



Sun ONE Studio 8 の新機能

Sun™ ONE Studio 8

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.
650-960-1300

Part No. 817-2903-10
2003 年 5 月 , Revision A

Copyright © 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

この配布には、第三者が開発したソフトウェアが含まれている可能性があります。

フォント技術を含む第三者のソフトウェアは、著作権法により保護されており、提供者からライセンスを受けているものです。

libdwarf and lidredblack are Copyright 2000 Silicon Graphics Inc. and are available under the GNU Lesser General Public License from <http://www.sgi.com>.

本製品の一部は、カリフォルニア大学からライセンスされている Berkeley BSD システムに基づいていることがあります。UNIX は、X/Open Company Limited が独占的にライセンスしている米国ならびに他の国における登録商標です。

Sun、Sun Microsystems、Forte、Java、iPlanet、NetBeans および docs.sun.com は、米国およびその他の国における米国 Sun Microsystems, Inc. (以下、米国 Sun Microsystems 社とします) の商標もしくは登録商標です。

サンロゴマークおよび Solaris は、米国 Sun Microsystems 社の登録商標です。

Netscape および Netscape Navigator は、米国ならびに他の国における Netscape Communications Corporation の商標または登録商標です。

すべての SPARC の商標はライセンス規定に従って使用されており、米国および他の各国における SPARC International, Inc. の商標または登録商標です。SPARC の商標を持つ製品は、Sun Microsystems, Inc. によって開発されたアーキテクチャに基づいています。

このマニュアルに記載されている製品および情報は、米国の輸出規制に関する法規の適用および管理下にあり、また、米国以外の国の輸出および輸入規制に関する法規の制限を受ける場合があります。核、ミサイル、生物化学兵器もしくは原子力船に関連した使用またはかかる使用者への提供は、直接的にも間接的にも、禁止されています。このソフトウェアを、米国の輸出禁止国へ輸出または再輸出すること、および米国輸出制限対象リスト(輸出が禁止されている個人リスト、特別に指定された国籍者リストを含む)に指定された、法人、または団体に輸出または再輸出することは一切禁止されています。

すべての SPARC 商標は、米国 SPARC International, Inc. のライセンスを受けて使用している同社の米国およびその他の国における商標または登録商標です。SPARC 商標が付いた製品は、米国 Sun Microsystems 社が開発したアーキテクチャに基づくものです。

本書は、「現状のまま」をベースとして提供され、商品性、特定目的への適合性または第三者の権利の非侵害の黙示の保証を含み、明示的であるか黙示的であるかを問わず、あらゆる説明および保証は、法的に無効である限り、拒否されるものとします。

原典：	<i>What's New Sun ONE Studio 8</i> Part No: 817-0921-10 Revision A
-----	--

© 2003 by Sun Microsystems, Inc.



目次

はじめに v

Sun ONE Studio 8, Compiler Collection の新機能 1

C コンパイラ 2

C++ コンパイラ 8

Fortran コンパイラ 18

コマンド行デバッガ dbx 25

区間演算 26

Sun Performance Library 26

dmake 28

パフォーマンス解析ツール 29

マニュアルについて 32

はじめに

このマニュアルでは、Sun ONE Studio 8, Compiler Collection のコンパイラとコマンド行ツールの新機能について説明します。

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 <i>filename</i> と入力します。 rm ファイル名 と入力します。
『』	参照する書名を示します。	『SPARCstorage Array ユーザーマニュアル』

書体または記号	意味	例
「 」	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。	<code>machinename% grep `^#define \ XV_VERSION_STRING`</code>
➤	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	<code>machine_name%</code>
UNIX の Bourne シェルと Korn シェル	<code>machine_name\$</code>
スーパーユーザー (シェルの種類を問わない)	<code>#</code>

コンパイラコレクションのツールとマニュアルページへのアクセス

コンパイラコレクションのコンポーネントとマニュアルページは、標準の `/usr/bin/` と `/usr/share/man` の各ディレクトリにインストールされません。コンパイラとツールにアクセスするには、`PATH` 環境変数にコンパイラコレクションのコンポーネントディレクトリを必要とします。マニュアルページにアクセスするには、`PATH` 環境変数にコンパイラコレクションのマニュアルページディレクトリが必要です。

`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 ONE Studio コンパイラコレクションコンポーネントが `/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。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ります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には

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。

```
/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には

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 `PATH` 環境変数に追加します。

