Sun Studio 9 の新機能

Sun™ Studio 9

Sun Microsystems, Inc.
www.sun.com

Part No. 817-7889-10
2004 年 7 月, Revision A

原典： What's New Sun Studio 9
Part No: 817-6691-10
Revision A

目次

はじめに

本書では、この Sun™ Studio 9 ソフトウェアリリースの、C や C++、Fortran コンパイラ、ライブラリ、ツールなどの新機能を説明しています。

書体と記号について

次の表と記述は、このマニュアルで使用している書体と記号について説明しています。

書体または記号	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コーディング例。	.login ファイルを編集します。 ls -a を使用してすべてのファイルを表示します。 machine_name% You have mail.
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して表わします。	machine_name% **su** Password:
AaBbCc123 または **ゴシック**	コマンド行の可変部分。実際の名前または実際の値と置き換えてください。	rm *filename* と入力します。 rm **ファイル名** と入力します。

書体または記号	意味	例
『 』	参照する書名を示します。	『SPARCstorage Array ユーザーマニュアル』
「 」	参照する章、節、または、強調する語を示します。	第 6 章「データの管理」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合、バックスラッシュは、継続を示します。	`machinename% grep '^#define \ XV_VERSION_STRING'`
➤	階層メニューのサブメニューを選択することを示します。	作成: 「返信」➤「送信者へ」

シェルプロンプトについて

シェル	プロンプト
UNIX の C シェル	machine_name%
UNIX の Bourne シェルと Korn シェル	machine_name$
スーパーユーザー (シェルの種類を問わない)	#

Sun Studio ソフトウェアおよびマニュアルページへのアクセス

コンパイラおよびツールは、標準の `/usr/bin/` および `/usr/share/man` の各ディレクトリにはインストールされません。コンパイラあるいはツールにアクセスするには、`PATH` 環境変数を正しく設定しておく必要があります (ix ページの「コンパイラとツールのマニュアルへのアクセス」を参照)。また、マニュアルページにアクセスするには、`MANPATH` 環境変数を正しく設定しておく必要があります (viii ページの「マニュアルページへのアクセス方法」を参照)。

PATH 変数についての詳細は、csh(1)、sh(1)、および ksh(1) のマニュアルページを参照してください。MANPATH 変数についての詳細は、man(1) のマニュアルページを参照してください。このリリースにアクセスするために PATH および MANPATH 変数を設定する方法の詳細は、『インストールガイド』を参照するか、システム管理者にお問い合わせください。

注 – この節に記載されている情報は Sun Studio のコンパイラおよびツールが /opt ディレクトリにインストールされていることを想定しています。製品ソフトウェアが /opt 以外のディレクトリにインストールされている場合は、システム管理者に実際のパスをお尋ねください。

コンパイラとツールへのアクセス方法

PATH 環境変数を変更して、コンパイラとツールにアクセスできるようにする必要があるかどうか判断するには以下を実行します。

▼ PATH 環境変数を設定する必要があるかどうか判断する

1. 次のように入力して、PATH 変数の現在値を表示します。

```
% echo $PATH
```

2. 出力内容から /opt/SUNWspro/bin を含むパスの文字列を検索します。

   パスがある場合は、PATH 変数はコンパイラとツールにアクセスできるように設定されています。パスがない場合は、次の指示に従って、PATH 環境変数を設定してください。

▼ PATH 環境変数を設定してコンパイラとツールにアクセスする

1. C シェルを使用している場合は、ホームの .cshrc ファイルを編集します。Bourne シェルまたは Korn シェルを使用している場合は、ホームの .profile ファイルを編集します。
2. 次のパスを PATH 環境変数に追加します。Sun ONE Studio ソフトウェアまたは Forte Developer ソフトウェアをインストールしている場合は、インストール先へのパスの前に、次のパスを追加します。

   /opt/SUNWspro/bin

マニュアルページへのアクセス方法

マニュアルページにアクセスするために MANPATH 変数を変更する必要があるかどうかを判断するには以下を実行します。

▼ MANPATH 環境変数を設定する必要があるかどうか判断する

1. 次のように入力して、dbx マニュアルページを表示します。

```
% man dbx
```

2. 出力された場合、内容を確認します。

   dbx(1) マニュアルページが見つからないか、表示されたマニュアルページがインストールされたソフトウェアの現バージョンのものと異なる場合は、この節の指示に従って MANPATH 環境変数を設定してください。

▼ MANPATH 変数を設定してマニュアルページにアクセスする

1. C シェルを使用している場合は、ホームの .cshrc ファイルを編集します。Bourne シェルまたは Korn シェルを使用している場合は、ホームの .profile ファイルを編集します。
2. 次のパスを MANPATH 環境変数に追加します。

   /opt/SUNWspro/man

統合開発環境へのアクセス方法

Sun Studio 9 統合開発環境 (IDE) には、C や C++、Fortran アプリケーションを作成、編集、構築、デバッグ、パフォーマンス解析するためのモジュールが用意されています。

IDE には、Sun Studio 9 のコアプラットフォームコンポーネントが必要です。コアプラットフォームコンポーネントを以下のどちらの場所にもインストールまたはマウントしていない場合は、インストールされているかかマウントされている場所 (*installation_directory*/netbeans/3.5R) を SPRO_NETBEANS_HOME 環境変数に設定する必要があります。

- デフォルトのインストールディレクトリの /opt/netbeans/3.5R
- Sun Studio 9 のコンパイラとツールコンポーネントと同じ場所 (たとえば、コンパイラとツールコンポーネントが /foo/SUNWspro にインストールされていて、コアプラットフォームコンポーネントが /foo/netbeans/3.5R にインストールされている)

IDE を起動するコマンドは、sunstudio です。このコマンドの詳細は、sunstudio(1) のマニュアルページを参照してください。

コンパイラとツールのマニュアルへのアクセス

マニュアルには、以下からアクセスできます。

- 製品マニュアルは、ご使用のローカルシステムまたはネットワークの製品にインストールされているマニュアルの索引から入手できます。
  file:/opt/SUNWspro/docs/ja/index.html

  製品ソフトウェアが /opt 以外のディレクトリにインストールされている場合は、システム管理者に実際のパスをお尋ねください。
- マニュアルは、docs.sun.com の Web サイトで入手できます。以下に示すマニュアルは、インストールされている製品のマニュアルの索引から入手できます (docs.sun.com Web サイトでは入手できません)。
  - 『Standard C++ Library Class Reference』
  - 『標準 C++ ライブラリ・ユーザーズガイド』
  - 『Tools.h++ クラスライブラリ・リファレンスマニュアル』
  - 『Tools.h++ ユーザーズガイド』
- リリースノートは、docs.sun.com で入手できます。
- IDE の全コンポーネントのオンラインヘルプは、IDE 内の「ヘルプ」メニューだけでなく、多くのウィンドウおよびダイアログにある「ヘルプ」ボタンを使ってアクセスできます。

  インターネットの docs.sun.com Web サイト (http://docs.sun.com) から、サンのマニュアルを参照したり、印刷したり、購入することができます。マニュアルが見つからない場合はローカルシステムまたはネットワークの製品とともにインストールされているマニュアルの索引を参照してください。

