What's New

Sun™ Studio 10

Sun Microsystems, Inc.
www.sun.com

Part No. 819-0488-10
January 2005, Revision A

Submit comments about this document at: http://www.sun.com/hwdocs/feedback

Contents

Before You Begin 5

Typographic Conventions 5

Shell Prompts 6

Supported Platforms 6

Accessing Sun Studio Software and Man Pages 7

Accessing Sun Studio Documentation 10

Accessing Related Solaris Documentation 13

Resources for Developers 13

Contacting Sun Technical Support 14

Sending Your Comments 14

1. Sun Studio 10 New Features and Enhancements 15

C Compiler 16

C++ Compiler 17

Examples of Template-Template Parameters 18

Nested Class Access Rules 19

Fortran Compiler 20

Binary File Sharing Between Big-endian and Little-endian Platforms 20

Command-line Debugger `dbx` 22

OpenMP API 23

Interval Arithmetic 24

Sun Performance Library 25

`dmake` 26

Performance Analysis Tools 27

Integrated Development Environment (IDE) 29

Documentation 29

2. Sun Studio 9 New Features and Enhancements 31

C Compiler 32

C++ Compiler 38

Fortran Compiler 43

Command-Line Debugger `dbx` 48

Interval Arithmetic 48

Sun Performance Library 48

`dmake` 49

Performance Analysis Tools 50

Integrated Development Environment (IDE) 53

Documentation 53

Before You Begin

The *What's New* describes the new features of the Sun Studio 10 software release and the Sun™ Studio 9 software release, which include new features in the C, C++, and Fortran compilers, libraries, and tools.

Typographic Conventions

TABLE P-1 Typeface Conventions

Typeface	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`

`AaBbCc123`	What you type, when contrasted with on-screen computer output	`%` **`su`**
		`Password:`

AaBbCc123	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this.
AaBbCc123	Command-line placeholder text; replace with a real name or value	To delete a file, type **`rm`** *filename*.

TABLE P-2 Code Conventions

Code Symbol	Meaning	Notation	Code Example
[ ]	Brackets contain arguments that are optional.	`O[`*n*`]`	`O4, O`
{ }	Braces contain a set of choices for a required option.	`d{y|n}`	`dy`
|	The "pipe" or "bar" symbol separates arguments, only one of which may be chosen.	`B{dynamic|static}`	`Bstatic`
:	The colon, like the comma, is sometimes used to separate arguments.	`R`*dir*`[:`*dir*`]`	`R/local/libs:/U/a`
…	The ellipsis indicates omission in a series.	`xinline=`*f1*`[,`*…fn*`]`	`xinline=alpha,dos`

Shell Prompts

Shell	Prompt
C shell	*machine-name*`%`
C shell superuser	*machine-name*`#`
Bourne shell and Korn shell	`$`
Superuser for Bourne shell and Korn shell	`#`

Supported Platforms

This Sun Studio release supports systems that use the SPARC® and x86 families of processor architectures: UltraSPARC®, SPARC64, AMD64, Pentium, and Xeon EM64T. The supported systems for the version of the Solaris Operating System you are running are available in the hardware compatibility lists at http://www.sun.com/bigadmin/hcl. These documents cite any implementation differences between the platform types.

In this document, the term “x86” refers to 64-bit and 32-bit systems manufactured using processors compatible with the AMD64 or Intel Xeon/Pentium product families. For supported systems, see the hardware compatibility lists.

Accessing Sun Studio Software and Man Pages

The Sun Studio software and its man pages are not installed into the standard /usr/bin/ and /usr/share/man directories. To access the software, you must have your PATH environment variable set correctly (see “Accessing the Software” on page 7). To access the man pages, you must have the your MANPATH environment variable set correctly (see “Accessing the Man Pages” on page 8.).

For more information about the PATH variable, see the csh(1), sh(1), ksh(1), and bash(1) man pages. For more information about the MANPATH variable, see the man(1) man page. For more information about setting your PATH variable and MANPATH variables to access this release, see the installation guide or your system administrator.

Note – The information in this section assumes that your Sun Studio software is installed in the /opt directory on Solaris platforms and in the /opt/sun directory on Linux platforms. If your software is not installed in the default directory, ask your system administrator for the equivalent path on your system.

Accessing the Software

Use the steps below to determine whether you need to change your PATH variable to access the software.

To Determine Whether You Need to Set Your `PATH` Environment Variable

1. **Display the current value of the `PATH` variable by typing the following at a command prompt.**

```
% echo $PATH
```

2. **On Solaris platforms, review the output to find a string of paths that contain `/opt/SUNWspro/bin`. On Linux platforms, review the output to find a string of paths that contain `/opt/sun/sunstudio10/bin`.**

   If you find the path, your `PATH` variable is already set to access the compilers and tools. If you do not find the path, set your `PATH` environment variable by following the instructions in the next procedure.

To Set Your `PATH` Environment Variable to Enable Access to the Compilers and Tools

- **On Solaris platforms, add the following to your `PATH` environment variable. If you have Forte Developer software, Sun ONE Studio software, or another release of Sun Studio software installed, add the following path before the paths to those installations.**

  `/opt/SUNWspro/bin`

- **On Linux platforms, add the following to your `PATH` environment variable.**

  `/opt/sun/sunstudio10/bin`

Accessing the Man Pages

Use the following steps to determine whether you need to change your `MANPATH` variable to access the man pages.

To Determine Whether You Need to Set Your `MANPATH` Environment Variable

1. **Request the `dbx` man page by typing the following at a command prompt.**

```
% man dbx
```

2. **Review the output, if any.**

   If the `dbx`(1) man page cannot be found or if the man page displayed is not for the current version of the software installed, follow the instructions in the next procedure for setting your `MANPATH` environment variable.

To Set Your `MANPATH` Environment Variable to Enable Access to the Man Pages

- **On Solaris platforms, add the following to your `MANPATH` environment variable.**

  ```
  /opt/SUNWspro/
  ```

- **On Linux platforms, add the following to your `MANPATH` environment variable.**

  ```
  /opt/sun/sunstudio10
  ```

Accessing the Integrated Development Environment

The Sun Studio 9 integrated development environment (IDE) provides modules for creating, editing, building, debugging, and analyzing the performance of a C, C++, or Fortran application.

The command to start the IDE is `sunstudio`. For details on this command, see the `sunstudio`(1) man page.

The correct operation of the IDE depends on the IDE being able to find the core platform. The `sunstudio` command looks for the core platform in two locations:

- The command looks first in the default installation directory, `/opt/netbeans/3.5V` on Solaris platforms and `/opt/sun/netbeans/3.5V` on Linux platforms.
- If the command does not find the core platform in the default directory, it assumes that the directory that contains the IDE and the directory that contains the core platform are both installed in or mounted to the same location. For example, on Solaris platforms, if the path to the directory that contains the IDE is `/foo/SUNWspro`, the command looks for the core platform in `/foo/netbeans/3.5V`. On Linux platforms, if the path to the directory that contains the IDE is `/foo/sunstudio10`, the command looks for the core platform in `/foo/netbeans/3.5V`.

If the core platform is not installed or mounted to either of the locations where the `sunstudio` command looks for it, then each user on a client system must set the environment variable `SPRO_NETBEANS_HOME` to the location where the core platform is installed or mounted (/*installation_directory*/`netbeans/3.5V`).

On Solaris platforms, each user of the IDE also must add /*installation_directory*/`SUNWspro/bin` to their `$PATH` in front of the path to any other release of Forte Developer software, Sun ONE Studio software, or Sun Studio software. On Linux platforms, each user of the IDE also must add /*installation_directory*/`sunstudio10/bin` to their `$PATH` in front of the path to any other release of Sun Studio software.

The path /*installation_directory*/`netbeans/3.5V/bin` should not be added to the user's `$PATH`.

Accessing Sun Studio Documentation

You can access the documentation at the following locations:

- The documentation is available from the documentation index that is installed with the software on your local system or network at `file:/opt/SUNWspro/docs/index.html` on Solaris platforms and at `file:/opt/sun/sunstudio10/docs/index.html` on Linux platforms.