```
/opt/SUNWspro/man
```

コンパイラコレクション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。
`/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++ ユーザズガイド』
- リリースノートは、`docs.sun.com` で入手できます。

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:	形式：HTML 場所： file:/opt/SUNWspro/docs/ja/index.html のマニュアル索引
『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』	
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
リリースノート	製品 CD 内の HTML ファイル

開発者向けのリソース

<http://www.sun.com/developers/studio> 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

- **Compiler Collection** 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です)。

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

Sun ONE Studio 8, Compiler Collection の新機能

この章では、Sun ONE Studio 8, Compiler Collection のコンパイラとコマンド行ツールの新機能について説明します。この新リリースでは、Sun の C、C++、Fortran のパフォーマンスと移植性の大幅な向上、C99 構文のサブセットのサポート、コマンド行デバッガ dbx における OpenMP™ プログラムのサポートに焦点を合わせて、機能が強化されています。

この章は、次の節で構成されます。

- C コンパイラ
- C++ コンパイラ
- Fortran コンパイラ
- コマンド行デバッガ dbx
- Sun Performance Library
- 区間演算
- パフォーマンス解析ツール
- マニュアルについて

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。各表は、2 つの欄、あるいは 3 つの欄で構成されています。

- 2 つの欄で構成されている表の場合：左の欄には新機能、右の欄にはその説明が示されます。
- 3 つの欄で構成されている表の場合：左の欄に新機能、中央の欄には関連するコマンドやオプション、右の欄には新機能の説明が示されます。

注 - この章で紹介している Sun ONE Studio 8, Compiler Collection 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、システムあるいはネットワーク上の該当するパスを、システム管理者にお問い合わせください。

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。以下の表に新機能を示します。

- 表 1 全般的な強化機能
- 表 2 コンパイルの高速化
- 表 3 パフォーマンスの向上
- 表 4 デバッグの簡略化

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 1 は、C コンパイラ的全般的な強化機能を示しています。

表 1 C コンパイラ的全般的な強化機能

機能	説明
変数スコープにおいてリンカーマップファイルが不要: <code>-xldscope</code>	ダイナミックライブラリのシンボルのエクスポートを 2 つの方法で制御できるようになりました。この機能はリンカースコープと呼ばれ、これまで、リンカーマップファイルによってサポートされていました。まず、コードに新しい宣言指定子を埋め込むことができるようになりました。 <code>_global</code> 、 <code>_symbolic</code> 、 <code>_hidden</code> をコードに直接埋め込むことによって、マップファイルを使用する必要がありません。また、コマンド行で <code>-xldscope</code> を指定し、変数スコープのデフォルトの設定を無効にすることができます。

表 1 C コンパイラの全般的な強化機能 (続き)

機能	説明
C99 機能の実装	<p>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 と表示) 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。各項目の先頭の数字は、C99 標準のサブセクション番号です。</p> <ul style="list-style-type: none"> • 6.2.5 _Bool • 6.2.5 _Complex 型 <p>このリリースでは、_Complex の部分実装をサポートしています。Solaris 7、8、9 の環境では、-lcplxsupp を指定してリンクする必要があります。</p> <ul style="list-style-type: none"> • 6.3.2.1 lvalue に制限されない、配列のポインタへの変換 • 6.4.4.2 16 進浮動小数点定数 • 6.5.2.5 複合定数 • 6.7.2 型指定子 • 6.10.6 STDC プラグマ • 6.10.8 __STDC_IEC_559、__STDC_IEC_559_COMPLEX マクロ
VIS TM Developers Kit のサポート:-xvis (SPARC®)	<p>VIS 命令セット Software Developers Kit (VSDK) で定義されているアセンブリ言語テンプレートを使用する場合には、-xvis=[yes no] オプションを使用します。</p> <p>VIS 命令セットは、SPARC v9 命令セットを拡張したものです。UltraSPARC® は 64 ビットプロセッサですが、特にマルチメディアアプリケーションでは、データのサイズが 8 または 16 ビットに制限されるケースが多数存在します。VIS 命令は 1 つの命令で 4 つの 16 ビットデータを処理することができるため、イメージング、線形代数、信号処理、音声、ビデオ、ネットワーキングなどの新しいメディアを扱うアプリケーションのパフォーマンスが大幅に向上します。</p> <p>VSDK の詳細については、http://www.sun.com/processors/vis を参照してください。</p>

表 1 C コンパイラの全般的な強化機能 (続き)

機能	説明
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 (SPARC) の強化	<p>-xprofile オプションに関して、以下の強化が行われています。</p> <ul style="list-style-type: none"> • 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の強化 <p>-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。</p>
UTF-16 文字列定数のサポート:-xustr	<p>ISO10646 UTF-16 文字列定数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定数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 ビット文字列定数の生成、認識を行いません。このオプションは、U"ASCII_string" 文字列定数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。</p>

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示しています。

表 2 コンパイルの高速化をサポートする新しい C 機能

機能	説明
