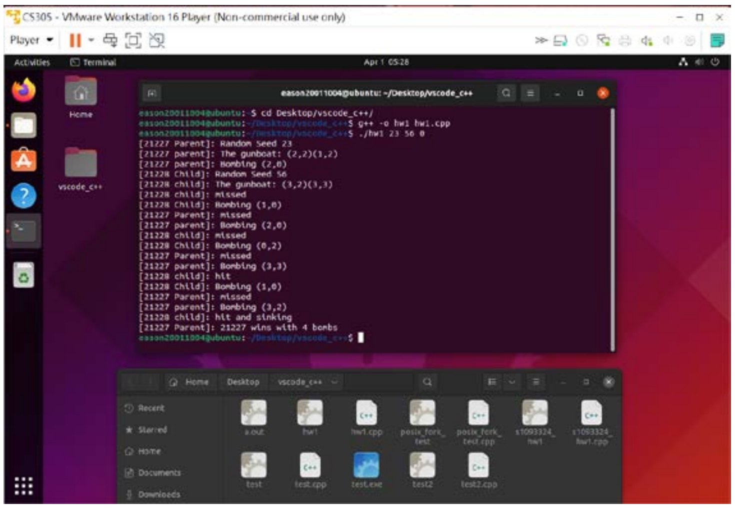
執行說明：



CS305 - VMware Workstation 16 Player (Non-commercial use only)
Activities
Terminal
Apr 1 05:28
eason20011004@ubuntu: ~/Desktop/vscode_c++
Home
vscode_c++

```
eason20011004@ubuntu:~$ cd Desktop/vscode_c++/
eason20011004@ubuntu:~/Desktop/vscode_c++$ g++ -o hw1 hw1.cpp
eason20011004@ubuntu:~/Desktop/vscode_c++$ ./hw1 23 56 0
[21227 Parent]: Random Seed 23
[21227 parent]: The gunboat: (2,2)(1,2)
[21227 parent]: Bombing (2,0)
[21228 Child]: Random Seed 56
[21228 child]: The gunboat: (3,2)(3,3)
[21228 child]: missed
[21228 Child]: Bombing (1,0)
[21227 Parent]: missed
[21227 parent]: Bombing (2,0)
[21228 child]: missed
[21228 Child]: Bombing (0,2)
[21227 Parent]: missed
[21227 parent]: Bombing (3,3)
[21228 child]: hit
[21228 Child]: Bombing (1,0)
[21227 Parent]: missed
[21227 parent]: Bombing (3,2)
[21228 child]: hit and sinking
[21227 Parent]: 21227 wins with 4 bombs
eason20011004@ubuntu:~/Desktop/vscode_c++$
```