注 – Sun では、本マニュアルに掲載した第三者の Web サイトのご利用に関しましては責任はなく、保証するものでもありません。また、これらのサイトあるいはリソースに関する、あるいはこれらのサイト、リソースから利用可能であるコンテンツ、広告、製品、あるいは資料に関して一切の責任を負いません。Sun は、これらのサイトあるいはリソースに関する、あるいはこれらのサイトから利用可能であるコンテンツ、製品、サービスのご利用あるいは信頼によって、あるいはそれに関連して発生するいかなる損害、損失、申し立てに対する一切の責任を負いません。

アクセシブルな製品マニュアル

マニュアルは、技術的な補足をすることで、ご不自由なユーザーの方々にとって読みやすい形式のマニュアルを提供しております。アクセシブルなマニュアルは以下の表に示す場所から参照することができます。製品ソフトウェアが /opt 以外のディレクトリにインストールされている場合は、システム管理者に実際のパスをお尋ねください。

マニュアルの種類	アクセシブルな形式と格納場所
マニュアル (サードパーティ製マニュアルは除く)	形式 : HTML (日本語版は PDF のみ) 場所 : http://docs.sun.com
サードパーティ製マニュアル • 『Standard C++ Library Class Reference』 • 『標準 C++ ライブラリ・ユーザーズガイド』 • 『Tools.h++ クラスライブラリ・リファレンスマニュアル』 • 『Tools.h++ ユーザーズガイド』	形式 : HTML 場所 : file:/opt/SUNWspro/docs/ja/index.html のマニュアル索引
Readme およびマニュアルページ	形式 : HTML 場所 : file:/opt/SUNWspro/docs/ja/index.html のマニュアル索引
オンラインヘルプ	形式 : HTML 場所 : IDE 内の「ヘルプ」メニュー
リリースノート	形式 : HTML 場所 : http://docs.sun.com

コンパイラとツールに関する関連マニュアル

以下の表は、file:/opt/SUNWspro/docs/ja/index.html および http://docs.sun.com から参照できるマニュアルの一覧です。製品ソフトウェアが /opt 以外のディレクトリにインストールされている場合は、システム管理者に実際のパスをお尋ねください。

マニュアルタイトル	内容の説明
Fortran プログラミングガイド	入出力、ライブラリ、パフォーマンス、デバッグ、並列処理などに関する、Solaris™ 環境における効果的な Fortran コードの書き方について説明しています。
Fortran ライブラリ・リファレンス	Fortran ライブラリと組み込みルーチンについて詳しく説明しています。
Fortran ユーザーズガイド	f95 コンパイラのコンパイル時環境とコマンド行オプションについて説明しています。従来の f77 のプログラムを f95 に移行するためのガイドラインも記載されています。
C ユーザーズガイド	cc コンパイラのコンパイル時環境とコマンド行オプションについて説明しています。
C++ ユーザーズガイド	CC コンパイラのコンパイル時環境とコマンド行オプションについて説明しています。
数値計算ガイド	浮動小数点演算における数値の精度に関する問題について説明しています。

関連する Solaris マニュアル

次の表では、docs.sun.com の Web サイトで参照できる関連マニュアルについて説明します。

マニュアルコレクション	マニュアルタイトル	内容の説明
Solaris Reference Manual Collection	マニュアルページの節を参照。	Solaris™ のオペレーティング環境に関する情報を提供しています。
Solaris Software Developer Collection	リンカーとライブラリ	Solaris™ のリンクエディタと実行時リンカーの操作について説明しています。
Solaris Software Developer Collection	マルチスレッドのプログラミング	POSIX と Solaris™ スレッド API、同期オブジェクトのプログラミング、マルチスレッド化したプログラムのコンパイル、およびマルチスレッド化したプログラムのツール検索について説明します。

開発者向けのリソース

`http://developers.sun.com/prodtech/cc` にアクセスし、Compiler Collection というリンクをクリックして、以下のようなリソースを利用できます。リソースは頻繁に更新されます。

- プログラミング技術と最適な演習に関する技術文書
- プログラミングに関する簡単なヒントを集めた知識ベース
- コンパイラとツールのコンポーネントのマニュアル、ソフトウェアと共にインストールされるマニュアルの訂正
- サポートレベルに関する情報
- ユーザーフォーラム
- ダウンロード可能なサンプルコード
- 新しい技術の紹介

`http://www.sun.co.jp/developers/` でも開発者向けのリソースが提供されています。

技術サポートへの問い合わせ

製品についての技術的なご質問がございましたら、以下のサイトからお問い合わせください (このマニュアルで回答されていないものに限ります)。

http://sun.co.jp/service/contacting

Sun Studio 9 の新機能と機能強化

Sun™ Studio 9 は、Sun™ Studio 8 の後継となる製品です。Sun Studio 9 リリースでは、新機能として、次のコンパイラ、ライブラリ、ツールに対するアップデートが含まれています。

- C コンパイラ
- C++ コンパイラ
- Fortran コンパイラ
- Sun Performance Library
- 分散 make ユーティリティ (`dmake`)
- dbx コマンド行デバッガ
- パフォーマンス解析ツール
- 統合開発環境 (IDE)
- マニュアル類

コンポーネントの新機能を示す表が、ほぼすべての節に掲載されています。表は 2 つの欄で構成され、左の欄が新機能の簡単な説明、右の欄がその詳しい内容です。

注 – この章で紹介している Sun Studio 9 のマニュアルについては、ソフトウェアとともにインストールされるマニュアル索引 `/opt/SUNWspro/docs/ja/index.html` を参照してください。`/opt` ディレクトリ以外の場所にソフトウェアがインストールされている場合は、ご使用のシステムあるいはネットワーク上の該当するパスを、システム管理者に確認してください。

C コンパイラ

この節では、新リリースの C コンパイラの新機能について説明します。次の各表に新機能を示します。

- 表 1 全般的な機能強化
- 表 2 ハードウェアプラットフォームのサポート強化
- 表 3 パフォーマンスおよび最適化オプションの改良
- 表 4 Lint ユーティリティの新しいセキュリティ検査機能

この節で示すコンパイラのオプションの詳細は、『C ユーザーズガイド』または CC(1) のマニュアルページを参照してください。

表 1 は、C コンパイラの全般的な機能強化の内容をまとめています。

表 1　　C コンパイラの全般的な機能強化

機能	内容の説明
C99 機能の実装	このリリースでは、以下の ISO/IEC 9899:1999 (このマニュアルでは C99 と表記) の機能のサポートが追加されています。本リリースでは、C99 の機能のサブセットが実装されています。ここでは、本リリースで実装された C99 の機能のみを示します。C コンパイラの過去および現在のリリースで実装されたすべての C99 の機能については、『C ユーザーズガイド』を参照してください。この Sun Studio 9 リリースで新たにサポートされた項目については、項目ごとに C99 規格のサブセクション番号を付記しています。 • 5.2.4.2.2: `FLT_EVAL_METHOD` マクロのサポート。このマクロおよび新しい `-flteval` コンパイルオプションは、浮動小数点式を long double として評価するか、あるいは式内の型とリテラルの組み合わせに基づいて評価するかを決定します。 • 6.4.3: 4 桁および 8 桁の汎用文字名 (UCN) のサポート。識別子や文字リテラル、文字列リテラルで、C の基本文字セットにない文字の指定に使用できます。UCN の \Unnnnnnnn は、8 桁の短い識別子が nnnnnnnn の文字を表します (ISO/IEC 10646 で規定)。同様に、汎用文字名の \unnnn は、4 桁の短い識別子が nnnn (8 桁の短い識別子は 0000nnnn) の文字を表します。 • 6.7.4: インライン関数と extern インライン関数のサポート • 6.7.8: 指示付きの初期化子のサポート。数値およびシステムプログラミングで一般的なスパース配列やスパース構造体を初期化する手段になります。

