# 各 function 設計及變數定義

```
void error_and_die(const char *msg){
     perror(msg);
     exit(EXIT_FAILURE);
}
```

設定 share memory 是否 error，error 即執行此 function

Function:direction:設定船的位置，x1, x2 為設定後的 gunboat 隨機座標位置，其方法由此 function 決定，dir 為下一位置可接受之方向

```
int direction(int x1 ,int y1)          //decide the boat direction
{
   int dir;   //direction
   if(x1 ==0)
   {
      if(y1 == 0)
         dir = rand()%2+1;
      else if(y1 == 3)
         dir = rand()%2;
      else
         dir = rand()%3;
   }
   else if(x1 == 3)
   {
      if(y1 == 0)
         dir = rand()%2+2;
      else if(y1 == 3)
      {
         int a;
         a = rand()%2;
         if(a == 0)
            dir = 0;
         else
            dir = 3;
      }
      else
      {
         int a;
         a = rand()%3;
         if(a == 0)
           dir = 0;
         else if (a == 1)
           dir = 2;
         else
           dir = 3;
      }
   }
   else if(y1 == 0 && x1 != 0 && x1 != 3 )
      dir = rand()%3+1;
   else if (y1 == 3 && x1 != 0 && x1 != 3 )
   {
         int a;
         a = rand()%3;
         if(a == 0)
           dir = 0;
         else if (a == 1)
           dir = 1;
         else
           dir = 3;
   }
   else
      dir = rand()%4;
   return dir;
}
```

此部分為判斷當隨機的 x1=0 時，角落及邊界下一位置可接受之方向

此部分為判斷當隨機的 x1=3 時，角落及邊界下一位置可接受之方向

此部分為判斷當隨機的 y1=0 時，該邊界下一位置可接受之方向

此部分為判斷當隨機的 y1=3 時，該邊界下一位置可接受之方向

此部分為令其他隨機之位置

```
void ship(int dir , int x1, int y1 ,int &x2 , int &y2)
{                                          //decide boat second point
   x2 = x1;
   y2 = y1;
   switch (dir)
   {
      case 0:
        y2--;
        break;
      case 1:
        x2++;
        break;
      case 2:
        y2++;
        break;
      case 3:
        x2--;
        break;
   }
}
```

此 function:ship 為藉由 function 得出之 dir 來隨機令 gunboat 的第二個座標位置

```
bool beatk(int x1,int y1,int x2,int y2,int hitx,int hity,int &atkp)
{                                          //judging whether to be hit
   if(hitx == x1 && hity == y1)
   {
        atkp = 1;
        return 1;
   }
   else if(hitx == x2 && hity == y2)
   {
        atkp = 2;
        return 1;
   }
   else
   {
      return 0;
   }
return 0;
}
```

Function beatk:判斷是否被轟炸，方式為將被攻擊的船之座標(x1y1),(x2 y2)，隨機砲擊之座標(hitx hity)加到此 function 之參數，並增加一個查看器(atkp)，功能為避免重複轟炸卻沉船的情況，如果隨機砲擊位置=船座標點 1，查看器(atkp)就會記錄為 1 且回傳是否被炸到，反之，如隨機砲擊位置=船座標點 2，則查看器(atkp)紀錄為 2 且回傳是否被炸到

Function:report:判斷轟炸後的結果，方式為將是否被擊中(hit)，當前船隻被擊中的次數(hitnum)，被擊中船隻之編號及名稱(pi,str)，是否被擊沉(down)，查看是否重複被轟炸的查看器(atkp1)(第一次被轟炸之結果),查看器 2(atkp)查看當次被轟炸結果是否和第一次結果一樣，加入到參數中，並進行判斷　　註**配合上面 function:beatk 一起於 main function 使用為判斷是否被攻擊 and 是否被擊沉之方法

```
void report(int &hitnum,int hit,int pi,string str,int &down,int atkp,int
&atkp1)
{                                          //report the state after shooting
   if(hit == 1)
   {
      hitnum++;
      if(hitnum == 2)
      {
         if(atkp == atkp1)
         {
            hitnum--;
            cout<<"["<<pi<<" "<<str<<"]"<<": hit"<<endl;
         }
         else
         {
            cout<<"["<<pi<<" "<<str<<"]"<<": hit and sinking"<<endl;
            down++;
         }
      }
      else if(hitnum == 1)
      {
         atkp1 = atkp;
         cout<<"["<<pi<<" "<<str<<"]"<<": hit"<<endl;
      }
   }
   else if(hit == 0)
   {
      cout<<"["<<pi<<" "<<str<<"]"<<": missed"<<endl;
   }
}
```