先用 cd 移動到資料所在資料夾
再用 g++ -o (檔名) (檔名).cpp 編譯
最後用 ./(檔名) (數字) (數字) 0 執行程式

數字部分為 0~99 整數
第三個參數固定為 0

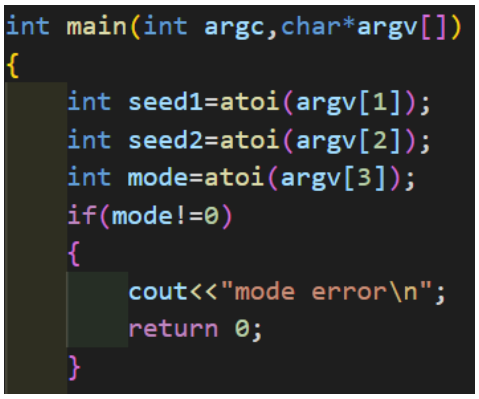
程式碼說明：

此為之後建立 shm 空間的名稱

```
#define dfder "datafolder"
```

這個 struct 是我要分享得內容 包成一包方便計算 shm 的儲存空間

```
struct mydata {
        int attackflag = 0;    → 決定誰攻擊
        bool csink=0;          ┐→ 判斷沉了沒
        bool psink=0;          ┘
        bool hitc1=0;          ┐
        bool hitc2=0;          │→ 擊中船的一半 or 全部
        bool hitp1=0;          │
        bool hitp2=0;          ┘
        int shoot[2]={0};      → 炸的位置
        int pboomcount=0;      ┐→ 炸彈量
        int cboomcount=0;      ┘
        pid_t cpid=0;          → 給 parent 知道child 的 pid
};
```



```
int main(int argc,char*argv[])
{
    int seed1=atoi(argv[1]);
    int seed2=atoi(argv[2]);
    int mode=atoi(argv[3]);
    if(mode!=0)
    {
        cout<<"mode error\n";
        return 0;
    }
```

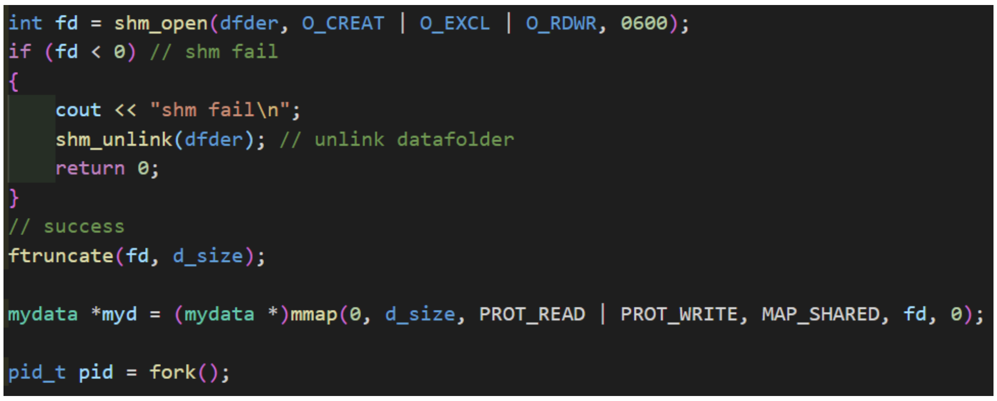
用 argv 讀入三個參數 分別為

parent process 的 seed、

child process 的 seed 和 mode

(seed 為 0~99 的數字

、mode 則固定為 0)

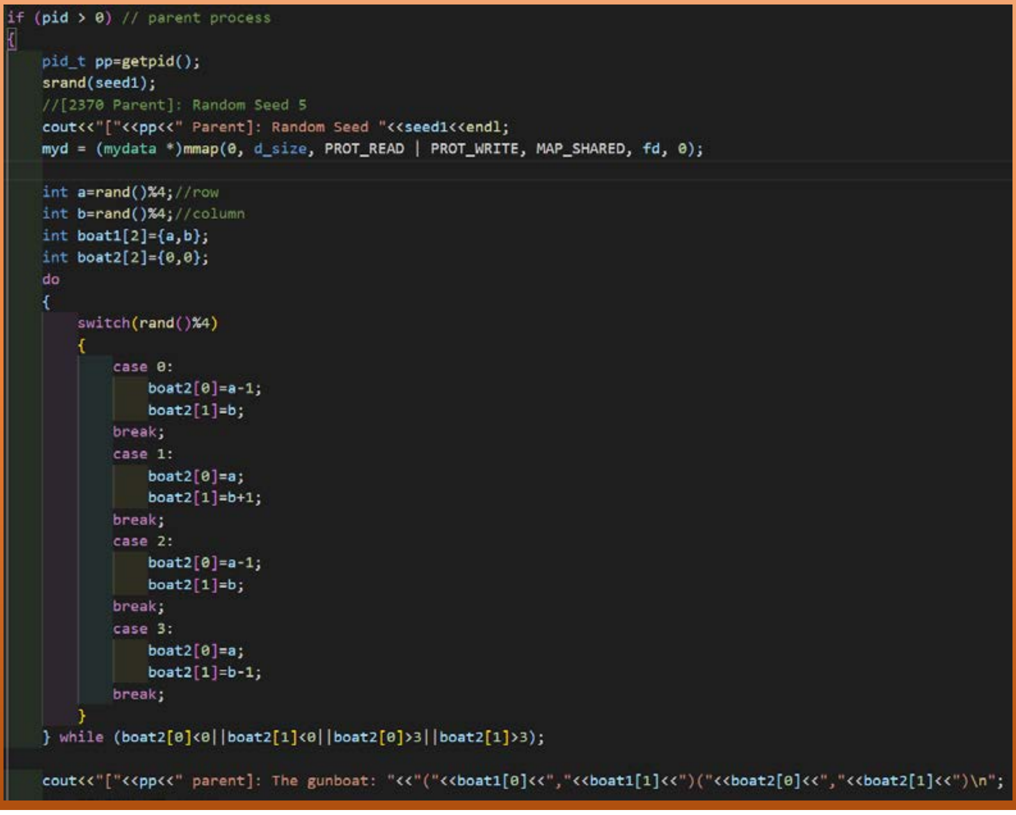


```
int fd = shm_open(dfder, O_CREAT | O_EXCL | O_RDWR, 0600);
if (fd < 0) // shm fail
{
    cout << "shm fail\n";
    shm_unlink(dfder); // unlink datafolder
    return 0;
}
// success
ftruncate(fd, d_size);

mydata *myd = (mydata *)mmap(0, d_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);

pid_t pid = fork();
```