  If your software is not installed in the `/opt` directory on a Solaris platform or the `/opt/sun` directory on a Linux platform, ask your system administrator for the equivalent path on your system.
- Most manuals are available from the `docs.sun.com`sm web site. The following titles are available through your installed software only:
  - *Standard C++ Library Class Reference*
  - *Standard C++ Library User's Guide*
  - *Tools.h++ Class Library Reference*
  - *Tools.h++ User's Guide*
- The release notes are available from the `docs.sun.com` web site.
- Online help for all components of the IDE is available through the Help menu, as well as through Help buttons on many windows and dialogs, in the IDE.

The `docs.sun.com` web site (`http://docs.sun.com`) enables you to read, print, and buy Sun Microsystems manuals through the Internet. If you cannot find a manual, see the documentation index that is installed with the software on your local system or network.

Note – Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with use of or reliance on any such content, goods, or services available on or through any such sites or resources.

Documentation in Accessible Formats

The documentation is provided in accessible formats that are readable by assistive technologies for users with disabilities. You can find accessible versions of documentation as described in the following table. If your software is not installed in the `/opt` directory, ask your system administrator for the equivalent path on your system.

Type of Documentation	Format and Location of Accessible Version
Manuals (except third-party manuals)	HTML at `http://docs.sun.com`
Third-party manuals: • *Standard C++ Library Class Reference* • *Standard C++ Library User's Guide* • *Tools.h++ Class Library Reference* • *Tools.h++ User's Guide*	HTML in the installed software through the documentation index at `file:/opt/SUNWspro/docs/index.html`
Readmes and man pages	HTML in the installed software through the documentation index at `file:/opt/SUNWspro/docs/index.html` on Solaris platforms, and at `file:/opt/sun/sunstudio10/docs/index.html` on Linux platforms,
Online help	HTML available through the Help menu in the IDE
Release notes	HTML at `http://docs.sun.com`

Related Documentation

The following table describes related documentation that is available at `file:/opt/SUNWspro/docs/index.html` and `http://docs.sun.com`. If your software is not installed in the `/opt` directory, ask your system administrator for the equivalent path on your system.

Document Title	Description
Dubugging a Program With dbx	Describes how to use the `dbx` command-line debugger to debug programs written in the C, C++, Fortran, and Java™ programming languages.
Fortran Programming Guide	Describes how to write effective Fortran code on Solaris™ environments; input/output, libraries, performance, debugging, and parallel processing.
Fortran Library Reference	Details the Fortran library and intrinsic routines
Fortran User's Guide	Describes the compile-time environment and command-line options for the `f95` compiler. Also includes guidelines for migrating legacy `f77` programs to `f95`.
C User's Guide	Describes the compile-time environment and command-line options for the `cc` compiler.
C++ User's Guide	Describes the compile-time environment and command-line options for the `CC` compiler.
Performance Analyzer	Describes how to use the Collector and Performance Analyzer to perform statistical profiling of a wide range of performance data and tracing of various system calls, and relate the data to program structure at the function, source line and instruction level.

Accessing Related Solaris Documentation

The following table describes related documentation that is available through the `docs.sun.com` web site.

Document Collection	Document Title	Description
Solaris Reference Manual Collection	See the titles of man page sections.	Provides information about the Solaris™ operating environment.
Solaris Software Developer Collection	*Linker and Libraries Guide*	Describes the operations of the Solaris™ link-editor and runtime linker.
Solaris Software Developer Collection	*Multithreaded Programming Guide*	Covers the POSIX® and Solaris™ threads APIs, programming with synchronization objects, compiling multithreaded programs, and finding tools for multithreaded programs.

Resources for Developers

Visit `http://developers.sun.com/prodtech/cc` to find these frequently updated resources:

- Articles on programming techniques and best practices
- A knowledge base of short programming tips
- Documentation of compilers and tools components, as well as corrections to the documentation that is installed with your software
- Information on support levels
- User forums
- Downloadable code samples
- New technology previews

You can find additional resources for developers at `http://developers.sun.com`.

Contacting Sun Technical Support

If you have technical questions about this product that are not answered in this document, go to:

`http://www.sun.com/service/contacting`

Sending Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. Submit your comments to Sun at this URL

`http://www.sun.com/hwdocs/feedback`

Please include the part number (819-0488-10) of your document.

CHAPTER 1

Sun Studio 10 New Features and Enhancements

Sun™ Studio 10 replaces the Sun™ Studio 9. New features in the Sun Studio 10 release include updates to the following compilers, libraries, and tools:

- C Compiler
- C++ Compiler
- Fortran Compiler
- Sun Performance Library
- Distributed `make` utility, `dmake`
- `dbx` Command-Line Debugger
- Performance Analysis Tools
- Integrated Development Environment (IDE)
- Documentation

In most sections, there is a table that lists the new features of that component. The table has two columns, where the left-hand column provides a short description of the feature, and the right-hand column has a longer description.

Note – To find the Sun Studio 10 documentation described in this chapter, see the documentation index installed with the product software at `/opt/SUNWspro/docs/index.html`. If your software is not installed in the `/opt` directory, contact your system administrator for the equivalent path on your system or network.

C Compiler

TABLE 1-1 C Compiler New Features

Feature	Description
OpenMP parallel programming API	The API is now enabled on 32-bit and 64-bit x86 based systems running the Solaris OS.
New `-xarch` option	`-xarch=amd64` specifies compilation for the 64-bit AMD instruction set. The C compiler now predefines `__amd64` and `__x86_64` when you specify `-xarch=amd64`.
New `-xtarget` option	`-xtarget=opteron` specifies the `-xarch`, `-xchip`, and `-xcache` settings for 32-bit AMD compilation.
New `-xregs` flag on x86 based systems	A new x86-only flag for the `-xregs` option, `-xregs=[no%]frameptr`, lets you use the frame-pointer register as an unallocated callee-saves register to increase the run-time performance of applications.
New `-Xarch=amd64` option for `lint`	The C utility `lint` now accepts a new option `-Xarch=amd64`. See the `lint`(1) man page for more information.
`-xarch=generic64` on x86 based systems	The existing `-xarch=generic64` option now supports the x86 platform in addition to the traditional SPARC platform.
`-xipo` on x86 based systems	The `-xipo` option is now available on x86 based systems.

Note – You must specify `-xarch=amd64` to the right of `-fast` and `-xtarget` on the command line to generate 64-bit code. For example, specify `cc -fast -xarch=amd64` or `cc -xtarget=opteron -xarch=amd64`. The new `-xtarget=opteron` option does not automatically generate 64-bit code. It expands to `-xarch=sse2`, `-xchip=opteron`, and `-xcache=64/64/2:1024/64/16`, which results in 32-bit code. The `-fast` option also results in 32-bit code because it is a macro which also defines `-xtarget=native`.

C++ Compiler

TABLE 1-2 C++ Compiler New Features

Feature	Description
OpenMP parallel programming API	The API is now enabled on 32-bit and 64-bit x86 based systems running the Solaris OS.
New `-xarch` option	`-xarch=amd64` specifies compilation for the 64-bit AMD instruction set. The C++ compiler now predefines `__amd64` and `__x86_64` when you specify `-xarch=amd64`.
New `-xtarget` option	`-xtarget=opteron` specifies the `-xarch`, `-xchip`, and `-xcache` settings for 32-bit AMD compilation.
New `-xregs` flag on x86 based systems	A new x86-only flag for the `-xregs` option, `-xregs=[no%]frameptr`, lets you use the frame-pointer register as an unallocated callee-saves register to increase the run-time performance of applications.
`-xarch=generic64` on x86 based systems	The existing `-xarch=generic64` option now supports the x86 platform in addition to the traditional SPARC platform.
`-xipo` on x86 based systems	The `-xipo` option is now available on x86 based systems.
Template-template parameters	You can specify a template definition with parameters that are themselves templates, rather than types or values. Recall that a template instantiated on a type is itself a type. For examples, see "Examples of Template-Template Parameters" on page 18.
Access rules for nested classes	In default mode, the C++ compiler in this release allows nested classes the same access to member classes that member functions have. For more information, see "Nested Class Access Rules" on page 19.

Note – You must specify `-xarch=amd64` to the right of `-fast` and `-xtarget` on the command line to generate 64-bit code. For example, specify `CC -fast -xarch=amd64` or `CC-xtarget=opteron -xarch=amd64.` The new `-xtarget=opteron` option does not automatically generate 64-bit code. It expands to `-xarch=sse2`, `-xchip=opteron`, and `-xcache=64/64/2:1024/64/16`, which results in 32-bit code. The `-fast` option also results in 32-bit code because it is a macro which also defines `-xtarget=native.`

Examples of Template-Template Parameters

The section provides two code examples, one that does not use template-template parameters and one that does.

