# Sun Studio 11 の新機能

Sun™ Studio 11

Sun Microsystems, Inc.
www.sun.com

Part No. 819-4752-10
2005 年 11 月, Revision A

Copyright © 2005 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。

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、Java、および JavaHelp 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。

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

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。

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**: *What's New : Sun Studio 11*
Part No: 819-3682-10
Revision A

Please Recycle

Adobe PostScript

# 目次

# はじめに

本書では、この Sun™ Studio 11 ソフトウェアおよび Sun Studio 10 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

**表 P-1**　書体と記号について

| 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、コード例。 | `.login` ファイルを編集します。<br>`ls -a` を実行します。<br>`% 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します。 | `% su`<br>`Password:` |
| *AaBbCc123* または**ゴシック** | コマンド行の可変部分。実際の名前や値と置き換えてください。 | `rm` *filename* と入力します。<br>`rm` **ファイル名** と入力します。 |
| 『 』 | 参照する書名を示します。 | 『Solaris 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。<br>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こえる場合に、継続を示します。 | `% grep '^#define \`<br>`XV_VERSION_STRING'` |

* 使用しているブラウザにより、これら設定と異なって表示される場合があります。

表 P-2　コードについて

| コードの記号 | 意味 | 記法 | コード例 |
|---|---|---|---|
| [ ] | 角括弧にはオプションの引数が含まれます。 | O[*n*] | -O4, -O |
| { } | 中括弧には、必須オプションの選択肢が含まれます。 | d{y\|n} | -dy |
| \| | 「パイプ」または「バー」と呼ばれる記号は、その中から 1 つだけを選択可能な複数の引数を区切ります。 | B{dynamic\|static} | -Bstatic |
| : | コロンは、コンマ同様に複数の引数を区切るために使用されることがあります。 | R*dir*[:*dir*] | -R/local/libs:/U/a |
| ... | 省略記号は、連続するものの一部が省略されていることを示します。 | -xinline=*f1*[,...*fn*] | -xinline=alpha,dos |

# 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 | $ |
| スーパーユーザー (シェルの種類を問わない) | # |

# サポートされるプラットフォーム

この Sun Studio のリリースは、SPARC® および x86 ファミリ (UltraSPARC®、SPARC64、AMD64、Pentium、Xeon EM64T) プロセッサアーキテクチャをサポートしています。サポートされるシステムの、Solaris オペレーティングシステムのバージョンごとの情報については、http://www.sun.com/bigadmin/hcl にあるハードウェアの互換性に関するリストで参照することができます。ここには、すべてのプラットフォームごとの実装の違いについて説明されています。

このドキュメントでは、x86 関連の用語は次のものを指します。

- 「x86」は、64 ビットおよび 32 ビットの、x86 と互換性のある製品を指します。
- 「x64」は、AMD64 または EM64T システムで、特定の 64 ビット情報を指します。
- 「32 ビット x86」は、x86 システムで特定の 32 ビット情報を指します。

サポートされるシステムについては、ハードウェアの互換性に関するリストを参照してください。

# 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

Sun Studio ソフトウェアおよびマニュアルページは、`/usr/bin/` と `/usr/share/man` ディレクトリにはインストールされません。ソフトウェア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 (vii ページの「ソフトウェアへのアクセス方法」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)、`ksh`(1)、および `bash`(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 のソフトウェアが Solaris プラットフォームでは `/opt` ディレクトリ、および Linux プラットフォームでは `/opt/sun` 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デフォルト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。

---

## ソフトウェアへのアクセス方法

`PATH` 環境変数を変更してソフトウェア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. Solaris プラットフォームでは、出力内容から /opt/SUNWspro/bin を含むパスの文字列を検索します。Linux プラットフォームでは、出力内容から /opt/sun/sunstudio11/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このパスがない場合は、次の手順に従って、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

- Solaris プラットフォームでは、次のパスを PATH 環境変数に追加します。以前に Forte Developer ソフトウェア、Sun ONE Studio ソフトウェア、または Sun Studio のほかのリリースをインストールしている場合は、インストール先のパスの前に、次のパスを追加します。

  /opt/SUNWspro/bin

- Linux プラットフォームでは、次のパスを PATH 環境変数に追加します。

  /opt/sun/sunstudio11/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 環境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 のマニュアルページを表示します。

```
% man dbx
```

2. 出力を確認します。

   dbx(1) の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ソフトウェアの現在の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 環境変数を設定してマニュアルページにアクセスする

- Solaris プラットフォームでは、次のパスを MANPATH 環境変数に追加します。

  /opt/SUNWspro/man

- Linux プラットフォームでは、次のパスを MANPATH 環境変数に追加します。

  /opt/sun/sunstudio11/man

## 統合開発環境へのアクセス方法

Sun Studio 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 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。

IDE が正しく動作するかどうかは、IDE がコアプラットフォームを検出できるかどうかに依存します。このため、sunstudio コマンドは、次の 2 つの場所でコアプラットフォームを探します。

- コマンドは、最初にデフォルトのインストールディレクトリを調べます。Solaris プラットフォームでは /opt/netbeans/3.5V11 ディレクトリ、および Linux プラットフォームでは /opt/sun/netbeans/3.5V11 ディレクトリです。
- このデフォルトのディレクトリでコアプラットフォームが見つからなかった場合は、IDE が含まれているディレクトリとコアプラットフォームが含まれているディレクトリが同じであるか、同じ場所にマウントされているとみなします。たとえば Solaris プラットフォームで、IDE が含まれているディレクトリへのパスが /foo/SUNWspro の場合は、/foo/netbeans/3.5V11 ディレクトリにコアプラットフォームがないか調べます。Linux プラットフォームでは、たとえば IDE が含まれているディレクトリへのパスが /foo/sunstudio11 の場合は、/foo/netbeans/3.5V11 ディレクトリにコアプラットフォームがないか調べます。

sunstudio が探す場所のどちらにもコアプラットフォームをインストールしていないか、マウントしていない場合、クライアントシステムの各ユーザーは、コアプラットフォームがインストールされているか、マウントされている場所 (*/installation_directory*/netbeans/3.5V11) を、SPRO_NETBEANS_HOME 環境変数に設定する必要があります。