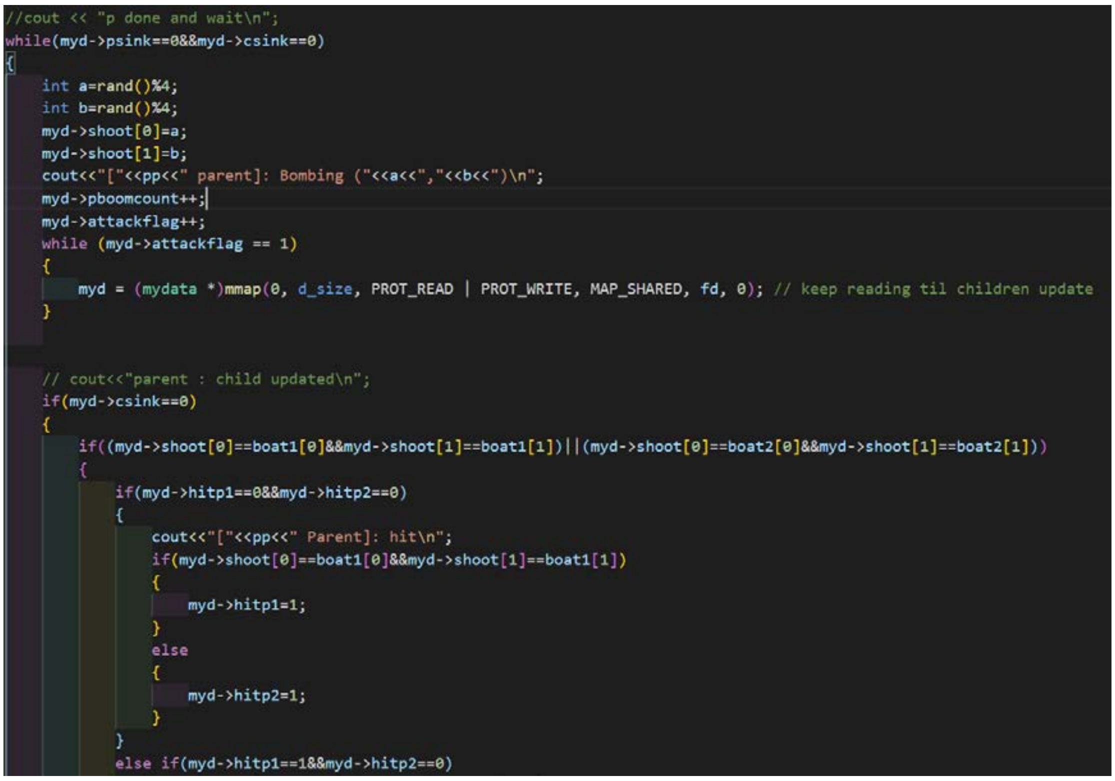
開啟一個空間大小和 struct mydata 相同的空間來分享資料，並且 fork()