プロファイリング の高速化 (SPARC)	<p>使用段階において、<code>-xprofile_ircache[=path]</code> と <code>-xprofile=collect use</code> を使用して、収集段階で保存されたコンパイルデータを再利用し、コンパイル時間を短縮することができます。</p> <p>大きなプログラムの場合、中間データが保存されるため、使用段階におけるコンパイル時間が大幅に短縮されます。ただし、データを保存するため、必要なディスク容量が大幅に増える可能性があります。</p>
プリコンパイル済 みヘッダ: <code>-xpch</code>	<p>C コンパイラのこのリリースでは、プリコンパイル済みヘッダという新機能が採用されています。アプリケーションのソースファイルが、大量のソースコードを含むインクルードファイルを共有する場合があります。プリコンパイル済みヘッダファイルは、このようなアプリケーションのコンパイル時間を短縮する目的で設計されています。プリコンパイル済みヘッダを使用することによって、1 つのソースファイルから一連のヘッダファイルに関する情報を収集し、そのソースファイルを再コンパイルする場合や、同じ一連のヘッダを持つ他のソースファイルをコンパイルする場合に、その情報を使用することができます。この機能を利用するには、指示語 <code>#pragma hdrstop</code> と <code>-xpch</code>、<code>-xpchstop</code> オプションを組み合わせ使用します。</p>
複数のプロセッサ の使用: <code>-xjobs=n</code> (SPARC)	<p><code>-xjobs=n</code> オプションを指定して、C コンパイラが作業を完了するために作成するプロセスの数を設定します。このオプションを使用して、マルチ CPU マシンにおけるビルド時間を短縮することができます。現在、<code>-xjobs</code> は、<code>-xipo</code> オプションとの組み合わせでのみ動作します。<code>-xjobs=n</code> を指定すると、手続き間オプティマイザは、複数のファイルをコンパイルするために起動することができるコードジェネレータインスタンスの最大数として <code>n</code> を使用します。</p>

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示しています。

表 3 パフォーマンスの向上をサポートする新しい C 機能

機能	説明
リンカーがサポートするスレッドローカルストレージを使用して行うパフォーマンスの向上： -xthreadvar	<p>リンカーでサポートされる新しいスレッドローカルストレージ機能を使用して、以下を行うことができます。</p> <ul style="list-style-type: none"> • スレッド固有データを割り当てるための POSIX インタフェースの高速実装の利用 • マルチプロセスプログラムのマルチスレッドプログラムへの変換 • スレッドローカルストレージを使用する Windows アプリケーションの Solaris への移植 • OpenMP プログラムにおけるスレッドプライベート変数の高速実装の利用 <p>スレッドローカル変数を宣言してスレッドローカルストレージを使用できるようになりました。スレッドローカル変数の宣言は、通常の変数宣言に変数指示子 <code>_thread</code> とコマンド行オプション <code>-xthreadvar</code> を追加して行います。</p>
ページフォルトの低減による実行時間の短縮： <code>-xF</code>	<p>新機能 <code>-xF</code> を使用して、リンカーによる変数と関数の並べ替えを最適化することができます。この最適化は、実行時間のパフォーマンスに悪影響を与える、以下の問題の解決に役立ちます。</p> <ul style="list-style-type: none"> • 関係のない変数がメモリー内で隣接することによって生じるキャッシュとページの競合 • 関係のある変数がメモリー内で隣接しないことが原因で生じる、必要以上に大きな作業セットサイズ • データ密度を低下させる変数の未使用のコピーが原因で生じる、必要以上に大きな作業セットサイズ

表 3 パフォーマンスの向上をサポートする新しい C 機能 (続き)

機能	説明
実行時間の短縮 : -xlinkopt (SPARC)	<p>-xlinkopt コマンドを指定して、再配置可能なオブジェクトファイルのリンク時間を最適化できるようになりました。</p> <p>-xlinkopt を指定すると、リンクされている .o ファイルを変更せずに最適化が行われます。最適化は実行可能プログラムでのみ行われます。-xlinkopt オプションは、プロファイルのフィードバックを使用して、プログラム全体をコンパイルする場合にもっとも効果的です。</p>
実行時間の短縮 : -xpagesize= <i>n</i> (SPARC)	<p>メモリー内のスタック用のページサイズを設定します。<i>n</i> には 8K、64K、512K、4M、32M、256M、2G、16G を指定するかデフォルト値を使用することができます。getpagesize(3C) によって返される、ターゲットのプラットフォームの Solaris 環境で有効なページサイズを指定する必要があります。有効なページサイズを指定しないと、要求は実行時に無視されます。pmap(1) または meminfo(2) を使用してターゲットプラットフォームのページサイズを決定することができます。</p> <p>この機能は Solaris 9 環境でのみ使用できることに注意してください。このオプションを使用してコンパイルしたプログラムは、Solaris 9 よりも前の Solaris 環境でリンクすることはできません。</p> <p>このオプションは、-xpagesize_stack、-xpagesize_heap のマクロです。</p>
ハードウェアカウンタベースのプロファイリング : -xhwcprof (SPARC)	<p>-xhwcprof=[enable disable] オプションを使用して、ハードウェアカウンタベースのプロファイリングのサポートをオン、オフすることができます。</p> <p>-xhwcprof を使用してプロファイリングのサポートをオンにすると、ハードウェアカウンタデータの参照と、失敗したイベントに関連する命令をツールが照合するための情報が生成されます。また、対応するデータ型と構造体のメンバーをシンボリック情報 (-g を指定して生成) と共に識別することができます。この情報はコードアドレス、ソース文、またはルーチンをベースとするプロファイルを使用して識別することは難しいため、パフォーマンス解析において役に立ちます。</p>

表 4 は、デバッグの簡略化をサポートする C コンパイラの新機能を示しています。

表 4 デバッグの簡略化をサポートする新しい C 機能

機能	説明
DWARF 形式のデバッグ情報： -xdebugformat	デバッグ情報の形式が <code>stabs</code> 形式から「 <i>DWARF Debugging Information Format</i> 」で規定されている DWARF 形式に移行中です。デバッグ情報を読み取るソフトウェアツールを保守するような場合、ツールを <code>stabs</code> 形式から DWARF 形式に移行させることができます。このリリースのデフォルト設定は、 <code>-xdebugformat=stabs</code> です ツールを移植する場合、 <code>-xdebugformat=dwarf</code> オプションを使用して DWARF 形式にアクセスすることができます。デバッグ情報を読み取るソフトウェアを保持しない場合や、上記の形式のデバッグ情報を必要とするツールがない場合には、このオプションを使用する必要はありません。
OpenMP プログラムのデバッグのサポート： -xopenmp=noopt	<code>dbx</code> を使用して OpenMP プログラムをデバッグする場合、 <code>-g</code> と <code>-xopenmp=noopt</code> を使用してコンパイルを行うと、複数の領域にブレークポイントを設定し、変数の内容を表示することができます。

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。以下の表に新機能を示します。

