Doc No: X3J16/96-0126
WG21/N0944
Date: July 10, 1996
Project: Programming Language C++
Reply-To: Stephen D. Clamage
stephen.clamage@eng.sun.com

Effect of `openmode` in IOStreams

IOStreams issue 27-604 concerns the effect of `openmode` in IOStream constructors and the `[i|o]fstream open` functions. In the absence of any explict flags, all `istream`s default to `in` mode, and all `ostream`s default to `out` mode, which is obviously correct.

```
ifstream in1("foo"); // input text mode
```

At issue is what should happen when the programmer adds additional mode flags. It appears that the following result is typical:

```
ifstream in2("foo", ios::binary); // binary mode, "in" not set
```

The effect of opening a stream with neither `in` nor `out` set is not currently defined. In order to open reliably an ifstream in binary mode, the programmer must write

```
ifstream in3("foo", ios::in|ios::binary); // binary input mode
```

Requiring the `ios::in` flag seems wrong, since an `ifstream` can usefully be opened only in input mode anyway. It makes more sense for any `istream` to set by default the `in` flag, and for any `ostream` to set by default the `out` flag. Where current practice is to require an explicit `in` or `out` flag, this recommended change is backward-compatible. Existing code with explicit flags will still work. The results of code like that opening `in2` above were not portable. The recommended new rule is much easier to explain and should be less surprising for all programmers. I do not believe we should be concerned about the possibility of existing code like `in2` that was written deliberately expecting the stream not to be readable.

Bidirectional (input/output) `fstream`s are a different story. A function might be declared to take an `fstream` parameter, yet might operate properly when passed a stream open only for reading (or writing). Nevertheless, neither an `ifstream` nor an `ofstream` can be passed to a function expecting an `fstream` parameter. Thus, a programmer might want to open an `fstream` for just reading or just writing, in particular because attempting to open a read-only file for read/write may fail. We should not make such a thing impossible. Bidirectional `fstream`s should therefore have no default input or output mode; the programmer should be required to provide the mode explicitly, and that is current practice.

IOStreams issue 27-803 and 804 concern the effect of `openmode trunc` in IOStream constructors and the `[io]fstream open` functions. Opening a file for output but without the `app` flag should result in truncating the file. Thus, the expression `out|trunc` should be equivalent to just `out`.

Finally, Table 105 "File open modes" in section 27.8.1.3 [lib.filebuf.members] is incomplete. All possible flag combinations need to be discussed. This proposal presents new wording for parts of Chapter 27 covering all of these points.

Proposal 1:

Modify the description of `basic_filebuf::open` in section 27.8.1.3 by changing Table 105 "File open modes" as shown below. Combinations of flags not shown in the table (such as neither `in` nor `out`, or both `trunc` and `app`) are invalid, and the attempted `open` operation fails.

Table 1: File open modes

`ios_base` Flag combination					stdio equivalent
`binary`	`in`	`out`	`trunc`	`app`	
		+			"w"
		+		+	"a"
		+	+		"w"
	+				"r"
	+	+			"r+"
	+	+		+	"a+"
	+	+	+		"w+"
+		+			"wb"
+		+		+	"ab"
+		+	+		"wb"
+	+				"rb"
+	+	+			"r+b"
+	+	+		+	"a+b"
+	+	+	+		"w+b"

Proposal 2:

Modify the semantics of `istringstream` (and `ostringstream`) constructors to say that the `in` (`out`) flag is always set automatically. (That is, the flags appear as default parameter values, but they are set regardless of the actual `openmode` passed in.)

[begin draft text, changes underlined —

27.7.2.1 basic_istringstream constructors [lib.istringstream.cons]

```
explicit basic_istringstream(ios_base::openmode which =
ios_base::in);
```

Effects: Constructs an object of class `basic_istringstream<charT,traits>`, initializing the base class with `basic_istream(&` *sb*`)` and initializing *sb* with `basic_stringbuf<charT,traits>(`*which*`|ios_base::in)` (27.7.1.1).

```
explicit basic_istringstream(const basic_string<charT>& str,
ios_base::openmode which = ios_base::in);
```

Effects: Constructs an object of class `basic_istringstream<charT,traits>`, initializing the base class with `basic_istream(&` *sb*`)` and initializing *sb* with `basic_stringbuf<charT,traits>(`*str*`, `*which*` |ios_base::in)` (27.7.1.1).

27.7.2.4 basic_ostringstream constructors [lib.ostringstream.cons]

```
explicit basic_ostringstream(ios_base::openmode which =
ios_base::out);
```

Effects: Constructs an object of class `basic_ostringstream`, initializing the base class with `basic_ostream(&` *sb*`)` and initializing *sb* with `basic_stringbuf<charT,traits>(`*which*`|ios_base::out)` (27.7.1.1).

```
explicit basic_ostringstream(const basic_string<charT>& str,
ios_base::openmode which = ios_base::out);
```

Effects: Constructs an object of class `basic_ostringstream<charT,traits>`, initializing the base class with `basic_ostream(&` *sb*`)` and initializing *sb* with `basic_stringbuf<charT,traits>(`*str*`, `*which*`|ios_base::out)` (27.7.1.1).

— end draft text]

Proposal 3:

Modify the semantics of `ifstream` (and `ofstream`) constructors and `open` functions to say that the `in` (`out`) flag is always set automatically. (That is, the flags appear as default parameter values, but they are set regardless of the actual `openmode` passed in.)

[begin draft text, changes underlined —

27.8.1.6 basic_ifstream constructors [lib.ifstream.cons]

```
explicit basic_ifstream(const char* s, openmode mode = in);
```

Effects: Constructs an object of class `basic_ifstream`, initializing the base class with `basic_istream(&`*sb*`)` and initializing *sb* with `basic_filebuf<charT,traits>()` (_lib.istream.cons_, 27.8.1.2), then calls `rdbuf()->open(`*s*`, `*mode*`|in)`.

27.8.1.7 Member functions [lib.ifstream.members]

```
void open(const char* s, openmode mode = in);
```

Effects: Calls `rdbuf()->open(`*s*`, `*mode*`|in)`. If `is_open()` returns `false`, calls `set-state(failbit)` (which may throw `ios_base::failure` (27.4.4.3)).

27.8.1.9 basic_ofstream constructors [lib.ofstream.cons]

```
explicit basic_ofstream(const char* s, openmode mode = out);
```

Effects: Constructs an object of class `basic_ofstream<charT,traits>`, initializing the base class with `basic_ostream(&` *sb*`)` and initializing *sb* with `basic_filebuf<charT,traits>()`(27.6.2.2, 27.8.1.2), then calls `rdbuf()->open(`*s*, *mode*`|out)`.

27.8.1.10 Member functions [lib.ofstream.members]

```
void open(const char* s, openmode mode = out);
```

Effects: Calls `rdbuf()->open(`*s*`, `*mode*`|out)`. If `is_open()` is then `false`, calls `set-state(failbit)` (which may throw `ios_base::failure` (27.4.4.3)).

— end draft text]

Note: The description of [i|o]strstream in Appendix D does not need any revisions regarding in and out flags. The stream always operates in a mode consistent with its declaration, since there is no problem of coordinating with an external device.