What's New

Sun™ Studio 8

Sun Microsystems, Inc.
www.sun.com

Part No. 817-5062-10
March 2004, 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 this Sun™ Studio 8 software release.

Sun Studio 8 includes an Integrated Development Environment (IDE) that provides modules for creating, editing, building, debugging, and analyzing the performance of a C, C++, or Fortran application. It includes a set of basic Java™ language support modules that can be enabled if needed for JNI (Java™ Native Interface) development.

In addition to the IDE, Sun Studio 8 includes the C, C++, and Fortran compilers and libraries released with the Sun ONE Studio 8, Compiler Collection.

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	`#`

Accessing Sun Studio Software and Man Pages

The compilers and tools and their man pages are not installed into the standard `/usr/bin/` and `/usr/share/man` directories. To access the compilers and tools, you must have your `PATH` environment variable set correctly (see "Accessing the

Compilers and Tools" 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), and ksh(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 compilers and tools are installed in the /opt directory. If your software is not installed in the /opt directory, ask your system administrator for the equivalent path on your system.

Accessing the Compilers and Tools

Use the steps below to determine whether you need to change your PATH variable to access the compilers and tools.

▼ 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. **Review the output to find a string of paths that contain /opt/SUNWspro/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

1. **If you are using the C shell, edit your home .cshrc file. If you are using the Bourne shell or Korn shell, edit your home .profile file.**

2. **Add the following to your PATH environment variable. If you have Sun ONE Studio software or Forte Developer software installed, add the following path before the paths to those installations.**

   /opt/SUNWspro/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

1. **If you are using the C shell, edit your home .cshrc file. If you are using the Bourne shell or Korn shell, edit your home .profile file.**

2. **Add the following to your MANPATH environment variable.**

   /opt/SUNWspro/man

Accessing the Integrated Development Environment

The Sun Studio 8 integrated development environment (IDE) provides modules for creating, editing, building, debugging, and analyzing the performance of a C, C++, or Fortran application.

The IDE requires the Core Platform component of Sun Studio 8. You must set the SPRO_NETBEANS_HOME environment variable to the location where the Core Platform component is installed or mounted (*installation_directory*/netbeans/3.5R) if the Core Platform component is not installed or mounted to one of the following locations:

- The default installation directory /opt/netbeans/3.5R

- The same location as the Compilers and Tools component of the Sun Studio 8 (for example, the Compilers and Tools component installed in `/foo/SUNWspro` and the Core Platform component in `/foo/netbeans/3.5R`

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

Accessing Compilers and Tools 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`.

  If your software is not installed in the `/opt` directory, 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 and does not endorse and is not responsible or liable for any content, advertising, products, or other materials on or available from such sites or resources. Sun will not be responsible or liable for any damage or loss caused or alleged to be 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`
Online help	HTML available through the Help menu in the IDE
Release notes	HTML at `http://docs.sun.com`

Related Compilers and Tools 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
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`.

Document Title	Description
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.
Numerical Computation Guide	Describes issues regarding the numerical accuracy of floating-point computations.

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. Email your comments to Sun at this address:

docfeedback@sun.com

Please include the part number (817-5062-10) of your document in the subject line of your email.

CHAPTER 1

Sun Studio 8 New Features

Sun™ Studio 8 replaces the Sun™ ONE Studio 8, Compiler Collection. The Sun ONE Studio 8, Compiler Collection was a command-line only release of the C, C++, and Fortran compilers, libraries, and tools. Sun Studio 8 includes the same compilers and libraries released with the Sun ONE Studio 8, Compiler Collection release and adds an Integrated Development Environment (IDE) and an updated Performance Analyzer.

This chapter describes the following new Sun Studio 8 features.

- Integrated Development Environment (IDE)
- Performance Analysis Tools
- Documentation

The Sun Studio 8 release includes the following compilers, libraries, and tools that are unchanged from the Sun ONE Studio 8, Compiler Collection release. New features for the compilers, libraries, and tools released with the Sun ONE Studio 8, Compiler Collection are described in Chapter 2.

- C Compiler
- C++ Compiler
- Fortran Compiler
- `dbx` Command-Line Debugger
- Sun Performance Library™
- Interval Arithmetic

In most sections, there is a table that lists the new features of that component. The table has either two columns or three columns:

- **Two-column table.** The left-hand column provides a short description of the feature, and the right-hand column has a longer description.
- **Three-column table.** The left-hand column provides a short description of the feature, the middle column lists the relevant command or option, and the right-hand column has a longer description of the new feature.

Note – To find the Sun Studio 8 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.

Integrated Development Environment (IDE)

The Sun Studio 8 IDE provides modules for creating, editing, building, debugging, and analyzing the performance of a C, C++, or Fortran application. It includes a set of basic Java™ language support modules that can be enabled if needed for JNI (Java™ Native Interface) development.

