked_y):已被擊中的座標

若轟炸的座標 = 敵人的戰艦座標 (78 行) {

要先檢查此點是否被轟過

c. 擊中，砲艇沈沒。判斷沈沒的條件是砲艇所有的位置均被擊中。

因題意(c.)所述, 重複擊中同一個以被擊中的敵人座標, num 不會減一

因為我一開始將 atked_x, atked_y 設為 (-1, -1)

所以將 85、86 行放在 79-84 行後面, 避免 num 沒有被-1

輸出: 若 num = 0, 代表所有敵人的座標均以擊中, 輸出 hit and sinking

若 num != 0, 代表被擊中, 但未擊沉, 輸出 hit

}

若未擊中, 輸出 missing

4. int ran()

```
101 int ran(){
102   return rand() % 4;
103 }
104
```

輸出隨機數字 0~3

5. int main(int argc, char *argv[])

(1)<修改於教授的 03-shm.pdf, p14、p15>

```
108         int r;
109         const char *memname="sample";
110         const size_t region_size = sysconf(_SC_PAGE_SIZE);
111
112         int fd = shm_open(memname, O_CREAT|O_TRUNC|O_RDWR,0666);
113         if(fd==-1) error_and_die("shm_open");
114
115         r=ftruncate(fd,region_size);
116         if(r!=0)  error_and_die("ftruncate");
117
118         int *ptr=(int *)mmap(0,region_size,PROT_READ|PROT_WRITE,MAP_SHARED,fd,0);
119         if(ptr == MAP_FAILED) error_and_die("mmap");
120         close(fd);
```

```
249         r = munmap(ptr,region_size);
250         if(r!=0) error_and_die("shm_unlink");
251 }
```

(2)

```
121         ptr[3] = -1; //initial, the num of the enemy's
122         ptr[4] = -1;  //initial, lock the next step if ptr[4] != 1
123         ptr[5] = 0;
124         ptr[6] = 0; //Winner's pid
125         ptr[7] = 0; //Winner's bomb
126         ptr[9] = 0; //cout who is win
127         char tmp1 = *argv[1],tmp2 = *argv[2];
128         // parent's and child's random seed
129         int pseed = tmp1 - '0', childseed = tmp2 - '0';
```

變數介紹:

ptr[3] = -1 ,-1 為初始值,用來表示**敵方剩餘**的〝**未被擊中**〞的數量

ptr[4] = -1 ,-1 為初始值,用來**鎖住下一個步驟(ptr[4] != -1)**,下一個步驟為〝**互相攻擊**〞

ptr[5] = 0 , 0 為初始值,用來控制**輸出的順序**

ptr[6] = 0 , 0 為初始值,用來儲存勝者的 pid

ptr[7] = 0 , 0 為初始值,用來儲存勝者使用的炸彈數量

ptr[9] = 0 , 0 為初始值,用來確保**上一個步驟(互相攻擊)**是否已完成,來決定是否輸出最後一行的勝者和砲彈數

ptr[3]、ptr[4]、ptr[5]、ptr[6]、ptr[7]、ptr[9]均在 Shared Memory 中

tmp1、tmp2 :用來儲存輸入的 p1、p2(亂數種子),因型態為 char,先將他暫存於 tmp1、tmp2 中

pseed、childseed : 將 tmp1、tmp2 轉換為 int 型態並分別存入 pseed、childseed 中

ps. 因為我只做基本功能,所以在程式碼中並未使用到 p3(第三個輸入的參數),直接輸入 0(p3)即可

(3)輸出 Random Seed、gunboat 座標

圖 (3) - 2

```
131         if(fork()==0){  //child
132            srand((unsigned)time(NULL)^childseed);
133            int x=ran(),y=ran(),x1,y1,pid_tmp = getpid(),atked_x = -1,atked_y = -1;
134            while(true){
135                 if(ptr[5] == 2){
136                  cout<<"["<<getpid()<<" Child]:Random Seed "<< argv[2];
137                  cout<<endl;
138                  cout<<"["<<getpid()<<" Child]:The gunboat:";
139
140                  point(x,y);
141                  anotherpoint(x,y,x1,y1);
142                  cout<<endl;
143                  ptr[4] = 1;  // start attack
144                  ptr[5] = 3;
145                 }
146                 if(ptr[5] == 3){
147                    ptr[5] = 0;
148                    break;
149                 }
150            }
```

圖 (3) - 3

```
189         else{  //parent process
190           srand((unsigned)time(NULL)^pseed);
191           int x=ran(),y=ran(),x1,y1,pid_tmp = getpid(),atked_x = -1,atked_y = -1;
192           while(true){
193            if(ptr[5] == 0){
194                 cout<<"["<<getpid()<<" Parent]:Random Seed "<< argv[1];
195                 cout<<endl;
196                 cout<<"["<<getpid()<<" Parent]:The gunboat:";
197                 point(x,y);
198                 anotherpoint(x,y,x1,y1);
199                 cout<<endl;
200                 ptr[5] = 1;
201            }
202            if(ptr[5] == 1){
203                 ptr[5] = 2;
204                 break;
205            }
206           }
```