This example does not use template-template parameters because MyClass<int> is a type.

```
template<typename T> class MyClass { ... };
std::list< MyClass<int> > x;
```

In this example, class template C has a parameter that is a class template, and object x is an instance of C using class template A as its argument. Member y of c has type A<int>.

```
// ordinary class template
template<typename T> class A {
    T x;
};
// class template having a template parameter
template < template<typename U> class V > class C {
    V<int> y;
// instantiate C on template
C<A> x;
```

Nested Class Access Rules

The C++ compiler, in default standard mode, now allows nested classes to access private members of the enclosing class.

The C++ standard says that nested classes have no special access to members of the enclosing class. However, most people feel this restriction is not justified because member functions have access to private members, so member classes should too. In the following example, function `foo` tries to access a private member of class `outer`. According to the C++ standard, the function has no access unless it is declared a friend function:

```
class outer {
    int i; // private in outer
    class inner {
        int foo(outer* p) {
            return p->i; // invalid
        }
    };
};
```

The C++ Committee is in the process of adopting a change to the access rules giving the same access to member classes that member functions have. Many compilers have implemented this rule in anticipation of the changed language rule.

To restore the old compiler behavior, disallowing the access, use the compiler option `-features=no%nestedaccess`. The default is `-features=nestedaccess`.

Fortran Compiler

TABLE 1-3 Fortran Compiler New Features

Feature	Description
OpenMP parallel programming API	The API is now enabled on 32-bit and 64-bit x86 based systems running the Solaris OS.
New `-xarch` option	`-xarch=amd64` specifies compilation for the 64-bit AMD instruction set. The Fortran compiler now predefines `__amd64` and `__x86_64` when you specify `-xarch=amd64`.
New `-xtarget` option	`-xtarget=opteron` specifies the `-xarch`, `-xchip`, and `-xcache` settings for 32-bit AMD compilation.
`-xarch=generic64` on x86 based systems	The existing `-xarch=generic64` option now supports the x86 platform in addition to the traditional SPARC platform.
`-xipo` on x86 based systems	The `-xipo` option is now available on x86 based systems.
Binary (unformatted) file sharing between big-endian and little-endian platforms	A new compiler flag `-xfilebyteorder` provides support of binary I/O files when moving between SPARC based systems and x86 based systems. The flag identifies the byte-order and byte-alignment of unformatted I/O files. For more information, see "Binary File Sharing Between Big-endian and Little-endian Platforms" on page 20

Note – You must specify `-xarch=amd64` to the right of `-fast` and `-xtarget` on the command line to generate 64-bit code. For example, specify `f95 -fast -xarch=amd64` or `f95 -xtarget=opteron -xarch=amd64`. The new `-xtarget=opteron` option does not automatically generate 64-bit code. It expands to `-xarch=sse2`, `-xchip=opteron`, and `-xcache=64/64/2:1024/64/16`, which results in 32-bit code. The `-fast` option also results in 32-bit code because it is a macro which also defines `-xtarget=native`.

Binary File Sharing Between Big-endian and Little-endian Platforms

A new compiler flag `-xfilebyteorder` provides support of binary I/O files when moving between SPARC based systems and x86 based systems. The flag identifies the byte-order and byte-alignment of unformatted I/O files.

The syntax of the flag is:

-xfilebyteorder=
{[little*max_align*:%all,*unitno*,*filename*}],[big*max_align*:{%all,*unitno*,*filename*}]
,[native:{%all,*unitno*,*filename*}]}:

max_align	Maximum byte alignment for the target platform. Values are 1, 2, 4, 8, and 16. The alignment applies to Fortran VAX structures and Fortran 95 derived types which use platform-dependent alignments for compatibility with C structures.
little*max_align*:{%all,*unitno*,*filename*}	List of files or unit numbers that are "little-endian" files used on a system where the maximum byte alignment is *max_align*. For example, little4 describes a 32-bit x86 file while little16 describes a 64-bit x86 file.
big*max_align*:{%all,*unitno*,*filename*}	List of files or unit numbers that are "big-endian" files used on a system where the maximum byte alignment is *max_align*.
native:{%all,*unitno*,*filename*}	List of files or unit numbers that are native files of the same byte order and alignment used by the compiling processor system
%all	Specifies all files and logical units except those opened as "SCRATCH" or named explicitly in this option. Can be used to describe default files not explicitly listed by this flag. %all can only appear once.
unitno	Fortran logical unit number opened by the program.
filename	Fortran file name opened by the program.

This option does not apply to files opened with STATUS=scratch. I/O operations done on these files are always with the byte-order and byte-alignment of the native processor.

The first default, when -xfilebyteorder is not specified on the compiler command line, is -xfilebyteorder=native:%all. The option must be specified with at least one argument. That is, at least one of the little:, big:, or native: parameters must be present.

Files not explicitly declared by this flag are assumed to be native files. For example, compiling with `-xfilebyteorder=little4:zfile.out` declares `zfile.out` to be a little-endian 32-bit x86 file with a 4-byte maximum data alignment rule, and all other files are native files.

When the byte-order specified for a file is the same as the native processor but a different alignment is specified, the appropriate padding will be used even though no byte swapping is done. For example, this would be the case when compiling with `-xarch=amd64` for 64-bit x86 and `-xfilebyteorder=little4:`*filename* is specified.

The declared types in data records shared between big-endian and little-endian platforms must have the same sizes. For example, a file produced by a SPARC executable compiled with `-xyptemap=integer:64,real:64,double:128` cannot be read by an x86 executable compiled with `-xtypemap=integer:64,real:64,double:64` since the default double precision data types will have different sizes.

Shared I/O files must not contain VAX `UNION/MAP` data structures since it is not possible for the compiler to know how the `UNION` data should be interpreted. Declaring a file containing `UNION` data with the `-xfilebyteorder` flag will result in a runtime error.

Command-line Debugger `dbx`

TABLE 1-4 `dbx` New Features

Feature	Description
AMD64 architecture support	64-bit `dbx` now supports the AMD64 architecture.

As in Sun Studio software for SPARC based systems, Sun Studio software for x86 based systems includes two `dbx` binaries, a 32-bit `dbx` that can debug 32-bit programs only, and a 64-bit `dbx` that can debug both 32-bit and 64-bit programs.

When you start `dbx`, it determines which of its binaries to execute. On the 64-bit Solaris OS, the 64-bit `dbx` is the default.

OpenMP API

TABLE 1-5 OpenMP API New Features

Feature	Description
Availability on x86 based systems running the Solaris 10 OS.	The same OpenMP API features already available for the Solaris OS on SPARC based systems are now available with the Sun Studio compilers on 32-bit or 64-bit x86 based systems running the Solaris 10 OS.
`libmtsk`	the multitasking library, `libmtsk`, is now a shared library and is part of the Solaris 10 OS.
Nested parallelism	Nested parallelism is supported in this release. It is disabled by default, and requires that you set the `OMP_NESTED` environment variable make a runtime call to the `omp_set_nested()` function to enable it. With nested parallelism enabled, calls to most `omp_` functions made from within a parallel region will not be ignored. Calls to adjust the parallel environment (for example, `omp_set_num_threads()` or `omp_set_dynamic()`) affect only the subsequent parallel regions at the same or inner nesting level encountered by the thread.
Default behavior for threads	The default behavior for threads is now SLEEP. The previous default was SPIN. To restore the previous behavior, use `SUNW_MP_THR_IDLE=SPIN.`

TABLE 1-5 OpenMP API New Features *(Continued)*

Feature	Description
`SUNW_MP_NUM_POOL_THREADS` environment variable	`SUNW_MP_NUM_POOL_THREADS` specifies the size (maximum number of threads) of the thread pool. The thread pool contains only non-user threads—threads that the `libmtsk` library creates. It does not include user threads such as the main thread. Setting `SUNW_MP_NUM_POOL_THREADS` to 0 forces the thread pool to be empty and all parallel regions will be executed by one thread. The value specified should be a non-negative integer. The default value is 1023. This environment variable can prevent a single process from creating too many threads, which is something that might happen, for example, with recursively nested parallel regions.
`SUNW_MP_MAX_NESTED_LEVELS` environment variable	`SUNW_MP_MAX_NESTED_LEVELS` specifies the maximum depth of active parallel regions. Any parallel region that has an active nested depth greater than `SUNW_MP_MAX_NESTED_LEVELS` will be executed by a single thread. The value should be a positive integer. The default is 4. The outermost parallel region has a depth level of 1.
`SUNW_MP_GUIDED_WEIGHT` environment variable	`SUNW_MP_GUIDED_WEIGHT` sets the weighting value used by `libmtsk` for loops with the `GUIDED` schedule. `libmtsk` uses the following formula to compute the chunk sizes for `GUIDED` loops: *chunk_size=num_unassigned_iterations*/(*weight***num_threads*) where *num_unassigned_iterations* is the number of iterations in the loop that have not yet been assigned to any thread, *weight* is a floating-point constant (default 2.0 in this release, 1.0 previously), and *num_threads* is the number of threads used to execute the loop. The value specified for `SUNW_MP_GUIDED_WEIGHT` must be a positive, non-zero floating-point constant. `libmtsk` will use that value as weight in the `GUIDED` chunk size calculation.