This release of the IDE is available on the Solaris™ Operating System (Solaris OS) (SPARC™ Platform Edition) versions 7, 8, and 9; and the Solaris OS (x86 Platform Edition) versions 7, 8, and 9.

The Sun Studio 8 software consists of two major components:

- The Sun Studio component, which includes the IDE, compilers, tools, and core platform
- The Java™ 2 Platform, Standard Edition (J2SE) technology on which the Core Platform runs

The Sun Studio 8 component is installed by default in the `/opt` directory, but an alternate location can be specified during installation. The J2SE technology is installed by default in the `/usr/jdk/j2sdk1.4.2_02` directory, but an alternate directory in `/usr` can be specified during installation.

The correct operation of the IDE depends IDE being able to find the core platform, and the core platform being able to find the J2SE technology.

Starting the IDE

The command to start the IDE is `sunstudio`. For details on this command, see the `sunstudio`(1) man page. (To display the `sunstudio`(1) man page, you must have */installation-directory*`/SUNWspro/man` in your `$MANPATH`.)

Performance Analysis Tools

The following sections list the new data collection and presentation features in the Sun Studio 8, release of the performance analysis tools. For more information, see the following man pages:

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

Data collection features

The following list describes new or changed data collection capabilities.

- The Performance Analyzer can now be used as a NetBeans module, integrated into the IDE, as well as a command from the shell.
- The Performance Analyzer now has a Collector window that allows selection of and recording of experiments from the GUI.
- Data collection error detection and reporting have been enhanced.

For more information, refer to the `collect`(1), `collector`(1) and `libcollector`(3) man pages.

Data presentation features

The following list describes new or changed data presentation capabilities.

- The Performance Analyzer shows a tick mark next to the scrollbar for annotated source and disassembly, indicating high-metric lines.
- The Performance Analyzer has additional flexibility in setting colors for functions in the timeline.
- The Performance Analyzer has additional flexibility in showing clock profiling data in the timeline for CPU idle states
- `er_print` and the Performance Analyzer now can coalesce data types from different object files, so that the type reports and displays do not show repeated entries for the same structure. The `er_print` formatting of the report for an object type is improved. A new `data_layout er_print` command and corresponding Analyzer tab were implemented.

- `er_print` now has an improved help message, with the commands listed in the same order as they appear on the man page.

For more information, refer to the `analyzer`(1) and `er_print`(1) man pages and the Performance Analyzer online help.

Documentation

This section describes Sun Studio 8 documentation new features.

- Sun Studio 8 product documentation is provided in formats that are readable by assistive technologies for users with disabilities. For more information, see "Documentation in Accessible Formats" on page 10.

CHAPTER 2

Sun ONE Studio 8, Compiler Collection New Features

This chapter describes the new features of the Sun ONE Studio 8, Compiler Collection compilers and command-line tools. The primary focus of this release is significant performance and portability updates to our C, C++ and Fortran language systems; and support for a subset of C99 syntax and for OpenMP™ programs in the `dbx` command-line debugger.

The compilers, libraries, and tools described in this chapter are included with the Sun Studio 8 release

This chapter has the following sections:

In most sections, there is a table that lists the new features of that component. The table has either two columns or three columns:

- **Two-column table.** The left column provides a short description of the feature, and the right column has a longer description.
- **Three-column table.** The left column provides a short description of the feature, the middle column lists the relevant command or option, and the right column has a longer description of the new feature.

Note – To find the Sun ONE Studio 8, Compiler Collection 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 Faster Compilation
- TABLE 2-3 Improved Performance
- TABLE 2-4 Easier Debugging

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
Linker mapfiles are no longer needed for variable scoping: `-xldscope`	There are now two different ways you can control the exporting of symbols in dynamic libraries. This facility is called linker scoping and is supported by linker mapfiles. First, you can now embed new declaration specifiers in code. By embedding `__global`, `__symbolic`, and `__hidden` directly in code, you no longer need to use mapfiles. Second, you can override the default setting for variable scoping by specifying `-xldscope` at the command line.

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

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 release of the C compiler. The sub-section number of the C99 standard is listed for each item. • 6.2.5 _Bool • 6.2.5 _Complex type This release supports a partial implementation of _Complex. You must link with `-lcplxsupp` on Solaris 7, 8, and 9 Operating Systems (OS). • 6.3.2.1 Conversion of arrays to pointers not limited to lvalues • 6.4.4.2 Hexadecimal floating-point literals • 6.5.2.5 Compound literals • 6.7.2 Type specifiers • 6.10.6 STDC pragmas • 6.10.8 __STDC_IEC_559 and __STDC_IEC_559_COMPLEX macros
Support for the VIS™ Developers Kit: `-xvis` (SPARC®)	Use the `-xvis=[yes|no]` option when you are using the assembly-language templates defined in the VIS instruction set Software Developers Kit (VSDK). The VIS instruction set is an extension to the SPARC v9 instruction set. Even though the UltraSPARC processors are 64-bit, there are many cases, especially in multimedia applications, when the data are limited to 8 or 16 bits in size. The VIS instructions can process four 16-bit data with one instruction so they greatly improve the performance of applications that handle new media such as imaging, linear algebra, signal processing, audio, video and networking. For more information on the VSDK, see *VIS Instruction Set User's Manual* at `http://www.sun.com/processors/documentation.html`.
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.