Solaris プラットフォームでは、Forte Developer ソフトウェア、Sun ONE Studio ソフトウェア、または他のバージョンの Sun Studio ソフトウェアがインストールされている場合、IDE の各ユーザーは、$PATH のそのパスの前に、

*/installation_directory*/SUNWspro/bin を追加する必要もあります。Linux プラットフォームでは、他のバージョンの Sun Studio ソフトウェアがインストールされている場合、IDE の各ユーザーは、$PATH のそのパスの前に、*/installation_directory*/sunstudio11/bin を追加する必要もあります。

$PATH には、*/installation_directory*/netbeans/3.5V11/bin のパスは追加しないでください。

# Sun Studio マニュアルへのアクセス方法

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。
  Solaris プラットフォーム: file:/opt/SUNWspro/docs/ja/index.html
  Linux プラットフォーム:
  file:/opt/sun/sunstudio11/docs/ja/index.html

  製品ソフトウェアが Solaris プラットフォームで /opt、Linux プラットフォームで /opt/sun 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[sm] 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 Web サイト (http://docs.sun.com) から、Sun 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<br>場所: `http://docs.sun.com` |
| サードパーティ製マニュアル<br>• 『Standard C++ Library Class Reference』<br>• 『標準 C++ ライブラリ・ユーザーズガイド』<br>• 『Tools.h++ クラスライブラリ・リファレンスマニュアル』<br>• 『Tools.h++ ユーザーズガイド』 | 形式: HTML<br>場所: `file:/opt/SUNWspro/docs/ja/index.html` のマニュアル索引 |
| Readme | 形式: HTML<br>場所: `http://developers.sun.com/prodtech/cc/documentation/ss11/ja/mr/READMEs` の開発者ポータル |

| マニュアルの種類 | アクセシブルな形式と格納場所 |
|---|---|
| マニュアルページ | 形式: HTML<br>場所: `file:/opt/SUNWspro/docs/ja/index.html` (Solaris プラットフォーム)<br>`file:/opt/sun/sunstudio11/docs/ja/index.html` (Linux プラットフォーム)<br>のマニュアル索引 |
| オンラインヘルプ | 形式: HTML<br>場所: IDE 内の「ヘルプ」メニューおよび「ヘルプ」ボタン |
| リリースノート | 形式: HTML<br>場所: `http://docs.sun.com` |

# 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 | 説明 |
|---|---|
| dbx コマンドによるデバッグ | C、C++、Fortran および Java™ プログラミング言語でプログラムのデバッグを行うための、dbx コマンド行デバッガの使用方法を説明しています。 |
| 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。 |

| マニュアルタイトル | 説明 |
| --- | --- |
| 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。 |
| プログラムのパフォーマンス解析 | コレクタおよびパフォーマンスアナライザを使用して、広範囲のパフォーマンスデータの統計的プロファイリングと多数のシステムコールの監視を行う方法を説明しています。また、そのデータを関数、ソース行、命令レベルでプログラム構造に関連付ける方法についても説明しています。 |

# 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 | 説明 |
| --- | --- | --- |
| Solaris Reference Manual Collection | マニュアルページのセクションのタイトルを参照。 | 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、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、ソフトウェアととも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://developers.sun.com` 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://jp.sun.com/service/contacting`

# 第1章

# Sun Studio 11 の新機能と機能強化

Sun™ Studio 11 リリースには、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 11 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` (Solaris プラットフォーム) および `/opt/sun/sunstudio11/docs/ja/index.html` (Linux プラットフォーム) 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、C++、および Fortran の共通機能

C、C++、または Fortran コンパイラで、次の新機能を利用できます。各言語に固有の新機能の詳細については、次の新機能リストを参照してください。

新機能の詳細については、各言語のユーザーガイドまたはマニュアルページを参照してください。

**表 1-1** C コンパイラの新機能

| 機能 | 説明 |
|---|---|
| x86 開発用の新しい `-xarch` フラグ | x86 プラットフォームでの開発用に、`-xarch` オプションで、次の新しいフラグがサポートされるようになりました。<br>• `amd64a`<br>• `pentium_proa`<br>• `ssea`<br>• `sse2a` |
| x86 サポート用の `-xpagesize` オプション | `-xpagesize`、`-xpagesize_heap`、`-xpagesize_stack` の各オプションが、SPARC プラットフォームに加えて x86 プラットフォームで使用できるようになりました。 |
| x86 メモリーモデルのサポート | 新しい `-xmodel` オプションでは、64 ビット AMD アーキテクチャーでカーネル、スモール、ミディアムのメモリーモデルを指定できます。 |
| SSE/SSE2 積分メディア組み込み関数のサポート | このリリースでは、SSE2 128 ビット XMM レジスタの積分メディア命令の組み込み関数をサポートします。ソースコードで `sunmedia_intrin.h` ヘッダーファイルをインクルードし、`-xbuiltin` オプションを指定すると、この機能を利用できます。さらに、これらの組み込み関数では SSE2 サポートが必要なため、`-xarch=sse2`、`-xarch=amd64`、`-xtarget=opteron` などのオプションを指定します。 |
| SSE2 プラットフォーム用の新しい `-xvector` フラグ | `-xvector` オプションでは、ベクタライブラリ関数の呼び出しの自動生成や、SIMD (Single Instruction Multiple Data) 命令の生成が可能です。 |
| SPARC プラットフォーム用バイナリオプティマイザ | 新しい `-xbinopt` オプションを使用すると、`binopt`(1) バイナリオプティマイザによる最適化用に、コンパイラがバイナリファイルを前処理できます。 |

