1093304 作業系統_hw1 說明報告

設計理念：

```
2  #include <sys/types.h>
3  #include <sys/file.h>
4  #include <sys/mman.h>
5  #include <sys/wait.h>
6  using namespace std;
7
8  struct share_memory
9  {
10     int x, y, last_pid, turn; //砲彈要攻打的xy座標、紀錄最後攻打的是父還是子程序、回合(-1 == 未開始、0 == 子回合、1 == 父回合、2 == 結束)
11     bool hit; //記錄炸射狀態
12 };
13
14 int main(int argc, char* argv[])
15 {
16     const char* memname = "plog";
17     int fd = shm_open(memname, O_CREAT | O_TRUNC | O_RDWR, 0666); //建立share memory
18     if (fd < 0) ///若fd < 0則代表開啟有問題
19     {
20         cout << "shm open error\n";
21         shm_unlink(memname); //刪除記憶體共享檔案
22         return 0;
23     }
24
25     ftruncate(fd, sizeof(share_memory)); //將引數fd指定的檔案大小改為引數share_memory指定的大小
26     //把文件內容映射到一段記憶體上，對這段記憶體的讀取等同對文件的讀取
27     share_memory* shm = (share_memory*)mmap(NULL, sizeof(share_memory), PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
28     shm->turn = -1; //初始回合設為0代表未開始
29     int p1, p2, p3; //前2數為父子的亂數種
30     cout << "> prog1 ";
31     cin >> p1 >> p2 >> p3;
32
33     pid_t pid = fork(); //建立子程序
34     if (pid > 0) //若為父程序
35     {
36         srand(p1); //放入亂數種並印出
37         cout << "[" << getpid() << " Parent]: Random Seed: " << p1 << endl;
38     }
39     else if (pid == 0) //若為子程序
40     {
41         srand(p2); //放入亂數種並印出
42         cout << "[" << getpid() << " Child]: Random Seed: " << p2 << endl;
43     }
44
```

第 8～12 行：建立一個 struct，裡面包含砲彈要攻打的 x y 座標、紀錄最後攻打的是父還是子程序、回合和記錄炸射狀態。

第 17 行：建立 share memory。

第 27 行：把文件內容映射到一段記憶體上。

第 28 行：將初始回合設為-1 (-1=未開始、0=子回合、1=父回合、2=結束) 。

第 33 行：fork()建立子程序。

第 34～43 行：分別在父子程序中放入對應的亂數種。

```
44
45     const int dir[5]{ 0, 1, 0, -1, 0 }; //方向，找上下左右用
46     int Map[4][4]{}, x = rand() % 4, y = rand() % 4, ex, ey; //4 x 4地圖、亂數找船頭的座標、船尾座標
47     Map[x][y] = 1; //船頭所在位置設為1
48     for (int i = 0; i < 4; i++) //找船尾位置
49     {
50         if (x + dir[i] < 0 || x + dir[i] == 4 || y + dir[i + 1] < 0 || y + dir[i + 1] == 4) //若要制定船尾的座標超出範圍
51         {
52             continue; //則直接找下個方向
53         }
54
55         Map[ex = x + dir[i]][ey = y + dir[i + 1]] = 1; //找到船尾的位置並將其設為1
56         break;
57     }
```