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

Feature	Description
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 nonunique 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.

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

TABLE 2-2 New C Features That Support Faster Compilation

Feature	Description
Faster profiling (SPARC)	Use `-xprofile_ircache[=`*path*`]` with `-xprofile=collect|use` to improve compilation time during the use phase by reusing compilation data saved from the collect phase. With large programs, compilation time in the use phase can improve significantly because the intermediate data is saved. The saved data could increase disk space requirements considerably.
Precompiled headers: `-xpch`	This release of the compiler introduces the new precompiled-header feature. The precompiled-header file is designed to reduce compile time for applications in which source files share a common set of include files containing a large amount of source code. A precompiled header works by collecting information about a sequence of header files from one source file, and then using that information when recompiling that source file, and when compiling other source files that have the same sequence of headers. You can take advantage of this feature through the `-xpch` and `-xpchstop` options in combination with the `#pragma hdrstop` directive.
Using multiple processors: `-xjobs=`*n* (SPARC)	Specify the `-xjobs=`*n* option to set how many processes the compiler creates to complete its work. This option can reduce the build time on a multi-CPU machine. Currently, `-xjobs` works only with the `-xipo` option. When you specify `-xjobs=`*n*, the interprocedural optimizer uses *n* as the maximum number of code generator instances it can invoke to compile different files.

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

TABLE 2-3 New C Features That Support Improved Performance

Feature	Description
Improving runtime with linker supported thread-local storage: `-xthreadvar`	Use the new linker supported thread-local storage facility of the compiler to do the following: • Utilize a fast implementation for the POSIX interfaces for allocating thread-specific data. • Convert multi-process programs to multi-thread programs. • Port Windows applications using thread-local storage to Solaris. • Utilize a fast implementation for the threadprivate variables in OpenMP programs. Thread-local storage is now available in the compiler through the declaration of thread-local variables. The declaration consists of a normal variable declaration with the addition of the variable specifier `__thread` and the command line option `-xthreadvar`.
Improving runtime by reducing page faults: `-xF`	Use the new functionality of `-xF` to enable the optimal reordering of variables and functions by the linker. This can help solve the following problems which negatively impact run-time performance: • Cache and page contention caused by unrelated variables that are near each other in memory. • Unnecessarily large work-set size as a result of related variables which are not near each other in memory. • Unnecessarily large work-set size as a result of unused copies of weak variables that decrease the effective data density.

TABLE 2-3 New C Features That Support Improved Performance *(Continued)*

Feature	Description
Improving runtime: `-xlinkopt` (SPARC)	The C++ compiler can now perform link time optimization on relocatable object files when you specify the `-xlinkopt` command. Specify `-xlinkopt` and the compiler performs some additional optimizations at link time without modifying the .o files that are linked. The optimizations appear only in the executable program. The `-xlinkopt` option is most effective when you use it to compile the whole program and with profile feedback.
Improving runtime: `-xpagesize=`*n* (SPARC)	Set the page size in memory for the stack. *n* can be `8K`, `64K`, `512K`, `4M`, `32M`, `256M`, `2G`, `16G`, or `default`. You must specify a valid page size for the Solaris OS on the target platform, as returned by `getpagesize`(3C). If you do not specify a valid page size, the request is silently ignored at run-time. You can use `pmap`(1) or `meminfo`(2) to determine page size at the target platform. Note that this feature is only available on Solaris 9 OS. A program compiled with this option does not link in earlier Solaris OS environments. This option is a macro for `-xpagesize_stack` and `-xpagesize_heap`.
Hardware counter-based profiling: `-xhwcprof` (SPARC)	Use the `-xhwcprof=[enable|disable]` option to enable compiler support for hardware counter-based profiling. When `-xhwcprof` is enabled, the compiler generates information that helps tools match hardware counter data reference and miss events with associated instructions. Corresponding data-types and structure-members can also be identified in conjunction with symbolic information (produced with -g). This information can be useful in performance analysis because it is not easily identified from profiles based on code addresses, source statements, or routines.