Interval Arithmetic

There are no new interval arithmetic features in this release.

Sun Performance Library

TABLE 1-6 Sun Performance Library New Features

Feature	Description
64-bit Solaris OS support	This release of Sun Performance Library includes support for the 64-bit Solaris OS on x86 based systems.

The 64-bit x86 version of Sun Performance Library is functionally identical to the SPARC v9 version, with the following exceptions:

- Quad-precision routines (`dqdoti`, `dqdota`) are not available.
- Interval BLAS routines are not available.
- Routines with 64-bit integer parameters are not available. For example, `DAXPY()` is available, but `DAXPY_64()` is not.

To link with the high performance amd64 optimized library, use the `-xarch=amd64` flag. For example:

```
f95 -xarch=amd64 example.f -xlic_lib=sunperf
```

dmake

TABLE 1-7 `dmake` New Features

Feature	Description
New `DMAKE_OUTPUT_MODE` environment variable	A new environment variable or makefile macro, `DMAKE_OUTPUT_MODE`, allows the format of the log file to be changed. By default, or when `DMAKE_OUTPUT_MODE` is set to `TXT1`, `dmake` prints additional lines of system information to the log file, and commands with output are repeated. When `DMAKE_OUTPUT_MODE` is set to `TXT2`, the system information is omitted and commands are never repeated. For details, refer to the `ENVIRONMENT/MACROS` section of the `dmake`(1) man page. (Note that the environment variable is incorrectly described in the man page; the correct values for `DMAKE_OUTPUT_MODE` are `TXT1` and `TXT2`.)
Unix2003 compliance	You can force Unix2003 compliance by setting `DMAKE_COMPAT_MODE=POSIX.`
Grid engine support	Specify grid engine support by setting `DMAKE_MODE=grid`.
Control of system overloading	Control system overloading with `DMAKE_ADJUST_MAX_JOBS`.
Improvements to memory usage	Improvements to memory usage are included in this release.

Performance Analysis Tools

TABLE 1-8 Performance Analysis Tools New Features

Feature	Description
Changes to experiment format	Changes have been made to the experiment format. The log now has an entry that gives the size of the targets in bits. Also, the version has changed from 9.1 to 9.2, so new experiments are not readable by older tools, but older experiments are readable using Sun Studio 10 tools.
`er_kernel` utility	A new `er_kernel` utility is now available on the Solaris 10 OS only. DTrace permissions are required to use this `er_kernel` utility.
Increased precision for performance metrics	The precision for percentage metrics in the Performance Analyzer and the `er_print` utility has increased from one to two decimal places.
Direct editing of the experiment Notes file	Direct editing of the experiment Notes file has been added to the Performance Analyzer.
New options to display function names	New options to display function names are now available in the Performance Analyzer and `er_print` command.
Enhanced metrics selection	Metrics selection has been enhanced in the Performance Analyzer. You can select or clear the display of all metrics at once.
Collector GUI changes	The menu used for following descendants has been moved to the Collect Experiment tab. In addition to the on and off options, the menu now supports the all option and extended hardware counter overflow profiling features.
Enhancements to hardware counter overflow profiling	Hardware counter overflow profiling has been enhanced to work with larger numbers of processors, including x86-based processors. The enhancement is available using the `collect -h` command, the `collector hwprofile` command in `dbx`, and the Performance Analyzer GUI.
New `appendfile` option	The `appendfile` option has been added to the `er_print` utility. This option allows output from the `er_print` utility to be appended to the end of an existing file.
Change in default behavior of `er_src` utility	The default behavior of the `er_src` utility has changed to be the same behavior as the following command: `er_src -source all -1 object`.
J2SE technology location	The Performance Analyzer and `collect` utility now use the default location of the J2SE technology where the product installer has installed it.

TABLE 1-8 Performance Analysis Tools New Features *(Continued)*

Feature	Description
New `collect -J java_args` option	The `collect -J java_args` option provides a means of passing flag arguments to the Java installation being used for profiling.
Sampling behavior changes during pause and resume	Sample data is generated prior to a pause and following a resume, but not when the collector is paused.
Pseudo function for JVM functions	The name of the pseudo function for Java Virtual Machine (JVM)* functions in Java Mode has been changed from `<JVM-Overhead>` to `<JVM-System>`.
`<Unknown>` subtypes	The names of the `<Unknown>` subtypes of Java functions has been changed to be more comprehensible.
`.er.rc` file paths	The paths of processed `.er.rc` files are now displayed in the Error/Warning Logs window for the Performance Analyzer and the `stderr` for the `er_print` and `er_src` utilities.
`JDK_1_4_2_HOME` environment variable	The environment variable `JDK_1_4_2_HOME`, which used to define the Java path to be used for data collection, is now obsolete.
Heap profiling	The heap profiling for Java programs is now obsolete since it will not be supported in JVM 1.5.
Extended options for `collect -j`	The `collect` utility will accept the values on or off and also a path to the Java installation to use for profiling.

* The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java™ platform.

Integrated Development Environment (IDE)

TABLE 1-9 IDE New Features

Feature	Description
Script execution capability	You can now execute scripts directly from the IDE.
ss_attach on Linux operating system	The ss_attach feature is now available in Sun Studio software running on the Linux operating system

Documentation

See the Latest News page on the developer portal at `http://developers.sun.com/prodtech/cc/support_index.html` for information that updates the Sun Studio 10 documentation.

CHAPTER **2**

Sun Studio 9 New Features and Enhancements

Sun™ Studio 9 replaces the Sun™ Studio 8. New features in the Sun Studio 9 release include updates to the following compilers, libraries, and tools:

- C Compiler
- C++ Compiler
- Fortran Compiler
- Sun Performance Library
- Distributed `make` utility, `dmake`
- `dbx` Command-Line Debugger
- Performance Analysis Tools
- Integrated Development Environment (IDE)
- Documentation

In most sections, there is a table that lists the new features of that component. The table has two columns, where the left-hand column provides a short description of the feature, and the right-hand column has a longer description.

Note – To find the Sun Studio 9 documentation described in this chapter, see the documentation index installed with the product software at `/opt/SUNWspro/docs/index.html`. If your software is not installed in the `/opt` directory, contact your system administrator for the equivalent path on your system or network.

C Compiler

This section lists the new features of the C compiler for this release. The new features are organized into the following tables:

- TABLE 2-1 General Enhancements
- TABLE 2-2 Enhanced Hardware Platform Support
- TABLE 2-3 Improved Performance and Optimization Options
- TABLE 2-4 New Security Checks through the Lint utility

For more information about the specific compiler options referenced in this section, see the *C User's Guide* or the `cc`(1) man page.

TABLE 2-1 lists the general enhancements of the C compiler.

TABLE 2-1 General Enhancements of the C Compiler

Feature	**Description**
Implementation of additional C99 features	This release adds support for the following ISO/IEC 9899:1999 (referred to as C99 in this document) features. The following list only details the C99 features implemented in this release, which is a subset of all the implemented C99 features. See the *C User's Guide* for a complete listing of all C99 features implemented over the past and current releases of the C compiler. The sub-section number of the C99 standard is listed for each new item supported in this Sun Studio 9 release. • 5.2.4.2.2: Support for the `FLT_EVAL_METHOD` macro. This macro, and a new `-flteval` compile option, determines whether the compiler evaluates floating point expressions as long doubles or whether they are evaluated based on the combination of types and constants in the expression. • 6.4.3: Support for four-digit and eight-digit Universal Character Names (UCN), which can be used in identifiers, character constants, and string literals to designate characters that are not in the C basic character set. The UCN \Unnnnnnnn designates the character whose eight-digit short identifier (as specified by ISO/IEC 10646 is nnnnnnnn. Similarly, the universal character name \unnnn designates the character whose four-digit short identifier is nnnn (and whose eight-digit short identifier is 0000nnnn. • 6.7.4: Support for inline functions and extern inline functions • 6.7.8: Support for designated initializers, which provide a method for initializing sparse arrays and structures, common in numerical and systems programming.