表 1　C コンパイラの全般的な機能強化 (続き)

機能	内容の説明
-features コンパイルオプションを追加することによる古いバイナリとの互換性の向上	古い C および C++ バイナリ (C/C++ 5.6 より古いもの) の動作を変更することなく、それらバイナリを新しい C および C++ バイナリとリンクさせることができます。新しいバイナリと、extern インライン関数を含む古い C および C++ ライブラリとの間で互換性を取る場合に -features=no%extinl オプションを使用してください。 規格に適合した動作を実現するには、最新のコンパイラを使って古いコードをコンパイルする必要があります。
スレーブスレッドのデフォルトのスタックサイズの拡大	スレーブスレッドのデフォルトのスタックサイズが拡大されました。スレーブスレッドのデフォルトのスタックサイズは、32 ビットアプリケーションの場合 4M バイト、64 ビットアプリケーションの場合 8M バイトです。スタックサイズは、環境変数 `STACKSIZE` を使用して設定します。
`-xprofile` の強化 (SPARC®)	`-xprofile` オプションに関して、以下の機能強化が行われています。 • 共有ライブラリのプロファイリングのサポート • `-xprofile=collect -mt` を使用して行うスレッドセーフなプロファイル収集 • 1 つのプロファイルディレクトリ内での複数のプログラムや共有ライブラリのプロファイリングのサポート強化 `-xprofile=use` を使用して、固有のベース名を持たない複数のオブジェクトファイルのデータが存在するプロファイルディレクトリ内のプロファイルデータを検出できるようになりました。オブジェクトファイルのプロファイルデータを見つけることができない場合には、`-xprofile_pathmap=`*collect-prefix*`:` *use-prefix* という新しいオプションを使用することができます。
UTF-16 文字列リテラルのサポート : `-xustr`	ISO10646 UTF-16 文字列リテラルを使用する多言語アプリケーションをサポートする必要がある場合には、`-xustr=ascii_utf16_ushort` を指定します。つまり、コードに 16 ビット文字から構成される文字列リテラルが含まれている場合にはこのオプションを使用します。このオプションを指定しないと、C++ コンパイラは 16 ビット文字列リテラルの生成、認識を行いません。このオプションは、U″ASCII_string″ 文字列リテラルを符号なし短精度の配列として認識することを可能にします。このような文字列は標準として規定されていないので、このオプションは標準に準拠しない C の認識を可能にします。
プリコンパイル済みヘッダーの自動生成	このリリースの C コンパイラでは、プリコンパイル済みのヘッダー機能が拡張され、コンパイラの側でプリコンパイル済みヘッダーファイルを自動的に生成できるようになっています。ただし、これまでどおり、プリコンパイル済みヘッダーファイルを手動で生成することもできます。コンパイラの新しい機能の詳細は、`cc`(1) のマニュアルページの `-xpch` オプションの説明をお読みください。`CCadmin`(1) のマニュアルページも参照してください。

表 2 は、コンパイルの高速化をサポートする C コンパイラの新機能をまとめています。

表 2　　ハードウェアプラットフォームのサポート強化

機能	内容の説明
SPARC プラットフォームをサポートするフラグの追加	`-xchip` および `-xtarget` オプションが、値として ultra3i および ultra4 をサポートするようになりました。このため、UltraSPARC IIIi および UltraSPARC IV 用に最適化されたアプリケーションを構築できます。
x86 プラットフォームをサポートするフラグの追加	C コンパイラが、x86 プラットフォームで動作するコード用に `-xarch`、`-xtarget`、および `-xchip` の新しいフラグをサポートするようになりました。これらの新しいフラグは、x86 プラットフォーム上での Solaris ソフトウェアによる `sse` および `sse2` 命令のサポートとの組み合わせで Pentium 3 および Pentium 4 チップを活用することを意図しています。新しいフラグは次のとおりです。 • `-xchip=pentium3` は Pentium 3 方式のプロセッサ用に最適化します。 • `-xchip=pentium4` は Pentium 4 方式のプロセッサ用に最適化します。 • `-xarch=sse` は、pentium_pro 命令セットアーキテクチャに sse 命令セットを追加します。 • `-xarch=sse2` は sse が許可する命令セットに sse2 命令セットを追加します。 • `-xtarget=pentium3` は、-xarch=sse、-xchip=pentium3、-xcache=16/32/4:256/32/4 に設定します。 • `-xtarget=pentium4` は、-xarch=sse2、-xchip=pentium4、-xcache=8/64/4:256/128/8 に設定します。 実際のコンパイルでの適切なオプションの組み合わせは、次のガイドラインに基づいて決定することができます。 • Solaris 9 update 6 以降が動作する Pentium 3 または Pentium 4 マシンで実行するアプリケーションを構築する場合は、コンパイルでそれぞれ `-xtarget=pentium3` または `-xtarget=pentium4` を使用する。 • Solaris 9 update 5 以前が動作する Pentium 3 または Pentium 4 マシンで実行するアプリケーションを構築する場合は、sse および sse2 命令がサポートされていないため、`-xarch=pentium_pro` (pentium3 あるいは pentium4 ではありません) に設定する。`-xtarget=pentium3` または `-xtarget=pentium4` を使用する場合は、ターゲットマシンに従って、`-xchip` および `-xcache` を同じ値に設定します。 • ターゲットマシンでの構築の場合は、`-fast` や `-xarch=native`、`-xtarget=native` を指定する。上記の適切な `-xchip`、`-xarch`、および `-xtarget` のフラグ設定に展開されます。

表 3 は、パフォーマンスの向上をサポートする C コンパイラの新機能をまとめています。

表 3　パフォーマンスおよび最適化オプションの改良

機能	内容の説明
コンパイラオプションのデフォルト値および展開の変更	次のコンパイルオプションのデフォルト値が変更されました。 • `-xarch` (SPARC プラットフォームの場合) : v8plus。新しいデフォルトでは、現在使用されているほぼあらゆるマシンで実行時のパフォーマンスが向上します。ただし、UltraSPARC 以前のコンピュータへの配備を意図したアプリケーションは、デフォルトでは、そうしたコンピュータ上で動作しなくなります。アプリケーションが UltraSPARC 以前のコンピュータ上で動作するようにするには、`-xarch=v8` でコンパイルしてください • `-xcode` (SPARC プラットフォーム) : v9 の場合 `abs44`、v8 の場合 `abs32`。 • `-xmemalign` (SPARC プラットフォーム) : v8 の場合 `8i`、v9 の場合 `8s`。 • `-xprefetch`　(SPARC プラットフォーム) : `auto,explicit`。基本的に非線形のメモリーアクセスパターンを持つアプリケーションには、この変更が良くない影響をもたらします。この変更を無効にするには、`-xprefetch=no%auto,no%explicit` を指定します。 次のオプションおよびマクロの展開が変更されました。 • `-fast` オプションが `-xlibmopt` にも展開されるようになりました (下記を参照)。 • `-O` マクロが、`-xO2` ではなく、`-xO3` に展開されるようになりました。このデフォルトの変更によって、実行時のパフォーマンスが向上します。ただし、あらゆる変数を自動的に volatile と見なすことを前提にするプログラムの場合、`-xO3` は不適切なことがあります。このことを前提とする代表的なプログラムとしては、専用の同期方式を実装するデバイスドライバや古いマルチスレッドアプリケーションがあります。回避策は、`-O` ではなく、`-xO2` を使ってコンパイルすることです。

