```
16      string infile = "blake.txt";
17      string outfile = "happy.tmp";
18      //./master.out A.txt B.tmp
19      if(argc > 1){
20          infile = argv[1];
21          outfile = argv[2];
22      }
23
24      const char* infile_name = infile.c_str();
25      const char* outfile_name = outfile.c_str();
```

依照使用者輸入的參數設定讀寫檔的檔名，如果沒有輸入參數的話就預設讀blake.txt，寫happy.tmp

並將字串轉成接下來要用的型態

```
27      int ifs = open(infile_name, O_RDWR, 0700);
28      if (ifs < 0)
29          cout << "Fail to open file.\n";
```

開啟等一下要讀的檔案

```
31      else{
32          int p[2];
33          pipe(p);
34          //child, write file(read from pipe)
35          if(fork() == 0){
36              close(0);
37              dup(p[0]);
38              close(p[0]);
39              close(p[1]);
40              execlp("./mmv.out", "./mmv.out", outfile_name, NULL);
41          }
```

建立pipe，且子process要執行mmv.out檔案，從pipe內讀取字串並寫入新的檔案

36~39行是將該process的標準輸入變成pipe的讀取端

```cpp
int main(int argc,char *argv[])
{
    int fout = open(argv[1], O_RDWR | O_CREAT, 0700);
    if (!fout){
        cout << "fail to open file.\n";
    }
    write(fout, "\\----Say Hello to s1083343!----\\\n", 34);
    string w,tmp;
    while(getline(cin,tmp))
    {
        w += tmp;
        w += "\n";
    }
    const char* w2=w.c_str();

    write(fout, w2, 200);
    close(fout);
    cout << "Successful (#" << getpid() << ")!\n";
}
```

此為mmv.cpp，argv[1]是寫檔的名字

接下來開始讀pipe內的資料，標準輸入已經改成了pipe的讀取端，所以cin會從pipe內讀出資料，也就是父process讀出的資料，最後再寫進檔案，並印出子process id

```
42          //parent, read file(write into pipe)
43          else{
44              char w1[200] = {};
45              read(ifs, w1, 200);
46              write(p[1], w1, 200);
47              close(p[0]);
48              close(p[1]);
49              close(ifs);
50          }
```

父process會讀檔並寫進pipe讓子process讀取