表 1-1　C コンパイラの新機能 (続き)

| 機能 | 説明 |
| --- | --- |
| 新しい SPARC の -xtarget および -xchip の値 | 新しい -xtarget フラグの ultra3iplus、ultra4plus、および ultraT1 を新しい -xchip フラグの ultra3iplus、ultra4plus、および ultraT1 と組み合わせると、UltraSPARC IIIiplus、UltraSPARC T1、および UltraSPARC IVplus プロセッサのコードを生成できます。 |
| STACKSIZE 環境変数の機能拡張 | STACKSIZE 環境変数の構文が拡張され、スレーブスレッドのスタックサイズの単位キーワードを指定できるようになりました。B はバイト、K はキロバイト、M はメガバイト、G はギガバイトです。<br>たとえば、setenv STACKSIZE 8192 ではスレーブスレッドのスタックサイズが 8M バイトに設定されます。1235B ではスレーブスレッドのスタックサイズが 1235 バイトに設定されます。1235G では 1235Gバイトに設定されます。接尾辞の文字が付かない場合のデフォルトは、キロバイトのまま変化ありません。 |
| OpenMP の自動スコープ宣言 | C および C++ プログラムで自動スコープ宣言が可能になりました。この機能については、Sun Studio の『OpenMP API ユーザーズガイド』の第 3 章に説明があります。 |

# C コンパイラ

C コンパイラでは、2 ページの「C、C++、および Fortran の共通機能」に示した機能に加えて、次の新機能も提供されます。

**表 1-2**　C コンパイラの新機能

| 機能 | 説明 |
|---|---|
| デバッガ情報の新しいデフォルト形式 | C コンパイラでは、デフォルトでデバッガ情報が DWARF 形式で生成されるようになりました。この変更は透過的なため、dbx およびパフォーマンスアナライザソフトウェアで DWARF 形式を問題なく扱うことができます。<br>`-xdebugformat=stabs` を指定することで、スタブ形式でデバッガ情報を生成することもできます。 |
| 新しい 2 つのプログラム | • c99<br>`c99 (implicit \| no%implicit)` プログラムを使用すると、暗黙的な関数宣言を発見できます。<br>• `[no_]warn_missing_parameter_info`<br>`[no_]warn_missing_parameter_info` プログラムを使用すると、パラメータ型情報が含まれない関数宣言を発見できます。 |

# C++ コンパイラ

C++ コンパイラでは、2 ページの「C、C++、および Fortran の共通機能」に示した機能に加えて、次の新機能も提供されます。

**表 1-3**　C++ コンパイラの新機能

| 機能 | 説明 |
|---|---|
| デバッガ情報の新しい形式 | C++ コンパイラで、デバッガ情報が DWARF 形式で生成されるようになりました。デフォルトはスタブ形式のままですが、新しいオプション `-xdebugformat` を `-xdebugformat=dwarf` に設定することで DWARF データを生成できます。 |
| 関数テンプレートからの依存静的関数の呼び出し | C++ 標準では、テンプレートパラメータに依存する関数呼び出しが参照できるのは、外部リンケージを持つ参照可能な関数宣言のみになっています。アプリケーションコードが関数テンプレートから依存静的関数を呼び出していて、コンパイラにこの規則を無視させる必要がある場合には、`-features=[no%]tmplrefstatic` を指定します。 |

# Fortran コンパイラ

2 ページの「C、C++、および Fortran の共通機能」に示したもの以外に、このリリースの Fortran 95 コンパイラに新機能はありません。

# コマンド行デバッガ dbx

表 1-4　dbx の新機能

| 機能 | 説明 |
|---|---|
| Linux オペレーティングシステムにおける AMD64 アーキテクチャのサポート | 64 ビット dbx が AMD64 アーキテクチャをサポートするようになりました。Linux OS では、64 ビットの dbx で 32 ビットプログラムをデバッグできません。Linux OS で 32 ビットプログラムをデバッグするには、dbx コマンドに -x exec32 オプションを付けて 32 ビットの dbx を起動します。 |
| Sun Studio C コンパイラの DWARF シンボリックデバッグ情報生成のサポート | C コンパイラからの DWARF シンボリックデバッグ情報がデフォルトで使用されます。 |
| thr_create および thr_exit イベント | thr_create イベントは、スレッドまたは *thread_id* で指定されたスレッドが作成されたときに発生します。thr_exit イベントは、スレッドが終了したときに発生します。 |
| step_abflow 環境変数 | この環境変数が stop に設定されていると、シングルステップ実行時に dbx が longjmp()、siglongjmp() で停止し、文をスローします。ignore に設定されていると、dbx は longjmp() および siglongjmp() の異常制御フロー変更を検出しません。 |
| intercept および unintercept コマンドの構文修正 | 構文に -set オプションが含まれるようになりました。このオプションは、インターセプトリストと除外リストの両方をクリアし、リストを指定した型のみをスローするインターセプトまたは除外に設定します。 |
| Java™ 2 プラットフォーム、Standard Edition v 5.0 アップデート 3 の javac コンパイラでコンパイルされた Java プログラムのデバッグサポート | J2SE 5.0 アップデート 3 でコンパイルされた Java プログラムのデバッグが、次の制限付きでサポートされました。<br>- generic の一部サポート<br>- Autoboxing と Unboxing のサポートなし<br>- 静的インポートのサポートなし |

# OpenMP API

表 1-5　OpenMP API の新機能

| 機能 | 説明 |
| --- | --- |
| OpenMP 2.5 | OpenMP の実装が 2.5 仕様にアップグレードされました。詳細については、OpenMP の Web サイト (`http://www.openmp.org/`) を参照してください。 |
| C++ 自動スコープ宣言 | C および Fortran 95 に加えて、C++ プログラムでも変数の自動スコープ宣言が可能になりました。自動スコープ宣言については、『OpenMP ユーザーズガイド』の第 3 章に説明があります。 |