- 表 5 全般的な強化機能
- 表 6 コンパイルの高速化
- 表 7 移植の簡略化
- 表 8 パフォーマンスの向上
- 表 9 警告とエラー制御の追加

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 5.5) の全般的な強化機能を示しています。

表 5 C++ コンパイラの全般的な強化機能

機能	説明
テンプレートキャッシュが不要： <code>-instances</code>	<p>C++ コンパイラのこのリリースでは、テンプレートのインスタンス化について大幅に機能強化されています。デフォルトのテンプレートインスタンス化モデルを使用するプログラムは 1 つのディレクトリ内で 2 つ以上のプログラムを作成しなければならないという制限はなくなりました。</p> <p>別のインスタンス化モデルを使用するプログラムのほとんどが、<code>-instances=static</code> を使用して新しいデフォルトのインスタンス化モデルを使用できるようになりました。</p> <p>このテンプレートのインスタンス化についての機能強化と変更によって、テンプレートのキャッシュが不要になり、コンパイル時間が短縮されます。また、静的関数の重複が防止され、実行可能プログラムのサイズが小さくなります。</p>
変数スコープにおいてリンカーマップファイルが不要： <code>-xldscope</code>	<p>ダイナミックライブラリのシンボルのエクスポートを 2 つの方法で制御できるようになりました。この機能はリンカーサポートと呼ばれ、これまで、リンカーマップファイルによってサポートされていました。まず、コードに新しい宣言指定子を埋め込むことができるようになりました。</p> <p><code>_global</code>、<code>_symbolic</code>、<code>_hidden</code>、をコードに直接埋め込むことによって、マップファイルを使用する必要がありません。</p> <p>また、コマンド行で <code>-xldscope</code> を指定し、変数スコープのデフォルトの設定を無効にすることができます。</p>
強力な新しいマクロ診断機能： <code>-xdumpmacros</code>	<p>このリリースでは、アプリケーション内でのマクロの動作の追跡に役立つ 2 つのプラグマとコンパイラオプションが新たに導入されています。システムヘッダで定義されるマクロの動作も追跡することができます。</p> <p>コマンド行で <code>-xdumpmacros</code> オプションを使用して、マクロの定義を確認することができます。また、プログラム内のマクロの定義場所、マクロが未定義な場所、マクロの使用場所も確認することができます。新たに導入されたプラグマ <code>dumpmacros</code>、<code>end_dumpmacros</code> をソースプログラム内で直接使用し、診断範囲を絞ることができます。</p>

表 5 C++ コンパイラの全般的な強化機能 (続き)

機能	説明
VIS Developers Kit のサポート: <code>-xvis</code>	VIS 命令セット Software Developers Kit (VSDK) で定義されているアセンブリ言語テンプレートを使用する場合には、 <code>-xvis=[yes no]</code> オプションを使用します。デフォルトは、 <code>-xvis=no</code> です。 VSDK の詳細については、 http://www.sun.com/processors/vis を参照してください。
C99 ランタイムライブラリ、環境のサポート: <code>-xlang</code>	C99 標準 (ISO/IEC 9899:1999, Programming Language - C) をサポートするオペレーティングシステムでは、 <code>-xlang=c99</code> を使用して、C ライブラリ関数を呼び出す C、C++ のコードの C99 実行時の動作を指定することができます。C99 の動作の中には、C 複素数型のように、C コンパイラで <code>-xc99=%all</code> オプションを使用する必要があるものと、 <code>printf</code> のように、その必要がないものがあります。 コンパクトモード (<code>-compat=4</code>) では、C99 をサポートすることはできません。
UTF-16 文字列定数のサポート: <code>-xustr</code>	ISO10646 UTF-16 文字列定数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、 <code>-xustr=ascii_utf16_ushort</code> 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定数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 ビット文字列定数の生成、認識を行いません。このオプションは、U "... " 文字列定数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。
OpenMP™ のサポート機能強化: <code>-xopenmp</code>	明示的な並列化のための OpenMP インタフェースの実装が続いています。 <code>-xopenmp</code> オプションの詳細については、CC(1) のマニュアルページを参照してください。 OpenMP 機能が強化され、以下が実現されています。 <ul style="list-style-type: none"> • OpenMP データ句内でのクラスオブジェクトの使用 • クラスメンバ関数内での OpenMP プラグマの使用

表 5 C++ コンパイラの全般的な強化機能 (続き)

機能	説明
-xprofile の機能強化	<p>-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。</p> <ul style="list-style-type: none"> • 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の強化

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示しています。

表 6 コンパイルの高速化をサポートする新しい C++ 機能

機能	説明
構文チェックの高速化 : <code>-xe</code>	<p><code>-xe</code> を指定すると、構文と意味上の誤りのみがチェックされ、オブジェクトコードは生成されません。</p> <p>コンパイル時にオブジェクトファイルを生成する必要がある場合には、<code>-xe</code> オプションを使用します。たとえば、コードの一部を削除し、エラーの原因を分離したい場合には、<code>-xe</code> を使用して、編集とコンパイルのサイクルを高速化することができます。</p>
プロファイリングの高速化: <code>-xprofile_ircache</code>	<p>使用段階において、<code>-xprofile_ircache[=path]</code> と <code>-xprofile=collect use</code> を使用して、収集段階で保存されたコンパイルデータを再利用し、コンパイル時間を短縮することができます。</p> <p>大きなプログラムの場合、中間データが保存されるため、使用段階におけるコンパイル時間が大幅に短縮されます。ただし、データを保存するため、必要なディスク容量が大幅に増える可能性があります。</p>
冗長なテンプレートのインスタンス化の停止 : <code>-instlib=filename</code>	<p><code>-instlib=filename</code> を使用して、ライブラリ、現在のオブジェクト内でのテンプレートインスタンスの重複生成を防ぐことができます。プログラムが多数のインスタンスをライブラリと共有している場合には、<code>-instlib=filename</code> を使用して、コンパイルに要する時間が短縮されるかどうかを確認してみてください。</p> <p>引数 <i>filename</i> を使用して、既存のテンプレートインスタンスが含まれるライブラリを指定します。この引数を指定する場合には、スラッシュ (<code>/</code>) 文字を含める必要があります。現在のディレクトリの相対パスを指定する場合には、ドットとスラッシュ (<code>/</code>) を使用します。<code>-instlib=filename</code> オプションはデフォルト値を持たないため、使用する場合のみ指定します。このオプションは複数指定することができます。</p>