如被擊中(hit=1)，則擊中次數+1 並判斷該船隻被擊中幾次。

1 次:則輸出 hit 且令 atkp1 = atkp(因為是第一次被轟炸，故將第一次擊中結果紀錄)

2 次:判斷當次被轟炸結果是否和第一次相符，如一樣則 hitnum-1 且輸出 hit，如不一樣則等於第二次有被轟炸到且是不同位置，此時船隻沉沒，輸出 hit and sinking，並令 down+1，代表該船隻沉沒

如 hit=0(沒被炸到)則輸出 missed

Function:atk 攻擊用的 function，方式為將隨機轟炸之位置(shotx shoty)，砲擊船隻之編號及名稱(pi,str)，該船隻砲擊次數(shotnum) 加入到參數中，並令(shotx shoty)為隨機位置，射擊次數+1，並輸出 bombing

```
void atk (int &shotx, int &shoty, int &shotnum, int pi , string str)
{
  shotx = rand()%4;
  shoty = rand()%4;
  shotnum++;
  cout<<"["<<pi<<" "<<str <<"]: bombing("<<shotx<<","<<shoty<<")"<<endl;
}
```

```
int main(int argc, char *argv[])
{
  int r;
  signal(SIGCHLD,SIG_IGN);

  const char *memname="sample";
  const size_t region_size = sysconf(_SC_PAGE_SIZE);

  int fd = shm_open(memname, O_CREAT|O_TRUNC|O_RDWR,0666);
  if(fd==-1)
     error_and_die("shm_open");

  r=ftruncate(fd,region_size);

  if(r!=0)
     error_and_die("ftruncate");

  int *shm=(int *)mmap(0,region_size,PROT_READ|PROT_WRITE,MAP_SHARED,fd,0);

  if(shm == MAP_FAILED)
      error_and_die("mmap");

//------------------------------share memory-------------------------
```

**Shame memory，參考及修改教授教材之 03-shm.pdf 及 shm.c

```
    int pid ;
    pid = fork();
    char ps = *argv[1],cs = *argv[2];
    int parseed = (int)ps -'0',chiseed = (int)cs -'0';
    shm[2] = 0;  // judging whether to jump
    shm[3] = 0;  // save pi
    shm[4] = 0;  // save shotnum
    shm[5] = -1;  // step of call Random seed and The gunboat
    shm[6] = 0;  // step of shot
    shm[7] = 0;  // judging whether to end
```

定義:pid=fork -creat child process 必要步驟  ps,cs -給定之 random seed

Parsed,chiseed -random seed 值由字元轉 int

shm[] share memory:存取 parent 及 child 雙方狀態及資料的共用區域

shm[2]  判斷對方是否被擊沉

shm[3]  存取勝利者之 pid 編號

shm[4]  存取勝利者之砲擊次數

shm[5]  由此和 while 迴圈來判斷 parent or child 誰先輸出 random seed 和 gunboat 位置

shm[6]  由此和 while 迴圈來判斷砲擊順序

shm[7]  由此來等待雙方 process 結束並輸出結果，避免 zombie 產生

**我並沒有做進階功能，故 p3 輸入會默認為 0

# Parent process－random seed and gunboat output

```
255  else //parent
256  {
257      string str = ("Parent");
258      int x1, y1, x2 = 0, y2 = 0,dir , pi = getpid();
259      srand(time(NULL) ^parseed);
260      cout<< [ <<pi<<" Parent" << ] << : Random Seed  <<argv[1]<<endl;
261      //Random seed
262
263      while(true)  //the while loop is telling information of parent
264      {
265         if(shm[5] == 0)
266         {
267            shm[5]--;
268         }
269         else if(shm[5] == -2)
270         {
271            x1 = rand()%4;
272            y1 = rand()%4;
273            dir = direction(x1,y1);
274            ship(dir, x1, y1, x2, y2);
275            cout<<"["<<pi<<" Parent" <<"]"<<": The gunboat: ";
276            cout<<"("<<x1<<","<<y1<<") ("<<x2<<","<<y2<<")"<<endl;
277            shm[5] = -3;
278         }
279         //tell where boat
280
281         else if(shm[5] == -4)
282            break;
283      }
284
```