表 3　パフォーマンスおよび最適化オプションの改良 (続き)

機能	内容の説明
最適化コンパイルオプションの追加	新しいコンパイルオプションは次のとおりです。 • `-xlibmopt` および `-xnolibmopt` : `-xlibmopt` オプションは、最適化された数学ルーチンのライブラリを使用するようコンパイラに指示します。このオプションを使用するときは `-fround=nearest` を指定することによって、デフォルトの丸めモードを使用する必要があります。数学ルーチンライブラリは最高のパフォーマンスが得られるように最適化されており、通常、高速なコードを生成します。この結果は、通常の数学ライブラリが生成する結果と少し異なることがあります。その場合、通常、異なるのは最後のビットです。 このライブラリは、コマンド行で新しい `-xnolibmopt` オプションを指定することによって明示的に無効にすることができます。 • `-xipo_archive` : 新しい `-xipo_archive` オプションは、`-xipo` を付けてコンパイルされ、実行可能ファイルを生成する前にアーカイブライブラリ (.a) に存在するオブジェクトファイルを使って、リンカーに渡すオブジェクトファイルを最適化するよう指示します。コンパイル中に最適化されたライブラリに含まれるオブジェクトファイルはすべて、その最適化されたバージョンに置き換えられます。 • `-xprefetch_auto_type` : 新しいオプション `-xprefetch_auto_type` を利用することによって、直接メモリーアクセスに対してプリフェッチが生成されるのと同じ方法で、-xprefetch_level=[1|2|3] が指示するループに対して間接プリフェッチを生成することができます。 `-xdepend`、`-xrestrict`、`-xalias_level` などのオプションと組み合わせると、`-xprefetch_auto_type` のもたらす最適化のメリットを増すことができます。これらのオプションはメモリーの別名のあいまいさを排除する情報を生成するのに役立つため、間接プリフェッチ候補の計算の積極性に影響し、自動的な間接プリフェッチの挿入が促進されます。

表 4 は、lint ユーティリティに含まれている新しい検査機能をまとめています。

表 4 Lint ユーティリティの新しいセキュリティ検査機能

機能	内容の説明
lint の新しい -errsecurity オプション	Sun Studio 9 リリースの lint ユーティリティには、新しいセキュリティ検査機能が追加されています。コンパイルの前に新しい -errsecurity オプションを使用して、セキュリティに問題がないかコードを検査することができます。 -errsecurity[={core | standard | extended | %none}] lint -errsecurity=core このレベルでは、たいていの場合安全でない、または検査することの難しいソースコードの構文がないかどうかを検査します。このレベルの検査には、以下があります。 • printf() および scanf() 系の関数での変数書式文字列の使用 • scanf() 関数における非結合文字列 (%s) 形式の使用 • 安全な使用法のない関数の使用 (gets()、cftime()、ascftime()、creat()) • O_CREAT と組み合わせた open() の不正使用 このレベルで警告が生成されるソースコードはバグと考えてください。問題のコードを変更することを推奨します。どんな場合でも、単純明快でより安全な別の方法があります。 lint -errsecurity=standard このレベルは、core レベルの検査に加えて、安全かもしれないが、より良い別の方法がある構文の検査があります。新しく作成したコードの検査には、このレベルを推奨します。このレベルで追加される検査には、以下があります。 • strlcpy() 以外の文字列コピー関数の使用 • 脆弱な乱数関数の使用 • 安全でない関数を使った一時ファイルの生成 • fopen() を使ったファイルの作成 • シェルを呼び出す関数の使用 このレベルで警告を生成するソースコードは、新しいコードまたは大幅に修正したコードに書き換えてください。ただし、従来のコードに含まれるこうした警告に対処することと、アプリケーションを不安定にするリスクとのバランスを検討してください。

表 4　Lint ユーティリティの新しいセキュリティ検査機能 (続き)

機能	内容の説明
lint の新しい -errsecurity オプション (続き)	lint -errsecurity=extended このレベルでは、core および standard レベルの検査を含む完全な検査が行われます。また、状況によっては安全でない可能性がある構文について、多数の警告が生成されます。このレベルの検査は、コードを見直す際の一助になりますが、許容しうるソースコードが守る必要のある基準と考える必要はありません。このレベルで追加される検査には、以下があります。 • ループ内での getc() または fgetc() の呼び出し • パス名競合になりがちな関数の使用 • exec() 系の関数の使用 • stat() と他の関数との間の競合 このレベルで警告が生成されるコードを見直して、安全上の潜在的な問題があるかどうかを判定することができます。 -errsecurity の値が指定されていない場合、コンパイラは -errsecury=%none に設定します。-errsecurity は指定されているが、引数が指定されていない場合は、-errsecurity=standard に設定します。

C++ コンパイラ

この節では、新リリースの C++ コンパイラの新機能について説明します。次の各表に新機能を示します。

- 表 5 全般的な機能強化
- 表 6 ハードウェアプラットフォームのサポート強化
- 表 7 最適化オプションの新規追加と強化

この節で示すコンパイラのオプションの詳細については、『C++ ユーザーズガイド』または CC(1) のマニュアルページを参照してください。

表 5 は、C++ コンパイラ (バージョン 5.6) の全般的な機能強化の内容をまとめています。

表 5　C++ コンパイラの全般的な機能強化

機能	内容の説明
インライン関数の外部リンケージ	C++ 規格では、static 宣言しない限り、インライン関数は非インライン関数のように外部リンケージを持つことになっています。C++ では、5.6 になって初めて、デフォルトでインライン関数に外部リンケージを持たせるようになりました。インライン関数をライン外で生成する必要がある場合は (たとえば、そのアドレスが必要な場合など)、コピー 1 つだけが最終プログラムにリンクされます。以前は、コピーを必要とするオブジェクトファイルが、それぞれローカルリンケージを持つ専用のコピーを持っていました。 この `extern` インライン関数の実装方法は、プログラムの動作が従来と同様に規格に適合しており、以前のコンパイラバージョンによって作成されたバイナリファイルと互換性があります。古いバイナリはインライン関数のローカルコピーを複数持っていることがありますが、新しいコードでは、`extern` インライン関数のコピー数はあっても 1 つになります。 この `extern` インライン関数の実装方法は、このリリースに含まれている C 5.6 コンパイラを使用する C99 版のインライン関数と互換性があります。すなわち、extern インライン関数に関する C および C++ 規則に従うことによって、同じインライン関数を C および C++ ファイルの両方で定義でき、その場合でも、最終プログラムでの外部関数のコピー数は 1 つだけになります。