# 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

表 1-6　Sun Performance Library の新機能

| 機能 | 説明 |
|---|---|
| x86 プラットフォームでの Sun Perflib | このリリースの Sun Performance Library には、64 ビット x86 システムの Solaris OS 用のライブラリが含まれています。64 ビット x86 版の Sun Performance Library は、次の点を除き、SPARC v9 版と機能的に同じです。<br>• Quad 精度ルーチン (dqdoti、dqdota) は使用できません。<br>• 区間 BLAS ルーチンは使用できません。<br>• 64 ビット整数のルーチンは使用できないため、DAXPY() は使用できますが、DAXPY_64() は使用できません。<br>• x86 システム用の Solaris OS では、Portable Performance Library 機能を使用できません。<br><br>頻繁に呼び出される amd64 ライブラリ内の多くの BLAS カーネルが最適化されました。一部の内部 FFT ルーチンも、さらに最適化されています。SPARC v9 バージョンと同様に、多数のルーチンが並列化されました。 |
| SPARC プラットフォームでの Sun Perflib | 最新の UltraSPARC プロセッサ用 BLAS および FFT が改良されました。 |

# dmake

表 1-7　dmake の新機能

| 機能 | 説明 |
|---|---|
| 互換モード | GNU make との互換性のため、`-x SUN_MAKE_COMPAT_MODE` コマンド行オプションおよび `SUN_MAKE_COMPAT_MODE` 環境変数が追加されました。 |

# パフォーマンス解析ツール

**表 1-8**　パフォーマンス解析ツールの新機能

| 機能 | 説明 |
|---|---|
| 表示されるタブの操作性向上 | 柔軟性を高めるため、アナライザのタブメカニズムを設計し直しました。読み込まれた 1 つ以上の実験に対応するタブのみが表示され、特に大規模な実験ではすべてのタブが表示されるのではなく、デフォルトセットのタブが表示されます。デフォルトのタブは、.er.rc ファイルに tabs 指令付きで設定します。「データ表示方法の設定」ダイアログボックスの「タブ」タブを使用すると、表示されるタブを追加または削除できます。 |
| 「フィルタ」ダイアログボックスの「詳細」タブ | 「フィルタ」ダイアログボックスに「詳細」タブが表示され、フィルタ式を入力できるようになりました。式は、AND および OR 演算子、および「関数」タブ、「データオブジェクト」タブ、「データレイアウト」タブ、または「メモリーオブジェクト」タブの 1 つまたは複数の選択内容を反映したフレーズを使用して作成することもできます。 |
| 「タイムライン」タブがフィルタリングを反映 | 「タイムライン」タブには、現在のフィルタ設定に適合したイベントのみが表示されます。 |
| 派生プロセス処理の改善 | アナライザおよび `er_print` ユーティリティは、`.er.rc` ファイル内の en_desc on\|off 指令を処理します。この指令に on が指定されているとすべての派生実験がただちに読み取られ、指令に `off` が指定されていると元の実験のみが読み取られます。 |
| アナライザの「新規ウィンドウ」の動作改善 | 「新規ウィンドウ」ツールバーボタンをクリックするか、「ファイル」→「新規ウィンドウ作成」を選択して表示されるアナライザの追加ウィンドウが、明確に区別して表示されるようになりました。読み込まれる実験は同じですが、フィルタリング、メトリック、並べ替えなどはウィンドウごとに独立して設定できます。 |
| 新しいメモリーオブジェクトタブとレポート | キャッシュ行、ページなどのパフォーマンスデータを表示する新しいタブが、アナライザで利用できるようになりました。メモリーオブジェクト用には、いくつかの新しい `er_print` コマンドを使用できます。「データの表示方法の設定」ダイアログボックスで「タブ」タブの「カスタムオブジェクトを追加」をクリックすると、カスタムメモリーオブジェクトを作成できます。 |
| Linux でのハードウェアカウンタのプロファイリング | サポートされる Linux システムで、ハードウェアカウンタのオーバーフロープロファイリングを利用できます。このサポートには、Perfctr パッチのインストールが必要です。 |
| MPI プロファイリング処理の改善 | MPI の LAM および MPICH バージョンのプロセスランクを指定する変数が認識されます。 |

表 1-8　パフォーマンス解析ツールの新機能 (続き)

| 機能 | 説明 |
|---|---|
| Java モードから表示モードへの変更 | Java モードが表示モードに置き換えられました。表示モード設定の user、expert、および machine は、Java モード設定の on、expert、および off 設定に対応しています。表示モードは Java プログラム、特に OpenMP 以外のプログラミングモデルに適しています。`javamode` コマンドも使用できますが、警告が表示されます。 |
| `er_print` コマンドの変更 | `er_print` ユーティリティのさまざまなコマンドが変更されました。データオブジェクトに影響を与えるコマンドの名前が変更され、メトリックに関するコマンド処理の一貫性が高められました。新しい `procstat` コマンドは、データ処理に関する情報を出力します。新しい `filters` コマンドを使用すると、フィルタ式を指定できます。フィルタの定義、およびメモリーオブジェクトインデックスの計算用として、新しい式の文法が追加されました。 |
| 「関数」タブ、「データオブジェクト」タブ、「データレイアウト」タブ、タブでの複数選択 | 「関数」タブ、「データオブジェクト」タブ、「データレイアウト」タブ、および「メモリーオブジェクト」タブで、複数の項目を選択できるようになりました。 |
| フィルタリングの改善 | メトリックスを表示する実験の選択、およびサンプル、スレッド、LWP、CPU のフィルタリングに加えて、式を指定し、真と評価されたデータレコードを表示できるようになりました。 |

# 統合開発環境 (IDE)

表 1-9　IDE の新機能

| 機能 | 説明 |
|---|---|
| J2SE 5.0 アップデート 3 のサポート | IDE が J2SE 5.0 アップデート 3 で動作するようになりました。 |

# マニュアル類

Sun Studio 11 のマニュアル類の最新情報については、`http://developers.sun.com/prodtech/cc/support_index.html` の開発者向けポータルサイトにある「Latest News」ページをお読みください。