定義: x1,y1-船隻位置 1　x2,y2-船隻位置 2　dir 是以 x1,y1 為基礎，x2,y2 位置可接受之方向

pi parent pid 編號

srand 設置一個隨機種子，保證每次 rand 值都不一樣 **種子利用給定之亂數種子碼

輸出 random seed（因 parent 是第一個輸出的故可以直接輸出無須判斷順序）

透過 while 及 shm[5]判斷與 child 的輸出順序，當 shm[5]=0 代表 parent 的 random seed 已輸出，令 shm[5]=-1，當 shm[5]=-1，child 讀取到 shm[5]=-1 且判斷後輸出 child 的 random seed, 而後令 shm[5]=-2，parent 讀取 shm[5]之值，此時 shm[5]=-2，因此滿足 else if(shm[5] == -2)之條件，令 x1,x2 為隨機位置, 並利用 dir 和 function:direction 及 function ship 決定 x2,y2 之位置，隨後輸出 parent gunboat 位置，並令 shm[5]=-3，child 讀取到後做了一樣的動作，且令 shm[5]=-4，當 parent 讀取到 shm[5]=-4 後跳出迴圈　**造成順序的方式為使用 while(true)持續讀取 shm[5]值之變化來排序

# Child process －random seed and gunboat output

```
187  if(pid == 0)  //child
188  {
189     string str = ("Child");
190     int x1, y1, x2 = 0, y2 = 0,dir,pi = getpid();
191     srand(time(NULL) ^ chiseed);
192     while(true)  //the while loop is telling information of child
193     {
194        if(shm[5] == -1)
195        {
196           cout<<"["<<pi<<" Child" <<"]"<<": Random Seed "<<argv[2]<<endl;
197           shm[5] = -2;
198           //Random seed
199        }
200        else if(shm[5] == -3)
201        {
202           x1 = rand()%4;
203           y1 = rand()%4;
204           dir = direction(x1,y1);
205           ship(dir, x1, y1, x2, y2);
206           cout<<"["<<pi<<" Child" <<"]"<<": The gunboat: ";
207           cout<<"("<<x1<<","<<y1<<") ("<<x2<<","<<y2<<")"<<endl;
208           shm[5] = -4;
209        }
210        if(shm[5] == -4)
211          break;

212
213     //tell where boat
214     }
```

定義: x1,y1-船隻位置 1　x2,y2-船隻位置 2　dir 是以 x1,y1 為基礎，x2,y2 位置可接受之方向
srand 設置一個隨機種子，保證每次 rand 值都不一樣 **種子利用給定之亂數種子碼

透過 while 及 shm[5]判斷與 parent 的輸出順序，當 shm[5]=-1 代表 parent 的 random seed 已輸出 child 讀取到 shm[5]=-1 且判斷後輸出 child 的 random seed, 而後令 shm[5]=-2，parent 讀取 shm[5]之值，此時 shm[5]=-2，因此滿足 else if(shm[5] == -2)之條件，令 x1,x2 為隨機位置, 並利用 dir 和 function:direction 及 function ship 決定 x2,y2 之位置，隨後輸出 parent gunboat 位置，並令 shm[5]=-3，child 讀取到後做了一樣的動作，且令 shm[5]=-4，並且跳出迴圈　**造成順序的方式為使用 while(true)持續讀取 shm[5]值之變化來排序