Parent process :

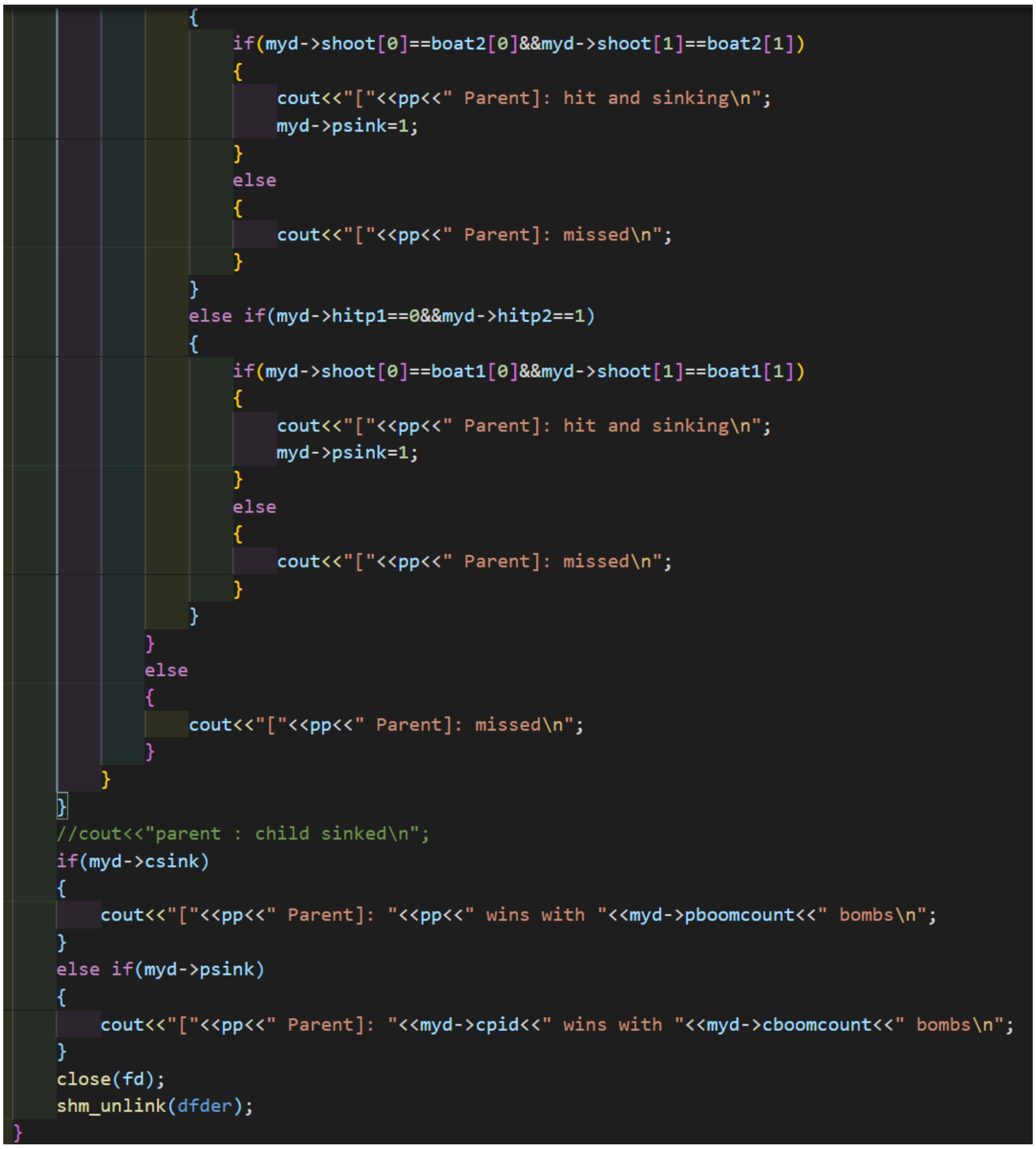
先用 getpid 取的 pid 並印出，創建自己的 struct，跟 shm 同步，之後設定小船位置，do while 部分是為了防止設到範圍外，然後再隨機選擇 4*4 地圖的其中一格進行第一次轟炸



```
if (pid > 0) // parent process
{
    pid_t pp=getpid();
    srand(seed1);
    //[2370 Parent]: Random Seed 5
    cout<<"["<<pp<<" Parent]: Random Seed "<<seed1<<endl;
    myd = (mydata *)mmap(0, d_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);

    int a=rand()%4;//row
    int b=rand()%4;//column
    int boat1[2]={a,b};
    int boat2[2]={0,0};
    do
    {
        switch(rand()%4)
        {
            case 0:
                boat2[0]=a-1;
                boat2[1]=b;
            break;
            case 1:
                boat2[0]=a;
                boat2[1]=b+1;
            break;
            case 2:
                boat2[0]=a-1;
                boat2[1]=b;
            break;
            case 3:
                boat2[0]=a;
                boat2[1]=b-1;
            break;
        }
    } while (boat2[0]<0||boat2[1]<0||boat2[0]>3||boat2[1]>3);

    cout<<"["<<pp<<" parent]: The gunboat: "<<"("<<boat1[0]<<","<<boat1[1]<<")("<<boat2[0]<<","<<boat2[1]<<")\n";
```