UTF-16 のサポート強化	UTF-16 文字列リテラルのサポートは、バージョン 5.5 の C++ コンパイラで導入されました。このリリースでは、文字列の U"x" 構文に似た U′x′ 構文を使用する UTF-16 文字リテラルもサポートされるようになっています。UTF-16 文字リテラルを認識できるようにするには、同じ `-xustr` オプションが必要です。
	
	このリリースではまた、UTF-16 の文字および文字列リテラル内の数値エスケープもサポートしています。これは、通常の文字および文字列リテラル内の数値エスケープに似ています。次に例を示します。
	
	`U"ab\123ef" // 文字の 8 進表現`
	`U'\x456'  // 文字の 16 進表現`
	
	詳細は、C++ マニュアルページの CC(1) の -xustr の説明をお読みください。

表 5　C++ コンパイラの全般的な機能強化 (続き)

機能	内容の説明
プリコンパイル済みヘッダーの自動生成	このリリースの C++ コンパイラでは、プリコンパイル済みのヘッダー機能が拡張され、コンパイラの側でプリコンパイル済みヘッダーファイルを自動的に生成できるようになっています。ただし、これまでどおり、プリコンパイル済みヘッダーファイルを手動で生成することもできます。コンパイラの k の新しい機能の詳細は、cc(1) のマニュアルページの -xpch オプションの説明をお読みください。また、CCadmin(1) のマニュアルページも参照してください。

表 6 は、コンパイルの高速化をサポートする C++ コンパイラの新機能をまとめています。

表 6　ハードウェアプラットフォームのサポート強化

機能	内容の説明
SPARC プラットフォームをサポートするフラグの追加	-xchip および -xtarget オプションが、値として ultra3i および ultra4 をサポートするようになりました。このため、UltraSPARC IIIi および UltraSPARC IV 用に最適化されたアプリケーションを構築できます。
x86 プラットフォームをサポートするフラグの追加	C++ コンパイラが、x86 プラットフォームで動作するコード用に -xarch、-xtarget、および -xchip の新しいフラグをサポートするようになりました。これらの新しいフラグは、x86 プラットフォーム上での Solaris ソフトウェアによる sse および sse2 命令のサポートとの組み合わせで Pentium 3 および Pentium 4 チップを活用することを意図しています。新しいフラグは次のとおりです。 • -xchip=pentium3 は Pentium 3 方式のプロセッサ用に最適化します。 • -xchip=pentium4 は Pentium 4 方式のプロセッサ用に最適化します。 • -xarch=sse は、pentium_pro 命令セットアーキテクチャに sse 命令セットを追加します。 • -xarch=sse2 は sse が許可する命令セットに sse2 命令セットを追加します。 • -xtarget=pentium3 は、-xarch=sse、-xchip=pentium3、-xcache=16/32/4:256/32/4 に設定します。 • -xtarget=pentium4 は、-xarch=sse2、-xchip=pentium4、-xcache=8/64/4:256/128/8 に設定します。

表 6　ハードウェアプラットフォームのサポート強化 (続き)

機能	内容の説明
x86 プラットフォームをサポートするフラグの追加 (続き)	実際のコンパイルで適切なオプションの組み合わせは、次のガイドラインに基づいて決定することができます。 • Solaris 9 update 6 以降が動作する Pentium 3 または Pentium 4 マシンで実行するアプリケーションを構築する場合は、コンパイルでそれぞれ `-xtarget=pentium3` または `-xtarget=pentium4` を使用する。 • Solaris 9 update 5 以前が動作する Pentium 3 または Pentium 4 マシンで実行するアプリケーションを構築する場合は、sse および sse2 命令がサポートされていないため、`-xarch=pentium_pro` (pentium3 あるいは pentium4 ではありません) に設定する。`-xtarget=pentium3` または `-xtarget=pentium4` を使用する場合は、ターゲットマシンに従って、`-xchip` および `-xcache` を同じ値に設定する。 • ターゲットマシンでの構築の場合、`-fast` や `-xarch=native`、`-xtarget=native` を指定すると、上記の適切な `-xchip`、`-xarch`、および `-xtarget` のフラグ設定に展開されます。

表 7 は、移植の簡略化をサポートする C++ コンパイラの新機能をまとめています。

表 7　最適化オプションの新規追加と強化

機能	内容の説明
コンパイラオプションのデフォルト値および展開の変更	次のコンパイルオプションのデフォルト値が変更されました。 • `-xarch` (SPARC プラットフォームの場合) : v8plus。新しいデフォルトでは、現在使用されているほぼあらゆるマシンで実行時のパフォーマンスが向上します。ただし、UltraSPARC 以前のコンピュータへの配備を意図したアプリケーションは、デフォルトでは、そうしたコンピュータ上で動作しなくなります。アプリケーションが -UltraSPARC 以前のコンピュータ上で動作するようにするには、`-xarch=v8` でコンパイルしてください • `-xcode` (SPARC プラットフォーム) : v9 の場合 `abs44`、v8 の場合 `abs32`。 • `-xmemalign` (SPARC プラットフォーム) : v8 の場合 `8i`、v9 の場合 `8s`。 • `-xprefetch` (SPARC プラットフォーム) : `auto,explicit`。基本的に非線形のメモリーアクセスパターンを持つアプリケーションには、この変更が良くない影響をもたらします。この変更を無効にするには、`-xprefetch=no%auto,no%explicit` を指定します。

表 7　最適化オプションの新規追加と強化 (続き)

機能	内容の説明
コンパイラオプションのデフォルト値および展開の変更 (続き)	次のマクロの展開が変更されました。 • -O マクロが、-xO2 ではなく、-xO3 に展開されるようになりました。このデフォルトの変更によって、実行時のパフォーマンスが向上します。ただし、あらゆる変数を自動的に volatile と見なすことを前提にするプログラムの場合、-xO3 は不適切なことがあります。このことを前提とする代表的なプログラムとしては、専用の同期方式を実装するデバイスドライバや古いマルチスレッドアプリケーションがあります。回避策は、-O ではなく、-xO2 を使ってコンパイルすることです。
ループ最適化コンパイルオプションの新規追加	C++ コンパイラが、計算の並列化が可能なループを最適化するための次のオプションをサポートするようになりました。これらのオプションは、最適化レベルとして -xO3 以上を指定する場合にのみ効果があります。 • -xautopar • -xvector • -xdepend 詳細は、C++ マニュアルページの CC(1) の -xautopar、-xvector、および -xdepend の説明をお読みください。
関数別の最適化レベル制御機能の追加	#pragma opt 指令とコマンド行オプションの -xmaxopt を組み合わせて、コンパイラが個々の関数に適用する最適化レベルを指定することができます。この組み合わせは、たとえば、スタックフレームの削除などのコード拡張を避けるために、特定の関数について最適化レベルを下げたり、あるいはその逆に、特定の関数について最適化レベルを上げたりする必要がある場合に有用です。
ループの先読み命令の最適化	-xprefetch_auto_type : 新しいオプション -xprefetch_auto_type を利用することによって、直接メモリーアクセスに対してプリフェッチが生成されるのと同じ方法で、-xprefetch_level=[1|2|3] が指示するループに対して間接プリフェッチを生成することができます。 -xdepend、-xrestrict、-xalias_level などのオプションと組み合わせると、-xprefetch_auto_type のもたらす最適化のメリットを増すことができます。これらのオプションはメモリーの別名のあいまいさを排除する情報を生成するのに役立つため、間接プリフェッチ候補の計算の積極性に影響し、自動的な間接プリフェッチの挿入が促進されます。