TABLE 2-4 lists the new features of the C compiler that support easier debugging

TABLE 2-4 New C Features That Support Easier Debugging

Feature	Description
DWARF-format debugger information: `-xdebugformat`	The C compiler is migrating the format of debugging information from the stabs format to the DWARF format as specified in *DWARF Debugging Information Format*. If you maintain software that reads debugging information, you now have the option to transition your tools from the stabs format to the DWARF format. The default setting for this release is `-xdebugformat=stabs`. Use the `-xdebugformat=dwarf` option as a way of accessing the new format for the purpose of porting tools. There is no need to use this option unless you maintain software that reads debugging information, or unless a specific tool tells you that it requires debugging information in one of these formats.
Support for debugging OpenMP programs: `-xopenmp=noopt`	If you are debugging an OpenMP program with `dbx`, compile with `-g` and `-xopenmp=noopt` so you can breakpoint within parallel regions and display the contents of variables.

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 Faster Compilation
- TABLE 2-7 Easier Porting
- TABLE 2-8 Improved Performance
- TABLE 2-9 Added Warning and Error Controls

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 5.5).

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

Feature	Description
Template cache no longer needed: `-instances`	This release of the C++ compiler improves template instantiation significantly. Programs that use the default template instantiation model are no longer restricted from building more than one program in a directory. Most programs that rely on an alternate instantiation model, with `-instances=static`, can now use the new default instantiation model. The improvements and changes to template instantiation can either improve compile time by avoiding a template cache or reduce executable size by avoiding duplicate static functions.
Linker mapfiles no longer needed for variable scoping: `-xldscope`	There are now two different ways you can control the exporting of symbols in dynamic libraries. This facility is called linker scoping and has been supported by linker mapfiles for some time. First, you can now embed new declaration specifiers in code. By embedding `__global`, `__symbolic`, and `__hidden` directly in code, you no longer need to use mapfiles. Second, you can override the default setting for variable scoping by specifying `-xldscope` at the command line.
Powerful new diagnostics for macros: `-xdumpmacros`	This release introduces two new pragmas and a new compiler option designed to help you track the behavior of macros in your application. This includes macros defined in system headers. You can use the `-xdumpmacros` option at the command line to see the macro definitions and also to see where macros are defined, undefined, and used in your program. To narrow your focus, use the new `dumpmacros` and `end_dumpmacros` pragmas directly in the source.
Support for VIS Developers Kit: `-xvis`	Use the `-xvis=[yes|no]` option when you are using the assembly-language templates defined in the VIS instruction set Software Developers Kit (VSDK). The default is `-xvis=no`. For more information on the VSDK, see `http://www.sun.com/processors/vis`
Support for C99 runtime libraries and environment: `-xlang`	On operating systems that support the C99 standard (ISO/IEC 9899:1999, Programming Language - C), `-xlang=c99` specifies C99 runtime behavior for C and C++ code that invokes C library functions. Some C99 behavior, like the C complex type, depends on the use of the `-xc99=%all` option with the C compiler, and some behavior, like `printf`, does not. C99 support is not available in compat mode `(-compat=4)`.

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

Feature	Description
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"..." 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++.
Expanded support for OpenMP™: `-xopenmp`	The C++ compiler continues its implementation of the OpenMP interface for explicit parallelization. See the `CC`(1) man page for specific details of the `-xopenmp` option. The compiler has expanded OpenMP functionality to allow the following: • Class objects are permitted in OpenMP data clauses. • OpenMP pragmas are permitted in class member functions.
Improved `-xprofile`	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.

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

TABLE 2-6 New C++ Features That Support Faster Compilation

Features	Description
Speeding up syntax checking: `-xe`	When you specify `-xe`, the compiler checks only for syntax and semantic errors and does not produce any object code. Use the `-xe` option if you do not need the object files produced by compilation. For example, if you are trying to isolate the cause of an error message by deleting sections of code, you can speed the edit and compile cycle by using `-xe`.
Faster profiling: `-xprofile_ircache`	Use `-xprofile_ircache[=`*path*`]` with `-xprofile=collect|use` to improve compilation time during the use phase by reusing compilation data saved from the collect phase. With large programs, compilation time in the use phase can improve significantly because the intermediate data is saved. The saved data could increase disk space requirements considerably.
Stopping redundant template instantiations: `-instlib=`*filename*	Use `-instlib=`*filename* to inhibit the generation of template instances that are duplicated in a library and the current object. In general, if your program shares large numbers of instances with libraries, try `-instlib=`*filename* and see whether or not compilation time improves. Use the *filename* argument to specify the library that you know contains the existing template instances. The filename argument must contain a forward slash '/' character. For paths relative to the current directory, use dot-slash './'. The `-instlib=`*filename* option has no default and is only used if you specify it. This option can be specified multiple times and accumulates.

