What's New

Sun™ Studio 11

Sun Microsystems, Inc.
www.sun.com

Part No. 819-3682-10
November 2005, Revision A

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

Contents

Before You Begin

The *What's New* describes the new features of the Sun™ Studio 11 software release and the Sun Studio 10 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, these x86 related terms mean the following:

- "x86" refers to the larger family of 64-bit and 32-bit x86 compatible products.
- "x64' points out specific 64-bit information about AMD64 or EM64T systems.
- "32-bit x86" points out specific 32-bit information about x86 based systems.

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 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/sunstudio11/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 path to your PATH environment variable. If you have previously installed Forte Developer software, Sun ONE Studio software, or another release of Sun Studio software, add the following path before the paths to those installations.**

  /opt/SUNWspro/bin

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

  /opt/sun/sunstudio11/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 , follow the instructions in the next procedure to set your `MANPATH` environment variable.

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

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

  `/opt/SUNWspro/man`

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

  `/opt/sun/sunstudio11/man`

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/3.5V11` on Solaris platforms and `/opt/sun/netbeans/3.5V11` 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/3.5V11`. On Linux platforms, if the path to the directory that contains the IDE is `/foo/sunstudio11`, the command looks for the core platform in `/foo/netbeans/3.5V11`.

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/3.5V11`).

On Solaris platforms, each user of the IDE also must add */installation_directory*`/SUNWspro/bin` to their `$PATH` in front of the path to any other release of Forte Developer software, Sun ONE Studio software, or Sun Studio software. On Linux platforms, each user of the IDE also must add */installation_directory*`/sunstudio11/bin` to their `$PATH` in front of the path to any other release of Sun Studio software.

The path */installation_directory*`/netbeans/3.5V11/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/sunstudio11/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	HTML on the developer portal at `http://developers.sun.com/prodtech/cc/documentation/ss11/mr/README`s
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/sunstudio11/docs/index.html` on Linux platforms.
Online help	HTML available through the Help menu and Help buttons 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 (819-3682-10) of your document.

CHAPTER 1

Sun Studio 11 New Features and Enhancements

The Sun™ Studio 11 release includes 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 11 documentation described in this chapter, see the documentation index installed with the product software at `/opt/SUNWspro/docs/index.html` on Solaris platforms and at `/opt/sun/sunstudio11/docs/index.html` on Linux platforms. If your software is not installed in the `/opt` directory, contact your system administrator for the equivalent path on your system or network.

Common C, C++, and Fortran Features

The following new features are available through the C, C++ or Fortran compilers. See the following new-feature lists for information on new features that are specific to each language.

For more information about the these new features, see the user's guide or man page for each language.

TABLE 1-1 C Compiler New Features

Feature	Description
New `-xarch` Flags For x86 Development	The `-xarch` option now supports the following new flags for development on the x86 platform: • `amd64a` • `pentium_proa` • `ssea` • `sse2a`.
Support For x86 `-xpagesize` Options	The `-xpagesize`, `-xpagesize_heap`, `-xpagesize_stack` options are now enabled for x86 platforms as well as SPARC platforms.
Support for x86 Memory Models	The new `-xmodel` option lets you specify the kernel, small, or medium memory models on the 64-bit AMD architecture.
Support For SSE/SSE2 Integral Media Intrinsics	This release supports intrinsic functions for SSE2 128-bit XMM register integral media-instructions. Include the `sunmedia_intrin.h` header file in the source code and specify the `-xbuiltin` option to take advantage of these functions. Furthermore, these intrinsic functions require SSE2 support so specify options such as `-xarch=sse2`, `-xarch=amd64`, or `-xtarget=opteron`.
New `-xvector` Flags for x86 SSE2 Platforms	The `-xvector` option enables automatic generation of calls to the vector library functions and/or the generation of the SIMD (Single Instruction Multiple Data) instructions.
Binary Optimizer for SPARC Platforms	A new `-xbinopt` option allows the compiler to prepare the binary file for further optimization by the `binopt`(1) binary optimizer.