表 7　最適化オプションの新規追加と強化 (続き)

機能	内容の説明
制限付きポインタの最適化	C++ は、C99 で導入された `restrict` キーワードをサポートしていません。ただし、この C++ コンパイラでは、C コンパイラのオプション `-xrestrict` を受け付けるようになっています。 このオプションは、コンパイル時に関数に関して、ポインタ型の関数パラメータが同じかオーバーラップするオブジェクトを参照していないという認識をします。このことは C++ 標準ライブラリに含まれる一部関数には当てはまらないため、C よりも C++ でより問題となります。

Fortran コンパイラ

表 8 は、以下をはじめとする Fortran コンパイラの新機能と機能強化の内容をまとめています。

- Solaris OS x86 プラットフォーム版 f95 向けの新しいコンパイル機能
- 実行時のパフォーマンスの向上
- Fortran 2003 の新しいコマンド行組み込み関数
- f95 コンパイラのコマンド行オプションのデフォルト値の変更
- デフォルトの SPARC アーキテクチャの変更
- OpenMP ライブラリの強化
- f95 コンパイラの新しいコマンド行オプション

この節で示すコンパイラのオプションの詳細については、『Fortran ユーザーズガイド』または f95(1) のマニュアルページを参照してください。

表 8 Fortran コンパイラの新規および機能強化

機能	内容の説明
Solaris OS x86 プラットフォーム版 f95 向けの新しいコンパイル機能	-xtarget 値として generic、native、386、486、pentium、pentium_pro、pentium3、pentium4 のいずれかを付けてコンパイルし、Solaris x86 プラットフォーム用の実行可能ファイルを生成できます。x86 プラットフォームでのデフォルトは -xtarget=generic です。 x86 プラットフォームの場合、次の f95 機能はまだ実装されていません。使用できるのは、SPARC プラットフォーム上のみです。 • 区間演算 (コンパイラオプション -xia および -xinterval) • Quad (128 ビット) 演算 • IEEE 組み込みモジュール の IEEE_EXCEPTIONS、IEEE_ARITHMETIC、および IEEE_FEATURES • sun_io_handler モジュール • -autopar、-parallel、-explitipar、openmp などの並列化オプション 次の f95 コマンド行オプションは、x86 プラットフォームでのみ使用できます。SPARC プラットフォーム では使用できません。 • -fprecision、-fstore、および -nofstore 次の f95 コマンド行オプションは、SPARC プラットフォームでのみ使用できます。x86 プラットフォーム では使用できません。 • -xcode、-xmemalign、-xprefetch、-xcheck、-xia、-xinterval、-xipo、-xjobs、-xlang、-xlinkopt、-xloopinfo、-xpagesize、-xprofile_ircache、-xreduction、-xvector、-depend、-openmp、-parallel、e--autopar、-explicitpar、-vpara、-XlistMP また x86 プラットフォームでは、-fast オプションの展開に、-nofstore というオプションが追加されるようになりました。

表 8　Fortran コンパイラの新規および機能強化 (続き)

機能	内容の説明
Solaris OS x86 プラットフォーム版 f95 向けの新しいコンパイル機能 (続き)	Fortran コンパイラが、x86 プラットフォームで動作するコード用に -xarch、-xtarget、および -xchip の新しいフラグをサポートするようになりました。これらの新しいフラグは、x86 プラットフォーム上での Solaris ソフトウェアによる sse および sse2 命令のサポートとの組み合わせで Pentium 3 および Pentium 4 チップを活用することを意図しています。新しいフラグは次のとおりです。 • -xchip=pentium3 は Pentium 3 方式のプロセッサ用に最適化します。 • -xchip=pentium4 は Pentium 4 方式のプロセッサ用に最適化します。 • -xarch=sse は、pentium_pro 命令セットアーキテクチャに sse 命令セットを追加します。 • -xarch=sse2 は sse で許可されている命令セットに sse2 命令セットを追加します。 • -xtarget=pentium3 は、-xarch=sse、-xchip=pentium3、-xcache=16/32/4:256/32/4 に設定します。 • -xtarget=pentium4 は、-xarch=sse2、-xchip=pentium4、-xcache=8/64/4:256/128/8 に設定します。 • -fns は、pentium3 または pentium4 プロセッサでのみ有効です。-xarch が sse か sse2 でない場合、-fns=yes は無視されます。そうではなく、SSE および SSE2 浮動小数点演算命令の場合は、アンダーフローをゼロにフラッシュし (FTZ)、非正規化オペランドをゼロとして扱う (DAZ) ことを意味します。- fns=yes は、従来の x86 浮動小数点演算命令には影響しません。たとえば long double 型のオペランドまたは結果に対する浮動小数点演算では、従来の x86 浮動小数点演算命令が使用され、-fns=yes の影響を受けません。 x86 に関する特記事項 : Solaris x86 SSE/SSE2 プラットフォームで実行するために -xarch=sse または -xarch=sse2 を付けてコンパイルしたプログラムは、SSE/SSE2 対応のプラットフォームでのみ実行する必要があります。SSE/SSE2 に対応していないプラットフォームでそうしたプログラムを実行すると、セグメント例外が発生したり、明示的な警告メッセージなしに不正な結果が発生したりすることがあります。SSE/SSE2 でコンパイルされたバイナリが SSE/SSE2 に対応していないプラットフォームで実行されることのないようにするための OS およびコンパイラに対するパッチが、後日提供される可能性があります。SSE/SSE2 対応の x86 プラットフォーム としては、Pentium 4 互換のプラットフォームで動作する Solaris 9 update 6 などがあります。 このことは、.il インラインアセンブリ言語関数を使用しているプログラムや、SSE/SSE2 命令を利用している __asm() アセンブラコードにも当てはまります。 こうしたプラットフォーム向けにコンパイルされたバイナリを実行するにあたっては、ターゲットの実行時プラットフォームが SSE/SSE2 対応であるかどうかを事前にシステム管理者に確認してください。

表 8　Fortran コンパイラの新規および機能強化 (続き)