表 6 コンパイルの高速化をサポートする新しい C++ 機能 (続き)

機能	説明
関数の生成: -template=geninlin nefuncs	通常、呼び出されたインラインテンプレート関数がインライン化できない場合を除いて、C++ コンパイラはインラインテンプレート関数を生成しません。しかし、 -template=geninlinefuncs を使用することによって、明示的にインスタンス化されたクラステンプレートの、かつて生成されたことのないインラインメンバー関数をインスタンス化することができます。これらの関数の連結は常にローカルになります。
プリコンパイル済み ヘッダ: -xpch	C++ コンパイラのこのリリースでは、プリコンパイル済みヘッダという新機能が採用されています。アプリケーションのソースファイルが、大量のソースコードを含むインクルードファイルを共有する場合があります。プリコンパイル済みヘッダファイルは、このようなアプリケーションのコンパイル時間を短縮する目的で設計されています。プリコンパイル済みヘッダを使用することによって、1 つのソースファイルから一連のヘッダファイルに関する情報を収集し、そのソースファイルを再コンパイルする場合や、同じ一連のヘッダを持つ他のソースファイルをコンパイルする場合に、その情報を使用することができます。この機能を利用するには、指示語 #pragma hdrstop と -xpch、-xpchstop オプションを組み合わせで使用します。
複数のプロセッサの使 用: -xjobs=n	-xjobs=n オプションを指定して、C++ コンパイラが作業を完了するために作成するプロセスの数を設定します。このオプションを使用して、マルチ CPU マシンにおけるビルド時間を短縮することができます。現在、-xjobs は、-xipo オプションとの組み合わせでのみ動作します。-xjobs=n を指定すると、手続き間オブティマイザは、複数のファイルをコンパイルするために起動することができるコードジェネレータインスタンスの最大数として n を使用します。

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示しています。

表 7 移植の簡略化をサポートする新しい C++ 機能

機能	説明
移植の簡略化： -xmemalign	<p>-xmemalign オプションを使用して、C++ コンパイラがデータの整列に関して行う仮定を制御することができます。メモリアクセスの際のデータの整列ミスの原因となるコードを制御し、整列ミスの発生時のプログラムの動作を制御することによって、コードをより簡単に Solaris 環境に移植することができます。</p> <p>また、-xmemalign オプションを使用して、必要以上に整列されたデータのパフォーマンスを向上させたり、通常以上にデータが入れられた構造体にアクセスすることができます。</p>
文字型の符号の設定： -xchar	<p>-xchar[={signed s unsigned u}] オプションは、文字型が符号なしと定義されているシステムからのコードの移植を簡略化する目的でのみ用意されています。このようなシステムからの移植を行う場合以外は、このオプションを使用しないようにしてください。文字型の符号が重要であるコードのみを再記述し、符号付きまたは符号なしを明示的に示す必要があります。</p>
移植されたコードのデバッグ： -xport64	<p>コードを 64 ビット環境に移植する際に、新しい -xport64 オプションが役に立ちます。このオプションを使用して、V7 (ILP32) のような 32 ビットアーキテクチャから、V9 (LP64) のような 64 ビットアーキテクチャにコードを移植する際に多く発生する、値 (ポインタを含む) の丸め、符号拡張、ビットパッキングの変更などの問題の警告を行うことができます。</p> <p>また、オプション -xnocastwarn を使用すると、明示的なキャストがデータの丸めの原因となっている場合に、64 ビットコンパイルモードにおけるデータ丸めの警告をオフにすることができます。</p>

表 8 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示しています。

表 8 パフォーマンスの向上をサポートする新しい C++ 機能

機能	説明
リンカーによってサポートされる、データのスレッドローカルストレージ: <code>-xthreadvar</code> (SPARC)	<p>リンカーでサポートされる新しいスレッドローカルストレージ機能を使用して、以下を行うことができます。</p> <ul style="list-style-type: none"> スレッド固有データを割り当てるための POSIX インタフェースの高速実装の利用 マルチプロセスプログラムのマルチスレッドプログラムへの変換 スレッドローカルストレージを使用する Windows アプリケーションの Solaris への移植 OpenMP におけるスレッドプライベート変数の高速実装の利用 <p>スレッドローカル変数を宣言してスレッドローカルストレージを使用できるようになりました。スレッドローカル変数の宣言は、通常の変数宣言に変数指示子 <code>_thread</code> とコマンド行オプション <code>-xthreadvar</code> を追加して行います。</p>
ページフォルトの低減: <code>-xF</code>	<p>新機能 <code>-xF</code> を使用して、リンカーによる変数と関数の並べ替えを最適化することができます。この最適化は、実行時間のパフォーマンスに悪影響を与える以下の問題の解決に役立ちます。</p> <ul style="list-style-type: none"> 関係のない変数がメモリー内で隣接することによって生じるキャッシュとページの競合 関係のある変数がメモリー内で隣接しないことが原因で生じる、必要以上に大きな作業セットサイズ データ密度を低下させる変数の未使用のコピーが原因で生じる、必要以上に大きな作業セットサイズ

表 8 パフォーマンスの向上をサポートする新しい C++ 機能 (続き)

機能	説明
新しいプラグマ	<p>コードの最適化の強化に役立つ 4 つのプラグマが新たにサポートされるようになりました。これらのプラグマの詳細については、『C++ ユーザーズガイド』を参照してください。</p> <ul style="list-style-type: none"> • <code>#pragma does_not_read_global_data</code> • <code>#pragma does_not_return</code> • <code>#pragma does_not_write_global_data</code> • <code>#pragma rarely_called</code>
実行時間の短縮 : -xlinkopt	<p>-xlinkopt オプションを指定して、再配置可能なオブジェクトファイルのリンク時間を最適化できるようになりました。CC(1)のマニュアルページを参照してください。</p> <p>-xlinkopt を指定すると、リンクされている .o ファイルを変更せずに最適化が行われます。最適化は実行可能プログラムでのみ行われます。-xlinkopt オプションは、プロファイルのフィードバックを使用して、プログラム全体をコンパイルする場合にもっとも効果的です。</p>
実行時間の短縮 : -xpagesize=n	<p>-xpagesize=n オプションを使用してスタックとヒープの優先ページサイズを設定することができます。n の値として 8K、64K、512K、4M、32M、256M、2G、16G を指定するか、デフォルト値を使用することができます。getpagesize(3C) によって返される、ターゲットのプラットフォームの Solaris 環境で有効なページサイズを指定する必要があります。有効なページサイズを指定しないと、要求は実行時に無視されます。pmap(1) または meminfo(2) を使用してターゲットプラットフォームのページサイズを決定することができます。</p> <p>この機能は Solaris 9 環境でのみ使用することができます。このオプションを使用してコンパイルしたプログラムは、Solaris 9 よりも前の Solaris 環境でリンクすることはできません。</p>