TABLE 2-6 New C++ Features That Support Faster Compilation *(Continued)*

Features	Description
Generating functions: `-template=geninlinefuncs`	Usually, the C++ compiler does not generate an inline template function unless the function is called and cannot be inlined. However, you can specify `-template=geninlinefuncs` and the compiler instantiates inline member functions of the explicitly instantiated class template which were not generated previously. Linkage for these functions is local in all cases.
Precompiled headers: `-xpch`	This release of the compiler introduces the new precompiled-header feature. The precompiled-header file is designed to reduce compile time for applications in which source files share a common set of include files containing a large amount of source code. A precompiled header works by collecting information about a sequence of header files from one source file and then using that information when recompiling that source file and when compiling other source files that have the same sequence of headers. You can take advantage of this feature through the `-xpch` and `-xpchstop` options in combination with the `#pragma hdrstop` directive.
Using multiple processors: `-xjobs=`*n*	Specify the `-xjobs=`*n* option to set how many processes the compiler creates to complete its work. This option can reduce the build time on a multi-CPU machine. Currently, `-xjobs` works only with the `-xipo` option. When you specify `-xjobs=`*n*, the interprocedural optimizer uses n as the maximum number of code generator instances it can invoke to compile different files.

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

TABLE 2-7 New C++ Features That Support Easier Porting

Feature	Description
Simplified porting: `-xmemalign`	Use the `-xmemalign` option to control the assumptions the compiler makes about the alignment of data. By controlling the code generated for potentially misaligned memory accesses and by controlling program behavior in the event of a misaligned access, you can more easily port your code to the Solaris Operating System (OS). The `-xmemalign` option is also used to improve performance for data that is aligned more than necessary and to access structures that are packed more than normal.
Setting the sign of char: `-xchar`	The `-xchar[={signed|s|unsigned|u}]` option is provided solely for the purpose of easing the migration of code from systems where the char type is defined as unsigned. Do not use this option unless you are migrating from such a system. Only code that relies on the sign of a char type needs to be rewritten to explicitly specify signed or unsigned.
Debugging ported code: `-xport64`	Use the new `-xport64` option to help you port code to a 64-bit environment. Specifically, this option warns against problems such as truncation of values (including pointers), sign extension, and changes to bit-packing that are common when you port code from a 32-bit architecture such as V7 (ILP32) to a 64-bit architecture such as V9 (LP64). An additional option, `-xnocastwarn`, is also now available to disable truncation warnings in 64-bit compilation mode when an explicit cast is the cause of data truncation.

TABLE 2-8 lists the new features of the C++ compiler that support improved performance:

TABLE 2-8 New C++ Features That Support Improved Performance

Feature	Description
Linker supported thread-local storage of data: `-xthreadvar` (SPARC)	Use the new linker supported thread-local storage facility of the compiler to do the following: • Utilize a fast implementation for the POSIX interfaces for allocating thread-specific data. • Convert multi-process programs to multi-thread programs. • Port Windows applications using thread-local storage to Solaris. • Utilize a fast implementation for the threadprivate variables in OpenMP. Thread-local storage is now available in the compiler through the declaration of thread-local variables. The declaration consists of a normal variable declaration with the addition of the variable specifier `__thread` and the command line option `-xthreadvar`.
Reducing page faults: `-xF`	Use the new functionality of `-xF` to enable the optimal reordering of variables and functions by the linker. This can help solve the following problems that negatively impact runtime performance: • Cache and page contention caused by unrelated variables that are near each other in memory. • Unnecessarily large work-set size as a result of related variables which are not near each other in memory. • Unnecessarily large work-set size as a result of unused copies of weak variables that decrease the effective data density.

TABLE 2-8 New C++ Features That Support Improved Performance *(Continued)*

Feature	Description
New pragmas	The C++ compiler now supports four new pragmas that you can use to help improve the optimization of your code. See the *C++ User's Guide* for complete descriptions of these pragmas: • `#pragma does_not_read_global_data` • `#pragma does_not_return` • `#pragma does_not_write_global_data` • `#pragma rarely_called`
Improving runtime: `-xlinkopt`	The C++ compiler can now perform link-time optimization on relocatable object files when you specify the `-xlinkopt` option. See the `CC`(1) man page. Specify `-xlinkopt` and the compiler performs some additional optimizations at link time without modifying the `.o` files that are linked. The optimizations appear only in the executable program. The `-xlinkopt` option is most effective when you use it to compile the whole program and with profile feedback.
Improving runtime: `-xpagesize=`*n*	Use the `-xpagesize=`*n* option to set the preferred page size for the stack and the heap. *n* can be 8K, 64K, 512K, 4M, 32M, 256M, 2G, 16G, or default. You must specify a valid page size for the Solaris Operating Environment on the target platform, as returned by `getpagesize`(3C). If you do not specify a valid page size, the request is silently ignored at runtime. You can use `pmap`(1) or `meminfo`(2) to determine page size of the target platform. This feature is only available on the Solaris 9 OS. A program compiled with this option do not link in earlier Solaris OS environments.

