# Sun C / Sun ANSI C Differences B

## Introduction

This appendix describes the differences between the previous (K&R) Sun C and Sun ANSI C, as implemented on SunOS 5.x.

"Sun C Incompatibilities with Sun ANSI C" on page B-620 describes previous Sun C features that are incompatible with Sun ANSI C. These differences should be addressed when porting source code written for the Sun C compiler to Sun ANSI C.

"Keywords" on page B-628 lists reserved words used by the ANSI C standard, Sun ANSI C, Sun C, and those defined by the Sun ANSI and Sun C preprocessors.

# Sun C Incompatibilities with Sun ANSI C

*Table B-1 Sun C Incompatibilities with Sun ANSI C (Sheet 1 of 9)*

| Topic | Sun C | Sun ANSI C |
|---|---|---|
| `envp` argument to `main()` | Allows `envp` as third argument to `main()`. | Allows this third argument; however, this usage is not strictly conforming to the ANSI C standard. |
| keywords | Treats the identifiers `const`, `volatile`, and `signed` as ordinary identifiers. | `const`, `volatile`, and `signed` are keywords. |
| `extern` and `static` functions declarations inside a block | Sun C promotes these function declarations to file scope. | The ANSI standard does not guarantee that block scope function declarations are promoted to file scope. |
| identifiers | Allows dollar signs (`$`) in identifiers. | `$`Not allowed. |
| `long float` types | Accepts `long float` declarations and treats these as `double`(s). | Does not accept these declarations. |
| multi-byte character constants | `int mc = 'abcd';`<br>yields<br>`abcd` | `int mc = 'abcd';`<br>yields<br>`dcba` |
| integer constants | Accepts 8 or 9 in octal escape sequences. | Does not accept 8 or 9 in octal escape sequences. |
| assignment operators | Treats the following operator pairs as two tokens, and as a consequence, permits whitespace between them:<br>`*=, /=, %=, +=, -=, <<=, >>=, &=, ^=, \|=` | Treats them as single tokens, and therefore disallows whitespace in-between. |
| `unsigned` preserving semantics for expressions | Supports *unsigned* preserving. That is, `unsigned char/short`(s) are converted into `unsigned int`(s). | Supports *value-preserving*. That is, `unsigned char/short`(s) are converted into `int`(s). |

*Table B-1 Sun C Incompatibilities with Sun ANSI C (Sheet 2 of 9)*

| Topic | Sun C | Sun ANSI C |
|---|---|---|
| single / double precision calculations | Promotes the operands of floating point expressions to `double`.<br><br>In Sun C, functions which are declared to return `floats` always promote their return values to `doubles`. | Allows operations on `floats` to be performed in single precision calculations.<br><br>Allows `float` return types for these functions. |
| name spaces of `struct/union` members | Allows `struct`, `union`, and arithmetic types using member selection operators (`'.'`, `'->'`) to work on members of other `struct`(s) or `unions`. | Requires that every unique `struct/union` have its own unique namespace. |
| a `cast` as an `lvalue` | Supports casts as `lvalue`(s). For example:<br><br>`(char *)ip = &char;` | Does not support this feature. |
| implied `int` declarations | Supports declarations without an explicit type specifier. A declaration such as `num;` is treated as implied `int`. For example:<br><br>`num;`<br>`/* num implied as an int */`<br><br>`int num2;`<br>`/* num2 explicitly declared an int */` | The `num;` declaration (without the explicit type specifier `int`) is not supported and generates a syntax error. |
| empty declarations | Allows empty declarations:<br><br>`int;` | Except for *tags*, disallows empty declarations. |
| type specifiers on type definitions | Allows type specifiers such as `unsigned, short, long` on `typedef` declarations. For example:<br><br>`typedef short small;`<br>`unsigned small x;` | Does not allow type specifiers to modify `typedef` declarations. |

*Table B-1 Sun C Incompatibilities with Sun ANSI C (Sheet 3 of 9)*

| Topic | Sun C | Sun ANSI C |
|---|---|---|
| types allowed on bitfields | Allows bitfields of all integral types, including unnamed bit fields.<br><br>**NOTE:** The ABI requires support of unnamed bit fields and the other integral types, as provided in Sun C. | Supports bitfields only of the type `int`, `unsigned int` and `signed int`. Other types are undefined. |
| treatment of tags in incomplete declarations | Because Sun C ignores the incomplete type declaration, in the below example `f1` refers to the outer `struct`:<br><br>`struct x { .   .   . } s1;`<br>`{`<br>`      struct x;`<br>`      struct y {struct x f1; }`<br>`s2;`<br>`      struct x { .   .   . };`<br>`}` | In an ANSI conforming implementation, an incomplete `struct` or `union` type specifier hides an enclosing declaration with the same tag. |
| mis-match on `struct/union/enum` declarations | Allows a mismatch on the `struct/enum/union` type of a tag in nested `struct/union` declarations. In the example below the second declaration is treated as a `struct`:<br><br>`struct x { .   .   . } s1;`<br>`{`<br>`      union x s2;`<br>`      .   .   .`<br>`}` | Will treat the inner declaration as a new declaration, hiding the outer tag. |
| labels in expressions | Treats labels as `(void *) lvalues`. | Does not allow labels in expressions. |
| `switch` condition type | Allows `float`(s) and `double`(s) by converting them to `int`(s). | Evaluates only integral types (`int`, `char`, and enumerated) for the switch condition type. |
| syntax of conditional inclusion directives | The Sun C preprocessor ignores trailing tokens after an `#else` or `#endif` directive. | Disallows such constructs. |