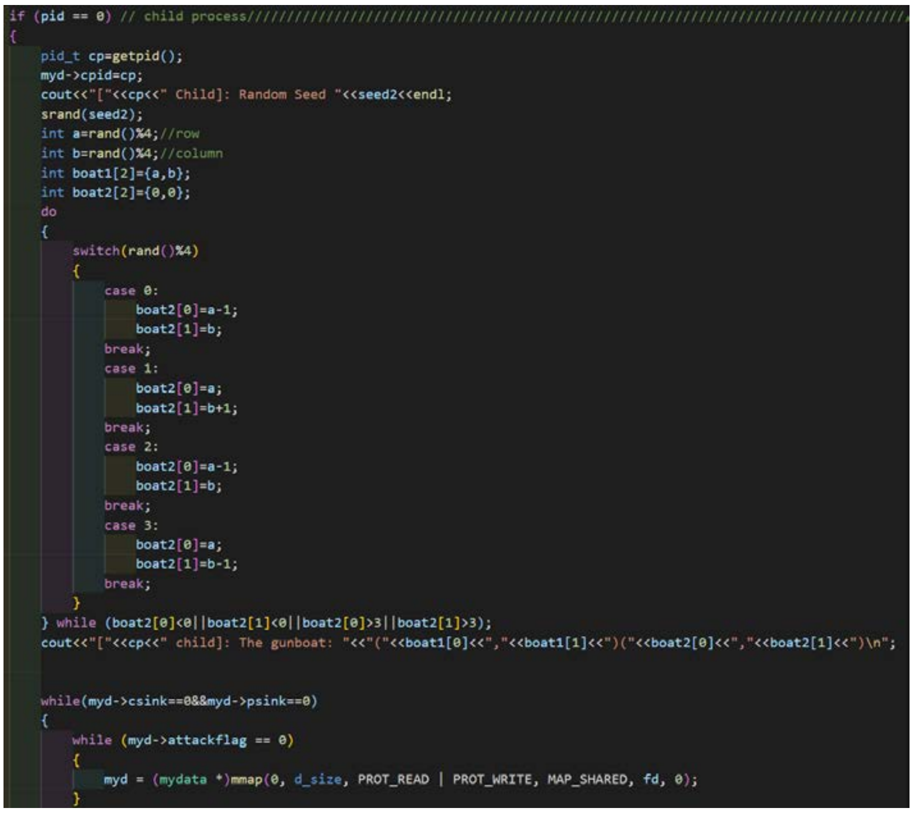
```
//cout << "p done and wait\n";
while(myd->psink==0&&myd->csink==0)
{
    int a=rand()%4;
    int b=rand()%4;
    myd->shoot[0]=a;
    myd->shoot[1]=b;
    cout<<"["<<pp<<" parent]: Bombing ("<<a<<","<<b<<")\n";
    myd->pboomcount++;
    myd->attackflag++;
    while (myd->attackflag == 1)
    {
        myd = (mydata *)mmap(0, d_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0); // keep reading til children update
    }


    // cout<<"parent : child updated\n";
    if(myd->csink==0)
    {
        if((myd->shoot[0]==boat1[0]&&myd->shoot[1]==boat1[1])||(myd->shoot[0]==boat2[0]&&myd->shoot[1]==boat2[1]))
        {
            if(myd->hitp1==0&&myd->hitp2==0)
            {
                cout<<"["<<pp<<" Parent]: hit\n";
                if(myd->shoot[0]==boat1[0]&&myd->shoot[1]==boat1[1])
                {
                    myd->hitp1=1;
                }
                else
                {
                    myd->hitp2=1;
                }
            }
            else if(myd->hitp1==1&&myd->hitp2==0)
```