TABLE 2-9 lists the newly added error and warning controls of the C++ compiler:

TABLE 2-9 Newly Added Error and Warning Controls of the C++ Compiler

Feature	Description
Filtering warning messages: `-erroff`	You can now use the new `-erroff` option to suppress warning messages from the compiler front-end. Neither error messages nor messages from the driver are affected. You can also use `-erroff` to single out a particular warning message so that either it alone is suppressed or it alone is issued.
Aborting compilation: `-errtags`, `-errwarn`	You can now use the `-errtags` compiler option and the `-errwarn` compiler option to stop compilation if the compiler issues a particular warning. Set `-errtags=yes` to find the tag for a particular warning, and then specify `-errwarn=tag` where tag is the unique identifier returned by `-errtags` for a particular warning message. You can also abort compilation if any warning is issued by specifying `-errwarn=%all`. See also `-xwe` in the `CC`(1) man page.
Improved filtering for standard-library names: `-filt=[no%]stdlib`	The `-filt=[no%]stdlib` option is set by default and simplifies names from the standard library in both the linker and compiler error messages. This makes it easier for you to recognize the name of standard-library functions. Specify `-filt=no%stdlib` to disable this filtering.

Fortran Compiler

The Sun ONE Studio 8, Compiler Collection release provides a Fortran 95 compiler, `f95`, with compatibility support for legacy Fortran 77 programs. See the chapter "FORTRAN 77 Compatibility: Migrating to Fortran 95" in the *Fortran User's Guide* for details on porting legacy Fortran 77 programs to the Fortran 95 compiler.

TABLE 2-10 lists the new features of the Fortran 95 compiler. See the *Fortran User's Guide*, *Fortran Programming Guide*, and the *Fortran Library Reference* for details.

TABLE 2-10 Fortran 95 Compiler New Features

Feature	Option	Description
Fortran 2000 features		The following features appearing in the Fortran 2000 draft standard, which can be found in PDF format at `http://www.dkuug.dk/jtc1/sc22/open/n3501.pdf`, have been implemented in this release of Fortran 95 compiler: • Exceptions and IEEE Arithmetic • Interoperability with C • `PROTECTED` Attribute • `ASYNCHRONOUS` I/O Specifier
Enhanced compatibility with legacy f77		A number of new features enhance the Fortran 95 compiler's compatibility with legacy Fortran 77 compiler, f77. These include: • Variable format expressions (VFE's) • Long identifiers • `-arg=loc` • `-vax` compiler option

TABLE 2-10 Fortran 95 Compiler New Features *(Continued)*

Feature	Option	Description
I/O error handlers		Two new functions enable you to specify your own error handling routine for formatted input on a logical unit. When a formatting error is detected, the runtime I/O library calls the specified user-supplied handler routine with data pointing at the character in the input line causing the error. The handler routine can supply a new character and allow the I/O operation to continue at the point where the error was detected using the new character; or take the default Fortran error handling. The new routines, `SET_IO_ERR_HANDLER`(3f) and `GET_IO_ERR_HANDLER` (3f), are module subroutines and require `USE SUN_IO_HANDLERS` in the routine that calls them. See the man pages for these routines for details.
Unsigned integers		With this release, the Fortran 95 compiler accepts a new data type, `UNSIGNED`, as an extension to the language. Four `KIND` parameter values are accepted with `UNSIGNED`: 1, 2, 4, and 8, corresponding to 1-, 2, 4-, and 8-byte unsigned integers, respectively. The form of an unsigned integer constant is a digit-string followed by the upper or lower case letter U, optionally followed by an underscore and `KIND` parameter.

TABLE 2-10 Fortran 95 Compiler New Features *(Continued)*

Feature	Option	Description
Preferred stack/heap page size	`-xpagesize`	A new compiler option, `-xpagesize`, enables the running program to set the preferred stack and heap page size at program startup. For example, `-xpagesize=4M` sets the preferred Solaris 9 Operating System (OS) stack and heap page sizes to 4 megabytes. Choose from a set of preset values. Stack or heap page sizes can be set individually with `-xpagesize_stack` and `-xpagesize_heap`. This feature is only available on Solaris 9 OS. A program compiled with this flag fails to link in earlier Solaris OS environments.)
Faster profiling	`xprofile_ircache=`*path*	This release introduces the new command-line option `-xprofile_ircache=`*path*, to speed up the use compilation phase during profile feedback. With this flag specified, the compiler saves intermediate data on path during the collect compilation phase, `-xprofile=collect`, for reuse later during the `-xprofile=use` phase, eliminating the need to regenerate this information. For large programs this could amount to a significant savings in compile time in the `-xprofile=use` phase.
Enhanced "known libraries"	`-xknown_lib`	The `-xknown_lib` option has been enhanced to include more routines from the Basic Linear Algebra Subprograms library, BLAS, and introduces three sub-options. The compiler recognizes calls to select BLAS library routines and is free to optimize appropriately for the Sun Performance Library implementation.