# Parent process-砲擊及被砲擊部分

```
287     int hitnum = 0, down=0, shotnum =0 ,atkp = 0,atkp1=0;
288     while(true)                 // atk and be atked loop
289     {
290         if(shm[6] == 0)
291         {
292            int shotx=0,shoty=0;
293            atk(shotx, shoty, shotnum, pi, str);
294            shm[0] = shotx ;
295            shm[1] = shoty ;
296            shm[6] = 1;
297            //atk
298
299         }
300         if(shm[2] == 1)
301         {
302            shm[3] = pi;
303            shm[4] = shotnum;
304            shm[7] = 1;
305            break;
306         }
307         // judging whether to end
308
309         else if(shm[6] == -1)
310         {
311            int hitx = shm[0],hity = shm[1];
312            bool hit =0;
313            hit = beatk(x1, y1, x2, y2, hitx, hity,atkp);
314            report(hitnum, hit, pi, str,down,atkp,atkp1);
315            shm[6] = 0;
316            //be atked
317
318            if(down == 1)
319            {
320               shm[2] = 1;
321               break;
322            }
323            //judging whether to be atked
324
325         }
326     }
```

砲擊 step.1

砲擊 step.4

定義:被擊中次數(hitnum)　是否被擊沉(down)　砲擊次數(shotnum)
　　查看是否重複被轟炸(atkp,atkp1)

持續進行砲擊和被砲擊，直到其中一方結束

當 shm[6]=0 會執行砲擊相關程式，呼叫 atk 進行攻擊 **見上方說明 並將砲擊的位置存於 shm[0]和 shm[1]供 child 操作，並令 shm[6]=1 使 child 知道 parent 已砲擊完畢

利用 shm[2]判斷對方是否被擊沉,如是，則 parent 勝利，將 parent 資訊存於 shm[3]及 shm[4] &&此 if 放這邊原因為 parent 先砲擊，再判斷對方是否被擊沉，如被擊沉直接跳出迴圈，則不會到執行到下方的被轟炸程式導致 bug

當 shm[6]=-1 代表已被砲擊，執行以下被砲擊之程式。 令 hitx,hity 讀取 child 砲擊的位置(shm[0]和 shm[1]), hit=0(重置是否被砲擊的判斷因子) 呼叫 beatk 和 report 來計算是否被砲擊，被砲擊次數，是否被重複砲擊以及是否倒下 **見上方說明 並令 shm[6]=0 使 parent 再次砲擊，形成一個循環

判段是否被擊沉(if (down=1))
是的話 shm[2]=1，告知對方已被擊沉且跳出迴圈

# Parent process-砲擊及被砲擊部分

```
217     int hitnum = 0, down=0, shotnum =0 ,atkp =0,atkp1=0;
218     while(true)                    // atk and be atked loop
219     {
220        if(shm[6] == 1)
221        {
222           int hitx = shm[0],hity = shm[1];
223           bool hit =0;
224           hit = beatk(x1, y1, x2, y2, hitx, hity,atkp);
225           report(hitnum, hit, pi, str,down,atkp,atkp1);
226
227           //be atked
228           if(down == 1)
229           {
230              shm[2] = 1;
231              break;
232           }
233           //judging whether to be atked
234
235           int shotx=0,shoty=0;
236           atk(shotx, shoty, shotnum, pi, str);
237           shm[0] = shotx ;
238           shm[1] = shoty ;
239           shm[6] = -1;
240           //atk
241        }
242        if(shm[2] == 1)
243        {
244           shm[3] = pi;
245           shm[4] = shotnum;
246           shm[7] = 1;
247           break;
248        }
249        // judging whether to end
250
251     }
252     exit(0);
253  }
```

砲擊 step.2

砲擊 step.3

定義:被擊中次數(hitnum)　是否被擊沉(down)　砲擊次數(shotnum)
　　查看是否重複被轟炸(atkp, atkp1)

持續進行砲擊和被砲擊，直到其中一方結束

當 shm[6]=1 代表已被砲擊，執行以下被砲擊之程式。 令 hitx, hity 讀取 child 砲擊的位置(shm[0]和 shm[1]), hit=0(重置是否被砲擊的判斷因子) 呼叫 beatk 和 report 來計算是否被砲擊，被砲擊次數，是否被重複砲擊以及是否倒下 **見上方說明