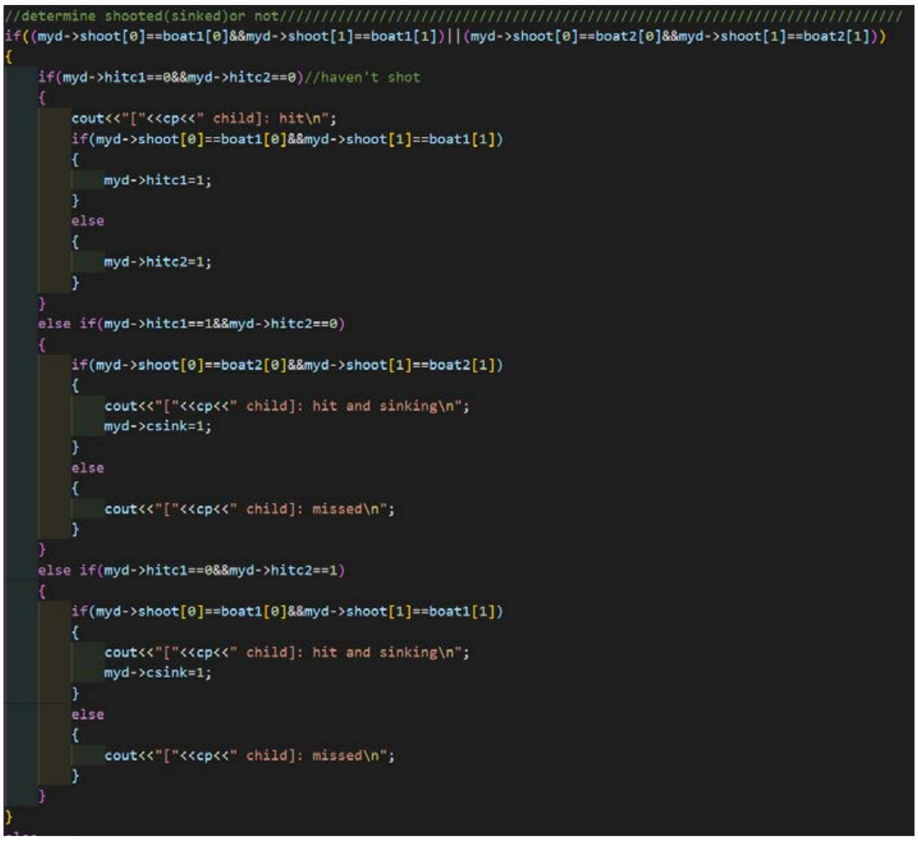
```
                {
                    if(myd->shoot[0]==boat2[0]&&myd->shoot[1]==boat2[1])
                    {
                        cout<<"["<<pp<<" Parent]: hit and sinking\n";
                        myd->psink=1;
                    }
                    else
                    {
                        cout<<"["<<pp<<" Parent]: missed\n";
                    }
                }
                else if(myd->hitp1==0&&myd->hitp2==1)
                {
                    if(myd->shoot[0]==boat1[0]&&myd->shoot[1]==boat1[1])
                    {
                        cout<<"["<<pp<<" Parent]: hit and sinking\n";
                        myd->psink=1;
                    }
                    else
                    {
                        cout<<"["<<pp<<" Parent]: missed\n";
                    }
                }
            }
            else
            {
                cout<<"["<<pp<<" Parent]: missed\n";
            }
        }
    }
    //cout<<"parent : child sinked\n";
    if(myd->csink)
    {
        cout<<"["<<pp<<" Parent]: "<<pp<<" wins with "<<myd->pboomcount<<" bombs\n";
    }
    else if(myd->psink)
    {
        cout<<"["<<pp<<" Parent]: "<<myd->cpid<<" wins with "<<myd->cboomcount<<" bombs\n";
    }
    close(fd);
    shm_unlink(dfder);
}
```