# 第2章

# Sun Studio 10 の新機能と機能強化

Sun™ Studio 10 は、Sun™ Studio 9 の後継となる製品です。Sun Studio 10 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 10 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

表 2-1　C コンパイラの新機能

| 機能 | 説明 |
|---|---|
| OpenMP 並列プログラミング API | Solaris OS が動作する 32 ビットおよび 64 ビット両方の x86 システムで API が使用できるようになりました。 |
| -xarch オプションの追加 | -xarch=amd64 は、64 ビット AMD 命令セット用のコンパイルを指示します。-xarch=amd64 が指定されると、__amd64 および __x86_64 が事前定義されるようになりました。 |
| -xtarget オプションの追加 | -xtarget=opteron は 32 ビット AMD コンパイル用の -xarch、-xchip、および -xcache 設定を指示します。 |
| x86 システム向けの -xregs フラグの追加 | -xregs オプションの新しい x86 専用フラグ、-xregs=[no%]frameptr では、未割り当ての呼び出し先保存レジスタとしてフレームポインタレジスタを使用して、アプリケーションの実行時パフォーマンスの向上を図ることができます。 |
| lint 用の -Xarch = amd64 オプションの追加 | C ユーティリティの lint が、新しいオプションの -Xarch=amd64 を受け付けるようになりました。詳細は lint(1) のマニュアルページを参照してください。 |
| x86 システム向けの -xarch=generic64 | 既存の -xarch=generic64 オプションが、従来の SPARC プラットフォームに加えて x86 プラットフォームをサポートするようになりました。 |
| x86 システム向けの -xipo | x86 システムで -xipo オプションが使用できるようになりました。 |

**注 –** 64 ビットコードを生成するには、コマンド行で -fast および -xtarget の右側に -xarch=amd64 を指定する必要があります。たとえば、cc -fast -xarch=amd64 または cc -xtarget=opteron -xarch=amd64 というように指定します。新しい -xtarget=opteron オプションは、自動的には 64 ビットコードを生成しません。このオプションは、-xarch=sse2、-xchip=opteron 、-xcache=64/64/2:1024/64/16 に展開されて、32 ビットコードになります。-xtarget=native と定義されているマクロのため、-fast オプションもまた 32 ビットコードになります。

# C++ コンパイラ

表 2-2　C++ コンパイラの新機能

| 機能 | 説明 |
|---|---|
| OpenMP 並列プログラミング API | Solaris OS が動作する 32 ビットおよび 64 ビット両方の x86 システムで API が使用できるようになりました。 |
| -xarch オプションの追加 | -xarch=amd64 は、64 ビット AMD 命令セット用のコンパイルを指示します。-xarch=amd64 が指定されると、__amd64 および __x86_64 が事前定義されるようになりました。 |
| -xtarget オプションの追加 | -xtarget=opteron は 32 ビット AMD コンパイル用の -xarch、-xchip、および -xcache 設定を指示します。 |
| x86 システム向けの -xregs フラグの追加 | -xregs オプションの新しい x86 専用フラグ、-xregs=[no%]frameptr では、未割り当ての呼び出し先保存レジスタとしてフレームポインタレジスタを使用して、アプリケーションの実行時パフォーマンスの向上を図ることができます。 |
| x86 システム向けの -xarch=generic64 | 既存の -xarch=generic64 オプションが、従来の SPARC プラットフォームに加えて x86 プラットフォームをサポートするようになりました。 |
| x86 システム向けの -xipo | x86 システムで -xipo オプションが使用できるようになりました。 |
| テンプレートテンプレートパラメータ | 型や値ではなくそれ自体がテンプレートのパラメータを持つテンプレート定義を指定することができます。型でインスタンス化されたテンプレートはそれ自体が型であることを思い出してください。たとえば 14 ページの「テンプレートテンプレートパラメータの使用例」を参照してください。 |
| 入れ子クラスのアクセス規則 | デフォルトモードで、入れ子のクラスから、包含しているクラスの private メンバーにアクセスできるようになりました。詳細は、15 ページの「入れ子クラスのアクセス規則」を参照してください。 |

**注 –** 64 ビットコードを生成するには、コマンド行で -fast および -xtarget の右側に -xarch=amd64 を指定する必要があります。たとえば、CC -fast -xarch=amd64 または CC -xtarget=opteron -xarch=amd64 というように指定します。新しい -xtarget=opteron オプションは、自動的には 64 ビットコードを生成しません。このオプションは、-xarch=sse2、-xchip=opteron 、-xcache=64/64/2:1024/64/16 に展開されて、32 ビットコードになります。-xtarget=native と定義されているマクロのため、-fast オプションもまた 32 ビットコードになります。

# テンプレートテンプレートパラメータの使用例

ここでは、テンプレートテンプレートパラメータを使用していないコードとテンプレートテンプレートパラメータを使用しているコードの 2 つのコード例を紹介します。

この例では、テンプレートテンプレートパラメータを使用していません。
MyClass<int> は型です。

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

この例のクラステンプレート C には、クラステンプレートのパラメータがあり、オブジェクト x は、クラステンプレートの A をその引数として使用する C のインスタンスです。c のメンバー y は、A<int> 型です。

```
// 通常のクラステンプレート
template<typename T> class MyClass {
    T x;
};
// テンプレートパラメータを持つクラステンプレート
template < template<typename U> class V > class C {
    V<int> y;
// テンプレートで C をインスタンス化
C<A> x;
```

# 入れ子クラスのアクセス規則

デフォルトの標準モードで、入れ子のクラスから、包含しているクラスの private メンバーにアクセスできるようになりました。

C++ 規格では、入れ子のクラスには、その入れ子を包含しているクラスのメンバーに対するアクセス特権はないことになっています。しかしながら、メンバー関数は private メンバーにアクセス可能であるため、この制限は妥当ではなく、メンバークラスもアクセスできるべきです。次の例の関数 `foo` は、`outer` クラスの private メンバーへのアクセスを試みます。C++ 規格によれば、フレンド関数宣言されないかぎり、この関数はアクセスできません。