TABLE 2-10 Fortran 95 Compiler New Features *(Continued)*

Feature	Option	Description
Link-time optimization	`-xlinkopt`	Compile and link with the new `-xlinkopt` flag to invoke a post-optimizer to apply a number of advanced performance optimizations on the generated binary object code at link time. This option is most effective when used to compile the whole program with profile feedback.
Initialization of local variables	`-xcheck=init_local`	A new extension to the `-xcheck` option flag enables special initialization of local variables. Compiling with `-xcheck=init_local` initializes local variables to a value that is likely to cause an arithmetic exception if it is used before it is assigned by the program. Memory allocated by the `ALLOCATE` statement will also be initialized in this manner. `SAVE` variables, module variables, and variables in `COMMON` blocks are not initialized.
Enhanced `-openmp` option	`-openmp`	The `-openmp` option flag has been enhanced to facilitate debugging OpenMP programs. To use `dbx` to debug your OpenMP application, compile with `-openmp=noopt -g` You will then be able to use dbx to breakpoint within parallel regions and display contents of variables.
Multi-process compilation	`-xjobs=`*n*	Specify `-xjobs=`*n* with `-xipo` and the interprocedural optimizer will invoke at most *n* code generator instances to compile the files listed on the command line. This option can greatly reduce the build time of large applications on a multi-CPU machine.

TABLE 2-10 Fortran 95 Compiler New Features *(Continued)*

Feature	Option	Description
Making assertions with `PRAGMA ASSUME`	`-xassume_control`	The `ASSUME` pragma is a new feature in this release of the compiler. This pragma gives hints to the compiler about conditions the programmer knows are true at some point in a procedure. This might help the compiler to do a better job optimizing the code. The programmer can also use the assertions to check the validity of the program during execution. The new -xassume_control flag determines how the `ASSUME` pragmas are processed.
OpenMP support for explicitly threaded programs.	-xopenmp	The implementation of the OpenMP API in this release supports programs that are explicitly threaded.

dbx Command-Line Debugger

TABLE 2-11 lists the new features in this release of the dbx command-line debugger. For more information about these features, see the *Debugging a Program With* dbx manual.

TABLE 2-11 dbx New Features

Feature	Description
Debug programs with mixed-language code	dbx now supports the following C99 language types: • complex • imaginary • double complex • double imaginary • long double complex • long double imaginary You can print the values of variables and expressions involving these types.
Support for debugging OpenMP programs	dbx now supports debugging of OpenMP programs in Fortran 95, C+++, and C. dbx can display threads, stacks, functions, parameters, and variables correctly in the presence of OpenMP code generated by the Fortran 95 compiler, the C++ compiler, and the C compiler.
New `-stop` option for `detach` command	The `detach -stop` command detaches dbx from the target program and leaves the process in a stopped state. The `-stop` option allows temporary application of other `/proc`-based debugging tools that may be blocked due to exclusive access.
New `-resumeone` event modifier	The new `-resumeone` modifier for event handlers helps with conditions with function calls in multi-threaded programs.

Interval Arithmetic

There are no new interval arithmetic features in this compiler collection 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 2-12 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 2-12 Sun Performance Library New Features

Feature	Description
Performance improvements	This release of Sun Performance Library includes the following performance improvements. • BLAS and FFT Performance Improvements: Improved GEMM performance of small problem sizes for US-III, and improved FFT performance of small problem sizes when using 32-bit FFT routines in V9 libraries • Sparse Solver Performance Improvements: Enhanced single-CPU performance of Sun Performance Library sparse solver, and parallelized Sun Performance Library sparse solver • Sparse BLAS Performance Improvements: Parallelized sparse matrix-vector operations, and improved performance of small problem sizes
Portable library performance	Internal changes that simplify getting optimal performance have been made to this release of the Sun Performance Library. At runtime, a version of Sun Performance Library optimized for the SPARC hardware platform that the executable is being run on, is dynamically loaded. This only occurs when the shared library versions of Sun Performance Library are linked, which is the default.

TABLE 2-12 Sun Performance Library New Features *(Continued)*

Feature	Description
Sparse solver new features	The sparse solver now includes Hermitian positive definite matrix support.
Combined parallelization models	This release of the Sun Performance Library includes combined parallelization models, which reduces the number of libraries shipped with the Sun Performance Libraries and reduces the size of the Sun Performance Library. Combining the parallelization models simplifies linking for serial or parallel behavior from Sun Performance Library.
Interval BLAS man pages moved to `man3pi` folder	The Interval BLAS man pages have been moved to the `man3pi` folder. For information on the Fortran 95 interfaces and types of arguments used in each Interval BLAS routine, see the section `3pi` man pages for the individual routines. For example, to display the man page for the `constructv_i.3pi` routine, type `man -s 3pi constructv_i`. Routine names must be lowercase.

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 `man dmake` for additional details.