之後進入迴圈，我的子父行程同步原理是使用一個 attackflag 存在 shm 中去控制該輪到誰行動，當 flag=1 時，代表 parent 做完動作，換 child 做，parent 持續讀取 shm 直到 child 做完，flag 變回 0，再換 parent 做(迴圈重複執行直到其中一艘船沉沒)

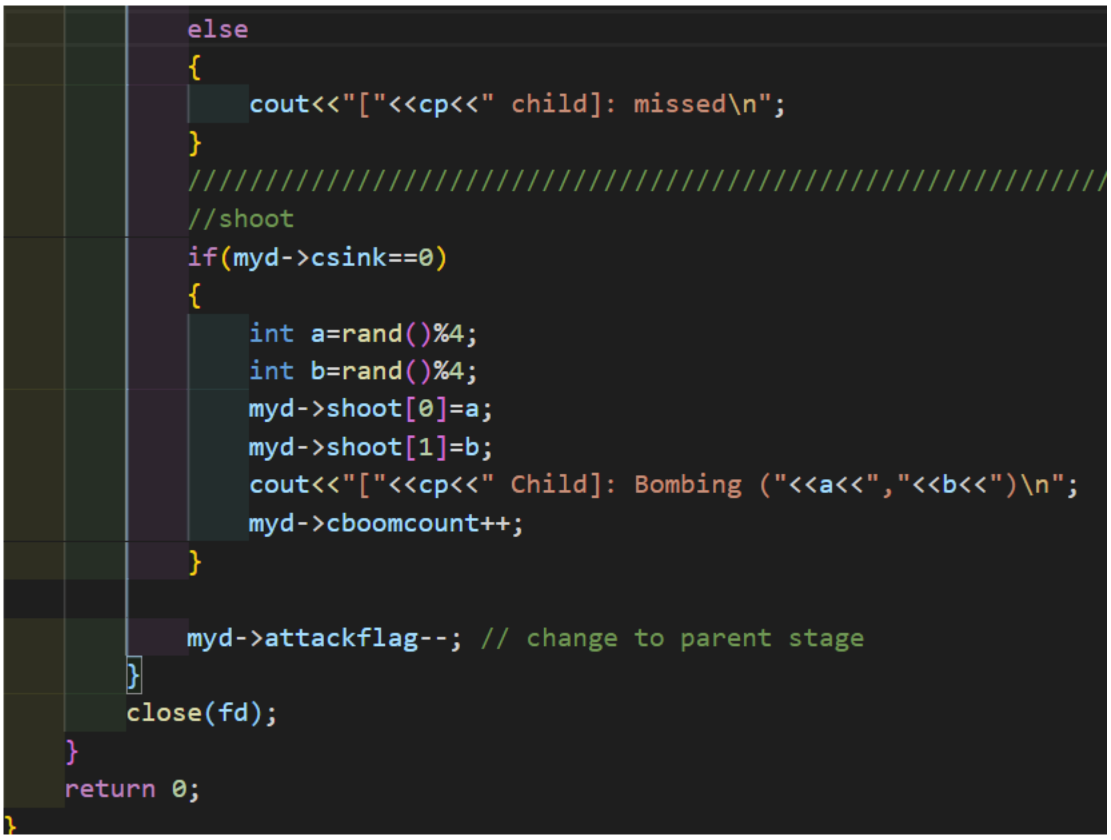
Child process :



```
if (pid == 0) // child process///////////////////////////////////////////////////////////////////////////////////
{
    pid_t cp=getpid();
    myd->cpid=cp;
    cout<<"["<<cp<<" Child]: Random Seed "<<seed2<<endl;
    srand(seed2);
    int a=rand()%4;//row
    int b=rand()%4;//column
    int boat1[2]={a,b};
    int boat2[2]={0,0};
    do
    {
        switch(rand()%4)
        {
            case 0:
                boat2[0]=a-1;
                boat2[1]=b;
            break;
            case 1:
                boat2[0]=a;
                boat2[1]=b+1;
            break;
            case 2:
                boat2[0]=a-1;
                boat2[1]=b;
            break;
            case 3:
                boat2[0]=a;
                boat2[1]=b-1;
            break;
        }
    } while (boat2[0]<0||boat2[1]<0||boat2[0]>3||boat2[1]>3);
    cout<<"["<<cp<<" child]: The gunboat: "<<"("<<boat1[0]<<","<<boat1[1]<<")("<<boat2[0]<<","<<boat2[1]<<")\n";


    while(myd->csink==0&&myd->psink==0)
    {
        while (myd->attackflag == 0)
        {
            myd = (mydata *)mmap(0, d_size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
        }
```



```
//determine shooted(sinked)or not/////////////////////////////////////////////////////////////////////////////
if((myd->shoot[0]==boat1[0]&&myd->shoot[1]==boat1[1])||(myd->shoot[0]==boat2[0]&&myd->shoot[1]==boat2[1]))
{