*Table B-1 Sun C Incompatibilities with Sun ANSI C (Sheet 4 of 9)*

| Topic | Sun C | Sun ANSI C |
|---|---|---|
| token pasting and the `##` preprocessor operator | Doesn't recognize the `##` operator. In Sun C, token pasting is accomplished by placing a comment between the two tokens being pasted:<br><br>`#define PASTE(A,B)A/*any comment*/B` | Defines `##` as the preprocessor operator that performs token pasting, as shown in the example below.<br><br>`#define PASTE(A,B)A##B`<br><br>Furthermore, the Sun ANSI C preprocessor doesn't recognize the Sun C method. Instead, it treats the comment between the two tokens as whitespace. |
| preprocessor rescanning | The Sun C preprocessor will recursively substitute:<br><br>`#define F(X) X(arg)`<br>`F(F)`<br><br>yields<br><br>`arg(arg)` | A macro will not be replaced if it is found in the replacement list during the rescan:<br><br>`#define F(X) X(arg)`<br>`F(F)`<br><br>yields:<br><br>`F ( arg )` |
| `typedef` names in formal parameter lists | Allows one to use `typedef` names as formal parameter names in a function declaration, and would, in effect, "hide" the `typedef` declaration. | Disallows the use of an identifier declared as a `typedef` name as a formal parameter. |
| implementation specific initializations of aggregates | Uses a bottom-up algorithm when parsing and processing partially elided initializers within braces:<br><br>`struct { int a[3]; int b; } \`<br>`w[] = { {1}, 2};`<br><br>yields<br><br>`sizeof(w) = 16`<br>`w[0].a = 1, 0, 0`<br>`w[0].b = 2` | Uses a top-down parsing algorithm. For example:<br><br>`struct { int a[3]; int b; } \`<br>`w[] = { {1}, 2};`<br><br>yields<br><br>`sizeof(w) = 32`<br>`w[0].a = 1, 0, 0`<br>`w[0].b = 0`<br>`w[1].a = 2, 0, 0`<br>`w[1].b = 0` |

*Table B-1 Sun C Incompatibilities with Sun ANSI C (Sheet 5 of 9)*

| Topic | Sun C | Sun ANSI C |
|---|---|---|
| comments spanning `include` files | Allows comments which start in an `#include` file to be terminated by the file that includes the first file. | Comments are replaced by a whitespace character in the translation phase of the compilation, which occurs before the `#include` directive is processed. |
| formal parameter substitution within a character constant | Substitutes characters within a character constant when it matches the replacement list macro:<br><br>`#define charize(c) 'c'`<br>`charize(Z)`<br><br>yields<br><br>`'Z'` | The character is not replaced:<br><br>`#define charize(c) 'c'`<br>`charize(Z)`<br><br>yields<br><br>`'c'` |
| formal parameter substitution within a string constant | The Sun C preprocessor will substitute a formal parameter when enclosed within a string constant:<br><br>`#define stringize(str) 'str'`<br>`stringize(foo)`<br><br>yields<br><br>`"foo"` | For string substitution in Sun ANSI C, the `#` preprocessor operator should be used.<br><br>`#define stringize(str) 'str'`<br>`stringize(foo)`<br><br>yields<br><br>`"str"` |

*Table B-1 Sun C Incompatibilities with Sun ANSI C (Sheet 6 of 9)*

| Topic | Sun C | Sun ANSI C |
|---|---|---|
| preprocessor built into the compiler "front-end" | Compiler calls `cpp(1)`. | Preprocessor (`cpp`) is built directly into `acomp`, so `cpp` is not directly involved (except in `-Xs`) mode.<br><br>Following are the components used in compiling:<br><br>Sun C  Sun ANSI C<br><br>`cpp`  `cpp (-Xs` mode only`)`<br>`ccom`  `acomp`<br>  `basicblk` (only with `-ql)`<br>`iropt`  `iropt`<br>`cg`  `cg`<br>`inline`  `inline`<br>`as`  `fbe`<br>`ld`  `ld`<br><br>Note: `iropt` and `cg` will be invoked only with the following options:<br><br>`-O -xO2 -xO3 -xO4 -xa -fast`<br><br>`inline` will be invoked only if an inline template file (*file*`.il`) is provided |
| line concatenation with backslash | Does not recognize the backslash character in this context. | Requires that a new-line character immediately preceded by a backslash character be spliced together. |
| trigraphs in string literals | Does not support this ANSI C feature. | |
| `asm` keyword | `asm` is a keyword. | `asm` is treated as an ordinary identifier. |