TABLE 2-1 General Enhancements of the C Compiler *(Continued)*

Feature	Description
Improved compatibility with old binaries through the new -features compile option	You can now link old C and C++ binaries (pre C/C++ 5.6) with new C and C++ binaries with no change of behavior for the old binaries. Use the -features=no%extinl compile option when you want compatibility between new binaries and old C and C++ binaries that contain extern inline functions. To get standard-conforming behavior, old code must be recompiled using the current compiler.
Larger default stack size for slave threads	The default stack size for slave threads is now larger. All slave threads have the same stack size, which is four megabytes for 32-bit applications and eight megabytes for 64-bit applications by default. The size is set with the `STACKSIZE` environment variable.
Improved `-xprofile` (SPARC®)	The `-xprofile` option offers the following improvements: • Support for profiling shared libraries • Thread-safe profile collection using `-xprofile=collect -mt` • Improved support for profiling multiple programs or shared libraries in a single profile directory With `-xprofile=use`, the compiler can now find profile data in profile directories that contain data for multiple object files with non unique basenames. For cases where the compiler is unable to find an object file's profile data, the compiler provides a new option `-xprofile_pathmap=`*collect-prefix*: *use-prefix*.
Support for UTF-16 string literals: `-xustr`	Specify `-xustr=ascii_utf16_ushort` if you need to support an internationalized application that uses ISO10646 UTF-16 string literals. In other words, use this option if your code contains a string literal composed of 16-bit characters. Without this option, the compiler neither produces nor recognizes 16-bit character string literals. This option enables recognition of the U"*ASCII_string*" string literals as an array of type unsigned short. Since such strings are not yet part of any standard, this option enables recognition of non-standard C.
Automatically generated precompiled headers	This release of the C compiler expands the precompiled header facility to include an automatic capability on the part of the compiler to generate the precompiled header file. You still have the option to manually generate the precompiled header file, but if you are interested in the new capability of the compiler, see the explanation for the `-xpch` option in the `cc`(1) manpage for more information. See also the `CCadmin`(1) manpage.

TABLE 2-2 lists the new features of the C compiler that support faster compilation.

TABLE 2-2 Enhanced Hardware Platform Support

Feature	Description
`More flags to support SPARC® platforms`	The `-xchip` and `-xtarget` options now support ultra3i and ultra4 as values so you can build applications that are optimized for the UltraSPARC IIIi and UltraSPARC IV processors.
More flags to support x86 platforms	The C compiler supports new flags for `-xarch`, `-xtarget`, and `-xchip` compile options for code that will run on x86 platforms. These new flags are designed to take advantage of Pentium 3 and Pentium 4 chips in combination with Solaris™ software support for `sse` and `sse2` instructions on the x86 platform. The new flags are as follows: • `-xchip=pentium3` optimizes for Pentium 3 style processor • `-xchip=pentium4` optimizes for Pentium 4 style processor • `-xarch=sse` adds the sse instruction set to the pentium_pro instruction set architecture • `-xarch=sse2` adds the sse2 instruction set to those permitted by sse • `-xtarget=pentium3` sets -xarch=sse, -xchip=pentium3, and -xcache=16/32/4:256/32/4 • `-xtarget=pentium4` sets -xarch=sse2, -xchip=pentium4, and -xcache=8/64/4:256/128/8 You can determine which combination of options is appropriate for your compilation by following these guidelines: • If you are building an application to run on a Pentium 3 or Pentium 4 machine with Solaris 9 update 6 or later, compile with `-xtarget=pentium3` or `-xtarget=pentium4`, as appropriate. • If you are building an application to run on a Pentium 3 or Pentium 4 machine with Solaris 9 update 5 or earlier, set `-xarch=pentium_pro` (not pentium3 or pentium4 as you might expect) because the Solaris 9 update 5 or earlier operating systems do not support sse and sse2 instructions. Set `-xchip` and `-xcache` to the same values that are used when `-xtarget=pentium3` or `-xtarget=pentium4`, depending on the target machine. • If you are building on the target machine, specifying `-fast`, `-xarch=native`, or `-xtarget=native` will automatically expand to the appropriate `-xchip`, `-xarch`, and `-xtarget` flag settings described above.

TABLE 2-3 lists the new features of the C compiler that support improved performance.

TABLE 2-3 Improved Performance and Optimization Options

Feature	Description
`New defaults and expansions for compiler options`	The defaults for the following compile options have changed: • `-xarch` on SPARC® platforms: v8plus. The new default yields higher run-time performance for nearly all machines in current use. However, applications that are intended for deployment on pre-UltraSPARC computers no longer execute using the default option; compile with `-xarch=v8` to ensure that the applications execute on pre-UltraSPARC computers. • `-xcode` on SPARC® platforms: `abs44` for v9 and `abs32` for v8. • `-xmemalign` on SPARC® platforms: `8i` for v8 and `8s` for v9 • `-xprefetch` on SPARC® platforms: `auto,explicit`. This change adversely affects applications that have essentially non-linear memory-access patterns. To disable the change, specify `-xprefetch=no%auto,no%explicit`. The expansions for the following option and macro have changed: • The `-fast` option now includes the new option `-xlibmopt` in its expansion (see below). • The `-O` macro now expands to `-xO3` instead of `-xO2`. The change in default yields higher run-time performance. However, `-x03` may be inappropriate for programs that rely on all variables being automatically considered volatile. Typical programs that might rely on this assumption are device drivers and older multi-threaded applications that implement their own synchronization primitives. The work-around is to compile with `-xO2` instead of `-O`.
`New optimization compile options`	The new compile options are as follows: • `-xlibmopt` and `-xnolibmopt`: The `-xlibmopt` option enables the compiler to use a library of optimized math routines. You must use default rounding mode by specifying `-fround=nearest` when using the `-xlibmopt` option. The math routine library is optimized for performance and usually generates faster code. The results may be slightly different from those produced by the normal math library. If so, they usually differ in the last bit. You can explicitly turn off this library by specifying the new `-xnolibmopt` option on the command line. • `-xipo_archive`: Use the new `-xipo_archive` option to enable the compiler to optimize object files passed to the linker with object files that were compiled with `-xipo` and that reside in the archive library (.a) before producing an executable. Any object files contained in the library that are optimized during the compilation are replaced with their optimized version.

TABLE 2-3 Improved Performance and Optimization Options *(Continued)*

Feature	Description
New optimization compile options *(continued)*	• -xprefetch_auto_type: Use the new -xprefetch_auto_type option to generate indirect prefetches for the loops indicated by the option -xprefetch_level=[1|2|3] in the same fashion that the prefetches for direct memory accesses are generated. Options such as -xdepend, -xrestrict, and -xalias_level can improve the optimization benefits of -xprefetch_auto_type. They affect the aggressiveness of computing the indirect prefetch candidates and therefore the aggressiveness of the automatic indirect prefetch insertion, because they help produce better disambiguation of memory-alias information.

TABLE 2-4 describes the new security-checking feature included in the `lint` utility.

TABLE 2-4 New Security Checks Through the Lint Utility

Feature	Description

New `-errsecurity` option for `lint`	The Sun Studio 9 release of the `lint` utility features a new security-checking facility. You can use the new `-errsecurity` option before compilation to check your code for security liabilities.
	
	`-errsecurity[={core | standard | extended | %none}]`
	
	`lint -errsecurity=core`
	Checks for source code constructs that are almost always either unsafe or difficult to verify. Checks at this level include:
	• Use of variable format strings with the `printf()` and `scanf()` family of functions
	• Use of unbounded string (%s) formats in `scanf()` functions
	• Use of functions with no safe usage: `gets()`, `cftime()`, `ascftime()`, `creat()`
	• Incorrect use of `open()` with `O_CREAT`
	Consider source code that produces warnings at this level to be a bug. The source code in question should be changed. In all cases, straightforward safer alternatives are available.
	