表 9 は、C++ コンパイラに新たに追加されたエラーと警告の制御を示しています。

表 9 C++ コンパイラに新たに追加されたエラーと警告の制御

機能	説明
警告メッセージのフィルタリング: <code>-erroff</code>	C++ コンパイラのフロントエンドから新しい <code>-erroff</code> オプションを使用して警告メッセージを抑制できるようになりました。エラーメッセージとドライバからのメッセージは影響を受けません。また、 <code>-erroff</code> を使用して特定の警告メッセージを選び、そのメッセージのみを抑制したり、発行されるようにすることができます。
コンパイルの中止: <code>-errtags, -errwarn</code>	C++ コンパイラが特定の警告メッセージを発行する場合、コンパイラオプション <code>-errtags</code> 、 <code>-errwarn</code> を使用してコンパイルを中止することができますようになりました。 <code>-errtags=yes</code> を設定して警告メッセージのタグを見つけ、 <code>-errwarn=tag</code> を指定します。 tag には、 <code>-errtags</code> が返す、そのメッセージ固有の識別子を指定します。 <code>-errwarn=%all</code> を指定すると、どの警告メッセージが発行された場合でもコンパイルを中止することができます。CC(1) のマニュアルページの <code>-xwe</code> も参照してください。
標準ライブラリ名のフィルタリングの強化 <code>:-filt=[no%]stdlib</code>	デフォルトで設定されるオプション <code>-filt=[no%]stdlib</code> は、リンカーエラーメッセージ、コンパイラエラーメッセージ中の標準ライブラリの名前を簡略化します。これによって、標準ライブラリ関数の名前を簡単に識別できるようになります。このフィルタリングを無効にするには、 <code>-filt=no%stdlib</code> を指定します。

Fortran コンパイラ

Sun ONE Studio 8, Compiler Collection は、Fortran 77 の従来のプログラムとの互換性をサポートする Fortran 95 コンパイラ、f95 を備えています。Fortran 77 の従来のプログラムの Fortran 95 コンパイラへの移行についての詳細は、『Fortran ユーザーズガイド』の「FORTRAN 77 の互換性：Fortran 95 への移行」の章を参照してください。

表 10 は、Fortran 95 コンパイラの新機能を示しています。詳細は、『Fortran ユーザーズガイド』、『Fortran プログラミングガイド』、『Fortran ライブラリ・リファレンス』を参照してください。

表 10 Fortran 95 コンパイラの新機能

機能	オプション	説明
Fortran 2000 の機能		Fortran 95 コンパイラの、このリリースでは、Fortran 2000 の標準のドラフト (PDF 形式の文書が http://www.dkuug.dk/jtc1/sc22/open/n3501.pdf に掲載されています) で規定されている以下の機能が実装されています。 <ul style="list-style-type: none">• 例外割り込みと IEEE 演算機能• C との相互運用性• PROTECTED 属性• ASYNCHRONOUS 入出力指示子
f77 との互換性の強化		多くの新機能によって、従来の Fortran 77 コンパイラである f77 と Fortran 95 コンパイラとの互換性が強化されています。以下のような新機能が導入されています。 <ul style="list-style-type: none">• 変数形式表現 (Variable Format Expression、VFE)• 長い識別子• -arg=loc• -vax コンパイラオプション

表 10 Fortran 95 コンパイラの新機能 (続き)

機能	オプション	説明
入出力エラーハンドラ		<p>ユーザーは 2 つの新しい機能を使用して、論理ユニットの書式付き入力のエラー処理ルーチンを独自に指定することができます。書式の誤りが検出されると、ランタイム入出力ライブラリは、エラーの原因となった入力行中の文字を示すデータを使用して、ユーザーが用意し、指定したハンドラルーチン呼び出します。ハンドラルーチンは新しい文字を供給し、その文字を使用してエラーの検出場所から入出力オペレーションを継続するか、デフォルトの Fortran エラー処理を使用することができます。</p> <p>新たに導入されたルーチン、 SET_IO_ERR_HANDLER(3f) と GET_IO_ERR_HANDLER (3f) はモジュールサブルーチンであり、呼び出し側のルーチンでは USE SUN_IO_HANDLERS が必要です。 これらのルーチンの詳細については、マニュアルページを参照してください。</p>

表 10 Fortran 95 コンパイラの新機能 (続き)

機能	オプション	説明
符号なし整数		<p>このリリースから、Fortran 95 コンパイラは、新しいデータ型である UNSIGNED を言語の拡張として使用できるようになりました。</p> <p>UNSIGNED では 1、2、4、8 という 4 つの KIND パラメータ値を使用することができ、それぞれ、1、2、4、8 バイトの符号なし整数に対応します。</p> <p>符号なし整数定数は、数字列の後ろに大文字または小文字の U を付けて表現します。オプションで、U の後ろに下線 (_) に続けて KIND パラメータを指定することができます。</p>
優先スタック、ヒープページサイズ	-xpagesize	<p>新しいコンパイラオプション</p> <p>-xpagesize を使用して、プログラムの起動時に優先スタック、ヒープページサイズを設定することができます。たとえば、-xpagesize=4M と指定し、Solaris 9 環境のスタックとヒープのページサイズを、4M バイトに設定することができます。サイズは、一連の事前設定値から選択します。</p> <p>スタック、ヒープのページサイズは、-xpagesize_stack、-xpagesize_heap を使用して個別に設定することもできます。</p> <p>この機能は Solaris 9 環境でのみ使用することができます。このフラグを使用してコンパイルしたプログラムは、Solaris 9 よりも前の Solaris 環境でリンクすることはありません。</p>

表 10 Fortran 95 コンパイラの新機能 (続き)

機能	オプション	説明
プロファイリングの 高速化	<code>xprofile_ircache=pat h</code>	<p>このリリースでは、プロファイルのフィードバック中のコンパイル段階を高速化する目的で、新しいコマンド行オプション <code>-xprofile_ircache=path</code> を導入しています。</p> <p>このフラグを指定すると、Fortran コンパイラは、収集コンパイル段階 <code>-xprofile=collect</code> において、<code>path</code> に指定された場所に中間データを保存し、<code>-xprofile=use</code> 段階中に再利用するため、この情報を再度生成する必要がなくなります。大規模なプログラムでは、<code>-xprofile=use</code> 段階のコンパイル時間を大幅に短縮することができます。</p>