機能	内容の説明
実行時のパフォーマンスの向上	今回のリリースでは、多くのアプリケーションの実行時のパフォーマンスが向上するとみられます。最良の結果を得るには、最適化レベルを高くして (-xO4 または -xO5) コンパイルしてください。これらのレベルでは、コンパイラが内部手続きや、形状引き継ぎ、割り付け、あるいはポインタ引数を持つ手続きをインライン化することができます。
Fortran 2003 の新しいコマンド行組み込み関数	Fortran 2003 規格草案では、コマンド行引数および環境変数を処理するための新しい組み込み関数が紹介されています。このリリースの f95 コンパイラには、これらの組み込み関数が実装されています。新しい組み込み関数は以下のとおりです。 • GET_COMMAND(command, length, status) command でプログラムを呼び出すコマンド行全体を返します。 • GET_COMMAND_ARGUMENT(number, value, length, status) value でコマンド行引数を返します。 • GET_ENVIRONMENT_VARIABLE(name, value, length, status, trim_name) 環境変数の値を返します。
コマンド行オプションのデフォルト値の変更	このリリースの f95 では、次のコマンド行オプションのデフォルト値が変更されています。 • -xprefetch のデフォルト値は -xprefetch=no%auto,explicit です。 • -xmemalign のデフォルト値は -xmemalign=8i です。ただし、-xarch=v9 および v9a の場合、デフォルト値は -xmemalign=8f になります。
デフォルトの SPARC アーキテクチャの変更	デフォルトの SPARC アーキテクチャが V7 でなくなりました。この Sun Studio 9 リリースでは、-xarch=v7 のサポートに制限があります。新しいデフォルトは V8PLUS (UltraSPARC) です。-xarch=v8 以上をサポートしているのは Solaris 8 OS だけであるため、-xarch=v7 によるコンパイルは、-xarch=v8 として扱われます。
OpenMP ライブラリの強化	OpenMP ライブラリが以下の点で機能強化されました。 • OMP_NUM_THREADS およびマルチタスクライブラリの最大スレッド数が 128 から 256 に増加しました。 • このリリースの Fortran 95 コンパイラに実装されている、共有メモリー並列プログラミング用の OpenMP API には、並列領域における変数の自動スコープ機能があります。詳細は、『OpenMP API ユーザーズガイド』を参照してください。 (このリリースでは、OpenMP は SPARC プラットフォームでのみ実装されています。)

表 8　Fortran コンパイラの新規および機能強化 (続き)

機能	内容の説明
f95 コンパイラの新しいコマンド行オプション	このリリースの f95 では、次のコマンド行オプションが新しく追加されています。詳細は f95(1) のマニュアルページを参照してください。 • `-xipo_archive={ none | readonly | writeback }` クロスファイル最適化でアーカイブ (.a) ライブラリを取り込むことができます (SPARC のみ)。 • `-xipo_archive=none` アーカイブファイルを処理しません。 • `-xipo_archive=readonly` 実行可能ファイルを生成する前に、アーカイブライブラリ (.a) に存在するオブジェクトファイル (-xipo でコンパイルしたファイル) を使ってリンカーに渡すオブジェクトファイルを最適化します。 • `-xipo_archive=writeback` 実行可能ファイルを生成する前に、アーカイブライブラリ (.a) に存在するオブジェクトファイル (-xipo でコンパイルしたファイル) を使ってリンカーに渡すオブジェクトファイルを最適化します。コンパイル中に最適化されたライブラリに含まれるオブジェクトファイルはすべて、その最適化されたバージョンに置き換えられます。`-xipo` の値が指定されていない場合、コンパイラは `-xipo_archive=none` に設定します。 • `-xprefetch_auto_type=[no%]indirect_array_access` 間接アクセスされるデータ配列に対して間接先読み命令を生成します (SPARC のみ)。 • `[no%]indirect_array_access` 直接メモリーアクセスに対して先読み命令が生成されるのと同じ方法で、`-xprefetch_level=[1|2|3]` オプションが指示するループに対して間接先読み命令を生成します (または生成しません)。 `-xprefetch_auto_type` の値が指定されていない場合、コンパイラは `-xprefetch_auto_type=[no%]indirect_array_access` に設定します。 `-xprefetch` オプションは、SPARC プラットフォームでのみ使用できます。 `-xdepend`、`-xrestrict`、`-xalias_level` などのオプションは、メモリー別名のあいまいさを排除する情報の生成に役立つため、間接先読み命令候補の計算の積極性に影響し、このため、自動的な間接先読み命令の挿入が促進されることがあります。 • `-xprofile_pathmap=collect_prefix:use_prefix` プロファイルデータファイルのパスマッピングを設定します。以前に `-xprofile=collect` を使ってコンパイルしたときに使用したディレクトリとは異なるディレクトリにプロファイリングする場合は、`-xprofile_pathmap` オプションと `-xprofile=use` オプションを併用してください。

コマンド行デバッガ dbx

Sun Studio 9 リリースの dbx には、次の新機能が追加されています。

- Linux プラットフォーム版での gcc および g++ コンパイラのサポート
- Solaris OS x86 プラットフォーム版での Fortran のサポート

区間演算

このリリースでは、区間演算の新機能はありません。

Sun Performance Library

Sun Performance Library™ は、線形代数問題とその他の数値を多く取り扱う問題の解決に使用する、最適化された高速数学サブルーチンのセットです。
Sun Performance Library は、Netlib (`http://www.netlib.org`) から利用できるパブリックドメインアプリケーション群に基づいています。これらのルーチンは、改良され、Sun Performance Library としてバンドルされています。

表 9 は、新リリースの Sun Performance Library の新機能をまとめています。詳細については、『Sun Performance Library User's Guide』とセクション 3p のマニュアルページを参照してください。

表 9 Sun Performance Library の新機能

機能	内容の説明

x86 用の Sun Performance Library をリリース	このリリースの Sun Performance Library には、Solaris および x86 両方のプラットフォーム用のライブラリが含まれています。また、次の 2 つのバージョンがあります。
	• SSE2 命令セットをサポートするシステム用の SSE2 命令を使用する高性能バージョン
	• SSE2 をサポートしていないシステムに適した互換性バージョン
	x86 版の Sun Performance Library は、次の点を除けば、SPARC 版と機能的に同じです。
	• Quad 精度ルーチン (`dqdoti`、`dqdota`) は使用できない
	• 区間 BLAS ルーチンは使用できない
	• x86 ライブラリはシングルスレッド
	• 32 ビットアドレッシングのみ使用可能
	• Solaris/x86 では、Portable Library Performance 機能は使用できません。
	SSE2 のサポートには、次のバージョンの Solaris/x86 が必要です。
	• Solaris 10 ビルド 48 以降
	• Solaris 9 update 5 ビルド 6 以降
	最適化済みの高性能 SSE2 ライブラリを使ってリンクするには、-xarch=sse2 フラグを使用します。次に例を示します。
	`f95 -xarch=sse2 example.f -xlic_lib=sunperf`
	または
	`cc -xarch=sse2 example.c -xlic_lib=sunperf`

dmake

dmake は、make(1) と互換性のあるコマンド行ツールです。dmake を使用して、分散、並列、または直列モードでターゲットを構築することができます。標準の make(1) ユーティリティを使用している場合、Makefile にほとんど変更を加えずに dmake に移行することができます。dmake は make ユーティリティのスーパーセットです。make を入れ子状態で使用する場合、トップレベルの Makefile から make を呼び出すには、$(MAKE) を使用する必要があります。dmake は Makefile を構文解析し、どのターゲットを同時に構築できるかを確認し、ユーザーが設定した数のホストにわたってターゲットを構築します。詳細は dmake(1) のマニュアルページを参照してください。表 10 は、Sun Studio 9 リリースの dmake の新機能をまとめています。

表 10 dmake の新機能

機能	内容の説明
Solaris 用 dmake のパフォーマンス、信頼性、使い勝手の向上	makefile パーサーが、前回バージョン比で 10 倍、GNU make 比で 3 倍高速になりました。構築も高速で安定性が増しています。ログファイルも見やすくなっています。
Linux 版 dmake の実装	Lunux 向けの完全な dmake 機能が実装され、直列、並列、分散モードで構築できます。この結果、makefile に大きな変更を加えることなく、Solaris アプリケーションを Lunux 上で構築できます。1 つのビルドを Linux と Solaris システムの両方に分散できます。