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

C++ 委員会は、メンバー関数が持っているのと同じアクセス権をメンバークラスに付与するという、アクセス規則変更を採用する過程にあります。この言語規則の変更を見越して、多くのコンパイラでこの規則が実装されています。

アクセスを許可しないという、古いコンパイラの動作に戻すには、コンパイラオプション `-features=no%nestedaccess` を使用します。デフォルトは `-features=nestedaccess` です。

# Fortran コンパイラ

表 2-3　Fortran コンパイラの新機能

| 機能 | 説明 |
|---|---|
| OpenMP 並列プログラミング API | Solaris OS が動作する 32 ビットおよび 64 ビット両方の x86 システムで API が使用できるようになりました。 |
| -xarch オプションの追加 | -xarch=amd64 は、64 ビット AMD 命令セット用のコンパイルを指示します。-xarch=amd64 が指定されると、__amd64 および __x86_64 が事前定義されるようになりました。 |
| -xtarget オプションの追加 | -xtarget=opteron は 32 ビット AMD コンパイル用の -xarch、-xchip、および -xcache 設定を指示します。 |
| x86 システム向けの -xarch=generic64 | 既存の -xarch=generic64 オプションが、従来の SPARC プラットフォームに加えて x86 プラットフォームをサポートするようになりました。 |
| x86 システム向けの -xipo | x86 システムで -xipo オプションが使用できるようになりました。 |
| ビッグエンディアンとリトルエンディアン式プラットフォーム間のバイナリ (書式なし) ファイルの共有 | 新しいコンパイラフラグの -xfilebyteorder は、SPARC システムと x86 システムとの間でバイナリ入出力ファイルの移動をサポートします。このフラグは、書式なし入出力ファイルのバイト順序とバイト列を特定します。詳細は、17 ページの「ビッグエンディアンとリトルエンディアン式プラットフォーム間のバイナリファイルの共有」を参照してください。 |

**注 –** 64 ビットコードを生成するには、コマンド行で -fast および -xtarget の右側に -xarch=amd64 を指定する必要があります。たとえば、f95 -fast -xarch=amd64 または f95 -xtarget=opteron -xarch=amd64 というように指定します。新しい -xtarget=opteron オプションは、自動的には 64 ビットコードを生成しません。このオプションは、-xarch=sse2、-xchip=opteron 、-xcache=64/64/2:1024/64/16 に展開されて、32 ビットコードになります。-xtarget=native と定義されているマクロのため、-fast オプションもまた 32 ビットコードになります。

# ビッグエンディアンとリトルエンディアン式プラットフォーム間のバイナリファイルの共有

新しいコンパイラフラグの `-xfilebyteorder` は、SPARC システムと x86 システムとの間でバイナリ入出力ファイルの移動をサポートします。このフラグは、書式なし入出力ファイルのバイト順序とバイト列を特定します。

このフラグの構文は次のとおりです。

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

| | |
|---|---|
| `max_align` | ターゲットプラットフォームの最大バイト列。値は、1、2、4、8、16 のどれかです。境界整列は、C 言語の構造体との互換性を維持するため、プラットフォーム依存のバイト列を使用する Fortran VAX 構造体と Fortran 95 派生型に適用されます。 |
| `little`*max_align*`:{%all,`*unitno*`,`*filename*`}` | 最大バイト列が *max_align* のシステムで使用する「リトルエンディアン」ファイルのファイル名またはその他装置番号。たとえば `little4` は 32 ビット x86 ファイルを表すのに対し、`little16` は 64 ビット x86 ファイルを表します。 |
| `big`*max_align*`:{%all,`*unitno*`,`*filename*`}` | 最大バイト列が *max_align* のシステムで使用する「ビッグエンディアン」ファイルのファイル名またはその他装置番号。 |
| `native:{%all,`*unitno*`,`*filename*`}` | コンパイルプロセッサシステムが使用するのと同じバイト順序およびバイト列のネイティブファイルのファイル名またはその他装置番号。 |
| `%all` | 「SCRATCH」として開かれるか、このオプションで明示的に指定する以外のすべてのファイルとその他論理装置。このフラグで明示的に指定しないデフォルトのファイルを表すのに使用できます。`%all` は、1 回だけ指定できます。 |
| *unitno* | プログラムが開く Fortran 論理装置番号。 |
| *filename* | プログラムが開く Fortran ファイル名。 |

このオプションは、STATUS=scratch で開くファイルには適用されません。これらのファイルに対する入出力処理は、つねにネイティブプロセッサのバイト順序、バイト列で行われます。

コンパイラコマンド行に -xfilebyteorder が指定されていない場合の最初のデフォルトは、-xfilebyteorder=native:%all です。このオプションには、引数を少なくとも 1 つ付ける必要があります。すなわち、little:、big:、native: のいずれか 1 つが存在する必要があります。

このフラグで明示的に宣言されていないファイルは、ネイティブファイルと見なされます。たとえば、-xfilebyteorder=little4:zfile.out を付けて zfile.out をコンパイルした場合と、このファイルは、4 バイトの最大データ整列規則を持つリトルエンディアンの 32 ビット x86 ファイルと宣言され、他のすべてのファイルはネイティブファイルになります。

ファイルに指定されたバイト順序はネイティブプロセッサと同じであるが、バイト列が異なる場合は、バイトスワップが行われないにしても、適切なパディングが使用されます。たとえば -xarch=amd64 を付けた、64 ビット x86 プラットフォーム向けのコンパイルで、-xfilebyteorder=little4:*filename* が指定された場合などがそうです。