既存のライブラリの 強化	<code>-xknown_lib</code>	<p><code>-xknown_lib</code> オプションが、Basic Linear Algebra Subprograms (BLAS) ライブラリからより多くのルーチンをインクルードし、3 つのサブオプションを導入することによって強化されています。</p> <p>Fortran コンパイラは、BLAS ライブラリルーチンを選択する呼び出しを認識し、Sun Performance Library の実装の適切な最適化を自由に行うことができます。</p>

表 10 Fortran 95 コンパイラの新機能 (続き)

機能	オプション	説明
リンク時間の最適化	-xlinkopt	<p>新たに導入された -xlinkopt フラグを使用してコンパイルとリンクを行い、最適化ツールを起動し、リンク時に生成されたバイナリオブジェクトコードに先進のパフォーマンス最適化を多数適用することができます。</p> <p>このオプションは、プロファイルのフィードバックを使用してプログラム全体をコンパイルする際に使用するのがもっとも効果的です。</p>
局所変数の初期化	-xcheck=init_local	<p>-xcheck オプションフラグの新しい拡張を使用して、局所変数の特別な初期化を行うことができます。</p> <p>-xcheck=init_local を指定してコンパイルを行うと、局所変数が、プログラムによって割り当てられる前に使用されると演算例外の原因となりうる値に初期化されます。ALLOCATE 文を使用して割り当てられるメモリーもこの方法で初期化されます。SAVE 変数、モジュール変数、COMMON ブロック内の変数は初期化されません。</p>

表 10 Fortran 95 コンパイラの新機能 (続き)

機能	オプション	説明
-openmp オプション の強化	-openmp	オプションフラグ -openmp が強化され、OpenMP プログラムのデバッグが簡単に行えるようになっています。dbx を使用して OpenMP アプリケーションのデバッグを行うには、 -openmp=noopt -g を使用してコンパイルを行います。 これによって、dbx を使用して、複数の領域でブレークポイントを設定し、変数の内容を表示することができるようになります。
複数のプロセスのコンパイル	-xjobs= <i>n</i>	-xjobs= <i>n</i> と -xipo を指定すると、手続き間の最適化ツールが <i>n</i> に指定した数のコードジェネレータインスタンスを起動し、コマンド行で指定されたファイルのコンパイルを行います。このオプションを使用して、マルチ CPU マシンにおける大規模なアプリケーションのビルド時間を大幅に短縮することができます。

表 10 Fortran 95 コンパイラの新機能 (続き)

機能	オプション	説明
PRAGMA ASSUME を 使用して行う表明	-xassume_control	プリAGMA ASSUME は、Fortran コンパイラのこのリリースの新機能の1つです。このプリAGMAは、プログラマーが把握している条件が手続き中のいくつかのポイントで真であることをコンパイラに暗示します。これによって、コンパイラがコードの最適化をよりうまく行えるようになります。また、プログラマーは、表明を使用して、プログラムの実行中にプログラムの正当性をチェックすることができます。新たに導入された -xassume_control フラグを使用して、ASSUME プリAGMAの処理の方法を定めることができます。
OpenMP による、明示的にスレッド化されたプログラムのサポート	-xopenmp	このリリースの OpenMP API の実装は、明示的にスレッド化されたプログラムをサポートします。

コマンド行デバッガ dbx

表 11 は、新リリースの dbx コマンド行デバッガの新機能を示しています。これらの機能に関する詳細は、『dbxコマンドによるデバック』を参照してください。

表 11 dbx の新機能

機能	説明
複数の言語が混在するコードで構成されるプログラムをデバック	dbx は、以下の C99 の型をサポートするようになりました。 <ul style="list-style-type: none">• complex• imaginary• double complex• double imaginary• long double complex• long double imaginary これらの型を含む変数や式の値を出力することができます。
OpenMP プログラムのデバックのサポート:	dbx は、Fortran 95、C++、C での OpenMP プログラムのデバックをサポートするようになりました。dbx は、Fortran 95 コンパイラ、C++ コンパイラ、C コンパイラによって生成された OpenMP のコードにおいてスレッド、スタック、関数、パラメータ、変数を正しく表示することができます。
detach コマンドの新しいオプション -stop	detach -stop コマンドは、ターゲットプログラムから dbx を切り離し、プロセスを停止状態にします。-stop オプションを使用することによって、排他アクセスによってアクセスが阻止される可能性のある他の /proc ベースのデバッグツールを、一時的に利用することができます。
新しいイベント修飾子 -resumeone	イベントハンドラの新しい修飾子 -resumeone は、マルチスレッドプログラムにおいて関数呼び出しを使用して状態を発生させる際に役立ちます。

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 12 は、新リリースの Sun Performance Library の新機能を示しています。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 12 Sun Performance Library の新機能

機能	説明
パフォーマンスの向上	<p>Sun Performance Library の新リリースでは、以下のようなパフォーマンスの向上が実現されています。</p> <ul style="list-style-type: none">• BLAS と FFT のパフォーマンスの向上: US-III の小規模な問題サイズの GEMM パフォーマンスと、V9 ライブラリの 32 ビット FET ルーチンを使用する場合の小規模な問題サイズの FET パフォーマンスが向上しています。• スパースソルバのパフォーマンスの向上: Sun Performance Library のスパースソルバのシングル CPU パフォーマンスが強化され、Sun Performance Library のスパースソルバが並列化されています。• スパース BLAS パフォーマンスの向上: スパースマトリックスベクトルオペレーションが並列化され、小規模な問題サイズのパフォーマンスが向上しています。

表 12 Sun Performance Library の新機能 (続き)

機能	説明
ポータブルライブラリのパフォーマンス	Sun Performance Library の新リリースでは内部変更が行われ、パフォーマンスの最適化を簡単に行えるようになっていました。実行時には、実行可能プログラムが動作する SPARC ハードウェアプラットフォーム用に最適化された Sun Performance Library のバージョンが動的に読み込まれます。これは、Sun Performance Library の共有ライブラリバージョンがリンクされている場合 (デフォルト) のみ実行されます。
スパースソルバの新機能	スパースソルバは、エルミート正定値行列をサポートするようになりました。
複合並列化モデル	Sun Performance Library の新リリースには複合並列化モデルが含まれており、Sun Performance Library に添付されるライブラリの数が増え、Sun Performance Library のサイズが小さくなっています。 並列化モデルを組み合わせることによって、Sun Performance Library を使用して直列、並列処理を行うためのリンキングが簡略化されます。
区間 BLAS のマニュアルページが man3pi フォルダに移動	区間 BLAS のマニュアルページが man3pi フォルダに移動しました。 区間 BLAS の各ルーチンで使用する Fortran 95 インタフェースと引数の型については、各ルーチンの 3pi マニュアルページを参照してください。たとえば、constructv_i.3pi ルーチンのマニュアルページを表示するには、man -s 3pi constructv_i と入力します。ルーチンの名前は小文字で入力する必要があります。