TABLE 1-1 C Compiler New Features *(Continued)*

Feature	Description
New SPARC `-xtarget` and `-xchip` Values	The new `-xtarget` flags `ultra3iplus`, `ultra4plus`, and `ultraT1` along with the new `-xchip` flags `ultra3iplus`, `ultra4plus`, and `ultraT1` provide code generation for the UltraSPARC IIIiplus, UltraSPARC T1, and UltraSPARC IVplus processors.
Enhancements to the `STACKSIZE` Environment Variable	The syntax of the `STACKSIZE` environment variable has been enhanced to accept a units keyword for denoting the slave thread stacksize: B for Bytes, K for Kilobytes, M for Megabytes, G for Gigabytes. For example, `setenv STACKSIZE 8192` sets the slave thread stack size to 8 MB. 1235B sets the slave thread stack size for 1235 Bytes. 1235G sets it for 1235 Gigabytes. The default for an integer value without a suffix letter is still Kilobytes.
OpenMP Autoscoping	Autoscoping is now available for C and C++ programs. This feature is described in chapter 3 of the Sun Studio *OpenMP API User's Guide*.

C Compiler

The C compiler also offers the following new features in addition to those features listed previously under "Common C, C++, and Fortran Features" on page 16.

TABLE 1-2 C Compiler New Features

Feature	Description
A New Default Format For Debugger Information	The C compiler now generates debugger information in the DWARF format by default. This change should be transparent, as the `dbx` and Performance Analyzer software readily accept and prefer the DWARF format. You can still generate debugger information in the stabs format by specifying `-xdebugformat=stabs`.
Two new pragmas	• c99 Use the `c99` (`implicit | no%implicit`) pragma to find implicit function declarations. • `[no_]warn_missing_parameter_info` Use the `[no_]warn_missing_parameter_info` pragma to find function declarations which contain no parameter-type information.

C++ Compiler

The C++ compiler also offers the following new features in addition to those features listed previously under "Common C, C++, and Fortran Features" on page 16.

TABLE 1-3 C++ Compiler New Features

Feature	Description
A New Format For Debugger Information	The C++ compiler can now generate debugger information in the DWARF format. The default is still the stabs format, but you can generate DWARF data by setting the new option `-xdebugformat` to `-xdebugformat=dwarf`.
Calling Dependent Static Functions From a Function Template	The C++ standard says that function calls that depend on a template parameter can refer only to visible function declarations having external linkage. Specify `-features=[no%]tmplrefstatic` if your application code depends on the compiler ignoring this rule and calling a dependent static function from a function template.

Fortran Compiler

There are no new features in the Fortran 95 compiler in this release beyond those listed previously under "Common C, C++, and Fortran Features" on page 16.

Command-line Debugger dbx

TABLE 1-4 dbx New Features

Feature	Description
Support for the AMD64 architecture on the Linux operating system	64-bit dbx now supports the AMD64 architecture. On the Linux OS, the 64-bit dbx cannot debug 32-bit programs. To debug a 32-bit program on the Linux OS, use the dbx command with the -x exec32 option to start the 32-bit dbx.
Support for the Sun Studio C compiler generating DWARF symbolic debugging information	The DWARF symbolic debugging information from the C compiler is used by default.
thr_create and thr_exit events	The thr_create event occurs when a thread, or a thread with the specified *thread_id*, has been created. The thr_exit event occurs when a thread has exited
step_abflow environment variable	When this environment variable is set to stop, dbx stops in longjmp(), siglongjmp(), and throw statements when single stepping.. When set to ignore, dbx does not detect abnormal control flow changes for longjmp() and siglongjmp().
Revised syntax for the intercept and unintercept commands	The syntax now includes the -set option, which clears both the intercept list and the excluded list, and sets the lists to intercept or exclude only throws of the specified types.
Support for debugging Java programs compiled with the javac compiler in Java™ 2 Platform, Standard Edition v 5.0 Update 3	Debugging Java programs compiled with J2SE 5.0 Update 3 is supported with the following limitations: - Partial support for generics - No support for autoboxing and unboxing - No support for static import

OpenMP API

TABLE 1-5 OpenMP API New Features

Feature	Description
OpenMP 2.5	The OpenMP implementation has been upgraded to the 2.5 specifications. See the OpenMP website http://www.openmp.org/ for details.
C++ Autoscoping	Automatic scoping of variables is now enabled for C++ programs as well as C and Fortran 95. Autoscoping is described in Chapter 3 of the *OpenMP User's Guide.*