圖(3) - 2 與圖(3) - 3 分別為 Child process 和 Parent process

首先,132、190 行分別指定了子行程和父行程的亂數種子

133、191 行分別隨機指定了子行程和父行程的第一個戰艦座標(x,y),以及宣告第二個戰艦座標(x1,y1),且利用 pid_tmp 暫時存取 pid 值,並將(atked_x,atked_y)均初始化為(-1,-1),因此時尚未被攻擊

為了達到圖(3) - 3 的輸出順序,首先我用了 **while(true)這個迴圈**(讓程式先**暫時卡在 192~206 行和 134~150 行**),等符合條件時再跳出(**避免程式往下個步驟走**),並在內部透過 ptr[5] = 0、1、2、3 來**控制輸出的順序**,若 194~199 行輸出了,則將 ptr[5] = 1,使之跳出 192 行的 while(true)迴圈,且 ptr[5] = 2,但父行程並不會接著做下半的步驟(攻擊),因為我並未讓 ptr[4] = 1。

當 ptr[5] = 2 時,代表**父行程的 Random Seed 和 gunboat 已輸出完畢**,輪到子行程輸出,輸出後將 ptr[5] = 3,並將 ptr[4] = 1。代表已將圖(3) - 1 的內容輸出完畢,可以進到互相攻擊的步驟了。

不過在 146~148 行中,我將 ptr[5] 變回 0,是為了接下來控制**〝攻擊的順序〞**

圖 (3) - 1

```
[3989 Parent]:Random Seed 5
[3989 Parent]:The gunboat:(1,0)(2,0)
[3990 Child]:Random Seed 10
[3990 Child]:The gunboat:(1,1)(1,2)
```

(4)互相攻擊

圖(4) - 1, Child Process

```
152          int num = 2;    // the point of the boat (initial num)
153          int bombnum = 0;
154          while(true){
155               if(ptr[4] == 1){
156                       if(ptr[3] != 0 && num != 0 && ptr[5] == 1){
157                               //-----------shootted-----------//
158                               cout<<"["<<getpid()<<" Child]:";
159                               attacked(ptr[1],ptr[2],x,y,x1,y1,num,atked_x,atked_y);
160                               ptr[3] = num;
161
162                               if(num!=0){  //child isn't sinking
163                               //-----------shooting----------//
164                               cout<<"["<<getpid()<<" Child]:bombing";
165                               ptr[1] = ran();  //x
166                               ptr[2] = ran();  //y
167                               point(ptr[1],ptr[2]);
168                               cout<<endl;
169                               bombnum++;
170                               }
171                               else{
172                                       // child is sinking
173                                       break;
174                               }
175                               ptr[5] = 2;
176                       }
177                       else if(ptr[3] == 0){
178                               ptr[6] = pid_tmp;
179                               ptr[7] = bombnum;
180                               ptr[9] = 1;
181                               break;
182                       }       //enemy is loss
183
184               }
185          }
```

圖(4) - 2, Parent Process

```
208          int num = 2; //The count of boat's point (initial)
209          int bombnum = 0;
210          while(true){
211               if(ptr[4] ==1){
212                 //-----------shooting----------//
213                 if(ptr[3] != 0 && num != 0 && ptr[5] == 0){
214                       cout<<"["<<getpid()<<" Parent]:bombing";
215                       ptr[1] = ran();  //x
216                       ptr[2] = ran();  //y
217                       point(ptr[1],ptr[2]);
218                       cout<<endl;
219                       ptr[5] = 1;
220                       bombnum++;
221                 }
222                 else if(ptr[3] == 0){  //enemy loss
223                       ptr[7] = bombnum;
224                       ptr[6] = pid_tmp;
225                       ptr[9] = 1;
226                       break;
227                 }
228                 if(ptr[3] != 0  && ptr[5] == 2){ // The enemy isn't loss
229                 //-----------shootted----------//
230                       cout<<"["<<getpid()<<" Parent]:";
231                       attacked(ptr[1],ptr[2],x,y,x1,y1,num,atked_x,atked_y);
232                       ptr[3] = num;
233                       ptr[5] = 0;
234                 }
235                 if(num == 0){ //Parent loss
236                       break;
237                 }
238               }
239          }
```

第 210、154 行的 while(true)與(3)的使用方式相同

第 208、209 行與第 152、153 行分別為 Parent process 與 Child Process 初始化剩餘戰艦的座標總數(num = 2) 和 使用的炸彈個數 (bombnum = 0)