	`lint -errsecurity=standard`
	Includes all checks from the core level plus constructs that may be safe, but have better alternatives available. This level is recommended when checking newly-written code. Additional checks at this level include:
	• Use of string copy functions other than `strlcpy()`
	• Use of weak random number functions
	• Use of unsafe functions to generate temporary files
	• Use of `fopen()` to create files
	• Use of functions that invoke the shell
	Replace source code that produces warnings at this level with new or significantly modified code. Balance addressing these warnings in legacy code against the risks of destabilizing the application.

TABLE 2-4 New Security Checks Through the Lint Utility *(Continued)*

Feature	Description
New `-errsecurity` option for `lint` *(continued)*	`lint -errsecurity=extended` Contains the most complete set of checks, including everything from the Core and Standard levels. In addition, a number of warnings are generated about constructs that may be unsafe in some situations. The checks at this level are useful as an aid in reviewing code, but need not be used as a standard with which acceptable source code must comply. Additional checks at this level include: • Calls to `getc()` or `fgetc()` inside a loop • Use of functions prone to pathname race conditions • Use of the `exec()` family of functions • Race conditions between `stat()` and other functions Review source code that produces warnings at this level to determine if the potential security issue is present. If you do not specify a setting for `-errsecurity`, the compiler sets it to `-errsecury=%none`. If you do specify `-errsecurity`, but not an argument, the compiler sets it to `-errsecurity=standard`.

C++ Compiler

This section lists the new features of the C++ compiler for this release. The new features are organized into the following tables:

- TABLE 2-5 General Enhancements
- TABLE 2-6 Enhanced Hardware Platform Support
- TABLE 2-7 New and Enhanced Optimization Options

For more information about the specific compiler options referenced in this section, see the *C++ User's Guide* or the `CC(1)` man page.

TABLE 2-5 lists the general enhancements of the C++ compiler (version 5.6).

TABLE 2-5 General Enhancements of the C++ Compiler

Feature	Description
Externally linked inline functions	The C++ standard states that `inline` functions have external linkage, like non-inline functions, unless declared static. C++ 5.6, for the first time, gives inline functions external linkage by default. If an inline function must be generated out of line (for example, if its address is needed), only one copy is linked into the final program. Previously, each object file that needed a copy had its own copy with local linkage. This implementation of `extern inline` functions is compatible with binary files created by earlier compiler versions, in the sense that program behavior is no less standard-conforming than before. The old binaries might have multiple local copies of inline functions, but new code will have at most one copy of an `extern inline` function. This implementation of `extern inline` functions is compatible with the C99 version of inline functions using the C 5.6 compiler that is included in this release. That is, following the C and C++ rules for extern inline functions, the same inline function can be defined in both C and C++ files, and only one copy of the external function will appear in the final program.
Enhanced UTF-16 support	Version 5.5 of the C++ compiler introduced support for UTF-16 string literals. This release expands support for UTF-16 character literals that use the syntax U'x', which is analogous to the U"x" syntax for strings. The same `-xustr` option is required to enable recognition of UTF-16 character literals. This release also supports numeric escapes in UTF-16 character and string literals, which are analogous to numeric escapes in ordinary character literals and strings. For example: U"ab\123ef" // octal representation of character U'\x456' // hexadecimal representation of character Refer to the description of -xustr in the C++ manpage CC(1) for details.

TABLE 2-5 General Enhancements of the C++ Compiler *(Continued)*

Feature	**Description**
Automatically generated precompiled header files	This release of the C++ compiler expands the precompiled header facility to include an automatic capability on the part of the compiler to generate the precompiled header file. You still have the option to manually generate the precompiled header file, but if you are interested in the new capability of the compiler, see the explanation for the `-xpch` option in the CC(1) manpage for more information. See also the CCadmin(1) manpage.

TABLE 2-6 lists the new features of the C++ compiler that support faster compilation.

TABLE 2-6 Enhanced Hardware Platform Support

Features	**Description**
`More flags to support SPARC® platforms`	The `-xchip` and `-xtarget` options now support ultra3i and ultra4 as values so you can build applications that are optimized for the UltraSPARC IIIi and UltraSPARC IV processors.
More flags to support x86 platforms	The C compiler supports new flags for `-xarch`, `-xtarget`, and `-xchip` compile options for code that will run on x86 platforms. These new flags are designed to take advantage of Pentium 3 and Pentium 4 chips in combination with Solaris™ software support for `sse` and `sse2` instructions on the x86 platform. The new flags are as follows: • `-xchip=pentium3` optimizes for Pentium 3 style processor • `-xchip=pentium4` optimizes for Pentium 4 style processor • `-xarch=sse` adds the sse instruction set to the pentium_pro instruction set architecture • `-xarch=sse2` adds the sse2 instruction set to those permitted by sse • `-xtarget=pentium3` sets -xarch=sse, -xchip=pentium3, and -xcache=16/32/4:256/32/4 • `-xtarget=pentium4` sets -xarch=sse2, -xchip=pentium4, and -xcache=8/64/4:256/128/8

TABLE 2-6 Enhanced Hardware Platform Support *(Continued)*

Features	Description
More flags to support x86 platforms *(continued)*	You can determine which combination of options is appropriate for your compilation by following these guidelines: • If you are building an application to run on a Pentium 3 or Pentium 4 machine with Solaris 9 update 6, compile with -xtarget=pentium3 or -xtarget=pentium4, as appropriate. • If you are building an application to run on a Pentium 3 or Pentium 4 machine with Solaris 9 update 5 or earlier, set -xarch=pentium_pro (not pentium3 or pentium4 as you might expect) because the Solaris 9 update 5 or earlier operating systems do not support sse and sse2 instructions. Set -xchip and -xcache to the same values that are used when -xtarget=pentium3 or -xtarget=pentium4, depending on the target machine. • If you are building on the target machine, specifying -fast, -xarch=native, or -xtarget=native will automatically expand to the appropriate -xchip, -xarch, and -xtarget flag settings described above.

TABLE 2-7 lists the new features of the C++ compiler that support easier porting:

TABLE 2-7 New and Enhanced Optimization Options

Feature	Description
New defaults and expansions for compiler options	The defaults for the following compile options have changed: • -xarch on SPARC® platforms: v8plus. The new default yields higher run-time performance for nearly all machines in current use. However, applications that are intended for deployment on pre-UltraSPARC computers no longer execute using the default option; compile with -xarch=v8 to ensure that the applications execute on pre-UltraSPARC computers. • -xcode on SPARC® platforms: abs44 for v9 and abs32 for v8. • -xmemalign on SPARC® platforms: 8i for v8 and 8s for v9 • -xprefetch on SPARC® platforms: auto,explicit. This change adversely affects applications that have essentially non-linear memory-access patterns. To disable the change, specify -xprefetch=no%auto,no%explicit. The expansions for the following macro has changed: • The -O macro now expands to -xO3 instead of -xO2. The change in default yields higher run-time performance. However, -x03 may be inappropriate for programs that rely on all variables being automatically considered volatile. Typical programs that might rely on this assumption are device drivers and older multi-threaded applications that implement their own synchronization primitives. The work-around is to compile with -xO2 instead of -O.

TABLE 2-7 New and Enhanced Optimization Options *(Continued)*

Feature	Description
New loop optimization compile options	The C++ compiler now supports the following options for optimization of loops whose computations can be parallelized. These options have an effect only if you specify an optimization level of -xO3 or higher. • `-xautopar` • `-xvector` • `-xdepend` Refer to the description of `-xautopar`, `-xvector`, and `-xdepend`, in the C++ manpage CC(1) for details.
New function-specific optimization-level control	You can combine the `#pragma opt` directive with the command-line option `-xmaxopt` to specify the level of optimization the compiler applies to individual functions. The combination is useful when you need to reduce the optimization level for specific functions, for example to avoid a code enhancement like elimination of stack frames, or to increase optimization level for specific functions.
Prefetch optimization for loops	`-xprefetch_auto_type`: Use the new `-xprefetch_auto_type` option to generate indirect prefetches for the loops indicated by the option -xprefetch_level=[1|2|3] in the same fashion that the prefetches for direct memory accesses are generated. Options such as `-xdepend`, `-xrestrict`, and `-xalias_level` can improve the optimization benefits of `-xprefetch_auto_type`. They affect the aggressiveness of computing the indirect prefetch candidates and therefore the aggressiveness of the automatic indirect prefetch insertion, because they help produce better disambiguation of memory-alias information
Restricted pointers optimization	C++ does not support the `restrict` keyword introduced in C99. But the C++ compiler now accepts the C compiler option `-xrestrict`. This option makes claims about functions in the compilation to the effect that function parameters of pointer type do not refer to the same or overlapping objects. This option is somewhat more dangerous for C++ than for C, because the claim is not true for some functions in the C++ standard library.

