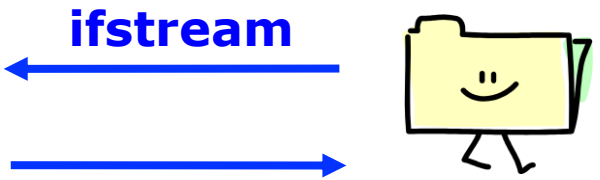
Lecture 3 - File Processing

Meng-Hsun Tsai
CSIE, NCKU

program



file

write_file.cpp

```
 1 #include <iostream>
 2 #include <string>
 3 #include <fstream>
 4 #include <cstdlib>
 5 using namespace std;
 6 int main()
 7 {
 8     string name;
 9     float proj, exam;
10     ofstream outFile("outfile", ios::out);
11     if(!outFile) {
12         cerr << "Failed opening" << endl;
13         exit(1);
14     }
15     cout << "Enter NAME PROJ EXAM
           each line.\n"
16         << "EOF to finish.\n" << "? ";
17     outFile <<
"Name\tProj\tExam\tTotal\n";
18     while(cin >> name >> proj >> exam) {
19         outFile << name << "\t" << proj << "\t" <<
             exam << "\t"
20             << proj*0.65 + exam*0.35 << endl;
21         cout << "? ";
22     }
23     cout << endl;
24     return 0;
25 }
```

```
> ./write_file
Enter NAME PROJ EXAM each line.
EOF to finish.
? Winner 99 100
? Loser 23 61
? ^D
> cat outfile
Name   Proj    Exam    Total
Winner 99      100     99.35
Loser  23      61      36.3
```

Creating a Sequential File

- Two arguments are passed to an ofstream object's constructor—the filename and the file-open mode (line 10).

```
10    ofstream outFile("outfile", ios::out);
```

- Existing files opened with mode ios::out are truncated—all data in the file is discarded.
- If the specified file does not yet exist, then the ofstream object creates the file, using that filename.

Creating a Sequential File (cont.)

- For an ofstream object, the file-open mode can be either ios::out to output data to a file or ios::app to append data to the end of a file.

Mode	Description
ios::app	***Append all output to the end of the file***
ios::ate	***Open a file for output and move to the end of the file (normally used to append data to a file). Data can be written anywhere in the file.***
ios::in	***Open a file for input.***
ios::out	***Open a file for output.***
ios::trunc	***Discard the file's contents (this also is the default action for ios::out).***
ios::binary	***Open a file for binary (i.e., nontext) input or output.***

Creating a Sequential File (cont.)

- An ofstream object can be created without opening a specific file—a file can be attached to the object later.
- For example, the statement
  - ofstream outFile;
- creates an ofstream object named outFile.
- The ofstream member function open opens a file and attaches it to an existing ofstream object as follows:
  - outFile.open(“outfile", ios::out);

Creating a Sequential File (cont.)

- The if statement in lines 11–14 uses the overloaded ios member function operator! to determine whether the open operation succeeded.
- Some possible errors are
  - attempting to open a nonexistent file for reading
  - attempting to open a file for reading or writing without permission
  - opening a file for writing when no disk space is available

```
11    if(!outFile) {
12        cerr << "Failed opening" << endl;
13        exit(1);
14    }
```

Creating a Sequential File (cont.)

- When end-of-file is encountered or bad data is entered, the while statement terminates.
- Ctrl-D in Unix and Ctrl-Z in Windows represent end-of-file.

```
18    while(cin >> name >> proj >> exam) {
19        outFile << name << "\t" << proj << "\t" <<
              exam << "\t"
20            << proj*0.65 + exam*0.35 << endl;
21        cout << "? ";
22    }
```

```
? ^D
```

Creating a Sequential File (cont.)

- Line 19 writes a set of data to the file outfile, using the stream insertion operator << and the outFile object associated with the file at the beginning of the program.

```
19          outFile << name << "\t" << proj << "\t" << exam << "\t"
20              << proj*0.65 + exam*0.35 << endl;
```

- Once the user enters the end-of-file indicator, main terminates.
- This implicitly invokes outFile's destructor, which closes the outfile file.
- You also can close the ofstream object explicitly, using member function close in the statement

read_file.cpp

```
 1 #include <iostream>
 2 #include <string>
 3 #include <fstream>
 4 #include <cstdlib>
 5 using namespace std;
 6 int main()
 7 {
 8     string name, headline;
 9     float proj, exam, total;
10     ifstream inFile("infile", ios::in);
11     if(!inFile) {
12         cerr << "Failed opening" << endl;
13         exit(1);
14     }
15     getline(inFile, headline);
16     cout << headline << endl;
17     while(inFile >> name >> proj >> exam >> total) {
18         cout << name << "\t" << proj << "\t"
19              << exam << "\t" << total << endl;
20     }
21     return 0;
22 }
```

```
> cat infile
Name    Proj    Exam    Total
Winner  99      100     99.35
Loser   23      61      36.3
> ./read_file
Name    Proj    Exam    Total
Winner  99      100     99.35
Loser   23      61      36.3
```

Reading Data from a Sequential File

- Creating an ifstream object opens a file for input.
- The ifstream constructor can receive the filename and the file open mode as arguments.
- Line 10 creates an ifstream object called inFile and associates it with the infile file.

```
10    ifstream inFile("infile", ios::in);
```

- Objects of class ifstream are opened for input by default.
- We could have used the statement

```
ifstream inFile("infile");
```

to open infile for input.

Reading Data from a Sequential File (cont.)

- Just as with an ofstream object, an ifstream object can be created without opening a specific file, because a file can be attached to it later.
- Each time line 17 executes, it reads another record from the file into the variables name, proj , exam and total.
- When the end of file has been reached, the ifstream destructor function closes the file and the program terminates.

```
17    while(inFile >> name >> proj >> exam >> total) {
18        cout << name << "\t" << proj << "\t"
19             << exam << "\t" << total << endl;
20    }
```