    if(myd->hitc1==0&&myd->hitc2==0)//haven't shot
    {
        cout<<"["<<cp<<" child]: hit\n";
        if(myd->shoot[0]==boat1[0]&&myd->shoot[1]==boat1[1])
        {
            myd->hitc1=1;
        }
        else
        {
            myd->hitc2=1;
        }
    }
    else if(myd->hitc1==1&&myd->hitc2==0)
    {
        if(myd->shoot[0]==boat2[0]&&myd->shoot[1]==boat2[1])
        {
            cout<<"["<<cp<<" child]: hit and sinking\n";
            myd->csink=1;
        }
        else
        {
            cout<<"["<<cp<<" child]: missed\n";
        }
    }
    else if(myd->hitc1==0&&myd->hitc2==1)
    {
        if(myd->shoot[0]==boat1[0]&&myd->shoot[1]==boat1[1])
        {
            cout<<"["<<cp<<" child]: hit and sinking\n";
            myd->csink=1;
        }
        else
        {
            cout<<"["<<cp<<" child]: missed\n";
        }
    }
}
```



```
                else
                {
                    cout<<"["<<cp<<" child]: missed\n";
                }
                ////////////////////////////////////////////////////
                //shoot
                if(myd->csink==0)
                {
                    int a=rand()%4;
                    int b=rand()%4;
                    myd->shoot[0]=a;
                    myd->shoot[1]=b;
                    cout<<"["<<cp<<" Child]: Bombing ("<<a<<","<<b<<")\n";
                    myd->cboomcount++;
                }

                myd->attackflag--; // change to parent stage
            }
            close(fd);
        }
        return 0;
}
```

Child process 和 parent 大同小異，差在開始執行並設定好船位置後就開始等待 parent 的第一次轟炸，後續就是互相轟炸直到其中一方被擊沉，結果存在 shm 中並由 parent 輸出