*Table B-1 Sun C Incompatibilities with Sun ANSI C (Sheet 7 of 9)*

| Topic | Sun C | Sun ANSI C |
|---|---|---|
| linkage of identifiers | Does not treat uninitialized `static` declarations as tentative declarations. As a consequence, the second declaration (in the example below) will generate a 'redeclaration' error:<br><br>`static int i = 1;`<br><br>`static int i;` | Treats uninitialized `static` declarations as tentative declarations. |
| name spaces | Distinguishes only three: `struct/union/enum` tags, members of `struct/union/enum`, and everything else. | Recognizes four distinct name spaces: label names, tags (the names that follow the keywords `struct`, `union` or `enum`), members (of `struct/union/enum`), and ordinary identifiers. |
| `long double` type | Not supported. | Allows `long double` type declaration. |
| floating point constants | The floating point suffixes, `f`, `l`, `F`, and `L`, are not supported. | |
| unsuffixed integer constants can have different types | The integer constant suffixes `u` and `U` are not supported. | |
| wide character constants | Does not accept the ANSI C syntax for wide character constants, as shown below:<br><br>`wchar_t wc = L'x';` | Supports this syntax. |
| `'\a'` and `'\x'` | Treats them as the characters 'a' and 'x'. | Treats `'\a'` and `'\x'` as special escape sequences. |
| Concatenation of string literals | The ANSI C concatenation of adjacent string literals is not supported in Sun C. | |

*Table B-1 Sun C Incompatibilities with Sun ANSI C (Sheet 8 of 9)*

| Topic | Sun C | Sun ANSI C |
|---|---|---|
| Wide-character string literal syntax | The ANSI C wide-character-string literal syntax shown in the example below is not supported:<br><br>`wchar_t *ws = L"hello";` | Supports this syntax. |
| pointers: `void *` *vs.* `char *` | The ANSI C `void*` feature is supported. | |
| unary plus operator | This ANSI C feature is not supported. | |
| function prototypes - ellipses | Not supported. | ANSI C defines the use of ellipses "..." to denote a variable argument parameter list. |
| type definitions | Disallows `typedefs` to be redeclared in an inner block by another declaration with the same type name. | Allows `typedefs` to be redeclared in an inner block by another declaration with the same type name. |
| initialization of `extern` variables | Does not support the initialization of variables explicitly declared as `extern`. | ANSI C treats the initialization of variables explicitly declared as `extern`, as definitions. |
| initialization of aggregates | Does not support the ANSI C initialization of unions or automatic structures. | |
| prototypes | This ANSI C feature (of prototyping) is not supported. | |
| syntax of preprocessing directive | Recognizes only those directives with a '`#`' in the first column. | ANSI C allows leading whitespace characters before a '`#`' directive. |
| the `#` preprocessor operator | The ANSI C `#` preprocessor operator is not supported. | |
| `#error` directive | This ANSI C feature is not supported by the preprocessor. | |

*Table B-1 Sun C Incompatibilities with Sun ANSI C (Sheet 9 of 9)*

| Topic | Sun C | Sun ANSI C |
|---|---|---|
| preprocessor directives | Supports two pragmas, `unknown_control_flow` and `makes_regs_inconsistent` along with the `#ident` directive. Moreover, the preprocessor will issue warnings when it finds unrecognized `pragmas`. | ANSI C does not specify its behavior for unrecognized `pragmas`.<br><br>(See the *C 2.0.1 Programmer's Guide* for recognized `pragmas`.) |
| predefined macro names | The ANSI C-defined macro names shown below are not defined in Sun C.<br><br>`__STDC__`<br>`__TIME__`<br>`__DATE__`<br>`__LINE__` | Defined. |

# Keywords

The three tables below list the keywords for the ANSI C Standard, the Sun ANSI C compiler, the Sun C compiler, and the preprocessor keywords that are therefore reserved words when using either compiler.

The first table lists the keywords defined by the ANSI C standard.

*Table B-2 ANSI C Standard Keywords*

| | | | |
|---|---|---|---|
| `auto` | `break` | `case` | `char` |
| `const` | `continue` | `default` | `do` |
| `double` | `else` | `enum` | `extern` |
| `float` | `for` | `goto` | `if` |
| `int` | `long` | `register` | `return` |
| `short` | `signed` | `sizeof` | `static` |
| `struct` | `switch` | `typedef` | `union` |
| `unsigned` | `void` | `volatile` | `while` |

Sun ANSI defines one additional keyword, `asm`. However, `asm` is not supported in `-Xc` mode.

Keywords in Sun C are listed below.

*Table B-3 Sun C (K&R) Keywords*

| | | | |
|---|---|---|---|
| asm | auto | break | case |
| char | continue | default | do |
| double | else | enum | extern |
| float | for | fortran | goto |
| if | int | long | register |
| return | short | sizeof | static |
| struct | switch | typedef | union |
| unsigned | void | while | |

The following are predefined macros defined by the preprocessor for both Sun C and Sun ANSI C. (The `-Xc` mode of the Sun ANSI C compiler does not define these names.)

*Table B-4 Preprocessor-Defined Keywords*

| | |
|---|---|
| sparc | sun |
| sun4 | unix |

The `-sys5` option predefines `__SYS5__`.