第 45～57 行：找船尾的座標。

```
58
59      //父子程序分別印出各自的船座標，因parent似乎執行較快，故讓child開啟parent的回合
60      cout << "[" << getpid() << " " << (pid ? "Parent" : (shm->turn = 1, "Child")) << "]: The gunboat: (" << x << "," << y << ")(" << ex << "," << ey << ")\n";
61      int bombNum = 0; //雙方各自使用的炸彈數
62      for (int shipHit = 0; shm->turn != 2;) //各自的船被擊中的次數，若為2則代表整艘被擊沉
63      {
64          if (pid > 0) //父程序會跑的if
65          {
66              if (shm->turn == 1) //若turn == 1，代表輪到父程序的回合
67              {
68                  if (bombNum) //一開始只會發射砲彈，除此之外每次到父的回合時，都會先判斷是否被擊中，再發射砲彈
69                  {
70                      if (Map[shm->x][shm->y] == 1) //若此時share memory中的座標就是父的船還未擊沉的部分
71                      {
72                          shm->hit = true; //被擊中，狀態設為true
73                          Map[shm->x][shm->y] = 0; //代表擊中，父的該位置設為0
74                          cout << "[" << getpid() << " Parent]: hit"; //輸出擊中的訊息
75                          if (++shipHit == 2) //因有擊中，故shipHit先 + 1，再判斷若船艦的2個部分皆已被擊沉
76                          {
77                              cout << " and sinking\n"; //則再輸出已擊沉
78                              shm->turn = 2; //此時轟炸結束
79                              break; //並跳出迴圈
80                          }
81
82                          cout << "\n";
83                      }
84                      else //否則代表未擊中
85                      {
86                          shm->hit = false; //未擊中，狀態設為false
87                          cout << "[" << getpid() << " Parent]: missed\n";
88                      }
89                  }
90
91                  shm->last_pid = getpid(); //每次轟炸前，share memory 先記錄當前是誰轟炸
92                  shm->x = rand() % 4; //隨機選擇轟炸座標
93                  shm->y = rand() % 4;
94                  bombNum++; //丟出一顆砲彈，砲彈數 + 1
95                  cout << "[" << getpid() << " Parent]: bombing (" << shm->x << "," << shm->y << ")\n"; //輸出發射砲彈的訊息
96                  shm->turn = 0; //砲彈發射後輪到子程序的回合
97              }
98          }
```

```
99          else if (pid == 0) //子程序會跑的if
100         {
101             if (shm->turn == 0) //若turn == 0，代表輪到子程序的回合
102             {
103
104                 if (Map[shm->x][shm->y] == 1) //若此時share memory中的座標就是子的船還未擊沉的部分
105                 {
106                     shm->hit = true; //被擊中，狀態設為true
107                     Map[shm->x][shm->y] = 0; //代表擊中，子的該位置設為0
108                     cout << "[" << getpid() << " Child]: hit"; //輸出擊中的訊息
109                     if (++shipHit == 2) //因有擊中，故shipHit先 + 1，再判斷若船艦的2個部分皆已被擊沉
110                     {
111                         cout << " and sinking\n"; //則再輸出已擊沉
112                         shm->turn = 2; //此時轟炸結束
113                         exit(0); //結束子程序
114                     }
115
116                     cout << "\n";
117                 }
118                 else //否則代表未擊中
119                 {
120                     shm->hit = false; //未擊中，狀態設為false
121                     cout << "[" << getpid() << " Child]: missed\n";
122                 }
123
124                 shm->last_pid = getpid(); //每次轟炸前，share memory 先記錄當前是誰轟炸
125                 shm->x = rand() % 4; //隨機選擇轟炸座標
126                 shm->y = rand() % 4;
127                 bombNum++; //丟出一顆砲彈，砲彈數 + 1
128                 cout << "[" << getpid() << " Child]: bombing (" << shm->x << "," << shm->y << ")\n"; //輸出發射砲彈的訊息
129                 shm->turn = 1; //砲彈發射後輪到父程序的回合
130             }
131         }
132     }
```

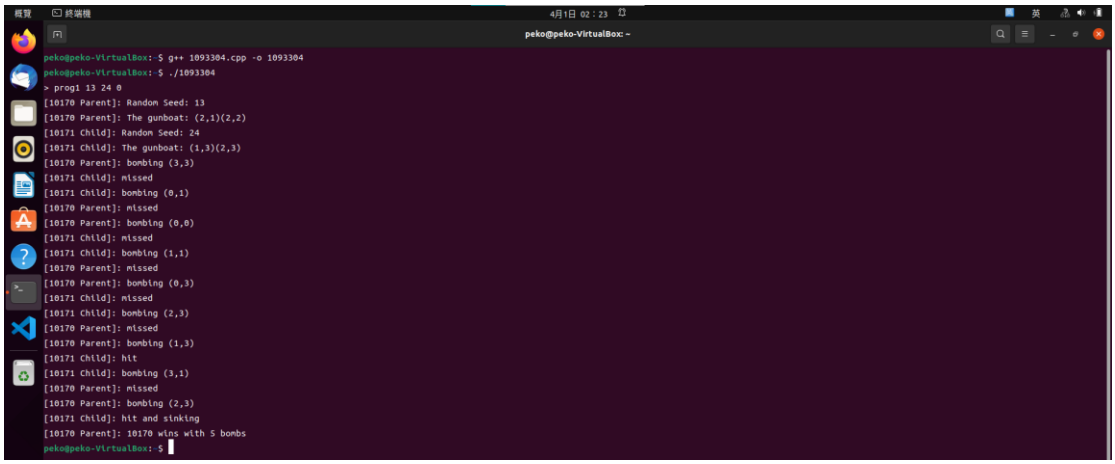
第 60 行：印出各自的船座標，其中當 child 執行到這行時，也同時開始 parent 的轟炸回合。

第 62～132 行：是一個迴圈，其中分成 2 個區塊：pid＞0 的父程序和 pid＝＝0 的子行程。兩部分幾乎相同，先判斷是否有沒有被轟炸到，再開始轟炸。例外是在第 68 行，因為 parent 一開始就要直接轟炸，所以當 bombNum == 0 時，不需要判斷有沒有被轟炸到。

```
134     if (pid > 0) //最後由父程序印出是誰獲勝和其所消耗的砲彈數
135     {
136         cout << "[" << getpid() << " Parent]: " << shm->last_pid << " wins with " << bombNum << " bombs\n";
137     }
138
139     munmap(shm, sizeof(share_memory)); //解除記憶體映射
140     shm_unlink(memname); //刪除記憶體共享檔案
141     return 0;
142 }
```

第 134～137 行：最後由父程序印出是誰獲勝和其所消耗的砲彈數。
第 139 和 140 行：程式收尾，釋放記憶體。

執行畫面：



開啟終端機，輸入 g++ 1093304.cpp -o 1093304，再輸入./1093304 後便可執行程式，輸入父子程序的亂數種和 0(基本功能)後，便可執行程式。由圖中可知，最後 parent 因為擊中 child 2 次而獲勝，過程中 parent 共用了 5 顆炸彈。