パフォーマンス解析ツール

表 11 は、Sun Studio 9 のパフォーマンス解析ツールにおけるデータの収集と表示の新機能をまとめています。詳細については、次のマニュアルページを参照してください。

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

表 11 は、Sun Studio 9 パフォーマンスアナライザの新機能と機能強化の内容をまとめています。

表 11　パフォーマンス解析ツールの新機能

機能	内容の説明
Linux ディストリビューションの追加	Sun Studio 9 for Solaris に加えて、Sun Studio 9 for Linux でも、パフォーマンスアナライザが利用できるようになりました。次の Linux オペレーティングシステムに対応しています。 • Java™ Desktop System 1.0 • SuSE Linux Enterprise Server 8 • RedHat Enterprise Linux 3 Linux ディストリビューションに er_kernel が含まれていないことを除けば、提供ユーティリティは両方のオペレーティングシステムで同じです。Linux の場合、collect コマンドの制限が多くなっています。使用できるのは時間プロファイリングおよびヒープトレースだけです。詳細は、collect のマニュアルページを参照してください。Linux 上でマルチスレッドアプリケーションをプロファイリングすることは可能ですが、現在のところ、RedHat 版 Linux オペレーティングシステムでのプロファイリングでは高い率で矛盾するデータが発生することが観察されています。
データ空間プロファイリング	SPARC プラットフォーム向けの C プログラムに対してデータ空間プロファイリングを行うことができます。データ空間プロファイルはキャッシュミスなどのメモリー関係のイベントの報告データをまとめたもので、メモリー関係のイベントが発生する命令だけではなく、イベントを発生させるデータオブジェクト参照についても報告します。 データ空間プロファイリング情報の解析結果は、次のようにコマンド行またはアナライザの GUI で表示することができます。 • `er_print` コマンドには、データ空間プロファイリング関係のオプションとして、`data_objects`、`data_osingle`、`data_olayout` という 3 つのオプションが新たに追加されています。 • アナライザには、データ空間プロファイリング関係のタブとして、「データオブジェクト」と「データレイアウト」という 2 つのタブが新たに追加されています。実験にデータ空間プロファイルが存在する場合は、これらのタブが自動的に表示されます。

表 11　パフォーマンス解析ツールの新機能 (続き)

機能	内容の説明
派生プロセス	派生プロセスの記録機能が強化され、fork や exec コマンド、その変形コマンドを使って作成されたプロセスばかりでなく、あらゆる派生プロセスを記録できるようになりました。この追加機能をサポートするため、collect -F コマンドに次の新しいオプションが追加されています。 `collect -F all` システムコールのように、`-F on` ではなく、`-F all` によって処理された派生プロセスには、コード文字「c」の付いた名前が付けられます。 派生プロセスのデータは、コマンド行ユーティリティ `er_print` を使って、またアナライザの GUI で明示的に選別表示できます。 詳細は、`collect`(1) のマニュアルページを参照してください。
データ収集出力のリダイレクト	collect コマンドに新しいオプション `collect -O` *file* が追加されました。このオプションは、`collect` からのすべての出力を *file* にリダイレクトします。生成されたターゲットからの出力はリダイレクトしません。
アナライザのコマンド行引数の強化	アナライザコマンド (起動スクリプト) が、長い引数の二重ハイフンを受け付けるようになりました。具体的には、`--jdkhome` および `--fontsize` です。
アナライザ API 共有ライブラリのパッケージ化	アナライザ API 用の共有ライブラリが独立したパッケージなりました。このため、単独かつ自由に配布できます。
`Collect` コマンドに対する notes ファイルのサポート	`collect` コマンドに新しいコマンド行オプション `collect -C` *comment* が追加されました。*comment* は、実験の notes ファイルに追加されます。最大 10 個の `-C` 引数を適用できます。
実験プレビューおよび実験ヘッダーでの notes 表示	実験プレビューおよび実験ヘッダーに、実験の notes ファイルの内容が表示されます。
ソースおよび逆アセンブリ表示の機能強化	注釈付きソースおよび逆アセンブリにおける、代替ソースコンテキストから得られたコードの取り扱いが改良されました。イタリック体で赤く表示されるインデックス行は、別のファイルからのコードの挿入位置を示します。「ソース」タブでインデックス行をクリックすると、「ソース」ウィンドウが開いて、その代替ソースファイルが表示されます。
`er_src` コマンドの機能強化	コマンド行ユーティリティの `er_src` が関数リストの表示、Java `.class` ファイルの処理、代替ソースコンテキストからのソースおよび逆アセンブリの表示を行えるようになりました。
Java メソッドの署名	Java の長い名前の形式には、関数名だけではなく、完全なメソッドの署名が表示されます。

表 11　パフォーマンス解析ツールの新機能 (続き)

機能	内容の説明
ヒープトレース時の mmap 呼び出しの取り込み	ヒープトレース時の mmap 呼び出しはメモリー割り当てとして処理されます。

統合開発環境 (IDE)

Sun Studio 9 リリースの IDE には、次の新機能が追加されています。

- ss_attach 機能 - プロセスの実行後に dbx デバッガを接続するのではなく、プログラムの実行開始とともにプログラムを捕捉し、dbx デバッガを接続してただちにデバッグを開始できます。
- ソースエディタの「クイックブラウズ」コンボボックス - ソースファイルのクラスメソッドや関数、#define、その他要素に簡単に移動できます。

マニュアル類

ここでは、Sun Studio 9 マニュアルの新機能について解説します。

- 『OpenMP API ユーザーズガイド』に新しい章が 2 つ追加されました。第 5 章は、Fortran 95 __AUTO 節での自動データスコープについて説明しています。第 6 章では、OpenMP プログラムのパフォーマンスの観点から、パフォーマンスを向上するテクニックに関する一般的な推奨事項を提供しています。
- 『C ユーザーズガイド』に 2 つの付録が追加されました。付録 A の「機能別コンパイラオプション」と付録 D の「C99 でサポートされている機能」です。付録 A の内容は、オプションリファレンスの章の最初の部分にあったものですが、参照しやすいよう、1 つの付録として独立させました。
- 『プログラムのパフォーマンス解析』マニュアルに、「注釈付きソースと逆アセンブリデータについて」という 1 章が追加されました。この章は、インデックス行やコンパイラのコメント、特殊な行 (アウトライン関数など) などのさまざまな種類の注釈と、注釈と元のソースの表示上の違いを識別する方法を説明しています。
- Sun の開発者向けポータルサイト (http://developers.sun.com/prodtech/cc) から、パフォーマンスアナライザ用のチュートリアルを入手できます。
- 『Sun WorkShop to Sun Studio Migration』ヘルプセットに、ファイルの比較とマージに関する新しいトピックが追加されました。

- この Sun Studio リリースに含まれているコンパイラとツールを取り上げた『コンパイラとツール』ヘルプセットが新登場しました。各トピックには、対応するコンポーネントに関する説明と関連マニュアルの一覧が記載されています。