判段是否被擊沉(if (down=1))
是的話 shm[2]=1，告知對方已被擊沉且跳出迴圈

因 child 被砲擊後接著會直接對 parent 砲擊，因此無須再寫判斷規定順序，可直接接續執行砲擊相關程式，呼叫 atk 進行攻擊 **見上方說明 並將砲擊的位置存於 shm[0]和 shm[1]供 child 操作，並令 shm[6]=-1 使 child 知道 parent 已砲擊完畢

利用 shm[2]判斷對方是否被擊沉, 如是，則 child 勝利，將 child 資訊存於 shm[3]及 shm[4]

迴圈結束即 child 砲擊及被砲擊作業完成，因此 exit 結束 process

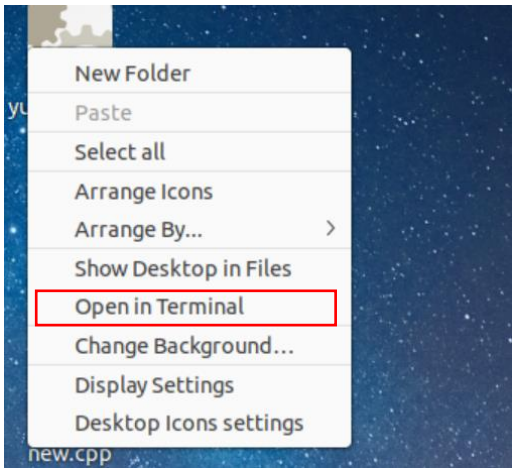
## Print grade and end

```
327     while(true)
328     {
329        if(shm[2] == 1 && shm[7] == 1)
330        {
331           cout<<"["<<pi<<" Parent" <<"]: "<<shm[3]<<" wins with "<<shm[4];
332           cout<<" bombs"<<endl;
333           break;
334        }
335
336        // get grade
337     }
338
339  }
340
341     close(fd);
342     r = munmap(shm,region_size);
343     if(r!=0)
344        error_and_die("shm_unlink");
345     return 0;
346     //end share memory
347
348 }
```

持續判斷是否有人被擊沉且雙方是否結束，是的話輸出成績 **這邊用 while 持續判斷原因是因為避免 child 來不及將資訊傳入 shm[3]及 shm[4]就輸出成績導致值為 0 wins with 0

關閉 share memory

# 如何編譯及操作




Step1:在 Ubuntu 桌面右鍵->選擇 Open Terminal

```
jacky@jacky-VirtualBox:~/Desktop$ g++ /home/jacky/Desktop/forkshm.cpp -o/home/j
acky/Desktop/forkshm.out -Wall
```

Step2:在 Terminal 鍵入以上指令 g++ /檔案路徑/檔案名稱.cpp -o/檔案路徑/檔案名稱.out -Wall Enter 來建置 out 檔

```
jacky@jacky-VirtualBox:~/Desktop$ ./forkshm.out 5 10 0
```

Step3:在 Terminal 鍵入以上指令 ./檔案名稱.out 輸入種子及模式

```
[13561 Parent]: missed
[13561 Parent]: bombing(1,0)
[13562 Child]: missed
[13562 Child]: bombing(2,1)
[13561 Parent]: missed
[13561 Parent]: bombing(0,2)
[13562 Child]: missed
[13562 Child]: bombing(0,2)
[13561 Parent]: missed
[13561 Parent]: bombing(2,3)
[13562 Child]: missed
[13562 Child]: bombing(0,2)
[13561 Parent]: missed
[13561 Parent]: bombing(1,1)
[13562 Child]: missed
[13562 Child]: bombing(1,2)
[13561 Parent]: missed
[13561 Parent]: bombing(2,2)
[13562 Child]: missed
[13562 Child]: bombing(3,0)
[13561 Parent]: missed
[13561 Parent]: bombing(2,2)
[13562 Child]: missed
[13562 Child]: bombing(0,0)
[13561 Parent]: missed
[13561 Parent]: bombing(1,3)
[13562 Child]: hit and sinking
[13561 Parent]: 13561 wins with 17 bombs
jacky@jacky-VirtualBox:~/Desktop$
```

Step4:執行成功