ビッグエンディアンとリトルエンディアン式プラットフォーム間で共有されるデータレコード内で宣言する型は、同じサイズである必要があります。たとえば、-xtypemap=integer:64,real:64,double:128 を付けてコンパイルした SPARC 実行可能ファイルの生成するファイルを、-xtypemap=integer:64,real:64,double:64 を付けてコンパイルした x86 実行可能ファイルが読み取ることはできません。これは、両者のデフォルトの倍精度データ型のサイズが異なるためです。

共有入出力ファイルに、UNION/MAP データ構造を含めてはなりません。これは、UNION データをどのように解釈すべきかの情報をコンパイラが持っていないためです。-xfilebyteorder フラグを付けて、UNION データを含むファイルを宣言すると、実行時エラーになります。

# コマンド行デバッガ dbx

表 2-4　dbx の新機能

| 機能 | 説明 |
| --- | --- |
| AMD64 アーキテクチャのサポート | 64 ビット dbx が AMD64 アーキテクチャをサポートするようになりました。 |

SPARC システム用の Sun Studio ソフトウェア同様、x86 システム用の Sun Studio ソフトウェアには、2 つの dbx バイナリが付属しています。1 つは、32 ビットプログラムのみをデバッグ可能な 32 ビット dbx、もう 1 つは、32 ビットと 64 ビット両方のデバッグが可能な 64 ビット dbx です。

dbx を起動すると、どちらのバイナリを実行すべきか自動的に判定されます。64 ビット Solaris OS では、デフォルトは 64 ビット dbx です。

# OpenMP API

表 2-5　OpenMP API の新機能

| 機能 | 説明 |
| --- | --- |
| Solaris 10 OS が動作する x86 システムに対応 | SPARC システムの Solaris OS 用にすでに提供されているのと同じ OpenMP API 機能が、Solaris 10 OS が動作する 32 ビットおよび 64 ビット x86 システムの Sun Studio コンパイラで使用できるようになりました。 |
| `libmtsk` | マルチタスクライブラリの `libmtsk` が共有ライブラリになりました。Solaris 10 OS に付属しています。 |
| 入れ子並列 | 入れ子並列に対応しました。デフォルトでは無効で、有効にするには、`OMP_NESTED` 環境変数を設定して、`omp_set_nested()` 関数に対する実行時呼び出しを行うように設定する必要があります。入れ子並列が有効な場合、並列領域内からの大部分の `omp_` 関数に対する呼び出しは無視されません。並列環境を調整するための呼び出し (たとえば `omp_set_num_threads()` または `omp_set_dynamic()`) は、スレッドが検出したのと同じか内側の入れ子レベルにある以降の並列領域にのみ関係します。 |
| スレッドのデフォルト動作 | スレッドのデフォルト動作が SLEEP になりました。以前は SPIN がデフォルトでした。以前の動作に戻すには、`SUNW_MP_THR_IDLE=SPIN` を使用します。 |

表 2-5　OpenMP API の新機能 (続き)

| 機能 | 説明 |
|---|---|
| SUNW_MP_NUM_POOL_THREADS 環境変数 | SUNW_MP_NUM_POOL_THREADS は、スレッドプールのサイズ (最大スレッド数) を指定します。スレッドプールには、ユーザー以外のスレッド、すなわち、libmtsk ライブラリが作成するスレッドでのみ構成されます。メインスレッドのようなユーザースレッドは含まれません。SUNW_MP_NUM_POOL_THREADS を 0 に設定すると、スレッドプールが強制的に空にされ、すべての並列領域が 1 つのスレッドで実行されます。この環境変数には、負以外の整数を指定します。デフォルト値は 1023 です。この環境変数は、1 つのプロセスがスレッドを作成しすぎないようにする働きをします。スレッドが多すぎると、たとえば、再帰的に入れ子にされた並列領域で問題が発生することがあります。 |
| SUNW_MP_MAX_NESTED_LEVELS 環境変数 | SUNW_MP_MAX_NESTED_LEVELS は、アクティブな並列領域の最大の深さを指定します。並列領域のアクティブな入れ子の深さレベルが SUNW_MP_MAX_NESTED_LEVELS よりも深い場合、その並列領域は単一のスレッドによって実行されます。この環境変数には、正の整数を指定します。デフォルトは 4 です。一番外側の並列領域の深さレベルは 1 です。 |
| SUNW_MP_GUIDED_WEIGHT 環境変数 | SUNW_MP_GUIDED_WEIGHT は、GUIDED スケジュールを持つループに対して libmtsk が使用する重み値を設定します。libmtsk は、次の式を使って、GUIDED ループのチャンクサイズを算出します。*chunk_size= num_unassigned_iterations/*(*weight*num_threads*)<br>*num_unassigned_iterations* は、スレッドにまだ割り当てられていないループの反復回数、*weight* は浮動小数点定数 (デフォルトは以前は 1.0 で、このリリースで 2.0)、*num_threads* は、ループの実行に使用するスレッド数です。SUNW_MP_GUIDED_WEIGHT には、ゼロ以外の正の浮動小数点定数を指定する必要があります。libmtsk は、GUIDED チャンクサイズの計算で、この値を重みとして使用します。 |

# 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

表 2-6　Sun Performance Library の新機能

| 機能 | 説明 |
|---|---|
| 64 ビット Solaris OS に対応 | x86 システム用の 64 ビット Solaris OS に対応しました。 |

64 ビット x86 版の Sun Performance Library は、次の点を除き、SPARC v9 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
- 64 ビット整数パラメータを持つルーチンは使用できない。たとえば DAXPY() は使用できますが、DAXPY_64() は使用できません。