Fortran Compiler

TABLE 2-8 lists the new and enhanced features of the Fortran compiler for this release, which include the following:

- New Compile Capability for f95 on Solaris™ OS x86 Platforms
- Improved Runtime Performance
- New Fortran 2003 command-line intrinsics
- Changed `f95` compiler command-line option defaults
- Change in Default SPARC® Architecture
- Enhancements to OpenMP Library
- New `f95` compiler command-line options

For more information about the specific compiler options referenced in this section, see the *Fortran User's Guide* or the `f95(1)` man page.

TABLE 2-8 Fortran Compiler New and Enhanced Features

Feature	Description
New Compile Capability for f95 on Solaris OS x86 Platforms	Compile with `-xtarget` values `generic`, `native`, `386`, `486`, `pentium`, `pentium_pro`, `pentium3`, or `pentium4`, to generate executables on Solaris x86 platforms. The default on x86 platforms is -xtarget=generic The following `f95` features are not yet implemented on x86 platforms and are only available on SPARC® platforms: • Interval Arithmetic (compiler options `-xia` and `-xinterval`) • Quad (128-bit) Arithmetic • IEEE Intrinsic modules IEEE_EXCEPTIONS, IEEE_ARITHMETIC, and IEEE_FEATURES • The sun_io_handler module • Parallelization options such as `-autopar`, `-parallel`, `-explitipar`, and `openmp`. The following `f95` command-line options are only available on x86 platforms and not on SPARC® platforms: • `-fprecision`, `-fstore`, `-nofstore` The following f95 command-line options are only available on SPARC® platforms and not on x86 platforms: • `-xcode`, `-xmemalign`, `-xprefetch`, `-xcheck`, `-xia`, `-xinterval`, `-xipo`, `-xjobs`, `-xlang`, `-xlinkopt`, `-xloopinfo`, `-xpagesize`, `-xprofile_ircache`, `-xreduction`, `-xvector`, `-depend`, `-openmp`, `-parallel`, e- `-autopar`, `-explicitpar`, `-vpara`, `-XlistMP` Also, on x86 platforms the `-fast` option expands to include the added option, `-nofstore`.

TABLE 2-8 Fortran Compiler New and Enhanced Features *(Continued)*

Feature	Description
New Compile Capability for f95 on Solaris OS x86 Platforms *(continued)*	The Fortran compiler supports new flags for `-xarch`, `-xtarget`, and `-xchip` compile options for code that will run on x86 platforms. These new flags are designed to take advantage of Pentium 3 and Pentium 4 chips in combination with Solaris™ software support for `sse` and `sse2` instructions on the x86 platform. The new flags are as follows: • `-xchip=pentium3` optimizes for Pentium 3 style processor • `-xchip=pentium4` optimizes for Pentium 4 style processor • `-xarch=sse` adds the sse instruction set to the pentium_pro instruction set architecture • `-xarch=sse2` adds the sse2 instruction set to those permitted by sse • `-xtarget=pentium3` sets -xarch=sse, -xchip=pentium3, and -xcache=16/32/4:256/32/4 • `-xtarget=pentium4` sets -xarch=sse2, -xchip=pentium4, and -xcache=8/64/4:256/128/8 • `-fns` is enabled only on pentium3 or pentium4 processors. When `-xarch` is not `sse` or `sse2`, `-fns=yes` is ignored. Otherwise, for SSE and SSE2 floating-point instructions, `-fns=yes` implies that underflows will be flushed to zero (FTZ) and that denormalized operands are treated as zero (DAZ). `-fns=yes` does not affect traditional x86 floating-point instructions. For example, floating-point operations on long double operands or results utilize traditional x86 floating-point instructions and these would not be affected by `-fns=yes`. *SPECIAL x86 NOTE:* Programs compiled with `-xarch=sse` or `-xarch=sse2` to run on Solaris™ x86 SSE/SSE2 platforms must be run only on platforms that are SSE/SSE2 enabled. Running such programs on platforms that are not SSE/SSE2-enabled could result in segmentation faults or incorrect results occurring without any explicit warning messages. Patches to the OS and compilers to prevent execution of SSE/SSE2-compiled binaries on platforms not SSE/SSE2-enabled could be made available at a later date. SSE/SSE2-enabled x86 platforms include Solaris 9 update 6 running on a Pentium 4 compatible processor. This warning extends also to programs that employ `.il` inline assembly language functions or `__asm()` assembler code that utilize SSE/SSE2 instructions. Contact your system administrator to determine if the target runtime platform is SSE/SSE2-enabled before attempting to run binaries compiled for these platforms.

TABLE 2-8 Fortran Compiler New and Enhanced Features *(Continued)*

Feature	Description
Improved runtime performance	Runtime performance for most applications should improve significantly with this release. For best results, compile with high optimization levels `-xO4` or `-xO5`. At these levels the compiler may now inline contained procedures, and those with assumed-shape, allocatable, or pointer arguments.
New Fortran 2003 command-line intrinsics	The Fortran 2003 draft standard introduces three new intrinsics for processing command-line arguments and environment variables. These have been implemented in this release of the f95 compiler. The new intrinsics are: • GET_COMMAND(command, length, status) Returns in command the entire command line that invoked the program. • GET_COMMAND_ARGUMENT(number, value, length, status) Returns a command-line argument in value. • GET_ENVIRONMENT_VARIABLE(name, value, length, status, trim_name) Returns the value of an environment variable.
Changed command-line option defaults	The following command-line option defaults have changed with this release of f95. • The default for `-xprefetch` is `-xprefetch=no%auto,explicit`. • The default for `-xmemalign` is `-xmemalign=8i`, except with `-xarch=v9` and `v9a` where the default is `-xmemalign=8f`.
Change in Default SPARC® Architecture	The default SPARC® architecture is no longer V7. Support for `-xarch=v7` is limited in this Sun Studio 9 release. The new default is `V8PLUS` (UltraSPARC). Compiling with `-xarch=v7` is treated as `-xarch=v8` because the Solaris 8 OS only supports `-xarch=v8` or better.
Enhancements to OpenMP Library	The OpenMP library has been enhanced as follows: • The maximum number of threads for `OMP_NUM_THREADS` and the multitasking library has increased from 128 to 256. • This release of the Fortran 95 compiler's implementation of the OpenMP API for shared-memory parallel programming features automatic scoping of variables in parallel regions. See the OpenMP API User's Guide for details. (OpenMP is only implemented on SPARC® platforms for this release.)

TABLE 2-8 Fortran Compiler New and Enhanced Features *(Continued)*

Feature	Description
New f95 compiler command-line options	The following f95 command-line options are new in this release. See the f95(1) man page for details. • `-xipo_archive={ none | readonly | writeback }` Allow crossfile optimization to include archive (.a) libraries. (SPARC® only) • `-xipo_archive=none` No processing of archive files. • `-xipo_archive=readonly` The compiler optimizes object files passed to the linker with object files compiled with -xipo that reside in the archive library (.a) before producing an executable. • `-xipo_archive=writeback` The compiler optimizes object files passed to the linker with object files compiled with -xipo that reside in the archive library (.a) before producing an executable. Any object filed contained in the library that were optimized during the compilation are replaced with their optimized version. If you do not specify a setting for -xipo, the compiler sets it to `-xipo_archive=none`. • `-xprefetch_auto_type=[no%]indirect_array_access` Generate indirect prefetches for a data arrays accessed indirectly. (SPARC® only) • `[no%]indirect_array_access` Does [Does not] generate indirect prefetches for the loops indicated by the option `-xprefetch_level=[1|2|3]` in the same fashion the prefetches for direct memory accesses are generated. If you do not specify a setting for `-xprefetch_auto_type`, the compiler sets it to `-xprefetch_auto_type=[no%]indirect_array_access`. The `-xprefetch` options are only available on SPARC® platforms Options such as `-xdepend`, `-xrestrict`, and `-xalias_level` can affect the aggressiveness of computing the indirect prefetch candidates and therefore the aggressiveness of the automatic indirect prefetch insertion due to better disambiguation of memory-alias information. • `-xprofile_pathmap=collect_prefix:use_prefix` Set path mapping for profile data files. Use the `-xprofile_pathmap` option with the `-xprofile=use` option when profiling into a directory that is not the directory used when previously compiling with `-xprofile=collect`.