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。詳細については、dmake の マニュアルページを参照してください。

表 13 dmake の新機能

機能	説明
dmake のメモリ使用量の減少	結果はさまざまな要因に左右されますが、ヒープメモリ量が 50% ～ 60% 減少しています。
整合性の向上	dmake と Solaris の make の整合性が実現されています。
dmake は、並列ジョブの限界を自動的に調整し、過負荷を防止できるようになりました。	<p>環境変数 DMAKE_ADJUST_MAX_JOBS を設定し、並列ジョブの限界を自動的に調整し、過負荷を防止することができます。</p> <ul style="list-style-type: none">• YES を設定すると、dmake は、システムの現在の負荷に従って並列ジョブの限界を調整します。システムが過負荷状態でない場合、dmake はユーザーが定義した限界値を使用します。システムが過負荷状態の場合、dmake は、現在の限界値をユーザーが定義した限界値よりも低く設定します。この環境変数を設定しない場合、dmake は、システムの現在の負荷に従って並列ジョブの限界を調整します。これは、dmake のデフォルトの設定です。• NO を指定すると、dmake は並列ジョブの限界の自動調整メカニズムをオフにします。

パフォーマンス解析ツール

表 14 は、Sun ONE Studio 8, Compiler Collection のパフォーマンス解析ツールのデータ収集と表示の新機能を示しています。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。

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

表 14 パフォーマンス解析ツールの新機能

機能	説明
Java™ プログラムのクロックベースのプロファイリングとハードウェアカウンタオーバーフロープロファイリングのサポート	クロックベースのプロファイリング、ハードウェアカウンタオーバーフロープロファイリング、同期遅延追跡、記憶域割り当てにおいて Java が完全にサポートされるようになりました。ターゲットのマシン表現、Java 表現のためにデータが収集されます。libcollector を呼び出すための Java API が用意されています。
クロックベースのプロファイリング	高クロック周波数をサポートするバージョンの Solaris 環境ではシステムクロック周波数の倍数のクロックを使用してクロックベースのプロファイリングを行う必要がありましたが、この制限はなくなりました。collect コマンドを引数を指定せずに実行すると、サポートされるプロファイリング間隔の範囲が返されます。
ロードオブジェクトのアーカイブ	ロードオブジェクトのアーカイブは、collect コマンドの -A オプション、または、dbx collector archive コマンドを使用して制御することができます。
データ収集を中断、再開するための API (Application Programming Interface)	個々のスレッドのデータ収集を中断、再開するための API が用意されています。
メモリアクセスイベントをカウントするハードウェアカウンタ	メモリアクセスイベントをカウントするハードウェアカウンタの名前の先頭に "+" を指定して、Collector によるプログラムカウントとイベントを発生させた仮想アドレスの検索を開始することができます。

表 14 パフォーマンス解析ツールの新機能 (続き)

機能	説明
CPU 別のフィルタリング	パフォーマンスアナライザと <code>er_print</code> ユーティリティに CPU 別のフィルタリング機能が追加されています。この機能は、Solaris バージョン 9 よりも前のバージョンでは使用できません。この機能は、 <code>er_print</code> 、 <code>cpu_select</code> 、 <code>cpu_list</code> コマンドで使用することができます。
表示リスト	パフォーマンス解析ツールの表示リストには、ソース行とプログラムカウンタの計量値が表示されます。表示リストは、パフォーマンスアナライザの「行」タブと「PC」タブに表示されます。もしくは <code>er_print</code> ユーティリティで <code>lines</code> コマンド、 <code>pcs</code> コマンドを使って生成します。パフォーマンスアナライザの「概要」タブには、選択したソース行の計量値がすべて表示されます。ソース行とプログラムカウンタのサマリーパネルは、 <code>er_print</code> ユーティリティで <code>lsummary</code> コマンド、 <code>psummary</code> コマンドを使用して表示することができます。
「タイムライン」タブ	<p>「タイムライン」オプションダイアログボックスは「データ表示方法の設定」ダイアログボックスと合体し、「タイムライン」タブになりました。</p> <p>「タイムライン」タブでは、LWP、スレッド、CPU のデータバーを表示することができます。「データ表示方法の設定」ダイアログボックスの「タイムライン」タブを使用して、どのデータを表示するかを選択します。</p> <p>「タイムライン」タブでは、呼び出しスタックをルート関数またはリーフ関数用に調整することができます。また、可視フレームの数を設定することができます。選択は、「データ表示方法の設定」ダイアログボックスの「タイムライン」タブを使用して行います。</p>

表 14 パフォーマンス解析ツールの新機能 (続き)

機能	説明
オブジェクトの選択	ソース行、プログラムカウンタ、関数もオブジェクトとして選択できるようになりました。選択されたオブジェクトはメニューバーに表示され、オブジェクトの計量値は「概要」タブに表示されます。別のタブに移動するたびに、選択したオブジェクトが表示されます。とくに、ソースまたは逆アセンブリに切り替えると、選択した関数の最初の行または命令ではなく、選択した行または関数に移動します。
派生プロセスに関する実験	親プロセスに関する実験が読み込まれると、派生プロセスに関する実験が自動的に読み込まれます。ただし、派生プロセスのデータの表示は無効になっています。パフォーマンスアナライザの「データをフィルタ」ダイアログボックスを使用するか、 <code>er_print</code> ユーティリティで <code>experiment_select</code> コマンドを使用して派生プロセスに関する実験のデータの表示を有効にする必要があります。
「リーク一覧」タブ	「リーク一覧」タブには、リークデータまたは割り当てデータがグラフィカル表示されます。また、リークまたは割り当ての呼び出しスタックの表示と、呼び出しスタック上の関数のソースまたは逆アセンブリの選択による他のデータ表示を行うことができます。
Java モード	Java で記述されたアプリケーションに関する実験は、Java モードを <code>on</code> 、 <code>expert</code> 、 <code>off</code> のいずれかに設定し、表示することができます。

マニュアルについて

この節では、Sun ONE Studio 8, Compiler Collection のマニュアルの新機能について説明します。

- マニュアル『dbx コマンドによるデバック』に「OpenMP プログラムのデバック」という章が追加されました。この章では、OpenMP インタフェースを使用して明示的な並列化を行うアプリケーションのデバックを、コマンド行デバッガ dbx を使用して行う方法を説明しています。
- 『FORTRAN 77言語リファレンス』の旧リリースは、docs.sun.com から参照することができます。このマニュアルは新リリースでは更新されていません。
- Sun ONE Studio 8, Compiler Collection 製品マニュアルは、技術的な補足をすることで、身体の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。詳細は、x ページの「アクセシブルな製品マニュアル」を参照してください。