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
Sun Perflib on x86 platforms	This release of Sun Performance Library includes libraries for the Solaris OS on 64-bit 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; that is, DAXPY() is available, but DAXPY_64() is not. • The Portable Performance Library feature is not available on the Solaris OS on x86 based systems. Many frequently-called BLAS kernels in the amd64 library have been optimized. Some internal FFT routines have also been further optimized. Similar to the SPARC v9 version, many routines are parallelized.
Sun Perflib on SPARC platforms	Improvements to BLAS and FFT for the latest UltraSPARC processors.

dmake

TABLE 1-7 dmake New Features

Feature	Description
Compatibility Mode	The `-x SUN_MAKE_COMPAT_MODE` command line option and `SUN_MAKE_COMPAT_MODE` environment variable added for compatibility with GNU make.

Performance Analysis Tools

TABLE 1-8 Performance Analysis Tools New Features

Feature	Description
Improved control over tabs displayed	The Analyzer's tab mechanism has been redesigned for greater flexibility. Only those tabs applicable to at least one loaded experiment are available, and a default set of tabs is shown, rather than all tabs, especially for large experiments. You can set default tabs in a .er.rc file, with the tabs directive. You can add or remove displayed tabs using the Tabs tab in the Set Data Presentation dialog box.
Advanced tab in Filter dialog box	The Filter dialog box now has an Advanced tab that lets you type a filter expression. You can also build an expression using the AND and OR operators, and phrases that reflect single or multiple selections from the Function tab, DataObject tab, DataLayout tab, or MemoryObject tabs.
Timeline tab now respects fltering	The Timeline Tab shows only events that pass the current filter settings.
Improved handling of descendant processes	The Analyzer and the `er_print` utility process an en_desc on|off directive in a `.er.rc` file. If the directive specifies `on`, all descendant experiments are read immediately; if the directive specifies `off`, only the founder experiment is read.
Improved behavior of New Windows in Analyzer	Additional Analyzer windows opened by clicking the New Window toolbar button or choosing File → Create New Window, are now more cleanly separated from each other. They share the loaded experiments, but you can set filtering, metrics, sorting, and so forth, independently in each window.
New Memory Objects tabs and report	New tabs are available in the Analyzer to show performance data for cache-lines, pages, etc. Several new `er_print` commands are available for memory objects. You can create a custom memory object by clicking Add Custom Object in the Tabs tab of the Set Data Presentation dialog box.
Hardware counter profiling on Linux	Hardware counter overflow profiling is available on supported Linux systems. This support requires that you install the Perfctr patch.
Improved handling of MPI profiling	Additional variables specifying process rank for LAM and MPICH versions of MPI are recognized.

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

Feature	Description
Java mode has been replaced by View mode	Java mode has been replaced by View mode. The View mode settings user, expert, and machine correspond to the Java mode settings on, expert, and off settings. View mode is applicable to programming models other than Java programs, OpenMP, in particular. The `javamode` command is accepted with a warning.
`er_print` command changes	Various commands to the `er_print` utility have been changed. Commands affecting data objects have been renamed, and the handling of the commands concerning metrics has been made more consistent. The new `procstat` command prints information concerning the processing of the data. The new `filters` command lets you specify a filter expression. A new expression grammar has been added for defining a filter and computing a memory object index.
Multiple selection in the Functions tab, DataObjects tab, DataLayout tab, tabs	In the Functions tab, DataObjects tab, DataLayout tab, and MemoryObjects tabs, you can now select multiple items.
Improved filtering	In addition to selecting experiments and filtering on the samples, threads, LWPs, and CPUs for which you want to display metrics, you can now specify a filter expression that evaluates to true for any data record you want to include in the display.

Integrated Development Environment (IDE)

TABLE 1-9 IDE New Features

Feature	Description
Support for J2SE 5.0 Update 3	The IDE now runs with J2SE 5.0 Update 3

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 11 documentation.

CHAPTER 2

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 2-1 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 2-2 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 28.
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 29.

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 2-3 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 30

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 2-4 `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 2-5 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`.
`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.

TABLE 2-5 OpenMP API New Features *(Continued)*

Feature	Description
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 2-6 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 2-7 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 2-8 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 2-8 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)[1] 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.

1 The terms “Java Virtual Machine” and “JVM” mean a Virtual Machine for the Java™ platform.

Integrated Development Environment (IDE)

TABLE 2-9 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.