TABLE 2-13 `dmake` New Features

Feature	Description
`dmake` memory usage reduced	While results depend on many factors, memory heap usage has been reduced by 50% to 60%.
Increased consistency	`dmake` now consistent with Solaris make
`dmake` now automatically adjusts the limit of parallel jobs to prevent overloading	The environment variable `DMAKE_ADJUST_MAX_JOBS` can be set to automatically adjust the limit of parallel jobs to prevent overloading. • If set to `YES`, `dmake` adjusts the limit of parallel jobs according to the current loading of the system. If the system is not overloaded, `dmake` uses the limit defined by the user. If the system is overloaded, `dmake` sets the current limit lower than the limit defined by the user. If this variable is not set, `dmake` adjusts the limit of parallel jobs according to the current loading of the system. This setting is the `dmake` default. • `NO` Causes `dmake` to switch off the autoadjustment mechanism.

Performance Analysis Tools

TABLE 2-14 lists the new data collection and presentation features in the Sun ONE Studio 8, Compiler Collection release of the performance analysis tools. For more information, see the following man pages:

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

TABLE 2-14 Performance Analysis Tools New Features

Feature	Description
Support for clock-based profiling and hardware-counter overflow profiling for Java™ programs	The Java programming language is now fully supported for clock-based profiling and hardware-counter overflow profiling, as well as synchronization-delay tracing, and memory allocation tracing. Data is collected for both a machine representation of the target, and the Java representation. A Java API to `libcollector` is provided.
Clock-based profiling	Clock-based profiling is no longer restricted to multiples of the system clock resolution for versions of the Solaris Operating Environment that support a higher resolution clock. The range of profiling intervals supported is returned when you use the `collect` command with no arguments.
Archiving of loadobjects	Archiving of loadobjects can be controlled using the `-A` option of the `collect` command or the `dbx collector archive` command.
Application programming interface for pausing and resuming data collection	An API for pausing and resuming data collection for individual threads has been provided.
Hardware counters that count memory access events	For hardware counters that count memory access events, prefixing the counter name with "+" activates a search by the Collector for the program counter and virtual address that triggered the event.
Filtering by CPU	Filtering by CPU has been added to the Performance Analyzer and the `er_print` utility. This capability is not available for versions of the Solaris Operating Environment earlier than 9. The capability is implemented in the `er_print` commands `cpu_select` and `cpu_list`.

TABLE 2-14 Performance Analysis Tools New Features *(Continued)*

Feature	Description
Display lists	The performance tools display lists are ordered by metric value for source lines and program counters. These lists are shown in the Lines tab and PCs tab of the Performance Analyzer, and are generated by the `er_print` utility using the `lines` command and the `pcs` commands. The Summary tab in the Performance Analyzer displays all the metrics for the selected source line. The summary panels for lines and for program counters can be displayed in `er_print` utility with the `lsummary` command and the `psummary` command.
Timeline tab	The Timeline Options dialog box has been merged with the Set Data Presentation dialog box as a Timeline tab. The Timeline tab can display data bars for LWPs, for threads, or for CPUs. Use the Timeline tab of the Set Data Presentation dialog box to choose to display data for one of these three. The call stacks in the Timeline tab can be aligned on the root function or on the leaf function, and the number of visible frames can be set. Selections are made using the Timeline tab of the Set Data Presentation dialog box.
Selection of an object	Selection of an object has been extended to include source lines and program counters as well as functions. The selected object is displayed in the menu bar, and its metrics are displayed in the Summary tab. The selected object is displayed when you navigate from the current tab to another tab. In particular, switching to source or disassembly will position on the line or instruction selected, rather than always positioning on the first line or instruction of the selected function.
Experiments on descendant processes	Experiments on descendant processes are automatically loaded when the experiment for the founder process is loaded, but the display of their data is disabled. You must use the Filter Data dialog box in the Performance Analyzer or the `experiment_select` command in the `er_print` utility to enable data display for descendant experiments.
Leaklist tab	The Leaklist tab shows leak and allocation data graphically, and allows navigation between the call stack of a leak or allocation and the source or disassembly of the function on the call stack.
Java mode	Experiments on applications written in the Java programming language might be presented with Java mode set to `on`, `expert`, or `off`.

Documentation

This section describes Sun ONE Studio 8, Compiler Collection documentation new features.

- The *Debugging a Program With* `dbx` manual has a new chapter entitled "Debugging OpenMP Applications" The chapter describes how you can use the `dbx` command-line debugger to debug applications that use the OpenMP interface for explicit parallelization.
- Previous releases of the *FORTRAN 77 Language Reference* are available through the `docs.sun.com` web site. This manual has not been updated for this release.
- Sun ONE Studio 8, Compiler Collection product documentation is provided in formats that are readable by assistive technologies for users with disabilities. For more information, see "Documentation in Accessible Formats" on page 10.