Command-Line Debugger `dbx`

The following new features have been added to the Sun Studio 9 release of `dbx`:

- Support for gcc and g++ compilers on Linux platforms
- Support for Fortran on Solaris™ OS, x86 platform edition

Interval Arithmetic

There are no new interval arithmetic features in this release.

Sun Performance Library

Sun Performance Library™ is a set of optimized, high-speed mathematical subroutines for solving linear algebra problems and other numerically intensive problems. Sun Performance Library is based on a collection of public domain applications available from Netlib (at `http://www.netlib.org`). These routines have been enhanced and bundled as the Sun Performance Library.

TABLE 2-9 lists the new features in this release of the Sun Performance Library. See the *Sun Performance Library User's Guide* and the section 3p man pages for more information.

TABLE 2-9 Sun Performance Library New Features

Feature	Description

Sun Performance Library released for x86	This release of Sun Performance Library includes libraries for the Solaris/x86 platform. Two versions are available:
	• A high-performance version utilizing SSE2 instructions for systems that support that instruction set.
	• A compatibility version suitable for systems that do not support SSE2.
	The x86 version of Sun Performance Library is functionally identical to the SPARC® version, with the following exceptions:
	• Quad-precision routines (`dqdoti`, `dqdota`) are not available
	• Interval BLAS routines are not available
	• The x86 libraries are single-threaded
	• Only 32-bit addressing is available
	• The Portable Library Performance feature is not available on Solaris/x86
	The following versions of Solaris/x86 are required for SSE2 support:
	• Solaris 10 build 48 (or later)
	• Solaris 9 build 6 update 5 (or later)
	To link with the high performance SSE2 optimized library, use the -xarch=sse2 flag. For example:
	`f95 -xarch=sse2 example.f -xlic_lib=sunperf`
	or
	`cc -xarch=sse2 example.c -xlic_lib=sunperf`

dmake

`dmake` is a command-line tool, compatible with `make`(1). `dmake` can build targets in distributed, parallel, or serial mode. If you use the standard `make`(1) utility, the transition to `dmake` requires little if any alteration to your makefiles. `dmake` is a superset of the `make` utility. With nested makes, if a top-level makefile calls `make`, you need to use `$(MAKE)`. `dmake` parses the makefiles and determines which targets can be built concurrently and distributes the build of those targets over a number of hosts set by you. See the `dmake`(1) man page for additional details. TABLE 2-10 lists the new features of `dmake` in the Sun Studio 9 release.

TABLE 2-10 `dmake` New Features

Feature	Description
Performance, reliability, and usability improvements in dmake for Solaris	The makefile parser is 10 times faster than the previous version, and 3 times faster than GNU make. Builds run faster and are more stable. The log file is also more readable.
Linux `dmake` implementation	Full dmake functionality is implemented for Linux builds in serial, parallel, and distributed modes. Consequently, Solaris™ applications can be built on Linux without big changes in makefiles. One build can be distributed to both Linux and Solaris™ systems.

Performance Analysis Tools

TABLE 2-11 lists the new data collection and presentation features in the Sun Studio 9 release of the performance analysis tools. For more information, see the following man pages:

- `analyzer(1)`
- `collect`(1)
- `collector`(1)
- `er_print`(1)
- `er_src`(1)
- `libcollector`(3)

TABLE 2-11 lists the new and enhanced features in the Sun Studio 9 Performance Analyzer.

TABLE 2-11 Performance Analysis Tools New Features

Feature	Description
New Linux distribution	The Performance Analyzer is now available in Sun Studio 9 for Linux, in addition to Sun Studio 9 for Solaris™. The following Linux operating systems are supported: • Java™ Desktop System 1.0 • SuSE Linux Enterprise Server 8 • RedHat Enterprise Linux 3 The utilities available are the same on both operating systems, except that er_kernel is not included in the Linux distribution. The collect command is more restricted on Linux. Only clock-based profiling and heap tracing are available; for details, refer to the collect man page. Profiling of multithreaded applications is possible under Linux, but presently high data discrepancies are observed when profiling under the RedHat version of the Linux operating system.
Dataspace profiling	Dataspace profiling is now possible for C programs targeted to a SPARC® platform. A dataspace profile is a data collection in which memory-related events, such as cache misses, are reported against the data-object references that cause the events rather than just the instructions where the memory-related events occur. The analysis of dataspace profiling information, can be displayed on the command line or in the Analyzer GUI as follows: • The `er_print` command has three new options related to dataspace profiling: `data_objects`, `data_osingle`, and `data_olayout` • The Analyzer now includes two new tabs related to dataspace profiling, labelled "Data Objects" and "Data Layout". These tabs will show automatically if a dataspace profile is present in the experiment.

TABLE 2-11 Performance Analysis Tools New Features *(Continued)*

Feature	Description
Descendant processes	The recording of descendant processes has been enhanced to include the ability to record all descendant processes, not just processes created using the fork and exec commands and their variants. To support the enhanced functionality, the collect `-F` command now has a new option: `collect -F all`. Descendants processed by `-F all` but not by `-F on`, like system calls, are named with the code letter "c". The data for descendant processes can be explicitly selected for display using the command-line utility `er_print` or in the Analyzer GUI. For more information, refer to the `collect`(1) man page.
Data collection output redirection	The collect command has a new option, `collect -O` *file*, which redirects all output from `collect` to the named *file*. The command does not redirect the output from the spawned target.
Enhanced Analyzer command-line arguments	The analyzer command (launch script) now accepts double-dash for long argument—in particular, `--jdkhome` and `--fontsize`.
New packages for Analyzer API shared libraries	The shared libraries for the Analyzer API have been put into separate packages so that they can be distributed independently and freely.
Notes file support for `collect` command	The `collect` command has a new command-line option: `collect -C` *`comment`*. The *`comment`* is added to the notes file for the experiment. Up to 10 `-C` arguments may be applied.
Notes in experiment preview and experiment header	Experiment preview and experiment header will show the contents of any notes file in the experiment
Enhanced source and disassembly displays	Annotated source and disassembly has improved handling of code from alternate source contexts. Index lines, shown in red italics, indicate where code is inserted from another file. With the Source tab, clicking the mouse on an index line will open the Source window in the alternate source file.
Enhanced `er_src` command	The command-line utility `er_src` can now show a function list, process Java `.class` files, and show source and disassembly from alternate source contexts.
Java™ method signatures	The Java™ long name format shows full method signatures rather than just the function name alone.
Inclusion of `mmap` calls when heap tracing	Calls to `mmap` are treated as memory allocations when heap tracing.

Integrated Development Environment (IDE)

The following new features have been added to the Sun Studio 9 release of the IDE:

- A new `ss_attach` feature that lets you capture a program as it starts executing and attach the dbx Debugger to begin debugging it immediately, rather than attaching the Debugger after the process is running.
- The Quick Browse combo box in the Source Editor that lets you navigate to a class method, function, #define, or other element of a source file.

Documentation

This section describes Sun Studio 9 documentation new features.

- The *OpenMP API User's Guide* has been extended to include two new chapters. Chapter 5 describes automatic data scoping with the Fortran 95 __AUTO clause. Chapter 6 considers performance of OpenMP programs, and gives some general recommendations on techniques for improving performance.
- The *C User's Guide* contains two new appendices: *Appendix A, "Compiler Options Grouped by Function"*, and *Appendix D, "C99 Implementation-Defined Information"*. The contents of Appendix A was at the beginning of the options reference chapter but is now separated into an appendix to enhance its visibility.
- The *Performance Analyzer* manual has a new chapter entitled "Understanding Annotated Source and Disassembly Data". The chapter describes the different kinds of annotations, such as index lines, compiler commentary, special lines (such as outline functions), and how to identify display differences between annotations and original source.
- A tutorial for the Performance Analyzer is available on the Sun developer's portal, `http://developers.sun.com/prodtech/cc`
- The *Sun WorkShop to Sun Studio Migration* helpset has a new topic on file comparison and file merging.
- A new *Compilers and Tools* helpset covering the compilers and tools included in the Sun Studio release. Each topic briefly describes a component and lists the documentation for that component.