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=amd64 フラグを使用します。次に例を示します。

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

# dmake

**表 2-7** dmake の新機能

| 機能 | 説明 |
|---|---|
| DMAKE_OUTPUT_MODE 環境変数の導入 | 新しい環境変数または makefile マクロの DMAKE_OUTPUT_MODE を使用して、ログファイルの形式を変更することができます。デフォルトでは、または DMAKE_OUTPUT_MODE が TXT1 に設定されている場合、dmake はシステム情報からなる追加行をログファイルに出力し、出力を付けたコマンドが繰り返されます。DMAKE_OUTPUT_MODE が TXT2 に設定されている場合は、システム情報が省略され、コマンドは繰り返されません。詳細は、dmake(1) のマニュアルページの「ENVIRONMENT/MACROS」セクションを参照してください。(マニュアルページのこの環境変数の記述に誤りがあります。DMAKE_OUTPUT_MODE の値は正しくは、TXT1 および TXT2 です。) |
| UNIX 2003 準拠 | DMAKE_COMPAT_MODE=POSIX を設定することによって、強制的に UNIX2003 に準拠させることができます。 |
| Grid Engine のサポート | DMAKE_MODE=grid を設定することによって、Grid Engine をサポートするよう指示できます。 |
| システムの過負荷の制御 | DMAKE_ADJUST_MAX_JOBS を使ってシステムの過負荷を制御できます。 |
| メモリー使用の改善 | このリリースでは、メモリー使用の機能が改善されています。 |

# パフォーマンス解析ツール

表 2-8　パフォーマンス解析ツールの新機能

| 機能 | 説明 |
|---|---|
| 実験の形式の変更 | 実験の形式に変更が加えられました。ログに、ターゲットのサイズをビット単位で示すエントリが追加されました。また、バージョンが 9.1 から 9.2 に変更されたのに伴い、新しい実験は以前のツールで読み取れなくなりましたが、以前の実験は Sun Studio 10 のツールで読み取ることができます。 |
| `er_kernel` ユーティリティ | 新しい `er_kernel` ユーティリティが追加されました (Solaris 10 OS のみ)。この `er_kernel` ユーティリティを使用するには、DTrace へのアクセス権が必要です。 |
| パフォーマンスメトリックの精度の向上 | パフォーマンスアナライザおよび `er_print` の百分率メトリックの精度が小数点以下 1 桁から 2 桁に向上しました。 |
| 実験の注記ファイルの直接編集 | パフォーマンスアナライザに、実験の注記ファイルを直接編集するための機能が追加されました。 |
| 関数名を表示するオプションの追加 | パフォーマンスアナライザと `er_print` コマンドに、関数名を表示するためのオプションが新しく追加されました。 |
| メトリック選択機能の強化 | パフォーマンスアナライザのメトリック選択機能が強化されました。すべてのメトリックを一度に選択したり、選択解除したりできます。 |
| 収集 GUI の変更 | 派生プロセスに使用されていたメニューが「実験を収集」タブに移動されました。メニューは、on および off オプションに加えて、all オプションと拡張ハードウェアカウンタオーバーフロープロファイル機能もサポートするようになっています。 |
| ハードウェアカウンタオーバーフロープロファイル機能の強化 | ハードウェアカウンタオーバーフロープロファイル機能が強化され、x86 系プロセッサを含む多くのプロセッサで使用できるようになりました。この強化機能は、`dbx` で `collect -h` コマンドや `collector hwprofile` コマンドを使用することによって、またパフォーマンスアナライザの GUI から使用できます。 |
| `appendfile` オプションの追加 | `er_print` ユーティリティに、`appendfile` オプションが追加されました。このオプションを使用して、`er_print` ユーティリティからの出力を既存のファイルの最後に負荷することができます。 |
| `er_src` ユーティリティのデフォルト動作の変更 | `er_src` ユーティリティのデフォルト動作が、次のコマンドと同じ動作に変更されました。`er_src -source all -1 object` |
| J2SE テクノロジの場所 | パフォーマンスアナライザおよび `collect` ユーティリティが、製品のインストーラがデフォルトでインストールした場所の J2SE テクノロジを使用するようになりました。 |

表 2-8　パフォーマンス解析ツールの新機能 (続き)

| 機能 | 説明 |
|---|---|
| collect -J java_args オプションの追加 | collect -J java_args オプションは、プロファイリングに使用する Java にフラグ引数を渡す手段を提供します。 |
| 一時停止および再開での標本収集動作の変更 | 標本データは、一時停止の前と再開後に生成されますが、コレクタが一時停止しているときは生成されません。 |
| JVM 関数用の疑似関数 | Java モードにおける Java 仮想マシン (JVM)* 関数用の疑似関数名が、<JVM-Overhead> から <JVM-System> に変更されました。 |
| <Unknown> サブタイプ | Java 関数の <Unknown> サブタイプの名前がもっと分かりやすいものに変更されました。 |
| .er.rc ファイルのパス | 処理済みの .er.rc ファイルのパスが、パフォーマンスアナライザの場合は、「エラー/警告ログ」ウィンドウに表示されるようになりました。er_print および er_src ユーティリティの場合は、stderr に出力されます。 |
| JDK_1_4_2_HOME 環境変数 | データの収集に使用する Java パスを定義するための環境変数 JDK_1_4_2_HOME が廃止されました。 |
| ヒープのプロファイリング | JVM 1.5 でサポート廃止されるため、Java プログラムのヒーププロファイリングが廃止されました。 |
| collect -j のオプション拡張 | collect ユーティリティが、on または off 値を受け付けるほか、プロファイリングに使用する Java へのパスも受け付けるようになりました。 |

* 「Java 仮想マシン (JVM)」という用語は、Java プラットフォーム用仮想マシンを意味します。

# 統合開発環境 (IDE)

表 2-9　IDE の新機能

| 機能 | 説明 |
|---|---|
| スクリプト実行機能 | IDE から直接、スクリプトを実行できるようになりました。 |
| Linux オペレーティングシステムでの `ss_attach` | Linux オペレーティングシステムで動作する Sun Studio ソフトウェアでも、`ss_attach` 機能が使用できるようになりました。 |

# マニュアル類

Sun Studio 10 のマニュアル類の最新情報については、`http://developers.sun.com/prodtech/cc/support_index.html` の開発者向けポータルサイトにある「Latest News」ページをお読みください。