Introduction of std::in, std::out, and std::err as substitutes for stream macros in Standard Library Modules

SM
March 3, 2024

Abstract

import std; does not expose macros including C streams. The corresponding header files should be included. To improve the usage of standard library modules, this paper proposes global variables to address this issue: std::in, std::out, and std::err.

1. Rationale

In C++23, import std; was introduced to facilitate the use of standard libraries. A user can access all library features by one line and it helps to reduce compilation time. However, modules don't expose macros intentionally, so the corresponding header files are needed to be included again. P2465R3 suggests that non-macro approaches should be utilized other than feature test macros. This paper proposes global variables, std::in, std::out, and std::err as substitutes for stream macros.

2. Motivation and Proposal

AS IS	TO BE

`#include <cstdio> // just for stderr`
`// import <cstdio>; // alternative`
`import std;`
 	
`int main()`
`{`
`  std::println(stderr, "error");`
`}`
	`import std;`
	
	`int main()`
	`{`
	`  std::println(std::err, "error");`
	`}`

Instead of providing additional APIs to hide stream macros, a fundamental approach is necessary for other use cases.

This paper proposes std::in, std::out, and std::err as substitutes for stdin, stdout, and stderr, respectively. They are global variables with the type of std::FILE*.

Proposal (succinct but new)	Alternative (maybe familiar)
std::in	std::stdin
std::out	std::stdout
std::err	std::stderr

3. Stream Macros in GCC and MSVC

C streams are already global variables in GCC and they are redefined as macros.

GCC

```
// filename: stdio.h

/* Standard streams.  */
extern FILE *stdin;             /* Standard input stream.  */
extern FILE *stdout;            /* Standard output stream.  */
extern FILE *stderr;            /* Standard error output stream.  */
/* C89/C99 say they're macros.  Make them happy.  */
#define stdin stdin
#define stdout stdout
#define stderr stderr
```

MSVC

```
// filename: corecrt_wstdio.h

_ACRTIMP_ALT FILE* __cdecl __acrt_iob_func(unsigned _Ix);

#define stdin  (__acrt_iob_func(0))
#define stdout (__acrt_iob_func(1))
#define stderr (__acrt_iob_func(2))
```

4. Possible Interface

Option A: std module

```
// ...
export module std;
namespace std
{
```

```
  export extern std::FILE* in;
  export extern std::FILE* out;
  export extern std::FILE* err;
}
// ...
```

Option B: a new header file and std module

In this option, a user can include the header file if necessary without importing standard library modules.

```
// filename: cstreams
// ...

namespace std
{
  extern std::FILE* in;
  extern std::FILE* out;
  extern std::FILE* err;
}
```

```
// std module file
module

// ...
#include <cstreams>

export module std;

// ...
namespace std
{
  export using std::in;
  export using std::out;
  export using std::err;
}
```

5. Conclusion

The support of macros including C streams is a missing block in import std;. By introducing substitute variables for stream macros, the convenience of import std; can be improved.

6. References

[P2465R3]

Standard Library Modules std and std.compat
https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2022/p2465r3.pdf

[P2654R0]
Macros And Standard Library Modules
import should suffice
https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2023/p2654r0.pdf

[P2883R0]
offsetof Should Be A Keyword In C++26
Supporting standard C++23 macros in module std
https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2023/p2883r0.pdf

[P2884R0]
assert Should Be A Keyword In C++26
Supporting standard C++23 macros in module std
https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2023/p2884r0.pdf

How to use standard library macros with std module in C++23
https://stackoverflow.com/questions/75041883/how-to-use-standard-library-macros-with-std-module-in-c23