由 Parent process 先攻(213~221 行),當 ptr[3] != 0 (對方的戰艦尚未被擊沉)、num != 0(自己的戰艦尚未被擊沉)和 **ptr[5] = 0(控制輸出順序)**才進行攻擊。攻擊完後,ptr[5] = 1,使用炸彈數量+ 1(bombnum++),並換 Child process 輸出被攻擊後的**〝結果〞**,並輪到 Child 反擊**(若未被擊沉)**。

第 156~176 行,先檢查 ptr[3] != 0 (對方的戰艦尚未被擊沉,Parent 才能**發起攻擊**)、num != 0(自己的戰艦尚未被擊沉,才能**被** Parent 攻擊)和 **ptr[5] = 1(控制輸出順序)**,滿足以上條件後輸出可能的炸射結果,並將 ptr[3] = num(第 160 行,儲存**被炸射後**,剩餘的戰艦座標總數**給敵方判斷**是否仍需攻擊),若 Child 並未被擊沉(第 162 行,num != 0),發起反攻。反之,被擊沉就跳出迴圈,結束互相攻擊的階段,並將 ptr[5] = 2(換 Parent 輸出被攻擊的結果)。

第 222-227 行,如果敵方(Child process)已被擊沉(ptr[3] = 0,來自第 160 行),則將自己的 pid 和使用的砲彈個數分別存入 ptr[6]、ptr[7]中,並將 ptr[9] = 1,跳出迴圈(代表可以進入到最後的階段,輸出結果) 。

第 228-237 行,若敵方存活(ptr[3] != 0)且 ptr[5] = 2,輸出 Parent 被 Child 炸射後的結果,並將 ptr[3] = num(第 232 行,儲存**被炸射後**,剩餘的戰艦座標總數**給 Child 判斷**是否被擊沉)、ptr[5] = 0,進行新的一輪攻擊(循環)

第 235-237 行,若被擊沉(Parent 被 Child 擊沉)則跳出迴圈

第 177-182 行,接收到 Parent 被炸射後的結果,若 ptr[3] = 0,代表 Child 贏了, 將自己的 pid 和使用的砲彈個數分別存入 ptr[6]、ptr[7]中,並將 ptr[9] = 1,跳出迴圈(代表可以進入到最後的階段,輸出結果)

一輪的輸出結果

```
[3989 Parent]:bombing(0,3)
[3990 Child]:missing
[3990 Child]:bombing(1,0)
[3989 Parent]:hit
```

(5) 輸出最終結果(由父行程印出)

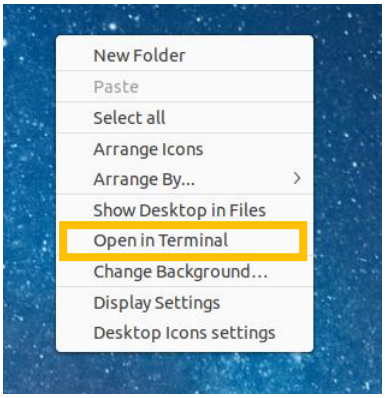
```
240             while(true){
241                     if(ptr[9] == 1){
242                     cout<<"["<<getpid()<<" Parent]:"<<ptr[6];
243                     cout<<" wins with "<<ptr[7]<<" bombs"<<endl;
244                     break;
245                     }
246             }
```

第 240 行的 while(true)與(3)的使用方式相同

當 ptr[9] = 1, 代表**上面的步驟均已完成**,即可輸出最後結果

6. 如何**編譯**

(1)在 Ubuntu 桌面右鍵 - > 選擇 Open Terminal



(2)輸入 g++ /home/yui/Desktop/yuitest.cpp -o/home/yui/Desktop/yuitest.out -Wall
(g++ /檔案路徑/檔案名稱.cpp -o/檔案路徑/檔案名稱.out -Wall),並按下 Enter

```
yui@yui-VirtualBox:~/Desktop$ g++ /home/yui/Desktop/yuitest.cpp -o/home/yui/Des
ktop/yuitest.out -Wall
```

(3)輸入 ./yuitest.out 5 10 0 (./檔案名稱.out 輸入 Random Seed 和 基本功能(模式))

```
yui@yui-VirtualBox:~/Desktop$ ./yuitest.out 5 10 0
```

5、10、0 分別為 p1、p2 以及 p3

(4)觀看結果

```
[5074 Parent]:Random Seed 5
[5074 Parent]:The gunboat:(0,3)(1,3)
[5075 Child]:Random Seed 10
[5075 Child]:The gunboat:(0,3)(0,2)
[5074 Parent]:bombing(3,2)
[5075 Child]:missing
[5075 Child]:bombing(0,3)
[5074 Parent]:hit
[5074 Parent]:bombing(0,3)
[5075 Child]:hit
[5075 Child]:bombing(0,0)
[5074 Parent]:missing
[5074 Parent]:bombing(3,1)
[5075 Child]:missing
[5075 Child]:bombing(1,3)
[5074 Parent]:hit and sinking
[5074 Parent]:5075 wins with 3 bombs
yui@yui-VirtualBox:~/Desktop$
```